SWI-Prolog i Python

Kiš, Martin

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Martin Kiš

SWI-Prolog i Python

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Martin Kiš

Matični broj: 0016136486

Studij: Informacijski sustavi

SWI-Prolog i Python

ZAVRŠNI RAD

Mentor/Mentorica:

Vlatka Sekovanić, mag. educ. inf.

Varaždin, rujan 2021.

Martin Kiš

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Cilj ovog rada je čitatelju prikazati mogućnosti koje nudi logički programski jezik SWI Prolog, poglavito kada se koristi zajedno sa objektno-orijentiranim jezikom poput Pythona. Prologov jedinstven način korištenja logike otvara nove mogućnoti prilikom kreiranja programskih rješenja te može olakšati rad i učiniti ga efikasnijim. Na primjeru programa koji igra stratešku računalnu igru, bit će prikazani korisni slučajevi u kojima je Prologov način rezolviranja učinio program efikasnijim i smanjio potrebno vrijeme za pretraživanje i iteriranje kroz program.

Ključne riječi: programiranje, SWI-Prolog, Python, PySwip

Sadržaj

1. Uvod

Ovaj rad istražuje mogućnosti izrade programskog rješenja primjenom kombinacije različitih programskih jezika. Kao konkretni primjeri, koriste se objektno-orijentirani jezik, Python, te logički jezik, SWI-Prolog. Njihova komplementarnost je omogućena preko Python biblioteke poznate kao PySwip koja korisniku omogućuje korištenje Prolog upita kroz sučelje Python-a.

Prolog i njegove razne iteracije su nešto drugačija vrsta programskog jezika od onih koji se uglavnom spominju poput Python-a, C++-a, Javascript-a i slično. Unatoč tome, od iznimne su važnosti zbog neizostavne uloge u izradi umjetne inteligencije i programa koji sudjeluju u procesima odlučivanja. Među poznatijim primjerima mogu se spomenuti IBM-ovo super računalo "Watson", NASA-in sustav "Warren's Abstract Machine" te sve veći broj aplikacija koji koristi podvrstu Prologa zvanu "Datalog".

Suvremeni informacijski svijet sadržava visoku količinu informacija u sebi te se javlja sve veća potencijalna korist od korištenja efikasnog logičkog programiranja. Uz relativno malo programskog koda moguće je jednostavno i brzo manipuliranje podacima od kojih su te informacije izgrađene. U ovom radu su, na primjeru programa koji upravlja odlukama unutar računalne igre, prikazane neke od mogućnosti koje Prolog otvara.

2. SWI-Prolog

Prolog je logički, deklarativni programski jezik koji radi na principu predikatne logike prvog reda. Korisnik definira klauzule (činjenice i pravila) unutar baze znanja te zatim definira upite koji se izvršavaju nad bazom znanja i vraćaju rezultate koji ispunjavaju sve pretpostavke upita.

2.1. Primjena Prologa

Zbog svojih mogućnosti, Prolog je koristan prilikom razvoja sustava koji manipuliraju velikom količinom podataka prilikom rada te mogu koristiti predikatnu logiku. Programeri koji razvijaju umjetnu inteligenciju, baze podataka i ekspertne sustave upravo radi toga upotrebljavaju logičke programske jezike.

2.1.1. Mogućnosti SWI-Prologa

Razlog Prologove uspješnosti u navedenim područjima su prednosti koje se ostvaruju korištenjem logičkog programiranja, a nisu dostižne korištenjem konvencionalnih programskih jezika. SWI-Prolog je jedna od mnogih podvrsta Prolog jezika, te je izgrađena s namjenom izrade aplikacija. Kao takva podvrsta, SWI-Prolog dolazi sa prednostima korisnima prilikom samog razvoja. Prema službenoj stranici SWI-Prologa, te prednosti su:

- **Skalabilnost** - SWI-Prolog koristi višedretveni način rada tako da može koristiti višejezgrene procesore koji postaju sve češći na današnjem tržištu. Jezik također koristi "Just In Time Indexing" sustav koji omogućuje ubrzano pretraživanje predikata koji imaju veći broj klauzula
- **Unificiranje** - kao relativno nova inačica Prologa, SWI-Prolog koristi visok broj nadogradnji i ekstenzija koje su razvijene od izlaska inicijalne verzije Prologa. Među njih spadaju korisne pojedinosti poput globalnih varijanti, ograničenja i tabličenja.
- **Pristupačnost** - SWI-Prolog nudi visok broj razvojnih alata koji se mogu kombinirati. Među njima su analiziranje koda, grafički modul za otklanjanje grešaka (debugger) te visok broj komplementarnih ekstenzija koje korisnik može dodavati i uređivati kako bi olakšao razvoj aplikacija
- **Paketi** - korisnici mogu kreirati, distribuirati i instalirati raznovrsne pakete za SWI-Prolog, omogućuju razvojnim programerima i Prolog zajednici da međusobno djele pakete te da si međusobno olakšavaju daljnje programiranje i rad sa SWI-Prologom

- **Fleksibilnost** - SWI-Prolog ima sučelja prema različitim programskim jezicima, neka su ugrađena, dok se drugima može pristupiti pomoću različitih paketa. Zbog toga, Prolog ima visok potencijal za interakciju s drugim programskim jezicima i korištenje njihovih sposobnosti
- **Prenosivost** - Razvojno okruženje prologa zauzima nisku količinu prostora, lako i brzo se može instalirati na različitim Windows i Linux operacijskim sustavima
- **Brzina** - stroj (engine) SWI-Prologa je robustan i brz, sposoban kompilirati visok broj linija u jako niskom vremenskom periodu

Zbog gore navedenih karakteristika, SWI-Prolog je odličan i lako upotrebljiv programski jezik te ga je lako inkorporirati u bilo koji razvojni proces koji zahtjeva njegove sposobnosti.

2.1.2. Ekspertni sustavi

Jedno od područja koje može koristiti mogućnosti Prologa su ekspertni sustavi, koji spadaju pod veću grupu sustava temeljenih na znanju. Ekspertni sustavi kombiniraju sve dostupne podatke kako bi složili kompleksne probleme. Cilj ekspertnog sustava je da nudi ista rješenja koja bi nudio i čovjek koji je ekspert na određenom području. Prednost sustava je manjak umaranja i mogućnost konstantnog rada bez prestanka te smanjene mogućnosti pogreške.

Prema Merrittu (2000.), ekspertni sustav treba zadovoljavati sljedeće značajke:

- **Unazadno ulančavanje ili ciljem vođeno razumijevanje** - tehnika kojimom se cilj rastavlja na manje ciljeve koji su lakši za dokazati
- **Nosivost sa nepoznatim** - sustav treba moći donositi odluke čak iako nisu precizno poznate sve činjenice
- **Unaprijedno ulančavanje ili podacima vođeno razumijevanje** - tehnika kojom se iz početnih podataka zaključuje problem pomoću ako-onda pravila
- **Prikaz podataka** - način spremanja i pristupanja podacima
- **Korisničko sučelje** - sučelje koje je lako razumljivo krajnjem korisniku
- **Objašnjenja** - sustav treba moći objasniti kako i zašto je došao do koje odluke

SWI-Prolog može ispuniti gore navedene značajke. Omogućeno mu je upravljanje podacima koje eksperti unose prilikom razvoja ekspertnog sustava te njegov logički način razmišljanja može biti korišten za rezolviranje različitih upita na više različitih načina, a proces razmišljanja moguće je pratiti preko korištenih klauzula i pravila. Zbog svoje brzine i skalabilnosti, SWI-Prolog nema poteškoća tijekom obrade iznimno visokih količina podataka kakve ekspertni sustav zahtjeva.

2.1.3. Inteligentne baze podataka

Još jedna od potencijalnih problemskih domena SWI-Prologa su inteligentne baze podataka. U današnjem svijetu, ljudi sakupljaju svaki dan sve više podataka te se javlja problem njihovog spremanja, organizacije i kasnijeg dohvaćanja.

Halper i Vasarhelyiju (1991.) su već 1991. tvrdili kako trenutni oblik spremanja podataka, unatoč činjenici da se kapacitet medija povećava s vremenom, nije optimalan te da ima visok broj poteškoća sa stajališta skalabilnosti. Kao zamjenu za postojeće baze podataka, koje koriste model procesiranja podataka, nudili su koncept inteligentnih baza podataka koje pokušavaju replicirati ljudski oblik procesuiranja informacija. Njihove baze podataka prilikom procesuiranja podataka su trebale imati sljedeća svojstva:

- **Prioritiziranje** - baza podataka bi trebala, prilikom konstantnog priljeva podataka, moći zaključiti koji podaci su višeg prioriteta
- **Nošenje sa nesigurnošću** - potrebno je moći prepoznati i preoblikovati podatke u slučaju da se pojave u nepoznatom obliku
- **Učiti bez učitelja** - baze same nadograđuju svoje znanje
- **Filtriranje i modeliranje** - treba moći prepoznati i odabrati najrelevantnije podatke ovisno o upitu
- **Brzina** - baze moraju biti efikasne prilikom svojeg rada, unatoč velikoj količini podataka
- **Jasnoća** - unutarnje strukture tablica trebaju nastati kao rezultat poznavanja vanjskog svijeta i njegovih potreba

SWI-Prologove sposobnosti mogu se primijeniti prilikom ispunjavanja zatraženih uvjeta. Njegova skalabilnost i brzina procesuiranja pokazale su se korisne prilikom manipulacija i procesiranja podacima. Prepoznavanje, filtriranje i modeliranje mogu se riješiti pomoću Prologove unutarnje logike i mogućnosti rezolviranja.

2.1.4. Umjetna inteligencija

Suvremeno društvo automatizira veliku količinu svojih poslova pomoću strojeva. Glavna prepreka prilikom automacije je napraviti stroj sposoban shvatiti svoj posao jednako dobro kako bi ga shvatio i pojedini radnik. Kako bi se unaprijedile mogućnosti automatizacije i razvili kompleksniji i sposobniji strojevi, potrebno ih je učiniti inteligentinijima i omogućiti im da sami donose pojedine zaključke i odluke bez vanjskog utjecaja čovjeka.

Rowe (1988.) tvrdi da umjetna inteligencija uključuje:

- komuniciranje između raučunala i čovjeka pomoću ljudskog jezika
- sposobnost računala da pamti međupovezane činjenice

- sposobnost računala da planira redoslijed događaja koji vode do cilja
- sposobnost računala da nudi prijedloge temeljem pravila
- sposobnost računala da detekitraju uzorke pomoću kamera
- sposobnost računala da pokreće sebe i druge objekte u stvarnom svijetu

Karakteristike SWI-Prologa ga čine korisnim alatom prilikom ostvarenja gore navedenih ciljeva. Prologov način rezolviranja problema pomoću klauzula (činjenica i pravila) čini ga sposobnim u pamćenju međupovezanih ciljeva i stvaranja redoslijeda događaja te nuđenju prijedloga rješenja temeljem zaključenog.

2.2. Programska struktura Prologa

Kao što je vidljivo iz prethodnog razmatranja logički jezici poput Prologa su najviše korišteni u razvijanju sustava koji koriste velike količine podataka prilikom svoga rada. Za razliku od konvencionalnih jezika, u kojima programer kreira algortime koji objašnjavaju kako obrađivati i manipulirati podacima, logički jezici testiraju veze između samih podataka pomoću predefiniranih pravila. Kako bi razumjeli logičko programiranje, trebamo razumjeti sintaksu programskog jezika i pripadajuću logiku zaključivanja.

2.2.1. Struktura

Radi razumijevanja Prologovog načina zaključivanja, potrebno je poznavati strukturu jednog Prolog programa. Ukratko, prema Clicksin i Mellish (2003.), programiranje u Prologu se svodi na tri koraka:

1. korisnik definira činjenice o objektima i njihovim vezama
2. korisnik definira pravila o objektima i njihovim vezama
3. korisnik postavlja pitanja o objektima i njihovim vezama

2.2.1.1. Baza znanja, predikati i klauzule

Svaki SWI-Prolog program ima bazu znanja koja sadrži klauzule – činjenice i pravila. Korisnik treba definirati predikate kojima će popuniti bazu. U Prologu, predikati se definiraju sljedećom strukturom:

```
voli(marko, marina).        % čitamo marko voli marinu
```

s time da je važno naglasiti da nazivi predikata moraju biti pisani malim početnim slovom, a završavati sa točkom koja označava kraj činjenice. Predikati se međusobno razlikuju osim po imenu i po svojoj kratnosti, tj. broju argumenata unutar zagrada. Ukoliko se u bazi znanja pojave dva predikata istog imena, ali različite kratnosti, Prolog će ih tretirati kao različite

klauzule. Argumenti unutar zagrada predikata podrazumijevaju različite tipove podataka kao što su varijable, atomi, brojevi, strukture i drugi, pri čemu je važno napomenuti da se tipovi podataka u Prologu zajednički nazivaju termi.

2.2.1.2. Upiti

Nakon što su u bazi znanja definirani predikati (klauzule – činjenice i pravila), u interaktivnom dijelu Prologa potrebno je definirati upite koji predstavljaju poseban oblik pravila koji se postavlja prilikom izvršavanja programa. Upiti u Prologu rade na principu ispunjavanja ciljeva koji se još zove usklađivanje ili matching. Drugim riječima, ako u programu postoje činjenice koje ispunjavaju cilj tada će Prolog odgovoriti „true“, a u suprotnom će odgovor biti „false“.

Na temelju prije definirane klauzule u bazi znanja sad možemo, na primjer, pitati Prolog koga voli Marko. Upit ima sljedeći oblik:

?- voli(marko,Y).

Odgovor glasi:

Y = marina

Drugim riječima, Prolog zaključuje da Marko voli Marinu.

2.2.2. Formalne metode

Nakon što su definirane činjenice i pravila, program mora biti sposoban donijeti pravilan zaključak iz dobivenog konteksta. U tu svrhu, SWI-Prolog koristi formalne metode, poglavito metodu logike prvog reda.

U širem smislu, prema NASA-inom članku ("What is formal methods", 2016.), formalne metode su matematičke tehnike i alati korišteni za specifikaciju, dizajniranje i verifikaciju programskih i sklopovskih sustava. Koristeći formalne metode, moguće je simbolički ispitati bilo kakav digitalni dizajn i omogućiti sigurnost i točnost koja je omogućena neovisno o ulaznim jedinicama podataka. Glavna poteškoća prilikom korištenja formalnih metoda je kompleksnost stvarnih sustava koja bi zahtjevala visoku količinu pravila prilikom pretvorbe u simboličke sustave. Kako bi se zaobišle poteškoće takvog pothvata, NASA u članku predlaže sljedeće korake:

- koristiti formalne metode samo na visokim razinama dizajna, gdje su svi detalji ionako apstraktni
- koristiti formalne metode samo kod kritično važnih komponenata
- analizirati digitalne modele gdje su varijable i njihovi dometi sniženi

- analizirati sistemske modele u hijerarhijskom obliku
- automatizirati verifikaciju što je moguće više

Razlog zbog kojega je potrebno koristiti formalne metode je upravo njihova sposobnost da osiguraju sigurnost neovisno u unesenim varijablama. Ne upotrebljavajući takve sigurnosne mjere može imati katastrofalne posljedice u kasnijem dijelovanju programa i njihovih funkcionalnosti.

Među poznatijim neuspjesima nastalima kao rezultat ne korištenja formalnih metoda u razvoju, u još jednom svojem članku ("Why is formal methods necessary?", 2016.), NASA navodi nekoliko primjera, među kojima su:

- neuspjeli let Ariane 5, financijski trošak od $850 milijuna dolara
- kvarovi u radu stroja za radioterapiju između 1985. i 1987., što je rezultiralo predoziranjem šestero ljudi i smrću dvoje
- kvarovi u tvornici procesora Intel-a koja je u 1995. korporaciju koštala više stotina milijuna dolara

Navedeni sustavi bili su nedovršeni ili su imali neprimjećene kvarove u radu. Čovjekova sposobnost verificiranja sustava se smanjuje sa povećanjem sustava. Današnji sustavi postaju sve kompleksniji te je potrebno koristiti nove metode verificiranja koje se ne umaraju i ne mogu pogriješiti. Pomoću programskijh jezika poput Prologa koji koriste logiku, bilo bi moguće napraviti verifikacijske sustave koji ne mogu pogriješiti. Glavna poteškoća je izgradnja takvog sustava do dovoljno detaljne mjere.

3. Python

Druga komponenta istražena u ovom radu su konvencionalni programski jezici, u ovom primjeru konkretno Python. Konvencionalni jezici imaju širu uporabu od logičkih te su češće spominjani i korišteni u realnim situacijama. Python je, radi svoje relativne jednostavnosti korištenja i upoznavanja, jedan od najpoznatijih programskih jezika od svoje prve inačice.

3.1. Primjena Pythona

Python je poznat kao programski jezik zbog visokog broja prednosti koje njegovo korištenje nudi. Glavna web stranica Pythona navodi sljedeće kao njegove prednosti:

- **jednostavnost** - sintaksa Pythona je jednostavnija od većine drugih jezika, no unatoč tome nudi veliki broj mogućnosti
- **veliki broj mogućnosti** - Python je izrazito fleksibilan te ima veliki broj dostupnih paketa i okvira (frameworka) koji su laki za instalaciju i uporabu u budućim projektima
- **edukacija** - zbog jednostavnosti sintakse, početnicima je lako shvatiti i ovladati kompleksnim programskim idejama
- **rasprostranjenost** - zbog svoje popularnosti, Python ima veliki broj izvora koji mogu pomoći korisniku u učenju njegovog korištenja

Zbog svega navedenog, Python se koristi u razvoju različitih aplikacija u raznovrsnim problemskim domenama. Njegove raznovrsne ekstenzije osiguravaju konstantnu prilagodljivost trendovima u industriji i mogućnost za suradnju i natjecanje sa drugim popularnim programskim jezicima.

Jedan od primjera usluga koje koriste Python kao glavni programski jezik je "ForecastWatch.com". Članak na Pythonovoj stranici je slavi kao jedan od uspješnih projekata koji je potpuno napravljen u Pythonu. Programeri iza projekta tvrde kako su Python koristili zbog njegove jednostavnosti i velike količine knjižica za sakupljanje, obradu i spremanje podataka koje su bile dostupne.

3.2. Programska struktura Pythona

Radi razumijevanja kako Python funkcionira i obavlja pojedine zadatke, potrebno je poznavati njegovu sintaksu i tipove podataka kojima vlada. Kasnije će lakše biti uočiti kako bi bilo moguće kombinirati ga sa logičkim jezikom poput SWI-Prologa.

3.2.1. Tipovi podataka

Python ima ugrađene tipove podataka koje korisnik može koristiti bez uporabe ikakvih vanjskih knjižica ili ekstenzija, ti tipovi su:

- **numerički** - korišteni za zapis razlčitih vrsta brojeva, razlikuju se integeri, floatovi, long integeri i kompleksni tipovi podataka
- **stringovi** - korišteni za zapis redoslijeda znakova
- **liste/n-torke** - dinamični skup stringova, brojeva i drugih lista/n-torki
- **rječnici** - skup različitih vrijednosti organizirane po principu ključ-vrijednost
- **boolean** - zastavice koje mogu imati True ili False vrijednost

Svi navedeni tipovi podataka imaju metode koje se mogu koristiti nad njima, što omogućuje korisnicima laku manipulaciju podacima nakon što ih dohvati. Moguće je pretvaranje.

Nad numeričkim tipovima podataka je moguće vršiti aritmetičke operacije poput uspoređivanja, zbrajanja, oduzimanja, množenja, dijeljenja, te, ako se uključi "math" knjižica s kojom Python dolazi, provoditi kompleksnije operacije poput faktoriranja, radnje s eksponentima i slično.

Kao i brojeve, stringove je moguće uspoređivati. Budući da su stringovi spremljeni u obliku n-torki, gdje je svaki znak stringa u jednom polju n-torke, moguće je provoditi operacije poput traženja, brisanja, dodavanja i uspoređivanja pojedinih vrijednosti polja, što korisnicima daje visoku kontrolu nad podacima zapisanima u obliku stringova.

Liste i n-torke sadržavaju različite elemente unutar svojih polja te omogućuju korisniku da sprema podatke u uredne redoslijede koji mu odgovaraju prilikom provođenja samog programa. Liste je moguće kombinirati s drugim listama, uspoređivati ih, množiti ih, provjeravati koje vrijednosti se pojavljuju u njima i slično. Budući da su polja indeksirana, korisnik može pristupati pojedinim elementima preko njihovog indeksa kako bi direktno upravljao podacima koji se nalaze unutar njega.

Python kao programski jezik korisnicima nudi visoku količinu kontrole nad podacima bez ikakve potrebe za dodavanjem vanjskih knjižica ili korištenjem ekstenzija. Sve postojeće preddefinirane operacije su lake i intuitivne za koristiti, što otvara mnoge mogućnosti za korištenje i kreiranje programa bez visoke količine potrebnog iskustva.

3.2.2. Izjave

Izjave su instrukcije koje Pythonov kompilator može izvršiti dok prolazi kroz njih. Kako bi mogao manipulirati podacima, korisnik pomoću izjava raznovrsnim varijablama daje vrijednosti, a zatim pomoću drugih izjava može upravljati tim varijablama i provoditi ih kroz raznovrsne algoritme koje isprogramira.

Prema službenoj dokumentaciji Pythona, program trenutno ima sljedeće vrste izjava:

- **dodjelne izjave** - služe za dodjeljivanje vrijednosti pojedinim varijablama
- **izražajne izjave** - izračunavaju i ispisuju vrijednosti
- *assert* **izjava** - korisne za lovljenje greški prilikom zadavanja tvrdnji
- *pass* **izjava** - korisna za rezerviranje mjesta, ne radi ništa prilikom izvršavanja
- *def* **izjava** - služi za definiranje korisnikovih vlastitih funkcija
- *del* **izjava** - rekurzivno briše vezu između imena varijable i njene vrijednosti
- *return* **izjava** - vraća zadanu vrijednost kada se izvrši
- *raise* **izjava** - vraća pogrešku, korisna za hvatanje greški
- *break* **izjava** - prekida izvođenje trenutne petlje
- *continue* **izjava** - skače na sljedeću iteraciju petlje bez dovršavanja trenutne
- *import* **izjava** - koristi se za uvođenje vanjskih knjižica u program, čime je korisniku omogućeno korištenje funkcija i klasa definiranih unutar tih knjižica
- *global* **izjava** - deklarira varijablu kao globalnu, čime će ona biti dostupna korištenju u trenutnom bloku koda
- *nonlocal* **izjava** - povezuje varijablu sa vrijednošću neke varijable koja nije globalna, ali je ranije definirana
- **izjave kontrola toka** - služe za provođenje raznovrsnih petlji, koje se iteriraju dok su ispunjeni uvjeti koje korisnik definira. Postoje *if*, *for*, *range*, *else* i *while* izrazi koji se svi na različite načine pokreću pojedine petlje.

Izjave omogućuju korisniku definiranje algoritama i funkcija kroz programski jezik, čime je moguće manipuliranje ranije navedenim tipovima podataka.

3.2.3. Izvođenje programa

Kada je program složen, Pythonov kompilator prolazi kroz sve izjave od početka do kraja python datoteke. Važno je da sve funkcije i varijable budu definirane prije njihova poziva unutar samog programa kako bi im se moglo pristupiti.

4. Međusobna kompatibilnost programa

Svaki programski jezik ima svoje prednosti, ali i svoje nedostatke. Iako je Python jednostavan za korištenje i ima dostupnu veliku količinu knjižica, sporiji je u svojem izvođenju od programskih jezika poput C-a ili C++-a. SWI-Prolog je koristan prilikom izvođenja formalnih metoda i brzih iteracija kroz velike količine podataka, gledajući odnose između objekata, ali ne nudi mnogo mogućnosti korisniku da manipulira dobivenim podacima te nije efikasan za definiranja raznovrsnih algoritama. Koristeći oba jezika istovremeno, programeri mogu umanjivati nedostatke i naglašavati prednosti Pythona i SWI-Prologa.

4.1. Usporedba jezika

Kako bismo mogli koristiti oba jezika, potrebno je poznavati i shvatiti njihove razlike. Python je konvencionalan programski jezik koji se može koristiti za raznovrsne svrhe. Ovisno o programeru i njegovom stilu pisanja programa, Python može biti proceduralan ili objektno-orijentiran, lako je prilagodljiv situaciji i lagan za korištenje čak i početnicima. Nasuprot Pythonu, SWI-Prolog je u potpunosti logički jezik koji zahtjeva poznavanje logike prvog reda za korištenje. Njegovo sučelje je nešto kompliciranije za koristiti te ima manje funkcija koje programeru pomažu u pisanju.

Oba jezika su fleksibilna sa stajališta vanjskih knjižica, programeri i Pythona i SWI-Prologa mogu međusobno definirati, distribuirati i koristiti vanjske knjižice drugih korisnika. Oba jezika je moguće urediti i prilagoditi visokim brojem ekstenzija i vanjskih opcija te oba jezika imaju visok broj razvojnih okruženja u kojima mogu raditi.

SWI-Prolog ima ograničen način zapisivanja podataka, prepoznaje jedino stringove i brojeve te stringovi moraju biti pisani malim slovima (velikim slovima se pišu varijable). Podaci se spremaju u bazu znanja u obliku atoma i nije moguće manipulirati njima van usporedbi i ispitivanja veza. S druge strane, Python ima relativno veći broj zapisivanja varijabli, može razaznavati veći broj različitih vrsta brojeva, a stringove može prikazivati u obliku n-torki.

Python također korisniku nudi relativno veći broj unaprijed ugrađenih funkcija za svaki pojedini tip podataka, čime je korisniku olakšano utjecati na ulazne i izlazne jedinice prilikom rada s njima. SWI-Prolog, iako ima neke ugrađene funkcije, je manje intuitivan i teže je koristiti se njegovim sposobnostima.

Izrada algoritama za filtriranje i prepoznavanje podataka je teža u Pythonu, od korisnika se zahtjeva da definira veliki broj funkcija radi raspoznavanja unesenih podataka. SWI-Prolog, koristeći gledište logike, može lakše prepoznati unesene vrijednosti i odlučiti kako ih iskoristiti

bez potrebe za izradom velikih količina funkcija i algoritama. Korisnik samo treba pravilno definirati i kombinirati upite kako bi program znao kako vladati ulaznim tokom podataka.

Budući da oba jezika imaju prednosti i nedostatke, prilikom rada je potrebno podijeliti poslove ovisno o programskom jeziku koji bi ih bolje obavljao. Kako bi razvojni programeri bili u mogućnosti razaznati koji zadaci bi bolje pristajali kojem jeziku, potrebno je da dobro poznaje sposobnosti i mogućnosti koje su mu dostupne.

4.2. PySwip

Kako bi se kombinirali Python i SWI-Prolog u radu, potreban im je most koji će osposobiti korištenje jednog jezika unutar drugog. PySwip je Python paket koji služi kao most između Pythona i SWI-Prologa, čime je omogućeno pristupati SWI-Prologovim funkcijama unutar Python programa. Kako bi se koristio, PySwip je potrebno instalirati pomoću Pythonove Pip skripte i zatim pomoću "import" izraza uvesti pyswip knjižicu u sam program.

Prema službenoj stranici projekta, PySwip trenutno omogućujue postavljanje SWI-Prologove upite unutar Pythona. Korisnici mogu stvarati vlastiti bazu znanja unutar samog Python programa pomoću PySwip-ove Prolog klase ili mogu konzultirati postojeće Prologove (.pl) datoteke u slučaju da se nalaze u istoj datoteci kao Python program.

Zahvaljujući PySwip-u, korisnici mogu lako inkorporirati SWI-Prologove mogućnosti preko Pythonovog razvojnog okruženja. Ovakav sastav omogućuje istovremeno korištenje obaju jezika kako bi se koristile njihove prednosti. Pythonove sposobnosti obrade podataka i fleksibilnosti glede korištenja korisnicima omogućuju korisnicima da prezentiraju podatke u drugačijim oblicima i manipuliraju njima na više načina. SWI-Prolog omogućuje Pythonu bržu pretragu, spremanje i dohvaćanje tih podataka te štedi vrijeme prilikom kompiliranja i izvršavanja samog programa, čime štedi resurse i čini cjelokupno programsko rješenje efikasnijim.

PySwip se može kombinirati i sa ostalim Pythonovim knjižicama, čime je moguće koristiti SWI-Prologove sposbnosti koristiti za različite projekte raznovrsnih problemskih domena.

Ovakvim međudjelovanjem moguće je stvarati jednostavnije i sposobnije sustave koji rješavaju kompliciranije i složenije zadatke te pokrijvaju veći broj područja, a istovremeno korisnicima smanjuju količinu posla i povećavaju brzinu izvođenja programa te njegovu čitljivost i kasniju nadogradivost.

5. Praktični primjer - automatizirano igranje računalne igre

Kao primjer aplikacije koja koristi mogućnosti SWI-Prologa i Pythona, napravljena je aplikacija koja automatizirano igra računalnu igru Darkest Dungeon. Pythonove mogućnosti korištene su radi sakupljanja informacija i upravljanja mišem i tipkovnicom, a SWI-Prolog je korišten u svrhe donošenja odluka o sljedećem potezu koji se treba izvršiti u igri.

5.1. Problemska domena - Darkest Dungeon

Darkest Dungeon je igra na poteze u kojoj igrač upravlja timom od četiri lika i navigira ih kroz različite tamnice i protiv raznovrsnih neprijatelja. Igraču je cilj održati likove u što boljem stanju dok prolaze nivoe, a da istovremeno sakuplja što više blaga i troši što manje resursa. Često treba riskirati ako želi dovršiti nivo ili ispuniti zadatak.

Igrač tijekom nivoa treba paziti na fizičko i mentalno zdravlje svojih likova, držati svjetlinu na dovoljno visokoj razini, pokušavati deaktivirati zamke, pravilno reagirati na različite predmete na koje nailazi u okolišu te koristiti stvari iz inventara na njima kako bi dobio pozitivne rezultate. U slučaju da se nađe u borbi sa neprijateljem, igrač treba strateški birati mete kako bi u što kraće vrijeme i uz što manje poteškoća pobijedio u borbi i očuvao likove u dovoljno dobrom stanju da nastave istraživati. Kako se napreduje kroz nivo, broj resursa dostupnih igraču se smanjuje, nivo stresa na likove se povećava te je sve teže održavati ekipu u zdravom stanju.

5.2. Ciljevi programskog rješenja

Program bi trebao moći kretati se kroz prolaze tamnice i pravilno reagirati na situacije u kojima se nalazi:

- tražiti pravilne puteve kroz hodnike
- držati razinu svjetline na određenoj razini
- liječiti likove od različitih vrsta negativnih statusa koje mogu dobiti
- birati pravilne mete unutar samih borba
- prioritizirati pravilne neprijatelje u borbi
- organizirati inventar tijekom prolaska kroz tamnicu

Neke ciljeve moguće je jednostavno izvršiti u sklopu samog Pythona, ali za biranje neprijatelja i interakcije sa predmetima u okolini, lakše je koristiti SWI-Prologove baze znanja i procese zaključivanja.

Slika 1 – Prikaz kruga borbe

5.3. Razvoj aplikacije

Aplikacija je složena kombiniranjem programskih jezika SWI-Prolog i Python. Prilikom razvoja, uz PySwip, korištena je nekolicina drugih Python knjižica u svrhu identificiranja informacija na ekranu i korištenja računalnog miša i tipkovnice:

- **pydirectinput, pyautogui, ahk** - knjižice za upravljanje pokretima miša, tipkovnice i pritiscima na tipke tipkovnice. ahk je knjižica AutoHotKey programa, koji je bio potreban za upravljanje tipkama miša
- **time** - ugrađena knjižica Pythona, ponekad se na ekranu odvijaju animacije koje dugo traju, pa je potrebno koristiti sleep funkciju
- **numpy, pyscreenshot, pyimagesearch, PIL** - knjižice za slikanje ekrana, uređivanje slika i lociranje pojedinih uzoraka na njemu. Ove knjižice su koristil za prepoznavanje predmeta na ekranu
- **pytesseract** - knjižica za prepoznavanje teksta

Koristeći Pythonove knjižice različitih programa, bilo je moguće sakupljati informacije i podatke koji su dostupni na ekranu.

Na primjer, "imagesearch" knjižica uzima sliku ekrana i na njoj traži uzorke koje joj korisnik zadaje. Kako bi je se koristilo, potrebno je imati raniju sliku na kojoj se nalazi uzorak. Funkcije najčešće korištene u izvođenju programa su bile "imagesearch", "imagesearcharea" i "imagesearch_from_folder". Prva funkcija uzima dva argumenta, putanju do slike koja se traži i preciznost, a kao rezultat vraća X i Y koordinate gornjeg lijevog kuta slike ako je pronađena, a [-1, -1] ako slika nije pronađena. "imagesearcharea" još prima dodatne argumente kojima korisnik može specificirati koji dio ekrana se pregledava. "imagesearch_from_folder" traži sve slike koje se nalaze u pojedinoj mapi i vraća popis svih slika sa X i Y koordinatama ako su pronađene ili [-1, -1] ako nisu.

Neki dijelovi programa ne zahtijevaju veliku količinu odlučivanja te ih je moguće riješiti pomoću jednostavnih Python ako...onda petlji.

Mogućnosti SWI-Prologa su korištene najviše prilikom interakcija sa predmetima u okolišu i same borbe, ali bio je korišten i kao normalna baza podataka za čuvanje znanja o koordinatama heroja i neprijatelja te držanja inventara.

5.3.1. Definiranje heroja

Prilikom početnog pokretanja, program pregledava sve pozicije heroja i zapisuje njihove značajke. Pomoću Pythonovih funkcija, program pozicionira pokazivač miša pomoću unaprijed definiranih vrijednosti na sve četiri pozicije heroja i sakuplja podatke o njima.

Slika 2 – Prikaz početne sobe misije

Kada je pozicioniran na pojedinog heroja, program pomoću "imagesearch_from_folder" funkcije saznaje o kojem heroju je riječ tako da traži njegovo ime negdje na ekranu. Kada je heroj odabran u igri, naziv njegove klase bit će desno od njegova portreta, te će ga tu locirati funkcija.

```
def findCurrentCharacter():
    global currentCharacter
    global p

    results = str(imagesearch_from_folder("./imagesForTheProgram/CharacterStuff/Portraits/",0.9))
    print()
    results = results[2:]
    results = results[:len(results)-1]
    listOfResults = results.split(", '")
    time.sleep(0.1)

    listOfResults = [i.replace("./imagesForTheProgram/CharacterStuff/Portraits/", "") for i in listOfResults]
    listOfResults = [i.replace("_pic.png'", "") for i in listOfResults]
    print()

    filteredList = []

    for j in listOfResults:
        if j.split(": ")[1] != "[-1, -1]":
            filteredList += [j]

    for k in filteredList:
        currentCharacter = k.split(":")[0]
```

Slika 3 - findCurrentCharacter funkcija

Jedini rezultat čije vrijednosti neće imati [-1, -1] vrijednost bit će rezultat slike heroja čije ime se trenutno nalazi na ekranu. Filtrirajući popis i koristeći Pythonove funkcije, string dobiven iz funkcije se smanjuje na samo ime lika, koje se zatim sprema u globalnu varijablu "currentCharacter"

Kada je pronađen određeni heroj, potrebno je saznati koji su mu trenutno dostupni potezi. Ponovno se poziva "imagesearch_from_folder" koja cilja specifičnu mapu u kojoj se nalaze samo ikone sposobnosti određenog lika.

```
def findCurrentCharactersAbilities():
    global currentCharacter
    global currentCharacterAbilities

    if currentCharacter == "":
        pass

    else:
        currentCharacterAbilities.clear()
        results = str(imagesearch_from_folder("./imagesForTheProgram/CharacterStuff/"+currentCharacter+"/" ,0.8))
        results = results[2:]
        results = results[:len(results)-1]
        listOfResults = results.split(", '")
        time.sleep(0.2)

        listOfResults = [i.replace("./imagesForTheProgram/CharacterStuff/"+currentCharacter+"/"+currentCharacter+"_", "") for i in listOfResults]
        listOfResults = [i.replace(".png'", "") for i in listOfResults]

        print()

        filteredList = []

        for j in listOfResults:
            #print(j.split(": ")[1] == "[-1, -1]")
            #print(j.split(": ")[1])
            if j.split(": ")[1] != "[-1, -1]":
                filteredList += [j]

        counter = 0
        for k in filteredList:
            counter = counter + 1

        for k in filteredList:
            #print(k)
            currentCharacterAbilities.append(k)
```

Slika 4 - Funkcija za dohvaćanje pojedinih sposobnosti lika

Ponovno se dobiveni popis filtrira kako bi ostali nazivi samo poteza čije ikone su pronađene na ekranu te se zatim njihovi nazivi spremaju u globalnu listu sposobnosti.

Kada su sakupljeni svi heroji prikazani na ekranu i njihove sposobnosti, vrši se petlja tijekom koje se sve vrijednosti iz globalnih n-torki unose u bazu znanja SWI-Prolog dijela programa.

```
for hero in allCharacters:
    forAdding = "currentlyAvailableCharacters(" + str(hero[0]) + ", " + hero[1] + ", "+ hero[2] + ", "+ hero[3]+ ", "+ hero[4]+")"
    p.assertz(forAdding)
    print(forAdding)
    for i in range(firstSkill, firstSkill + 4):
        forAdding = "currentlyAvailableSkills(" + hero[1] + ", " + allSkills[i] + ")"
        print(forAdding)
        p.assertz(forAdding)

    firstSkill = firstSkill + 4
```

Slika 5 - Unos svih heroja i njihovih sposobnosti u bazu znanja

Kako bi heroji bili uneseni u bazu znanja, proglašena je varijabla p kao instanca Prolog klase. Klasa ima definiranu operaciju "assertz()" koja služi za dodavanje predikata u bazu znanja. Kada je predikat postavljen u bazu znanja, moguće mu je pristupati pomoću upita u

kasnijim dijelovima programa. Budući da se operacije "assertz" i "retract" mogu koristiti tijekom korištenja programa, sve u bazi znanja je automatski dinamično.

5.3.2. Upravljanje inventarom

Baza znanja SWI-Prologa lako se može koristiti i za informacije o inventaru. Najprije je potrebno definirati sve vrste predmeta koje je moguće imati u inventaru.

```
def defineValuableItems():
    global p
    valuableItem = Functor("valuableItem", 2)
    valueAndHowManyStacks = Functor("valueAndHowManyStacks", 2)
    p.assertz("valuableItem(gold1, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(gold2, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(gold3, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(gold4, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(citrine, valueAndHowManyStacks(250,5))")
    p.assertz("valuableItem(jade, valueAndHowManyStacks(375,5))")
    p.assertz("valuableItem(onyx, valueAndHowManyStacks(500,5))")
    p.assertz("valuableItem(emerald, valueAndHowManyStacks(750,5))")
    p.assertz("valuableItem(sapphire, valueAndHowManyStacks(1000,5))")
    p.assertz("valuableItem(ruby, valueAndHowManyStacks(1250,5))")
    p.assertz("valuableItem(juteTapestry, valueAndHowManyStacks(4500,1))")
    p.assertz("valuableItem(puzzlingTrapezohedron, valueAndHowManyStacks(3500,1))")
    p.assertz("valuableItem(minorAntique, valueAndHowManyStacks(500,20))")
    p.assertz("valuableItem(rareAntique, valueAndHowManyStacks(1250,5))")
    p.assertz("valuableItem(consecratedPew, valueAndHowManyStacks(2500,1))")
    p.assertz("valuableItem(cometShard, valueAndHowManyStacks(0,99))")
```

Slika 6 - Definiranje vrijednosnih predmeta

```
def defineItems():
    global p
    item = Functor("item", 1)
    p.assertz("item(food_1)")
    p.assertz("item(food_2)")
    p.assertz("item(food_3)")
    p.assertz("item(shovel)")
    p.assertz("item(antivenom)")
    p.assertz("item(bandages)")
    p.assertz("item(medicinalHerbs)")
    p.assertz("item(skeletonKey)")
    p.assertz("item(holyWater)")
    p.assertz("item(laudanum)")
    p.assertz("item(torch)")
    p.assertz("item(firewood)")
    p.assertz("item(theBlood)")
    p.assertz("item(dogTreats)")

    print("Items defined")

def defineHeirlooms():
    global p
    item = Functor("heirloom", 2)
    p.assertz("heirloom(bust, 2)")
    p.assertz("heirloom(deed, 3)")
    p.assertz("heirloom(crest, 1)")
    p.assertz("heirloom(portrait, 4)")

    print("Heirlooms defined")

def defineQuestItems():
    global p
    questItem = Functor("questItem", 1)
    p.assertz("questItem(pick-axe)")
    p.assertz("questItem(consecratedEssence)")
    p.assertz("questItem(potentSalve)")
    p.assertz("questItem(handOfGlory)")
    p.assertz("questItem(pinealGland)")
    p.assertz("questItem(medicine)")
    p.assertz("questItem(oneGrainSack)")
    p.assertz("questItem(oneHolyRelic)")
    p.assertz("questItem(oneAncestorsRelic)")

    print("Quest items defined")
```

Slika 7 - Definiranje preostalih vrsta predmeta

Važno je pravilno definirati različite vrste predmeta kako program kasnije ne bi odbacio nešto što je potrebno za izvršavanje zadatka ili predmet koji bi mogao biti koristan pri liječenju heroja ili korištenju na različitim predmetima u prostoru.

Kad god se odvija interakcija sa inventarom, program poziva funkciju za dodavanje predikata u bazu znanja.

```
def whatsInTheInventory():
    removeEverythingInInventory()
    pressInventory()
    addSomethingGroup("QuestStuff")
    addSomethingGroup("InventoryStuff")
    addSomethingGroup("ValuableStuff")
    addSomethingGroup("HeirloomStuff")
```

Slika 8 - Funkcija za pregledavanje predmeta u inventaru

```
def addSomethingGroup(theStuffThatsBeingAdded):
    results = str(imagesearch_from_folder("./imagesForTheProgram/"+theStuffThatsBeingAdded+"/", 0.8))
    results = results[2:]
    results = results[:len(results)-1]
    listOfResults = results.split(", '")

    listOfResults = [i.replace("./imagesForTheProgram/"+theStuffThatsBeingAdded+"/"+theStuffThatsBeingAdded+"_", "") for i in listOfResults]
    listOfResults = [i.replace(".png'", "") for i in listOfResults]

    filteredList = []

    for j in listOfResults:
        if j.split(": ")[1] != "[-1, -1]":
            filteredList += [j]

    filteredList2 = [i.split(":")[0] for i in filteredList]

    for k in filteredList2:
        print(k)
        addToInventory(k)
```

Slika 9 - Funkcija za pronalaženje predmeta na ekranu

```
def addToInventory(thingToAdd):
    global p
    p.assertz("playerHas("+thingToAdd+")")
```

Slika 10 - Funkcija za dodavanje predmeta u bazu znanja

Ponovno se koriste funkcije imagesearch knjižice kako bi se locirali predmeti koji se nalaze u inventaru. Pretraga se vrši posebno nad predmetima različitih vrsta, ali svi predmeti se na kraju u bazu znanja dodaju pomoću predikata "playerHas()". Kasnije je predmete u inventaru moguće koristiti prilikom interakcija s "curioima" i pri liječenju likova.

Prilikom sakupljanja raznovrsnih blaga i drugih predmeta tijekom misije, može doći do manjka mjesta u inventaru. U tom slučaju je potrebno odbaciti neki predmet kako bi bilo moguće pokupiti trenutno ponuđene bolje predmete. Prilikom biranja predmeta koji će se odbaciti, poziva se Python funkcija koja traži predmet najmanje vrijednosti. Nije moguće odbaciti "quest" predmete jer su oni potrebni za izvršavanje trenutne zadaće.

```
def kickLeastValuableThing():
    somethingKicked = False

    valuableItem = Functor("valuableItem", 2)
    heirloom = Functor("heirloom", 2)
    playerHas = Functor("playerHas", 1)
    valueAndHowManyStacks = Functor("valueAndHowManyStacks", 2)

    genre = ""
    bestToKick = ""
    worthOfMostKickable = 10000

    X = Variable()
    Y = Variable()
    Z = Variable()

    q = Query(playerHas(X), valuableItem(X, valueAndHowManyStacks(Y, Z)))
    while q.nextSolution():
        if int(Y.value)*int(Z.value) < worthOfMostKickable:
            worthOfMostKickable = int(Y.value)*int(Z.value)
            bestToKick = str(X.value)
            genre = "valuable"
    q.closeQuery()

    if bestToKick == "":
        worthOfMostKickable = 5
        q = Query(playerHas(X), heirloom(X,Y))
        while q.nextSolution():
            if int(Y.value) < worthOfMostKickable:
                worthOfMostKickable = int(Y.value)
                bestToKick = str(X.value)
                genre = "heirloom"
        q.closeQuery()

    if bestToKick == "":
        q = Query(playerHas(X), item(X))
        while q.nextSolution():
            bestToKick = str(X.value)
            genre = "item"
        q.closeQuery()

    print(bestToKick)
```

Slika 11 - Funkcija za biranje najmanje vrijednog predmeta

Funkcija za izbacivanje prvo traži vrijednosni predmet u inventaru te će izbaciti onaj čija potencijalna vrijednost je najmanja (vrijednosni predmeti iste vrste se mogu slagati u stogove do određenog broja). Izbacivanje počinje od traženja vrijednosnih predmeta jer oni ne igraju nikakvu ulogu tijekom provođenja same misije.

Ako nije nađen neki vrijednosni predmet, postavlja se upit koji traži "heirloom" predmete. Oni također ne ispunjavaju nikakvu ulogu tijekom izvođenja same misije, ali su dosta rijeđi te se koriste za unaprijeđenje drugih mehanika igrice van samih misija pa ih je korisno zadržavati.

Sljedeći predmeti koji se biraju za izbacivanje su normalni predmeti koji imaju svrhe tijekom same misije. Njihovim izbacivanjem se onemogućuju pojedine interakcije koje bi se mogle pokazati korisne pa su zato na samom kraju.

Kada je odabran predmet za izbacivanje, jednostavno se poziva "retract()" operacija kojom se predmet miče iz inventure, a ubacuje se predmet koji će zauzeti njegovo mjesto.

```
def removeFromInventory(item):
    global p
    p.retract("playerHas("+item+")")
```

Slika 12 - Funkcija za brisanje predmeta iz baze znanja

5.3.3. Interakcije s predmetima (curioima)

Dok se heroji kreću kroz hodnik, imaju prilike naići na razne predmete u okolišu, predmeti imaju šanse sadržavati različite vrste potrepština i blaga ili mogu imati negativne posljedice za lika koji ih aktivira. Igrač ima opciju koristiti predmete iz svojeg inventara kako bi garantirao da će rezultat interakcije biti pozitivan. Predmeti reagiraju samo na određene stvari iz inventara dok će druge biti potrošene bez ikakvog utjecaja. Interakcije i veze između dređenih predmeta iz okoliša i predmeta iz inventure mogu se definirati u SWI-Prologovoj bazi prilikom pokretanja programa.

```
p.assertz("curio(dinnerCart)")
p.assertz("curio(makeshiftDiningTable)")
p.assertz("curio(moonshineBarrel)")
p.assertz("curio(occultScrawlings)")
p.assertz("curio(pileOfBones)")
p.assertz("curio(pileOfScrolls)")
```

Slika 13 - Dodavanje predmeta u SWI-Prolog bazu

```
p.assertz("item(antivenom)")
p.assertz("item(bandages)")
p.assertz("item(medicinalHerbs)")
p.assertz("item(skeletonKey)")
p.assertz("item(holyWater)")
p.assertz("item(laudanum)")
p.assertz("item(torch)")
```

Slika 14 - Dodavanje inventarskih predmeta u SWI-Prolog bazu

```
p.assertz("curioAndWhatWorksOnIt(boneAltar, nothing, positive)")
p.assertz("curioAndWhatWorksOnIt(dinnerCart, medicalHerbs, positive)")
p.assertz("curioAndWhatWorksOnIt(makeshiftDiningTable, medicalHerbs, positive)")
p.assertz("curioAndWhatWorksOnIt(moonshineBarrel, medicalHerbs, positive)")
p.assertz("curioAndWhatWorksOnIt(occultScrawlings, holyWater, positive)")
p.assertz("curioAndWhatWorksOnIt(pileOfBones, holyWater, positive)")
p.assertz("curioAndWhatWorksOnIt(pileOfScrolls, torch, positive)")
p.assertz("curioAndWhatWorksOnIt(rackOfBlades, bandage, positive)")
p.assertz("curioAndWhatWorksOnIt(beastCarcass, medicinalHerbs, positive)")
```

Slika 15 - Dodavanje odnosa između pojedinih predmeta, te vrstu rezultata koji se dobiva njihovom interakcijom

Slika 16 – Prikaz detektiranja predmeta u prostoru/"curio"

Python pomoću imagesearch knjižice detektira da se na ekranu otvorio prozor sa ponuđenim opcijama. Imagesearch funkcija traga za sličicom ruke. Ako je ruka locirana, pokreće se "imagesearch_from_folder" funkcija koja uspoređuje sve slike iz zadane mape sa trenutnim ekranom (zbog fonta u igrici, knjižica za detektiranje teksta je često griješila prilikom pretvaranja slike u tekst). Funkcija prepoznaje uzorak "Giant Oyster"-a i vraća X i Y koordinate gornjeg lijevog kuta te slike. Slika u samoj mapi je nazvana po predmetu te će uz ime slike stajati pozitivne koordinate. Filtriranjem popisa moguće je izvući samo ime slike i tako saznati o kojem predmetu je riječ.

Dobiveni predmet iz okoliša ("curio") se prosljeđuje u funkciju koja stvara upit. Upit u bazi znanja traži predmet koji ima pozitivnu interakciju sa danim "curiom" te provjerava postoji li taj predmet u korisnikovom inventaru.

Stvaranje upita u Pythonu zahtjeva od korisnika da proglasi sve predikate i njihove arnosti prije nego li ih spaja. Klasa "Query" ima funkciju "nextSolution()" koja vraća jedan

```
def checkIfYouHaveWhatYouNeedForCurio(investigatedCurio):
    curio = Functor("curio", 1)
    curioAndWhatWorksOnIt = Functor("curioAndWhatWorksOnIt", 3)
    playerHas = Functor("playerHas",1)

    X = Variable()

    itemNeeded = ""
    q = Query(curio(investigatedCurio), curioAndWhatWorksOnIt(investigatedCurio,X,"positive"), playerHas(X))

    while q.nextSolution():
        itemNeeded = str(X.value)
    q.closeQuery()

    return itemNeeded
```

Slika 17 - Prolog upit preko Python funkcije

rezultat koji odgovara svim napisanim uvjetima. U slučaju da korisnik nema potrebni predmet u inventaru, funkcija vraća prazan string.

Ako korisnik ima potrebni predmet, program će tragati za njegovom slikom, dobiti njene koordinate kao rezultat te proslijediti koordinate funkciji koja upravlja mišem.

Desni klik na predmet ga iskorištava i garantira pozitivnu interakciju sa "curiom".

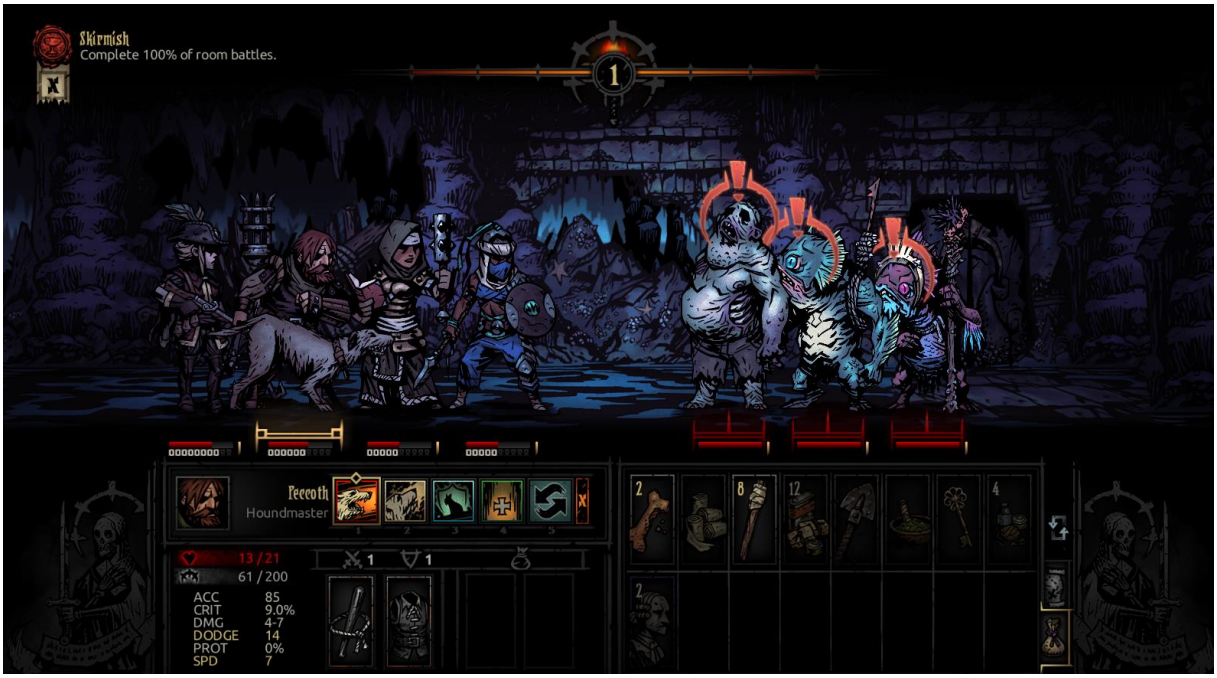
```
def useItemOnCurio(item):
    results = imagesearch("./imagesForTheProgram/InventoryStuff/InventoryStuff_"+item+".png")
    if results[0] != -1:
        rightClickMouse(results[0]+5, results[1]+5)
```

Slika 18 - Funkcija za pretragu slike pojedinog predmeta

Slika 19 – Prikaz rezultata korištenja dobrog predmeta

5.3.4. Borbe s neprijateljima

Kretanjem kroz tamnice, heroji često nailaze na neprijatelje koje je potrebno poraziti kako bi se nastavilo sa istraživanjem. Neprijatelji također dolaze u grupama te svaki ima svoje napade. Neki neprijatelji, zbog potencijalne štete koje mogu nanijeti, su od većeg prioriteta za napasti od drugih.



Slika 20 - Heroji nailaze na neprijatelje

Heroji imaju raznovrsne poteze koji mogu jačati druge članove u grupi, štititi druge članove grupe, liječiti druge članove grupe, napadati neprijatelje, ošamućivati neprijatelje ili slabiti neprijatelje. Potrebno je održavati zdravlje grupe, ali i istovremeno napadati neprijatelje.

Kako bi program znao što učiniti, potrebno je prije početka definirati sve vrste heroja, sve njihove napade te domete njihovih napada.

```
p.assertz("characterAbility(houndmaster, lick_Wound, selfheal)")
p.assertz("characterAbility(houndmaster, blackjack, stun)")
p.assertz("characterAbility(houndmaster, blackjack, stun)")
p.assertz("characterAbility(houndmaster, hounds_Rush, attack)")
p.assertz("characterAbility(houndmaster, hounds_Rush, bleed)")
p.assertz("characterAbility(houndmaster, hounds_Rush, markable)")
p.assertz("characterAbility(houndmaster, hounds_Harry, attack)")
p.assertz("characterAbility(houndmaster, hounds_Harry, bleed)")
p.assertz("characterAbility(houndmaster, target_Whistle, mark)")
p.assertz("characterAbility(houndmaster, target_Whistle, debuff)")
p.assertz("characterAbility(houndmaster, guard_Dog, buff)")
p.assertz("characterAbility(houndmaster, cry_Havoc, destress)")
```

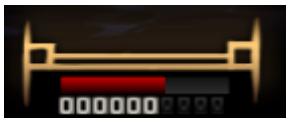
Slika 21 - Potezi "houndmaster" heroja

```
p.assertz("usableFromPosition(hounds_Rush, 2,4, 1,4)")
p.assertz("usableFromPosition(hounds_Harry, 1,4, 1,4)")
p.assertz("usableFromPosition(target_Whistle, 1,4, 1,4)")
p.assertz("usableFromPosition(cry_Havoc, 3,4, 1,4)")
p.assertz("usableFromPosition(guard_Dog, 1,4, 1,4)")
p.assertz("usableFromPosition(lick_Wound, 2,4, 1,4)")
p.assertz("usableFromPosition(blackjack, 1,2, 1,3)")
```

Slika 22 - Dometi "houndmaster"ovih poteza, prva dva broja su donja i gornja granica pozicija iz kojih se može koristiti potez, dok su druga dva pozicije na koje se može koristiti.

Nakon pokretanja programa, prije obavljanja bilo koje druge aktivnosti, program će pregledati sve postojeće heroje i, pomoću "imagesearch" i "imagesearch_from_folder" funkcija, zaključiti koji heroji se nalaze u grupi i koji potezi su im dostupni.

Unutar same borbe, na početku kruga pojedinog lika, program traži žutu oznaku koja se nalazi ispod heroja čiji je trenutni red.



Slika 23 - Žuta oznaka heroja

Pomoću lokacije oznake, program zaključuje u kojoj poziciji se heroj nalazi. Prvo se provodi funkcija koja dohvaća sve poteze koje heroj može provesti iz te pozicije.

```
def whatSkillsCanTheyUse(character, position):
    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    characterAbility = Functor("characterAbility", 3)
    usableFromPosition = Functor("usableFromPosition", 5)
    currentlyAvailableSkills = Functor("currentlyAvailableSkills", 2)

    print(character)

    Ability = Variable()
    MaxHealth = Variable()
    CurrHealth = Variable()
    Stress = Variable()
    AbilityType = Variable()
    LowFrom = Variable()
    HighFrom = Variable()
    LowTarget = Variable()
    HighTarget = Variable()

    usableMoves = []

    q = Query(currentlyAvailableSkills(character, Ability),
              usableFromPosition(Ability, LowFrom, HighFrom, LowTarget, HighTarget),
              characterAbility(character, Ability, AbilityType))
    while q.nextSolution():
        fromWhereLow = int(LowFrom.get_value())
        fromWhereHigh = int (HighFrom.get_value())
        if fromWhereLow <= position and fromWhereHigh >= position:
            usableMoves.append([str(Ability.value), str(AbilityType.value)])
    q.closeQuery()

    return usableMoves
```

Slika 24 - Upit koji vraća dostupne poteze

Kao ulazne argumente, funkcija uzima naziv heroja čiji je trenutni red i njihovu poziciju. Upit najprije pregledava trenutno dostupne poteze tom heroju (za koje je saznao tijekom inicijalnog pregleda svih heroja prilikom pokretanja programa), za svaki potez pregledava njihove domete i pregledava kojem heroju pripada. Unutar "while" petlje se u n-torku umeću svi potezi koji su upotrebljivi iz trenutne pozicije te se n-torka vraća prilikom izlaza iz funkcija.

Ako se među vraćenim potezima nalazi nešto upotrebljivo na herojima provodi se funkcija koja dohvaća vrijednosti trenutnog zdravstvenog stanja svih heroja te, ako je itko ispod zadane granice, liječi se onaj koji je u najlošijem stanju.

Za napade među vraćenim potezima, provodi se još jedan upit koji bira kojeg neprijatelja je najbolje napasti. Neprijatelji u zadnje dvije pozicije često imaju napade koji mogu slabiti heroje ili jačati druge neprijatelje, a istovremeno su lakši za poraziti od često oklopljenih neprijatelja u prednjim pozicijama.

Prije provođenja upita, program provodi još jednu imagesearch funkciju pomoću koje saznaje pozicije i oklopljenost neprijatelja te ih sprema u Prologovu bazu.

```
def countEnemiesAndStats(positions):
    enemies = []
    currentPositionOfSkull = [0,0]
    oldPositionOfSkull = 0
    number = 0
    canStress = ""
    hasProt = ""
    isCorpse = ""

    for position in positions:
        moveTo(position[1], position[2])
        currentPositionOfSkull = imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Enemy_Targetting.png", 0.8)
        protection = imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Prot.png", 0.8)
        stress = imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Stress.png", 0.8)
        corpse = imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Corpse.png", 0.8)
        if protection[0] != -1 and protection[0] > 945:
            hasProt = "prot"
        else:
            hasProt = "no_prot"
        if stress[0] != -1:
            canStress = "stress"
        else:
            canStress = "no_stress"
        if corpse[0] == -1:
            isCorpse = "notCorpse"
        else:
            isCorpse = "isCorpse"
        if currentPositionOfSkull[0] != oldPositionOfSkull and currentPositionOfSkull[0] != -1:
            number = number + 1
            oldPositionOfSkull = currentPositionOfSkull[0]

            enemies.append([number, position[1], position[2], hasProt, canStress, isCorpse])
        else:
            pass

    return enemies
```

Slika 25 - Funkcija koja prolazi kroz neprijateljske pozicije i pregledava njihova svojstva.

Kada su poznati svi dostupni potezi heroja i sve dostupne mete, provodi se Prolog upit koji traži najpovoljniju metu.

```
def whoToAttack(skill):
    returnMe = ["", [0,0]]
    usableFromPosition = Functor("usableFromPosition", 5)
    currentEnemies = Functor("currentEnemies", 6)

    LowFrom = Variable()
    HighFrom = Variable()
    LowTarget = Variable()
    HighTarget = Variable()
    PositionOfEnemy = Variable()
    Xcoordinate = Variable()
    Ycoordinate = Variable()
    Stress = Variable()
    Prot = Variable()
    Corpse = Variable()

    potentialVictims = []

    q = Query(usableFromPosition(skill[0], LowFrom, HighFrom, LowTarget, HighTarget),
              currentEnemies(PositionOfEnemy, Xcoordinate, Ycoordinate, Prot, Stress, Corpse))
    while q.nextSolution():
        ToWhereLow = int(LowTarget.get_value())
        ToWhereHigh = int(HighTarget.get_value())
        position = int(PositionOfEnemy.get_value())
        if ToWhereLow <= position and ToWhereHigh >= position:
            potentialVictims.append([int(PositionOfEnemy.value), int(Xcoordinate.value), int(Ycoordinate.value)])
    q.closeQuery()
    list = []
    for victim in potentialVictims:
        list.append([skill[0], [victim[1], victim[2]]])
    print(list)
    return list
```

Slika 26 - Upit za traženje mete

Funkcija za svaki dostupni potez vraća popis sa imenom tog poteza i koordinatama neprijatelja na kojima se potez može koristiti. Program bira potez sa vrha i izvršava ga, najprije pomoću naziva napada, preko imagesearch funkcije, dohvati koordinate ikone napda, pritisne ikonu napada, a zatim se pokazivač miša pomiče na koordinate iz popisa na kojima se nalazi neprijatelj i stišće se na njega.

Ovaj proces se ponavlja za svakog heroja dok borba nije gotova, nakon čega se nastavlja sa istraživanjem.

5.4. Nedostaci i moguća poboljšanja

Aplikacija nema direktne komunikacije sa programom, već je sve provedeno pregledom ekrana, korisnik ne može obavljati druge poslove u pozadini tijekom njenog izvođenja. Zbog korištenja imagesearch funkcija, potrebno je imati spremljene slike svih predmeta koje je potrebno prepoznati.

Programi za detektiranje teksta često krivo zaključuju vrijednosti zbog igričinih estetičkih razloga, potencijalno bi se umjesto konkretnih brojeva mogla gledati samo mjerila i crtice koja se nalaze ispod heroja i neprijatelja kako bi se saznalo zdravstveno stanje.

Imagesearch program uspoređuje uzorke u onoj rezoluciji u kojoj su mu spremljene slike, tako da povećanjem ili smanjenjem rezoluciji igrice dolazi do kvara i onemogućeno je prepoznavanje.

Stalno pregledavanje pozicija heroja i neprijatelja te njihovih stanja je vremenski zahtijevno te programu treba dugo vremena da završi određen nivo, potencijalnim preraspoređivanjem redoslijeda funkcija ili spremanjem stanja svih likova bi se moglo smanjiti vrijeme potrebno za provedbu jednoga kruga.

6. Zaključak

Razvojem tehnologije i ljudskog društva javlja se sve veća potreba za kompleksnijim sustavima koji obuhvaćaju više različitih problemskih domena i zahtijevaju velika ulaganja kako bi bila ostvariva. Sustavi trebaju savladavati složene zadatke, biti samostalniji, replicirati ljudsko znanje i koristiti se njime kako bi donosili odluke bez potrebe ljudskog utjecaja ili mogućnosti za pogrešku. Kako bi se izgradili ovakvi sustavi, potrebno je koristiti sva dostupna sredstva na njihov optimalan način, dok njihove manjkavosti treba nadoknaditi korištenjem drugih resursa.

U takvim sustavima bi logički programski jezici imali ključnu ulogu. Njihove sposobnosti shvaćanja odnosa među predmetima, stvaranja informacija, čitanja velikih količina podataka u kratko vrijeme i donošenja zaključaka su od velike koristi, ali, zbog poteškoća u drugim područjima, zahtijevaju potporu konvencionalnih višenamjenskih programskih jezika kako bi njihovi nedostaci bili umanjeni, a njihove prednosti optimizirane.

Potrebno je upoznati širu informatičku zajednicu sa mogućnostima koje jezici poput Prologa nude. Njihova uporaba zahtijeva drugačiji pristup od proceduralnih i objektno-orijentiranih jezika te je potrebno drugačije shvaćanje programiranja i logike. Iako njihova popularnost raste, logički programski jezici su manje zastupljeni i rijeđe spominjani te visok broj programera nije svijestan njihovih mogućnosti. Potrebno je povećati njihovu ulogu u programskim rješenjima i omogućiti više prilika da se demonstrira korist njihove uporabe prikažu svoje prednosti u kombinaciji sa općenitijim programskim jezicima.

Popis literature

[1] Clocksin, W.F., Mellish, C.S. (2003). *Programming in Prolog, Fifth Edition*

[2] NASA (2016). Why is formal methods necessary? < https://shemesh.larc.nasa.gov/fm/fm-why-new.html>. Pristupljeno 13. rujna 2021.

[3] Halper, F. B., Miklos, A. V., (1991). *Intelligent Databases: a program for research and development* < http://raw.rutgers.edu/MiklosVasarhelyi/Resume%20Articles/CHAPTERS%20IN%20BOOKS/C03.%20intelligent%20databases.pdf>. Pristupljeno 14. rujna 2021.

[4] Johnson, D., (2021). *Expert System in AI: What is, Applications & Example* < https://www.guru99.com/expert-systems-with-applications.html> Pristupljeno 14. rujna 2021.

[5] Kuhlman, D., (2013). *A Python Book: Beginning Python, Advanced Python, and Python Exercises* < http://www.davekuhlman.org/python_book_01.pdf>. Pristupljeno 12. rujna 2021.

[6] Lees, M., (2017). *How I made a python bot to automate a tactical mmorpg*, < https://medium.com/@martin.lees/how-i-made-a-python-bot-to-automate-a-tactical-mmorpg-9f6693350d10>. Pristupljeno 8. rujna 2021.

[7] Lloyd, J.W., (1984). *Foundations of Logic Programming: Second Extended Edition* < http://cgi.di.uoa.gr/~prondo/SEMANTICS/Lloyd.pdf>. Pristupljeno 15. rujna 2021.

[8] Merritt, D., (2000). *Building Expert Systems in Prolog*, < https://amzi.com/distribution/files/xsip_book.pdf>. Pristupljeno 14. rujna 2021.

[9] NASA (2016), What is formal methods? < https://shemesh.larc.nasa.gov/fm/fm-what.html>. Pristupljeno 13. rujna 2021.

[10] Python Software Foundation (2021). *Python Language Reference, Release 3.9.7.* < https://docs.python.org/3/reference/index.html>. Pristupljeno 13. rujna 2021.

[11] Python Software Foundation (2021). *Python Success Stories ForecastWatch* < https://www.python.org/about/success/forecastwatch/>. Pristupljeno 13. rujna 2021.

[12] Rowe, N. C., (2003). *Artificial Intelligence through Prolog* < https://doc.lagout.org/science/Artificial%20Intelligence/General/Artificial%20Intelligence%20Through%20Prolog%20-%20Neil%20C%20Rowe.pdf>. Pristupljeno 12. rujna 2021.

Popis slika

Prilog 1

Izvorni kod Python programa

```
from pyautogui import *
from python_imagesearch.imagesearch import *
from pyswip import *
import pyscreenshot
import pyswip
import numpy
import pydirectinput
import time
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract'
import PIL
from random import *
from ahk import AHK

###Prolog part

def prepareProlog():

    defineItems()
    defineQuestItems()
    defineHeirlooms()
    defineValuableItems()
    defineStatusEffects()
    defineWhatCuresWhat()
    defineCurios()
    defineEnemyCharacteristics()
    defineWhatWorksOnWhichCurio()
    defineAbilityTypes()
    defineClasses()
    defineAbilities()
    defineRanges()
    definePositions()

    #defineChangeableStuff()


###Atoms

def defineEnemyCharacteristics():
    p.assertz("enemyStat(stress)")
    p.assertz("enemyStat(no_stress)")

    p.assertz("enemyStat2(no_prot)")
    p.assertz("enemyStat2(prot)")

    p.assertz("enemyStat3(notCorpse)")
    p.assertz("enemyStat3(isCorpse)")
```

```
def definePositions():
    global p
    partyPosition = Functor("partyPosition", 3)
    p.assertz("partyPosition(4, 268, 431)")
    p.assertz("partyPosition(3, 392, 431)")
    p.assertz("partyPosition(2, 525, 431)")
    p.assertz("partyPosition(1, 663, 431)")

    p.assertz("enemyPosition(1, 945, 431)")
    p.assertz("enemyPosition(2, 1077, 431)")
    p.assertz("enemyPosition(3, 1199, 431)")
    p.assertz("enemyPosition(4, 1339, 431)")

    print("Positions defined")

def defineRanges():
    global p
    usableFromPosition = Functor("usableFromPosition", 5)

    p.assertz("usableFromPosition(absolution, 1,4, 1,4)")
    p.assertz("usableFromPosition(transform, 1,4, 1,4)")
    p.assertz("usableFromPosition(manacles, 2,3, 1,3)")
    p.assertz("usableFromPosition(beasts_Bile, 2,3, 2,3)")
    p.assertz("usableFromPosition(rage, 1,2, 1,3)")
    p.assertz("usableFromPosition(rake, 1,2, 1,2)")
    p.assertz("usableFromPosition(slam, 1,3, 1,2)")

    p.assertz("usableFromPosition(nervous_Stab, 1,4, 1,3)")
    p.assertz("usableFromPosition(festering_Vapours, 1,4, 1,4)")
    p.assertz("usableFromPosition(get_Down, 1,4, 1,4)")
    p.assertz("usableFromPosition(flashpowder, 1,4, 1,4)")
    p.assertz("usableFromPosition(fortifying_Vapours, 3,4, 1,4)")
    p.assertz("usableFromPosition(invigorating_Vapours, 3,4, 1,4)")
    p.assertz("usableFromPosition(protect_Me, 1,4, 1,4)")

    p.assertz("usableFromPosition(sniper_Shot, 3,4, 2,4)")
    p.assertz("usableFromPosition(suppressing_Fire, 3,4, 3,4)")
    p.assertz("usableFromPosition(snipers_Mark, 3,4, 2,4)")
    p.assertz("usableFromPosition(bola, 3,4, 1,2)")
    p.assertz("usableFromPosition(blindfire, 1,4, 1,4)")
    p.assertz("usableFromPosition(battlefield_Bandage, 3,4, 1,4)")
    p.assertz("usableFromPosition(rallying_Flare, 1,4, 1,4)")

    p.assertz("usableFromPosition(collect_Bounty, 1,3, 1,2)")
    p.assertz("usableFromPosition(mark_For_Death, 1,4, 1,4)")
    p.assertz("usableFromPosition(come_Hither, 1,4, 3,4)")
    p.assertz("usableFromPosition(uppercut, 1,2, 1,2)")
    p.assertz("usableFromPosition(flashbang, 2,4, 2,4)")
    p.assertz("usableFromPosition(finish_Him, 1,3, 1,3)")
    p.assertz("usableFromPosition(caltrops, 2,4, 3,4)")

    p.assertz("usableFromPosition(smite, 1,2, 1,2)")
    p.assertz("usableFromPosition(zealous_Accusation, 1,2, 1,2)")
    p.assertz("usableFromPosition(stunning_Blow, 1,2, 1,2)")
    p.assertz("usableFromPosition(bulwark_Of_Faith, 3,4, 1,4)")
```

```
p.assertz("usableFromPosition(battle_Heal, 1,4, 1,4)")
p.assertz("usableFromPosition(holy_Lance, 3,4, 2,4)")
p.assertz("usableFromPosition(inspiring_Cry, 1,4, 1,4)")

p.assertz("usableFromPosition(punish, 1,2, 1,2)")
p.assertz("usableFromPosition(rain_Of_Sorrows, 1,2, 3,4)")
p.assertz("usableFromPosition(exsanguinate, 1,2, 1,2)")
p.assertz("usableFromPosition(reclaim, 1,4, 1,4)")
p.assertz("usableFromPosition(redeem, 1,4, 1,4)")
p.assertz("usableFromPosition(endure, 1,4, 1,4)")
p.assertz("usableFromPosition(suffer, 1,4, 1,4)")

p.assertz("usableFromPosition(pick_To_The_Face, 1,3, 1,2)")
p.assertz("usableFromPosition(lunge, 3,4, 1,3)")
p.assertz("usableFromPosition(flashing_Daggers, 2,4, 2,3)")
p.assertz("usableFromPosition(shadow_Fade, 1,2, 1,4)")
p.assertz("usableFromPosition(thrown_Dagger, 2,4, 2,4)")
p.assertz("usableFromPosition(poison_Darts, 2,4, 1,4)")
p.assertz("usableFromPosition(toxin_Trickery, 1,4, 1,4)")

p.assertz("usableFromPosition(wicked_Hack, 1,2, 1,2)")
p.assertz("usableFromPosition(iron_Swan, 1,1, 4,4)")
p.assertz("usableFromPosition(barbaric_Yawp, 1,2, 1,2)")
p.assertz("usableFromPosition(if_It_Bleeds, 1,3, 2,3)")
p.assertz("usableFromPosition(breakthrough, 2,4, 1,3)")
p.assertz("usableFromPosition(adrenaline_Rush, 1,4, 1,4)")
p.assertz("usableFromPosition(bleed_Out, 1,1, 1,1)")

p.assertz("usableFromPosition(wicked_Slice, 1,3, 1,2)")
p.assertz("usableFromPosition(pistol_Shot, 2,4, 2,4)")
p.assertz("usableFromPosition(point_Blank_Shot, 1,1, 1,1)")
p.assertz("usableFromPosition(grapeshot_Blast, 2,3, 1,3)")
p.assertz("usableFromPosition(tracking_Shot, 1,4, 2,4)")
p.assertz("usableFromPosition(duelists_Advance, 2,4, 1,3)")
p.assertz("usableFromPosition(open_Vein, 1,3, 1,2)")

p.assertz("usableFromPosition(hounds_Rush, 2,4, 1,4)")
p.assertz("usableFromPosition(hounds_Harry, 1,4, 1,4)")
p.assertz("usableFromPosition(target_Whistle, 1,4, 1,4)")
p.assertz("usableFromPosition(cry_Havoc, 3,4, 1,4)")
p.assertz("usableFromPosition(guard_Dog, 1,4, 1,4)")
p.assertz("usableFromPosition(lick_Wound, 2,4, 1,4)")
p.assertz("usableFromPosition(blackjack, 1,2, 1,3)")

p.assertz("usableFromPosition(dirk_Stab, 1,4, 1,3)")
p.assertz("usableFromPosition(harvest, 2,3, 2,3)")
p.assertz("usableFromPosition(finale, 1,2, 1,4)")
p.assertz("usableFromPosition(solo, 3,4, 1,4)")
p.assertz("usableFromPosition(slice_Off, 2,3, 2,3)")
p.assertz("usableFromPosition(battle_Ballad, 3,4, 1,4)")
p.assertz("usableFromPosition(inspiring_Tune, 3,4, 1,4)")

p.assertz("usableFromPosition(chop, 1,2, 1,2)")
p.assertz("usableFromPosition(hew, 1,2, 1,2)")
p.assertz("usableFromPosition(purge, 1,1, 1,1)")
```

```
p.assertz("usableFromPosition(revenge, 1,4, 1,4)")
p.assertz("usableFromPosition(withstand, 1,3, 1,4)")
p.assertz("usableFromPosition(solemnity, 1,2, 1,4)")
p.assertz("usableFromPosition(intimidate, 1,1, 1,4)")

p.assertz("usableFromPosition(crush, 1,2, 1,3)")
p.assertz("usableFromPosition(rampart, 1,3, 1,2)")
p.assertz("usableFromPosition(bellow, 1,4, 1,4)")
p.assertz("usableFromPosition(defender, 1,4, 1,4)")
p.assertz("usableFromPosition(retribution, 1,3, 1,3)")
p.assertz("usableFromPosition(command, 1,4, 1,4)")
p.assertz("usableFromPosition(bolster, 1,4, 1,4)")

p.assertz("usableFromPosition(aimed_Shot, 3,4, 2,4)")
p.assertz("usableFromPosition(smokescreen, 3,4, 3,4)")
p.assertz("usableFromPosition(call_The_Shot, 3,4, 2,4)")
p.assertz("usableFromPosition(buckshot, 3,4, 1,2)")
p.assertz("usableFromPosition(sidearm, 1,4, 1,4)")
p.assertz("usableFromPosition(patch_Up, 3,4, 1,4)")
p.assertz("usableFromPosition(skeet_Shot, 1,4, 1,4)")

p.assertz("usableFromPosition(abyssal_Artillery, 3,4, 3,4)")
p.assertz("usableFromPosition(daemons_Pull, 2,4, 3,4)")
p.assertz("usableFromPosition(hands_From_The_Abyss, 1,2, 1,3)")
p.assertz("usableFromPosition(sacrificial_Stab, 1,3, 1,3)")
p.assertz("usableFromPosition(vulnerability_Hex, 1,4, 1,4)")
p.assertz("usableFromPosition(weakening_Curse, 1,4, 1,4)")
p.assertz("usableFromPosition(wyrd_Reconstruction, 1,4, 1,4)")

p.assertz("usableFromPosition(noxious_Blast, 2,4, 1,2)")
p.assertz("usableFromPosition(plague_Grenade, 3,4, 3,4)")
p.assertz("usableFromPosition(blinding_Gas, 3,4, 3,4)")
p.assertz("usableFromPosition(incision, 1,3, 1,2)")
p.assertz("usableFromPosition(battlefield_Medicine, 3,4, 1,4)")
p.assertz("usableFromPosition(emboldening_Vapours, 1,4, 1,4)")
p.assertz("usableFromPosition(disorienting_Blast, 2,4, 2,4)")

p.assertz("usableFromPosition(pierce, 1,3, 1,4)")
p.assertz("usableFromPosition(puncture, 1,4, 1,4)")
p.assertz("usableFromPosition(adders_Kiss, 1,1, 1,2)")
p.assertz("usableFromPosition(impale, 1,1, 1,4)")
p.assertz("usableFromPosition(expose, 1,3, 1,3)")
p.assertz("usableFromPosition(captivate, 2,3, 2,3)")
p.assertz("usableFromPosition(serpent_Sway, 1,3, 1,3)")

p.assertz("usableFromPosition(mace_Bash, 1,2, 1,2)")
p.assertz("usableFromPosition(judgement, 3,4, 1,4)")
p.assertz("usableFromPosition(dazzling_Light, 2,4, 1,3)")
p.assertz("usableFromPosition(divine_Grace, 3,4, 1,4)")
p.assertz("usableFromPosition(divine_Comfort, 2,4, 1,4)")
p.assertz("usableFromPosition(illumination, 1,3, 1,4)")
p.assertz("usableFromPosition(hand_Of_Light, 1,2, 1,3)")

print("Ranges defined")
```

```
def defineAbilities():
    global p
    characterAbility = Functor("characterAbility", 3)
    p.assertz("characterAbility(abomination, absolution, selfheal)")
    p.assertz("characterAbility(abomination, absolution, selfdestress)")
    p.assertz("characterAbility(abomination, manacles, stun)")
    p.assertz("characterAbility(abomination, manacles, attack)")
    p.assertz("characterAbility(abomination, beasts_Bile, blight)")
    p.assertz("characterAbility(abomination, beasts_Bile, attack)")
    p.assertz("characterAbility(abomination, transform, buff)")
    p.assertz("characterAbility(abomination, rage, attack)")
    p.assertz("characterAbility(abomination, rake, attack)")
    p.assertz("characterAbility(abomination, slam, movement)")
    p.assertz("characterAbility(abomination, slam, attack)")

    p.assertz("characterAbility(antiquarian, fortifying_Vapours, heal)")
    p.assertz("characterAbility(antiquarian, invigorating_Vapours, buff)")
    p.assertz("characterAbility(antiquarian, nervous_Stab, attack)")
    p.assertz("characterAbility(antiquarian, festering_Vapours, blight)")
    p.assertz("characterAbility(antiquarian, get_Down, buff)")
    p.assertz("characterAbility(antiquarian, get_Down, movement)")
    p.assertz("characterAbility(antiquarian, flashpowder, debuff)")
    p.assertz("characterAbility(antiquarian, flashpowder, reveal)")
    p.assertz("characterAbility(antiquarian, protect_Me, buff)")

    p.assertz("characterAbility(arbalest, battlefield_Bandage, heal)")
    p.assertz("characterAbility(arbalest, sniper_Shot, markable)")
    p.assertz("characterAbility(arbalest, sniper_Shot, attack)")
    p.assertz("characterAbility(arbalest, snipers_Mark, mark)")
    p.assertz("characterAbility(arbalest, suppressing_Fire, attack)")
    p.assertz("characterAbility(arbalest, rallying_Flare, buff)")
    p.assertz("characterAbility(arbalest, rallying_Flare, reveal)")
    p.assertz("characterAbility(arbalest, bola, movement)")
    p.assertz("characterAbility(arbalest, bola, attack)")
    p.assertz("characterAbility(arbalest, blindfire, attack)")

    p.assertz("characterAbility(bountyHunter, collect_Bounty, markable)")
    p.assertz("characterAbility(bountyHunter, collect_Bounty, attack)")
    p.assertz("characterAbility(bountyHunter, mark_For_Death, mark)")
    p.assertz("characterAbility(bountyHunter, mark_For_Death, debuff)")
    p.assertz("characterAbility(bountyHunter, come_Hither, mark)")
    p.assertz("characterAbility(bountyHunter, come_Hither, movement)")
    p.assertz("characterAbility(bountyHunter, uppercut, stun)")
    p.assertz("characterAbility(bountyHunter, uppercut, movement)")
    p.assertz("characterAbility(bountyHunter, flashbang, stun)")
    p.assertz("characterAbility(bountyHunter, flashbang, movement)")
    p.assertz("characterAbility(bountyHunter, finish_Him, attack)")
    p.assertz("characterAbility(bountyHunter, caltrops, bleed)")

    p.assertz("characterAbility(crusader, battle_Heal, heal)")
    p.assertz("characterAbility(crusader, inspiring_Cry, heal)")
    p.assertz("characterAbility(crusader, inspiring_Cry, destress)")
    p.assertz("characterAbility(crusader, smite, attack)")
    p.assertz("characterAbility(crusader, zealous_Accusation, attack)")
    p.assertz("characterAbility(crusader, stunning_Blow, stun)")
```

```
p.assertz("characterAbility(crusader, stunning_Blow, attack)")
p.assertz("characterAbility(crusader, holy_Lance, movement)")
p.assertz("characterAbility(crusader, holy_Lance, attack)")
p.assertz("characterAbility(crusader, bulwark_Of_Faith, buff)")

p.assertz("characterAbility(flagellant, punish, attack)")
p.assertz("characterAbility(flagellant, punish, bleed)")
p.assertz("characterAbility(flagellant, rain_Of_Sorrows, attack)")
p.assertz("characterAbility(flagellant, rain_Of_Sorrows, bleed)")
p.assertz("characterAbility(flagellant, exsanguinate, selfheal)")
p.assertz("characterAbility(flagellant, exsanguinate, attack)")
p.assertz("characterAbility(flagellant, exsanguinate, bleed)")
p.assertz("characterAbility(flagellant, reclaim, heal)")
p.assertz("characterAbility(flagellant, redeem, heal)")
p.assertz("characterAbility(flagellant, endure, destress)")
p.assertz("characterAbility(flagellant, suffer, heal)")
p.assertz("characterAbility(flagellant, suffer, cleanse)")

p.assertz("characterAbility(graverobber, lunge, attack)")
p.assertz("characterAbility(graverobber, lunge, movement)")
p.assertz("characterAbility(graverobber, thrown_Dagger, attack)")
p.assertz("characterAbility(graverobber, thrown_Dagger, markable)")
p.assertz("characterAbility(graverobber, toxin_Trickery, buff)")
p.assertz("characterAbility(graverobber, flashing_Daggers, attack)")
p.assertz("characterAbility(graverobber, flashing_Daggers, debuff)")
p.assertz("characterAbility(graverobber, pick_To_The_Face, attack)")
p.assertz("characterAbility(graverobber, pick_To_The_Face, piercing)")
p.assertz("characterAbility(graverobber, shadow_Fade, movement)")
p.assertz("characterAbility(graverobber, shadow_Fade, buff)")
p.assertz("characterAbility(graverobber, poison_Darts, blight)")
p.assertz("characterAbility(graverobber, poison_Darts, debuff)")

p.assertz("characterAbility(hellion, wicked_Hack, attack)")
p.assertz("characterAbility(hellion, iron_Swan, attack)")
p.assertz("characterAbility(hellion, adrenaline_Rush, selfheal)")
p.assertz("characterAbility(hellion, adrenaline_Rush, buff)")
p.assertz("characterAbility(hellion, barbaric_Yawp, stun)")
p.assertz("characterAbility(hellion, if_It_Bleeds, bleed)")
p.assertz("characterAbility(hellion, if_It_Bleeds, attack)")
p.assertz("characterAbility(hellion, breakthrough, attack)")
p.assertz("characterAbility(hellion, breakthrough, movement)")
p.assertz("characterAbility(hellion, bleed_Out, attack)")
p.assertz("characterAbility(hellion, bleed_Out, bleed)")

p.assertz("characterAbility(highwayman, point_Blank_Shot, attack)")
p.assertz("characterAbility(highwayman, point_Blank_Shot, movement)")
p.assertz("characterAbility(highwayman, tracking_Shot, attack)")
p.assertz("characterAbility(highwayman, tracking_Shot, buff)")
p.assertz("characterAbility(highwayman, tracking_Shot, reveal)")
p.assertz("characterAbility(highwayman, pistol_Shot, attack)")
p.assertz("characterAbility(highwayman, pistol_Shot, markable)")
p.assertz("characterAbility(highwayman, wicked_Slice, attack)")
p.assertz("characterAbility(highwayman, grapeshot_Blast, attack)")
p.assertz("characterAbility(highwayman, grapeshot_Blast, debuff)")
p.assertz("characterAbility(highwayman, duelists_Advance, attack)")
```

```
p.assertz("characterAbility(highwayman, duelists_Advance, movement)")
p.assertz("characterAbility(highwayman, open_Vein, attack)")
p.assertz("characterAbility(highwayman, open_Vein, bleed)")

p.assertz("characterAbility(houndmaster, lick_Wound, selfheal)")
p.assertz("characterAbility(houndmaster, blackjack, stun)")
p.assertz("characterAbility(houndmaster, blackjack, stun)")
p.assertz("characterAbility(houndmaster, hounds_Rush, attack)")
p.assertz("characterAbility(houndmaster, hounds_Rush, bleed)")
p.assertz("characterAbility(houndmaster, hounds_Rush, markable)")
p.assertz("characterAbility(houndmaster, hounds_Harry, attack)")
p.assertz("characterAbility(houndmaster, hounds_Harry, bleed)")
p.assertz("characterAbility(houndmaster, target_Whistle, mark)")
p.assertz("characterAbility(houndmaster, target_Whistle, debuff)")
p.assertz("characterAbility(houndmaster, guard_Dog, buff)")
p.assertz("characterAbility(houndmaster, cry_Havoc, destress)")

p.assertz("characterAbility(jester, inspiring_Tune, destress)")
p.assertz("characterAbility(jester, battle_Ballad, buff)")
p.assertz("characterAbility(jester, dirk_Stab, attack)")
p.assertz("characterAbility(jester, dirk_Stab, movement)")
p.assertz("characterAbility(jester, harvest, bleed)")
p.assertz("characterAbility(jester, harvest, attack)")
p.assertz("characterAbility(jester, slice_Off, bleed)")
p.assertz("characterAbility(jester, slice_Off, attack)")
p.assertz("characterAbility(jester, finale, attack)")
p.assertz("characterAbility(jester, finale, movement)")
p.assertz("characterAbility(jester, solo, movement)")
p.assertz("characterAbility(jester, solo, buff)")

p.assertz("characterAbility(leper, solemnity, selfheal)")
p.assertz("characterAbility(leper, solemnity, selfdestress)")
p.assertz("characterAbility(leper, chop, attack)")
p.assertz("characterAbility(leper, hew, attack)")
p.assertz("characterAbility(leper, purge, movement)")
p.assertz("characterAbility(leper, purge, attack)")
p.assertz("characterAbility(leper, revenge, buff)")
p.assertz("characterAbility(leper, withstand, buff)")
p.assertz("characterAbility(leper, intimidate, attack)")
p.assertz("characterAbility(leper, intimidate, reveal)")

p.assertz("characterAbility(manAtArms, rampart, stun)")
p.assertz("characterAbility(manAtArms, rampart, attack)")
p.assertz("characterAbility(manAtArms, rampart, movement)")
p.assertz("characterAbility(manAtArms, command, buff)")
p.assertz("characterAbility(manAtArms, bolster, buff)")
p.assertz("characterAbility(manAtArms, crush, attack)")
p.assertz("characterAbility(manAtArms, bellow, debuff)")
p.assertz("characterAbility(manAtArms, defender, buff)")
p.assertz("characterAbility(manAtArms, retribution, buff)")
p.assertz("characterAbility(manAtArms, retribution, attack)")

p.assertz("characterAbility(musketeer, patch_Up, heal)")
p.assertz("characterAbility(musketeer, aimed_Shot, markable)")
p.assertz("characterAbility(musketeer, aimed_Shot, attack)")
```

```
p.assertz("characterAbility(musketeer, call_The_Shot, mark)")
p.assertz("characterAbility(musketeer, smokescreen, debuff)")
p.assertz("characterAbility(musketeer, smokescreen, attack)")
p.assertz("characterAbility(musketeer, buckshot, movement)")
p.assertz("characterAbility(musketeer, buckshot, attack)")
p.assertz("characterAbility(musketeer, sidearm, attack)")
p.assertz("characterAbility(musketeer, skeet_Shot, buff)")
p.assertz("characterAbility(musketeer, skeet_Shot, reveal)")

p.assertz("characterAbility(occultist, wyrd_Reconstruction, heal)")
p.assertz("characterAbility(occultist, weakening_Curse, debuff)")
p.assertz("characterAbility(occultist, vulnerability_Hex, mark)")
p.assertz("characterAbility(occultist, abyssal_Artillery, attack)")
p.assertz("characterAbility(occultist, daemons_Pull, attack)")
p.assertz("characterAbility(occultist, daemons_Pull, movement)")
p.assertz("characterAbility(occultist, hands_From_The_Abyss, stun)")
p.assertz("characterAbility(occultist, hands_From_The_Abyss, attack)")
p.assertz("characterAbility(occultist, sacrificial_Stab, attack)")

p.assertz("characterAbility(plagueDoctor, noxious_Blast, attack)")
p.assertz("characterAbility(plagueDoctor, noxious_Blast, blight)")
p.assertz("characterAbility(plagueDoctor, disorienting_Blast, stun)")
p.assertz("characterAbility(plagueDoctor, disorienting_Blast, movement)")
p.assertz("characterAbility(plagueDoctor, plague_Grenade, attack)")
p.assertz("characterAbility(plagueDoctor, plague_Grenade, blight)")
p.assertz("characterAbility(plagueDoctor, blinding_Gas, stun)")
p.assertz("characterAbility(plagueDoctor, incision, attack)")
p.assertz("characterAbility(plagueDoctor, incision, bleed)")
p.assertz("characterAbility(plagueDoctor, battlefield_Medicine, heal)")
p.assertz("characterAbility(plagueDoctor, emboldening_Vapours, buff)")

p.assertz("characterAbility(shieldbreaker, pierce, attack)")
p.assertz("characterAbility(shieldbreaker, pierce, piercing)")
p.assertz("characterAbility(shieldbreaker, pierce, movement)")
p.assertz("characterAbility(shieldbreaker, puncture, attack)")
p.assertz("characterAbility(shieldbreaker, puncture, movement)")
p.assertz("characterAbility(shieldbreaker, adders_Kiss, blight)")
p.assertz("characterAbility(shieldbreaker, impale, blight)")
p.assertz("characterAbility(shieldbreaker, impale, movement)")
p.assertz("characterAbility(shieldbreaker, expose, reveal)")
p.assertz("characterAbility(shieldbreaker, expose, attack)")
p.assertz("characterAbility(shieldbreaker, captivate, attack)")
p.assertz("characterAbility(shieldbreaker, captivate, blight)")
p.assertz("characterAbility(shieldbreaker, captivate, markable)")
p.assertz("characterAbility(shieldbreaker, serpent_Sway, movement)")

p.assertz("characterAbility(vestal, divine_Grace, heal)")
p.assertz("characterAbility(vestal, divine_Comfort, heal)")
p.assertz("characterAbility(vestal, dazzling_Light, stun)")
p.assertz("characterAbility(vestal, dazzling_Light, attack)")
p.assertz("characterAbility(vestal, mace_Bash, attack)")
p.assertz("characterAbility(vestal, judgement, attack)")
p.assertz("characterAbility(vestal, illumination, reveal)")
p.assertz("characterAbility(vestal, illumination, debuff)")
p.assertz("characterAbility(vestal, hand_Of_Light, attack)")
```

```
    p.assertz("characterAbility(vestal, hand_Of_Light, buff)")

    print("Abilities defined")

def defineClasses():
    global p
    characterClass = Functor("characterClass", 1)
    p.assertz("characterClass(abomination)")
    p.assertz("characterClass(antiquarian)")
    p.assertz("characterClass(arbalest)")
    p.assertz("characterClass(bountyHunter)")
    p.assertz("characterClass(crusader)")
    p.assertz("characterClass(flagellant)")
    p.assertz("characterClass(graverobber)")
    p.assertz("characterClass(hellion)")
    p.assertz("characterClass(highwayman)")
    p.assertz("characterClass(houndmaster)")
    p.assertz("characterClass(jester)")
    p.assertz("characterClass(leper)")
    p.assertz("characterClass(manAtArms)")
    p.assertz("characterClass(musketeer)")
    p.assertz("characterClass(occultist)")
    p.assertz("characterClass(plagueDoctor)")
    p.assertz("characterClass(shieldbreaker)")
    p.assertz("characterClass(vestal)")

    print("Classes defined")

def defineAbilityTypes():
    global p
    abilityType = Functor("abilityType", 1)
    p.assertz("abilityType(heal)")
    p.assertz("abilityType(selfheal)")
    p.assertz("abilityType(destress)")
    p.assertz("abilityType(selfdestress)")
    p.assertz("abilityType(attack)")
    p.assertz("abilityType(bleed)")
    p.assertz("abilityType(blight)")
    p.assertz("abilityType(buff)")
    p.assertz("abilityType(debuff)")
    p.assertz("abilityType(stun)")
    p.assertz("abilityType(mark)")
    p.assertz("abilityType(movement)")
    p.assertz("abilityType(markable)")
    p.assertz("abilityType(piercing)")
    p.assertz("abilityType(cleanse)")
    p.assertz("abilityType(reveal)")

    print("Ability types defined")

def defineCurios():
    global p
    curio = Functor("curio", 1)
    p.assertz("curio(crate)")
    p.assertz("curio(sack)")
```

```
p.assertz("curio(ancientCoffin)")
p.assertz("curio(sarcophagus)")
p.assertz("curio(suitOfArmor)")
p.assertz("curio(sacrificialStone)")
p.assertz("curio(sconce)")
p.assertz("curio(bookshelf)")
p.assertz("curio(unlockedStrongbox)")
p.assertz("curio(discardedPack)")
p.assertz("curio(eldritchAltar)")
p.assertz("curio(heirloomChest)")
p.assertz("curio(shamblersAltar)")
p.assertz("curio(stackOfBooks)")
p.assertz("curio(alchemyTable)")
p.assertz("curio(altarOfLight)")
p.assertz("curio(confessionBooth)")
p.assertz("curio(decorativeUrn)")
p.assertz("curio(holyFountain)")
p.assertz("curio(ironMaiden)")
p.assertz("curio(lockedDisplayCabinet)")
p.assertz("curio(lockedSarcophagus)")
p.assertz("curio(boneAltar)")
p.assertz("curio(dinnerCart)")
p.assertz("curio(makeshiftDiningTable)")
p.assertz("curio(moonshineBarrel)")
p.assertz("curio(occultScrawlings)")
p.assertz("curio(pileOfBones)")
p.assertz("curio(pileOfScrolls)")
p.assertz("curio(rackOfBlades)")
p.assertz("curio(beastCarcass)")
p.assertz("curio(eerieSpiderweb)")
p.assertz("curio(leftLuggage)")
p.assertz("curio(mummifiedRemains)")
p.assertz("curio(oldTree)")
p.assertz("curio(pristineFountain)")
p.assertz("curio(shallowGrave)")
p.assertz("curio(travelersTent)")
p.assertz("curio(troublingEffigy)")
p.assertz("curio(barnacleCrustedChest)")
p.assertz("curio(bas-relief)")
p.assertz("curio(brackishTidePool)")
p.assertz("curio(eerieCoral)")
p.assertz("curio(fishIdol)")
p.assertz("curio(fishCarcass)")
p.assertz("curio(giantOyster)")
p.assertz("curio(shipsFigurehead)")

p.assertz("curio(animalisticShrine)")
p.assertz("curio(corruptedAltar)")
p.assertz("curio(infectedCorpse)")
p.assertz("curio(ironCrown)")
p.assertz("curio(protectiveWard)")
p.assertz("curio(chirugeonsSatchel)")
p.assertz("curio(foodstuffCrate)")
p.assertz("curio(reliquary)")
p.assertz("curio(shipmentCrate)")
```

```
    print("Curios defined")

def defineWhatWorksOnWhichCurio():
    global p
    curioAndWhatWorksOnIt = Functor("curioAndWhatWorksOnIt", 3)
    p.assertz("curioAndWhatWorksOnIt(crate, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(sack, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(ancientCoffin, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(sarcophagus, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(suitOfArmor, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(sacrificialStone, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(sconce, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(bookshelf, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(unlockedStrongbox, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(discardedPack, nothing, neutral)")
    p.assertz("curioAndWhatWorksOnIt(eldritchAltar, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(heirloomChest, skeletonKey, positive)")
    p.assertz("curioAndWhatWorksOnIt(shamblersAltar, torch, negative)")
    p.assertz("curioAndWhatWorksOnIt(stackOfBooks, torch, negative)")
    p.assertz("curioAndWhatWorksOnIt(alchemyTable, torch, positive)")
    p.assertz("curioAndWhatWorksOnIt(alchemyTable, medicinalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(altarOfLight, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(confessionBooth, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(decorativeUrn, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(decorativeUrn, shovel, negative)")
    p.assertz("curioAndWhatWorksOnIt(holyFountain, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(ironMaiden, medicinalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(lockedDisplayCabinet, skeletonKey, positive)")
    p.assertz("curioAndWhatWorksOnIt(lockedDisplayCabinet, shovel, positive)")
    p.assertz("curioAndWhatWorksOnIt(lockedSarcophagus, skeletonKey, positive)")
    p.assertz("curioAndWhatWorksOnIt(lockedSarcophagus, shovel, positive)")
    p.assertz("curioAndWhatWorksOnIt(boneAltar, nothing, positive)")
    p.assertz("curioAndWhatWorksOnIt(dinnerCart, medicalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(makeshiftDiningTable, medicalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(moonshineBarrel, medicalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(occultScrawlings, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(pileOfBones, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(pileOfScrolls, torch, positive)")
    p.assertz("curioAndWhatWorksOnIt(rackOfBlades, bandage, positive)")
    p.assertz("curioAndWhatWorksOnIt(beastCarcass, medicinalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(eerieSpiderweb, bandage, positive)")
    p.assertz("curioAndWhatWorksOnIt(leftLuggage, skeletonKey, positive)")
    p.assertz("curioAndWhatWorksOnIt(leftLuggage, antivenom, positive)")
    p.assertz("curioAndWhatWorksOnIt(mummifiedRemains, bandage, positive)")
    p.assertz("curioAndWhatWorksOnIt(oldTree, antivenom, positive)")
    p.assertz("curioAndWhatWorksOnIt(pristineFountain, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(shallowGrave, shovel, positive)")
    p.assertz("curioAndWhatWorksOnIt(travelersTent, nothing, positive)")
    p.assertz("curioAndWhatWorksOnIt(troublingEffigy, holyWater, positive)")
    p.assertz("curioAndWhatWorksOnIt(barnacleCrustedChest, shovel, positive)")
    p.assertz("curioAndWhatWorksOnIt(bas-relief, shovel, positive)")
    p.assertz("curioAndWhatWorksOnIt(brackishTidePool, antivenom, positive)")
    p.assertz("curioAndWhatWorksOnIt(eerieCoral, medicinalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(fishIdol, holyWater, positive)")
```

```
    p.assertz("curioAndWhatWorksOnIt(fishCarcass, medicinalHerbs, positive)")
    p.assertz("curioAndWhatWorksOnIt(giantOyster, shovel, positive)")
    p.assertz("curioAndWhatWorksOnIt(shipsFigurehead, nothing, positive)")

    p.assertz("curioAndWhatWorksOnIt(animalisticShrine, pick-axe, positive)")
    p.assertz("curioAndWhatWorksOnIt(corruptedAltar, consecratedEssence, positive)")
    p.assertz("curioAndWhatWorksOnIt(infectedCorpse, potentSalve, positive)")
    p.assertz("curioAndWhatWorksOnIt(ironCrown, handOfGlory, positive)")
    p.assertz("curioAndWhatWorksOnIt(protectiveWard, pinealGland, positive)")
    p.assertz("curioAndWhatWorksOnIt(chirugeonsSatchel, nothing, positive)")
    p.assertz("curioAndWhatWorksOnIt(foodstuffCrate, nothing, positive)")
    p.assertz("curioAndWhatWorksOnIt(reliquary, nothing, positive)")
    p.assertz("curioAndWhatWorksOnIt(shipmentCrate, nothing, positive)")

    print("Curio interactions defined")

def defineWhatCuresWhat():
    global p
    cures = Functor("cures", 2)
    p.assertz("cures(bandages, bleed)")
    p.assertz("cures(antivenom, blight)")
    p.assertz("cures(medicinalHerbs, debuff)")
    p.assertz("cures(laudanum, horror)")
    p.assertz("cures(theBlood, craving)")
    p.assertz("cures(theBlood, wasting)")
    p.assertz("cures(food_1, deathsDoor)")
    p.assertz("cures(food_2, deathsDoor)")
    p.assertz("cures(food_3, deathsDoor)")

    print("Cures defined")

def defineStatusEffects():
    global p
    statusEffect = Functor("statusEffect", 1)
    p.assertz("statusEffect(bleed)")
    p.assertz("statusEffect(blight)")
    p.assertz("statusEffect(debuff)")
    p.assertz("statusEffect(mark)")
    p.assertz("statusEffect(stun)")
    p.assertz("statusEffect(horror)")
    p.assertz("statusEffect(deathsDoor)")
    #p.assertz("statusEffect(craving)")
    #p.assertz("statusEffect(wasting)")

    print("Status effects defined")

def defineValuableItems():
    global p
    valuableItem = Functor("valuableItem", 2)
    valueAndHowManyStacks = Functor("valueAndHowManyStacks", 2)
    p.assertz("valuableItem(gold1, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(gold2, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(gold3, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(gold4, valueAndHowManyStacks(1,1750))")
    p.assertz("valuableItem(citrine, valueAndHowManyStacks(250,5))")
```

```
    p.assertz("valuableItem(jade, valueAndHowManyStacks(375,5))")
    p.assertz("valuableItem(onyx, valueAndHowManyStacks(500,5))")
    p.assertz("valuableItem(emerald, valueAndHowManyStacks(750,5))")
    p.assertz("valuableItem(sapphire, valueAndHowManyStacks(1000,5))")
    p.assertz("valuableItem(ruby, valueAndHowManyStacks(1250,5))")
    p.assertz("valuableItem(juteTapestry, valueAndHowManyStacks(4500,1))")
    p.assertz("valuableItem(puzzlingTrapezohedron, valueAndHowManyStacks(3500,1))")
    p.assertz("valuableItem(minorAntique, valueAndHowManyStacks(500,20))")
    p.assertz("valuableItem(rareAntique, valueAndHowManyStacks(1250,5))")
    p.assertz("valuableItem(consecratedPew, valueAndHowManyStacks(2500,1))")
    p.assertz("valuableItem(cometShard, valueAndHowManyStacks(0,99))")

    print("Valuable items defined")

def defineItems():
    global p
    item = Functor("item", 1)
    p.assertz("item(food_1)")
    p.assertz("item(food_2)")
    p.assertz("item(food_3)")
    p.assertz("item(shovel)")
    p.assertz("item(antivenom)")
    p.assertz("item(bandages)")
    p.assertz("item(medicinalHerbs)")
    p.assertz("item(skeletonKey)")
    p.assertz("item(holyWater)")
    p.assertz("item(laudanum)")
    p.assertz("item(torch)")
    p.assertz("item(firewood)")
    p.assertz("item(theBlood)")
    p.assertz("item(dogTreats)")

    print("Items defined")

def defineHeirlooms():
    global p
    item = Functor("heirloom", 2)
    p.assertz("heirloom(bust, 2)")
    p.assertz("heirloom(deed, 3)")
    p.assertz("heirloom(crest, 1)")
    p.assertz("heirloom(portrait, 4)")

    print("Heirlooms defined")

def defineQuestItems():
    global p
    questItem = Functor("questItem", 1)
    p.assertz("questItem(pick-axe)")
    p.assertz("questItem(consecratedEssence)")
    p.assertz("questItem(potentSalve)")
    p.assertz("questItem(handOfGlory)")
    p.assertz("questItem(pinealGland)")
    p.assertz("questItem(medicine)")
    p.assertz("questItem(oneGrainSack)")
    p.assertz("questItem(oneHolyRelic)")
```

```
    p.assertz("questItem(oneAncestorsRelic)")

    print("Quest items defined")

    ###End of Prolog part

    ###Utility

def firstCountdown():
    for x in range(2):
        time.sleep(1)
        print("Program starts in: ", 2 - x)
    time.sleep(1)

def shouldIContinue():
    playerInput = "q"
    firstTry = 1
    global keepDoingIt
    global dungeonCompleted

    while playerInput != "y" and playerInput != "n":

        if firstTry != 1:
            playerInput = input("It has to be y or n! \n")

        else:
            playerInput = input("Do you want to keep going? (y/n) \n")


        if playerInput == "y":
            keepDoingIt = True
            dungeonCompleted = False

        elif playerInput == "n":
            keepDoingIt = False

        firstTry = 0

def goodbyeMessage():

    print("Thank you for using this program! \n")

###

###Positioning and clicking

def shiftClick(xCoordinate, yCoordinate, delay = 0.3, sleep = 0.1):
    ahk = AHK()
    ahk.key_down("shift")
    if xCoordinate > 20 or yCoordinate > 20:
        pyautogui.moveTo(xCoordinate, yCoordinate, delay)
        ahk.click()
        ahk.click()
        time.sleep(sleep)
        ahk.click()
```

```
        ahk.click()
        time.sleep(sleep)
    ahk.key_up("shift")

def moveForABit():
    pydirectinput.keyDown("d")
    time.sleep(1)
    pydirectinput.keyUp("d")

def clickMouse(xCoordinate, yCoordinate, delay = 0.1, sleep = 0.1):
    ahk = AHK()
    if xCoordinate > 20 or yCoordinate > 20:
        pyautogui.moveTo(xCoordinate, yCoordinate, delay)
        ahk.click()
        ahk.click()
        time.sleep(sleep)
        ahk.click()
        ahk.click()
        time.sleep(sleep)

def holdDAndCheckForTrap():
    ahk = AHK()
    pydirectinput.keyDown("d")
    pydirectinput.keyDown("w")
    clickMouse(1073, 557)
    time.sleep(0.3)
    clickMouse(1073, 557)
    time.sleep(0.3)
    clickMouse(1073, 557)
    time.sleep(0.3)
    clickMouse(1073, 557)
    time.sleep(0.3)
    clickMouse(1073, 557)
    pydirectinput.keyUp("d")
    pydirectinput.keyUp("w")


def rightClickMouse(xCoordinate, yCoordinate, delay = 0.3, sleep = 0.1):

    ahk = AHK()
    if xCoordinate > 20 or yCoordinate > 20:
        pyautogui.moveTo(xCoordinate, yCoordinate, delay)
        ahk.right_click()
        ahk.right_click()
        time.sleep(sleep)
        ahk.right_click()
        ahk.right_click()
        time.sleep(sleep)

def moveMouse(x,y):

    if x >= 1 or y >= 1:
        pyautogui.moveTo(x,y,0.3)

def pressEsc():
```

```
    pydirectinput.press("esc")

def pressW():

    pydirectinput.press("w")

####

###Inventory and curio things

def clickOnReOrderParty():
    results = imagesearch("./imagesForTheProgram/Utility/Utility_Center_On_Party.png",0.8)
    clickMouse(results[0],results[1])

def lookAtTorch():

    global recentlyLooked

    if recentlyLooked == True:
        recentlyLooked = False
        return

    time.sleep(0.2)
    moveMouse(800,123)
    time.sleep(0.2)

    results = str(imagesearch_from_folder("./imagesForTheProgram/TorchStuff/",0.9))
    print()
    results = results[2:]
    results = results[:len(results)-1]
    listOfResults = results.split(", ")

    listOfResults = [i.replace("./imagesForTheProgram/TorchStuff/", "") for i in listOfResults]
    listOfResults = [i.replace(".png'", "") for i in listOfResults]
    print()

    filteredList = []

    recentlyLooked = True

    for j in listOfResults:
        if j.split(": ")[1] != "[-1, -1]":
            filteredList += [j]
    if len(filteredList) != 0:
        torchLevel = filteredList[0].split(":")[0]
        howManyTorchesNeeded = countLight(torchLevel)
        useTorches(howManyTorchesNeeded)

def useTorches(number):
    toUse = number
    while doIHave("torch") == True and toUse != 0:
        pydirectinput.press('t')
        canIkeepGoing = checkIfItExists("InventoryStuff/InventoryStuff_torch")
        if canIkeepGoing == False:
            removeFromInventory("torch")
```

```
            else:
                pass
            toUse = toUse - 1

        if number == toUse:
            print("You don't need to use torches right now")
        elif toUse != 0:
            print("You ran out of torches")
        else:
            print("light raised to max")

    def countLight(lightLevel):
        howMany = 0
        if lightLevel == "BlackAsPitch":
            howMany = 4
        elif lightLevel == "Dark":
            howMany = 3
        elif lightLevel == "Shadowy":
            howMany = 2
        elif lightLevel == "DimLight":
            howMany = 1
        return howMany

    def removeEverythingInInventory():
        global p
        p.retractall("playerHas(_)")

    def whatsInTheInventory():
        removeEverythingInInventory()
        pressInventory()
        addSomethingGroup("QuestStuff")
        addSomethingGroup("InventoryStuff")
        addSomethingGroup("ValuableStuff")
        addSomethingGroup("HeirloomStuff")

    def pressInventory():
        results = imagesearch("./imagesForTheProgram/Utility/Utility_Inventory.png")
        clickMouse(results[0]+5, results[1]+10)

    def addToInventory(thingToAdd):
        global p
        p.assertz("playerHas("+thingToAdd+")")

    def currentInventory():
        global p
        print("In your inventory: ")
        playerHas = Functor("playerHas", 1)

        A = Variable()

        q = Query(playerHas(A))
        while q.nextSolution():
            print(A.get_value())

        q.closeQuery()
```

```
def checkIfItExists(thing):
    results = imagesearch("./imagesForTheProgram/"+ thing+".png", 0.8)
    if results[0] != -1:
        return True
    else:
        return False

def addSomethingGroup(theStuffThatsBeingAdded):
    results =
str(imagesearch_from_folder("./imagesForTheProgram/"+theStuffThatsBeingAdded+"/", 0.8))
    results = results[2:]
    results = results[:len(results)-1]
    listOfResults = results.split(", '")

    listOfResults =
[i.replace("./imagesForTheProgram/"+theStuffThatsBeingAdded+"/"+theStuffThatsBeingAdde
d+"_", "") for i in listOfResults]
    listOfResults = [i.replace(".png'", "") for i in listOfResults]

    filteredList = []

    for j in listOfResults:
        if j.split(": ")[1] != "[-1, -1]":
            filteredList += [j]

    filteredList2 = [i.split(":")[0] for i in filteredList]

    for k in filteredList2:
        print(k)
        addToInventory(k)

def whatAmIPreparedToHeal():
    stuffToCure = []

    cures = Functor("cures", 2)
    playerHas = Functor("playerHas", 1)

    X = Variable()
    Y = Variable()

    q = Query(cures(X,Y), playerHas(X))
    while q.nextSolution():
        stuffToCure.append(str(Y.value))
    q.closeQuery()

    print(stuffToCure)

    return stuffToCure

def doIHave(item):
    answer = 0
    playerHas = Functor("playerHas", 1)
    q = Query(playerHas(item))
    while q.nextSolution():
```

```
        answer = answer + 1
    q.closeQuery()

    if answer == 0:
        return False
    else:
        return True

def removeFromInventory(item):
    global p
    p.retract("playerHas("+item+")")

def checkFor(item):
    res = imagesearch(".\imagesForTheProgram\InventoryStuff\InventoryStuff_"+item+".png")
    if res[0] == -1:
        removeFromInventory(item)

###

###Characters

def checkOnParty():
    global wasInFight

    if wasInFight == True:
        searchForAilmentsOnEveryone()
        wasInFIght = False
    lookAtTorch()
    lookAtHealth()

def lookAtHealth():

    positions = currentPositions()

    for position in positions:
        if doTheyNeedIt(position):
            cure("deathsDoor", position)


def searchForAilmentsOnOnePerson(position):

    whatCanBeCured = whatAmIPreparedToHeal()

    for affliction in whatCanBeCured:
        if affliction != "deathsDoor":
            if doTheyNeedIt(position, 0.6):
                cure(affliction, position)

def searchForAilmentsOnEveryone():

    clickOnReOrderParty()
    pressInventory()

    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    partyPosition = Functor("partyPosition",3)
```

```
    whatCanBeCured = whatAmIPreparedToHeal()

    print("What can be cured: ", whatCanBeCured)

    whatThePartyHas = lookForAfflictions(whatCanBeCured)
    positions = currentPositions()

    print("Looking for sickness")

    for affliction in whatCanBeCured:
        if affliction != "deathsDoor":
            for position in positions:
                if doTheyNeedIt(position):
                    cure(affliction, position)

    checkEverything()

def doTheyNeedIt(position, ratio = 0.5):

    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    partyPosition = Functor("partyPosition",3)

    X = Variable()
    Y = Variable()
    Z = Variable()
    A = Variable()
    B = Variable()
    C = Variable()

    curr = 0
    max = 0

    q = Query(partyPosition(X, position[0], position[1]), currentlyAvailableCharacters(X, Z, A,
B, C))
    while q.nextSolution():
        max = float(A.value)
        curr = float(B.value)
    q.closeQuery()
    print(curr/max)
    if curr/max < ratio:
        return True
    else:
        return False

def cure(affliction, position):
    item = ""

    cures = Functor("cures", 2)
    playerHas = Functor("playerHas", 1)

    X = Variable()

    q = Query(cures(X,affliction), playerHas(X))
    while q.nextSolution():
```

```
        item = str(X.value)
    q.closeQuery()

    clickMouse(position[0],position[1])
    if item != "":
        res =
imagesearch(".\imagesForTheProgram\InventoryStuff\InventoryStuff_"+item+".png")
        if res[0] != -1:
            rightClickMouse(res[0],res[1],0.05,0.05)
            checkUpCharactersHealth(position)
        checkFor(item)

def checkUpCharactersHealth(position):
    global currentCharacter
    global p

    findCurrentCharacter()

    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    partyPosition = Functor("partyPosition", 3)

    maxHealth, currHealth, currStress = findCurrentCharactersHealthAndStress()

    X = Variable()
    Y = Variable()
    Z = Variable()
    A = Variable()
    B = Variable()

    positionToSend = 0

    q = Query(partyPosition(X, position[0], position[1]), currentlyAvailableCharacters(X, Y, Z,
A, B))
    while q.nextSolution():
        positionToSend = int(X.value)
        forRemoving = "currentlyAvailableCharacters(" + str(X.value) + "," + str(Y.value) + "," +
str(Z.value) + "," + str(A.value) + "," + str(B.value) +")"
    q.closeQuery()
    p.retract(forRemoving)


p.assertz("currentlyAvailableCharacters("+str(positionToSend)+","+currentCharacter+","+str(
maxHealth)+","+str(currHealth)+","+ str(currStress)+")")

def currentPositions():
    allNumbers = []

    partyPosition = Functor("partyPosition",3)
    X = Variable()
    Y = Variable()
    Z = Variable()
    q = Query(partyPosition(X,Y,Z))
    while q.nextSolution():
        allNumbers.append([int(Y.value), int(Z.value)])
    q.closeQuery()
```

```
    return allNumbers

def lookForAfflictions(whatCanBeCured):
    diseasesFound = []

    for disease in whatCanBeCured:
        result = imagesearch(".\imagesForTheProgram\Afflictions\Affliction_"+disease+".png",
0.2)
        if result[0] != -1:
            diseasesFound.append([result,disease])


    filteredDiseases = []
    for disease in diseasesFound:
        if disease[0][0] > 198 and disease[0][1] > 572 and disease[0][0] < 726 and disease[0][1]
< 623:
            filteredDiseases.append(disease)

    print(filteredDiseases)
    return filteredDiseases

def currentCharactersAndSkills():
    global p

    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    currentlyAvailableSkills = Functor("currentlyAvailableSkills", 2)
    fail = Functor("fail", 0)

    A = Variable()
    B = Variable()
    C = Variable()
    X = Variable()
    Y = Variable()
    Z = Variable()

    q = Query(currentlyAvailableCharacters(A,B,X,Y,Z), currentlyAvailableSkills(B, C))


    while q.nextSolution():
        print("Position: ", A.get_value(), " | Hero: ", B.get_value(), " | Ability: ", C.get_value())

    q.closeQuery()

def findCurrentCharacter():
    global currentCharacter
    global p

    results =
str(imagesearch_from_folder("./imagesForTheProgram/CharacterStuff/Portraits/",0.9))
    print()
    results = results[2:]
    results = results[:len(results)-1]
    listOfResults = results.split(", '")
    time.sleep(0.1)
```

```
    listOfResults = [i.replace("./imagesForTheProgram/CharacterStuff/Portraits/", "") for i in
listOfResults]
    listOfResults = [i.replace("_pic.png'", "") for i in listOfResults]
    print()

    filteredList = []

    for j in listOfResults:
        if j.split(": ")[1] != "[-1, -1]":
            filteredList += [j]

    for k in filteredList:
        currentCharacter = k.split(":")[0]

def findCurrentCharactersAbilities():
    global currentCharacter
    global currentCharacterAbilities

    if currentCharacter == "":
        pass

    else:
        currentCharacterAbilities.clear()
        results =
str(imagesearch_from_folder("./imagesForTheProgram/CharacterStuff/"+currentCharacter+"/
" ,0.8))
        results = results[2:]
        results = results[:len(results)-1]
        listOfResults = results.split(", '")
        time.sleep(0.2)

        listOfResults =
[i.replace("./imagesForTheProgram/CharacterStuff/"+currentCharacter+"/"+currentCharacter
+"_", "") for i in listOfResults]
        listOfResults = [i.replace(".png'", "") for i in listOfResults]

        print()

        filteredList = []

        for j in listOfResults:
            #print(j.split(": ")[1] == "[-1, -1]")
            #print(j.split(": ")[1])
            if j.split(": ")[1] != "[-1, -1]":
                filteredList += [j]

        counter = 0
        for k in filteredList:
            counter = counter + 1

        for k in filteredList:
            #print(k)
            currentCharacterAbilities.append(k)

def removeExistingCharactersAndSkills():
```

```
    global allCharacters
    allCharacters.clear()
    removeCurrentCharacters()
    removeCurrentSkills()

def removeCurrentCharacters():
    global p
    p.retractall("currentlyAvailableCharacters(_,_,_,_,_)")

def removeCurrentSkills():
    global p
    p.retractall("currentlyAvailableSkills(_,_)")

def whoAreTheHeros():
    global p
    global currentCharacter
    global currentCharacterAbilities
    global allCharacters

    partyPosition = Functor("partyPosition", 3)
    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    currentlyAvailableSkills = Functor("currentlyAvailableSkills", 2)
    retractall = Functor("retractall", 1)

    removeExistingCharactersAndSkills()

    X = Variable()
    Y = Variable()
    Z = Variable()

    q = Query(partyPosition(X, Y, Z))

    position = 0

    allSkills = []

    while q.nextSolution():
        clickMouse(Y.value, Z.value)
        print("Looking at position: ", X.value)

        position = X.value
        findCurrentCharacter()
        print("Current character is: ", currentCharacter)
        maxHealth, currHealth, stress = findCurrentCharactersHealthAndStress()

        allCharacters.append([position, currentCharacter, maxHealth, currHealth, stress])

        findCurrentCharactersAbilities()

        for k in currentCharacterAbilities:
            print(k)
            allSkills.append(k.split(":")[0])

        currentCharacter = ""
        time.sleep(0.4)
```

```
    q.closeQuery()

    firstSkill = 0

    for hero in allCharacters:
        forAdding = "currentlyAvailableCharacters(" + str(hero[0]) + ", " + hero[1] + ", "+ hero[2]
+ ", "+ hero[3]+ ", "+ hero[4]+")"
        p.assertz(forAdding)
        print(forAdding)
        for i in range(firstSkill, firstSkill + 4):
            forAdding = "currentlyAvailableSkills(" + hero[1] + ", " + allSkills[i] + ")"
            print(forAdding)
            p.assertz(forAdding)

        firstSkill = firstSkill + 4

def findCurrentCharactersHealthAndStress():

    healthImage = pyscreenshot.grab(bbox=(305, 745, 380, 800))
    healthImage.save("checkThis.png")

    image = PIL.Image.open("checkThis.png")
    health = pytesseract.image_to_string(image)

    print(health)

    maxHealth = ""
    currHealth = ""
    currStress = ""

    try:
        maxHealth = health.split("/")[1].split("\n")[0]
    except Exception as ex:
        maxHealth = str(40)
    except ValueError as vex:
        maxHealth = str(40)

    try:
        currHealth = health.split("/")[0]
    except Exception as ex:
        currHealth = str(20)
    except ValueError as vex:
        currHealth = str(20)

    try:
        currStress = health.split("\n")[1].split("/")[0]
    except Exception as ex:
        currStress = str(50)
    except ValueError as vex:
        currStress = str(50)

    try:
        maxHealth = int(maxHealth)
        maxHealth = str(maxHealth)
```

```
    except Exception as ex:
        maxHealth = str(40)
    except ValueError as vex:
        maxHealth = str(40)

    try:
        currHealth = int(currHealth)
        currHealth = str(currHealth)
    except Exception as ex:
        currHealth = str(20)
    except ValueError as vex:
        currHealth = str(20)

    try:
        currStress = int(currStress)
        currStress = str(currStress)
    except Exception as ex:
        currStress = str(50)
    except ValueError as vex:
        currStress = str(50)


    print(maxHealth)
    print(currHealth)
    print(currStress)

    if int(maxHealth) > 90:
        maxHealth = str(40)
    if int(currHealth) > 90:
        currHealth = str(40)

    return maxHealth, currHealth, currStress


###

###Map cotnrols

def placeIntoRoom():
    global p
    p.retractall("partyLocation(_)")
    p.assertz("partyLocation(room)")

def placeIntoHallway():
    global p
    p.retractall("partyLocation(_)")
    p.assertz("partyLocation(hallway)")

def selectMap():
    result = imagesearch("./imagesForTheProgram/Utility/Utility_Map.png", 0.7)
    clickMouse(result[0]+5, result[1]+5)

def selectCenteringOnParty():
    result = imagesearch("./imagesForTheProgram/Utility/Utility_Center_On_Party.png", 0.7)
    clickMouse(result[0]+10, result[1]+10)
```

```
def leaveTheRoom():
    somethingClickable = False
    counter = 0

    while somethingClickable == False and counter < 5:
        results = str(imagesearch_from_folder("./imagesForTheProgram/BouncyStuff/", 0.8))
        results = results[2:]
        results = results[:len(results)-1]
        listOfResults = results.split(", ")

        for i in listOfResults:
            if i.split(": ")[1] != "[-1, -1]":
                somethingClickable = True
                x = int(i.split(": (")[1].split(",")[0])
                y = int(i.split(", ")[1].split(")")[0])
                print(x," ", y)
        counter = counter + 1

    if counter == 5:
        return [0,0]
    else:
        return [x+5, y+5]

def selectWhereToGo():
    megaZoomMap()
    coordinates = leaveTheRoom()
    if coordinates[0] == 0:
        return False
    else:
        clickMouse(coordinates[0],coordinates[1])

    moveMouse(400, 400)
    zoomMapProperly()
    whereToGo = findOutWhereToGo()

    print(whereToGo)

    print("I'm going: ", whereToGo)
    goBackToTheRoom()

    selectCenteringOnParty()
    moveToNextRoom(whereToGo)

    print("Moving")

    return True

def fullyZoomMap():

    pydirectinput.press("ADD", 6)

def megaZoomMap():
    pydirectinput.press("ADD", 6)
    pydirectinput.press("SUBTRACT", 4)
```

```
def bouncyZoomMap():
    pydirectinput.press("ADD", 6)
    pydirectinput.press("SUBTRACT", 3)

def zoomMapProperly():
    pydirectinput.press("ADD", 6)
    pydirectinput.press("SUBTRACT", 5)

def goBackToTheRoom():

    pydirectinput.press("w")

def moveToNextRoom(directions):
    global comingFrom

    allDirections = ["left", "right", "down", "up"]
    toPickFrom = []

    if len(directions) == 1:
        toPickFrom.append(directions[0][1])

    for direct in directions:
        if direct[0] == 2 and direct[0] != comingFrom:
            toPickFrom.append(direct[1])
    for direct in directions:
        if direct[0] == 1 and direct[0] != comingFrom:
            toPickFrom.append(direct[1])
    for direct2 in allDirections:
        found = False
        for direct in toPickFrom:
            if direct2 == direct:
                found = True
        if found == False:
            toPickFrom.append(direct2)

    if toPickFrom[0] == toPickFrom[1]:
        toPickFrom.pop(1)
    print(toPickFrom)
    bouncyZoomMap()

    moved = False

    while moved == False:
        for d in toPickFrom:
            x, y = getCoordinatesForMap(d)
            if d == "left" or d == "right":
                for i in range(4):
                    for j in range(4):
                        if moved == False:
                            x, y = getCoordinatesForMap(d)
                            moveMouse(x + (i * 30), y + (j * 30))
                            results =
imagesearch("./imagesForTheProgram/Utility/Utility_Move_To.png",0.8)
                            time.sleep(0.2)
```

```
                    if results[0] != -1:
                        moved = True
                        comingFrom = oppositeDirection(d)
                        clickMouse(x + (i * 30), y + (j * 30))
            else:
                for j in range(4):
                    for i in range(4):
                        if moved == False:
                            x, y = getCoordinatesForMap(d)
                            moveMouse(x + (i * 30), y + (j * 30))
                            results =
imagesearch("./imagesForTheProgram/Utility/Utility_Move_To.png",0.8)
                            time.sleep(0.2)
                            if results[0] != -1:
                                moved = True
                                comingFrom = oppositeDirection(d)
                                clickMouse(x + (i * 30), y + (j * 30))

def oppositeDirection(direction):
    opposite = ""
    if direction == "up":
        opposite = "down"
    elif direction == "down":
        opposite = "up"
    elif direction == "left":
        opposite = "right"
    elif direction == "right":
        opposite = "left"

    return opposite

def getCoordinatesForMap(direction):
    x1 = 0
    y1 = 0
    x2 = 0
    y2 = 0
    if direction == "up":
        x1 = 1015
        y1 = 660
        x2 = 1125
        y2 = 730
    elif direction == "down":
        x1 = 1015
        y1 = 770
        x2 = 1125
        y2 = 860
    elif direction == "left":
        x1 = 945
        y1 = 710
        x2 = 1040
        y2 = 810
    elif direction == "right":
        x1 = 1080
        y1 = 710
        x2 = 1140
```

```
        y2 = 810
    return x1, y1

def findOutWhereToGo():

    global comingFrom

    Up, EmptyUp = WhatsCompleted("up")
    Down, EmptyDown = WhatsCompleted("down")
    Left, EmptyLeft = WhatsCompleted("left")
    Right, EmptyRight = WhatsCompleted("right")
    print("Up: | Has uncompleted squares : ", Up, " | Has completed squares : ", EmptyUp)
    print("Down: | Has uncompleted squares : ", Down, " | Has completed squares : ",
EmptyDown)
    print("Left: | Has uncompleted squares : ", Left, " | Has completed squares : ", EmptyLeft)
    print("Right: | Has uncompleted squares : ", Right, " | Has completed squares : ",
EmptyRight)

    allDirections = [(Up, EmptyUp, "up"),
(Down,EmptyDown,"down"),(Left,EmptyLeft,"left"),(Right,EmptyRight,"right")]

    filteredDirections = []

    for direction in allDirections:
        print(direction)
        if (direction[0] == 0 and direction[1] == 1) or (direction[0] == 0 and direction[1] == 0) or
direction[2] == comingFrom:
            pass
        else:
            filteredDirections.append([direction[0] + direction[1], direction[2]])

    print(filteredDirections)

    return filteredDirections

def WhatsCompleted(direction):
    numberOfSquares = 0
    hasCompletedRooms = 0
    x1 = 0
    x2 = 0
    y1 = 0
    y2 = 0
    if direction == "up":
        x1 = 808+150
        y1 = 638
        x2 = 1354-150
        y2 = 751
    elif direction == "down":
        x1 = 811+150
        y1 = 764
        x2 = 1357-150
        y2 = 914
    elif direction == "left":
        x1 = 808
        y1 = 638+50
```

```
        x2 = 1058
        y2 = 917-50
    elif direction == "right":
        x1 = 1070
        y1 = 638+50
        x2 = 1350
        y2 = 914-50

    rooms = ["BattleRoom", "TreasureRoom", "TreasureRoomMovingTo", "UnknownRoom",
"EmptyRoom", "BattleRoomMovingTo", "UnknownRoomMovingTo", "EmptyRoomMovingTo"]
    completedRooms = ["CompletedRoom", "CompletedRoomMovingTo", "StartingRoom",
"StartingRoomMovingTo"]

    for room in rooms:
        if numberOfSquares == 0:
            numberOfSquares = specificSquares(room, x1,y1,x2,y2)

    for completedRoom in completedRooms:
        if hasCompletedRooms == 0:
            hasCompletedRooms = specificSquares(completedRoom,x1,y1,x2,y2)


    return numberOfSquares, hasCompletedRooms

def specificSquares(whichOne,x1,y1,x2,y2):

    numOfPics = 0
    image = "./imagesForTheProgram/MapStuff/MapStuff_" + whichOne + ".png"
    results = imagesearch(image, 0.7)

    if results[0] != -1 and results[0] <= x2 and results[0] >= x1 and results[1] <= y2 and
results[1] >= y1: #and results[0] > 485 and results[1] > 667 and results[0] < 1178 and
results[1] < 860:
        numOfPics = 1
    return numOfPics

def currentPartyLocation():
    global p

    currentLocation = ""

    partyLocation = Functor("partyLocation", 1)

    A = Variable()
    q = Query(partyLocation(A))
    while q.nextSolution():
        currentLocation = A.value
    q.closeQuery()

    print("Party is in:", currentLocation)
    print(currentLocation)

    return currentLocation

###
```

```
###Combat stuff

def findOutCurrentCharacter():
    global currentCharacter
    findCurrentCharacter()
    print(currentCharacter)

###

def checkEverything():
    print("Everything available: ")

    currentCharactersAndSkills()
    currentPartyLocation()
    currentInventory()

def initialSetupForDungeon():
    print("Initial dungeon setup")

    print("Checking inventory")
    whatsInTheInventory()

    print("Checking heroes")
    whoAreTheHeros()

    print("Begins in a room")
    placeIntoRoom()

    checkEverything()

def isTheObjectiveReached():
    global dungeonCompleted

    result = imagesearch("./imagesForTheProgram/Utility/Utility_Return_To_Hamlet.png", 0.9)
    if result[0] == -1:
        objectiveReached = False
    else:
        objectiveReached = True

    print("Objective reached: ", objectiveReached)

def isThePartyInCombat():

    moveMouse(945, 431)
    partyIsInCombat = False

    result =
imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Enemy_Targetting.png",
0.7)

    if str(result) == "[-1, -1]":
        partyIsInCombat = False
    else:
        partyIsInCombat = True
```

```
        time.sleep(4)

    print("Party is in combat: ", partyIsInCombat)
    print()

    return partyIsInCombat

def lookForCharacterMarker():

    results = [-1,-1]

    while results[0] == -1:
        results =
imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Current_Character.png",
0.8)

def removeAllEnemies():
    global p
    p.retractall("currentEnemies(_,_,_,_,_,_)")

def makeACombatMove():
    global currentCharacter
    global transformed

    partyIsInCombat = True

    while partyIsInCombat == True:

        lookForCharacterMarker()

        listOfThingsToTry = findOutWhatToDo()

        useAbilities(listOfThingsToTry)

    clickOnReOrderParty()
    transformed = False
    AbominationHumanForm()
    wasInFight = True

def findOutWhatToDo():
    print("Figuring out what to do")
    global currentCharacter
    whatToDo = []
    lookForCharacterMarker()
    time.sleep(1)

    character = findCurrentCharacterFromExisting()
    time.sleep(2)
    #findCharacter

    position = findOutPosition()
    positionToSend = position[0]

    skillsAvailableRn = whatSkillsCanTheyUse(character, positionToSend)
    print(skillsAvailableRn)
```

```
    checkCharacter = findCurrentCharacterFromExisting()

    if checkCharacter == character:
        whatToDo = settingUpPriorities(skillsAvailableRn)

    print(whatToDo)

    print("Finished")

    return whatToDo

def settingUpPriorities(skills):
    abilityNameAndTargetCoordinates = []
    findTargets()
    for skill in skills:
        decidedThingy = decideWhatToDoWithSkill(skill)
        for target in decidedThingy:
            abilityNameAndTargetCoordinates.append(target)
    return abilityNameAndTargetCoordinates

def decideWhatToDoWithSkill(skill):
    global abilityCounter
    decidedThingy = []

    if skill[1] == "heal":
        decidedThingy.append(whoToHeal(skill))
    elif skill[1] == "selfheal":
        decidedThingy.append(healSelf(skill))
    elif skill[1] == "destress":
        decidedThingy.append(whoToDestress(skill))
    elif skill[1] == "selfdestress":
        decidedThingy.append(destressSelf(skill))
    elif skill[1] == "buff" and abilityCounter == 3:
        abilityCounter = 0
        list = whoToBuff(skill)
        for triable in list:
            if triable[0] != "":
                decidedThingy.append(triable)
    elif skill[1] == "stun" and abilityCounter == 2:
        abilityCounter = abilityCounter + 1
        list = whoToStun(skill)
        for triable in list:
            if triable[0] != "":
                decidedThingy.append(triable)
    elif skill[1] == "blight" or skill[1] == "bleed":
        list = whoToDOT(skill)
        for triable in list:
            if triable[0] != "":
                decidedThingy.append(triable)
    elif skill[1] == "debuff":
        list = whoToDebuff(skill)
        for triable in list:
            if triable[0] != "":
                decidedThingy.append(triable)
```

```
            abilityCounter = abilityCounter + 1
        elif skill[1] == "attack":
            list = whoToAttack(skill)
            for triable in list:
                if triable[0] != "":
                    decidedThingy.append(triable)
        else:
            list = whoToAttack(skill)
            for triable in list:
                if triable[0] != "":
                        decidedThingy.append(triable)

        return decidedThingy

    def whoToAttack(skill):
        returnMe = ["", [0,0]]
        usableFromPosition = Functor("usableFromPosition", 5)
        currentEnemies = Functor("currentEnemies", 6)

        LowFrom = Variable()
        HighFrom = Variable()
        LowTarget = Variable()
        HighTarget = Variable()
        PositionOfEnemy = Variable()
        Xcoordinate = Variable()
        Ycoordinate = Variable()
        Stress = Variable()
        Prot = Variable()
        Corpse = Variable()

        potentialVictims = []

        q = Query(usableFromPosition(skill[0], LowFrom, HighFrom, LowTarget, HighTarget),
                  currentEnemies(PositionOfEnemy, Xcoordinate, Ycoordinate, Prot, Stress, Corpse))
        while q.nextSolution():
            ToWhereLow = int(LowTarget.get_value())
            ToWhereHigh = int(HighTarget.get_value())
            position = int(PositionOfEnemy.get_value())
            if ToWhereLow <= position and ToWhereHigh >= position:
                potentialVictims.append([int(PositionOfEnemy.value), int(Xcoordinate.value),
    int(Ycoordinate.value)])
        q.closeQuery()
        list = []
        for victim in potentialVictims:
            list.append([skill[0], [victim[1], victim[2]]])
        print(list)
        return list

    def whoToDebuff(skill):
        returnMe = ["", [0,0]]
        usableFromPosition = Functor("usableFromPosition", 5)
        currentEnemies = Functor("currentEnemies", 6)

        LowFrom = Variable()
        HighFrom = Variable()
```

```
    LowTarget = Variable()
    HighTarget = Variable()
    PositionOfEnemy = Variable()
    Xcoordinate = Variable()
    Ycoordinate = Variable()
    Stress = Variable()
    Prot = Variable()

    potentialVictims = []

    q = Query(usableFromPosition(skill[0], LowFrom, HighFrom, LowTarget, HighTarget),
            currentEnemies(PositionOfEnemy, Xcoordinate, Ycoordinate, Prot, Stress,
"notCorpse"))
    while q.nextSolution():
        ToWhereLow = int(LowTarget.get_value())
        ToWhereHigh = int(HighTarget.get_value())
        position = int(PositionOfEnemy.get_value())
        if ToWhereLow <= position and ToWhereHigh >= position:
            potentialVictims.append([int(PositionOfEnemy.value), int(Xcoordinate.value),
int(Ycoordinate.value)])
    q.closeQuery()
    if len(potentialVictims) != 0:
        returnMe = [skill[0], [potentialVictims[0][1], potentialVictims[0][2]]]
    list = []
    for victim in potentialVictims:
        list.append([skill[0], [victim[1], victim[2]]])
    print(list)
    return list

def whoToDOT(skill):
    returnMe = ["", [0,0]]
    usableFromPosition = Functor("usableFromPosition", 5)
    currentEnemies = Functor("currentEnemies", 6)

    LowFrom = Variable()
    HighFrom = Variable()
    LowTarget = Variable()
    HighTarget = Variable()
    PositionOfEnemy = Variable()
    Xcoordinate = Variable()
    Ycoordinate = Variable()
    Stress = Variable()

    potentialVictims = []

    q = Query(usableFromPosition(skill[0], LowFrom, HighFrom, LowTarget, HighTarget),
            currentEnemies(PositionOfEnemy, Xcoordinate, Ycoordinate, "prot", Stress,
"notCorpse"))
    while q.nextSolution():
        ToWhereLow = int(LowTarget.get_value())
        ToWhereHigh = int(HighTarget.get_value())
        position = int(PositionOfEnemy.get_value())
        if ToWhereLow <= position and ToWhereHigh >= position:
            potentialVictims.append([int(PositionOfEnemy.value), int(Xcoordinate.value),
int(Ycoordinate.value)])
```

```
    q.closeQuery()
    if len(potentialVictims) != 0:
        returnMe = [skill[0], [potentialVictims[0][1], potentialVictims[0][2]]]
    list = []
    for victim in potentialVictims:
        list.append([skill[0], [victim[1], victim[2]]])
    print(list)
    return list

def whoToStun(skill):
    returnMe = ["", [0,0]]
    usableFromPosition = Functor("usableFromPosition", 5)
    currentEnemies = Functor("currentEnemies", 6)

    LowFrom = Variable()
    HighFrom = Variable()
    LowTarget = Variable()
    HighTarget = Variable()
    PositionOfEnemy = Variable()
    Xcoordinate = Variable()
    Ycoordinate = Variable()
    Prot = Variable()

    potentialVictims = []

    q = Query(usableFromPosition(skill[0], LowFrom, HighFrom, LowTarget, HighTarget),
            currentEnemies(PositionOfEnemy, Xcoordinate, Ycoordinate, Prot, "stress",
"notCorpse"))
    while q.nextSolution():
        ToWhereLow = int(LowTarget.get_value())
        ToWhereHigh = int(HighTarget.get_value())
        position = int(PositionOfEnemy.get_value())
        if ToWhereLow <= position and ToWhereHigh >= position:
            potentialVictims.append([int(PositionOfEnemy.value), int(Xcoordinate.value),
int(Ycoordinate.value)])
    q.closeQuery()
    if len(potentialVictims) != 0:
        returnMe = [skill[0], [potentialVictims[0][1], potentialVictims[0][2]]]
    list = []
    for victim in potentialVictims:
        list.append([skill[0], [victim[1], victim[2]]])

    print(list)
    return list


def whoToBuff(skill):
    global transformed
    returnMe = ["", [0,0]]

    if skill[0] == "tracking_Shot":
        returnMe = [skill[0], findOutPositionCoordinatesEnemy(2)]
        if returnMe[1] == []:
            retunrMe = ["", [0,0]]
    if skill[0] == "emboldening_Vapours":
```

```
        returnMe = [skill[0], XandYofFriendlyTarget(1)]
    else:
        if skill[0] == "transform":
            if transformed == False:
                AbominationBeastForm()
                transformed = True
            if transformed == True:
                AbominationHumanForm()
                transformed = False
        throw, send = findOutPosition()
        returnMe = [[skill[0], [send[0],send[1]]],[skill[0], XandYofFriendlyTarget(1)]]
    return returnMe

def AbominationHumanForm():
    global p
    p.retract("currentlyAvailableSkills(abomination, rage)")
    p.retract("currentlyAvailableSkills(abomination, rake)")
    p.retract("currentlyAvailableSkills(abomination, slam)")

    p.assertz("currentlyAvailableSkills(abomination, absolution)")
    p.assertz("currentlyAvailableSkills(abomination, manacles)")
    p.assertz("currentlyAvailableSkills(abomination, beasts_Bile)")

def AbominationBeastForm():
    global p
    p.retract("currentlyAvailableSkills(abomination, absolution)")
    p.retract("currentlyAvailableSkills(abomination, manacles)")
    p.retract("currentlyAvailableSkills(abomination, beasts_Bile)")

    p.assertz("currentlyAvailableSkills(abomination, rage)")
    p.assertz("currentlyAvailableSkills(abomination, rake)")
    p.assertz("currentlyAvailableSkills(abomination, slam)")

def destressSelf(skill):
    returnMe = ["", [0,0]]
    max, curr, stress = findCurrentCharactersHealthAndStress()
    if int(stress) > 80:
        throw, send = findOutPosition()
        returnMe = [skill[0], send]

    return returnMe

def whoToDestress(skill):
    returnMe = ["", [0,0]]
    biggestStress = 0
    everyone = checkEveryonesHealth()
    position = 0
    for guy in everyone:
        if int(guy[2]) >  biggestStress:
            biggestStress = int(guy[2])
            position = guy[0]
    if biggestStress > 80:
        returnMe = [skill[0], XandYofFriendlyTarget(position)]

    return returnMe
```

```
def healSelf(skill):
    returnMe = ["", [0,0]]
    max, curr, stress = findCurrentCharactersHealthAndStress()
    if int(curr)/int(max) < 0.5:

        throw, send = findOutPosition()
        returnMe = [skill[0], send]

    return returnMe

def XandYofFriendlyTarget(position):
    coordinates = [0,0]
    if position == 4:
        coordinates = [268, 431]
    elif position == 3:
        coordinates = [392, 431]
    elif position == 2:
        coordinates = [525, 431]
    elif position == 1:
        coordinates = [663, 431]

    return coordinates

def whoToHeal(skill):
    returnMe = ["", [0,0]]
    smallestRatio = float(1)
    everyone = checkEveryonesHealth()
    position = 0
    for guy in everyone:
        if float(guy[1]) < smallestRatio:
            smallestRatio = float(guy[1])
            position = int(guy[0])
    if skill[0] == "battlefield_Medicine":
        if smallestRatio < 0.1:
            returnMe = [skill[0], XandYofFriendlyTarget(position)]
    else:
        if smallestRatio < 0.5:
            returnMe = [skill[0], XandYofFriendlyTarget(position)]

    return returnMe

def whatSkillsCanTheyUse(character, position):
    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    characterAbility = Functor("characterAbility", 3)
    usableFromPosition = Functor("usableFromPosition", 5)
    currentlyAvailableSkills = Functor("currentlyAvailableSkills", 2)

    print(character)

    Ability = Variable()
    MaxHealth = Variable()
    CurrHealth = Variable()
    Stress = Variable()
    AbilityType = Variable()
```

```
    LowFrom = Variable()
    HighFrom = Variable()
    LowTarget = Variable()
    HighTarget = Variable()

    usableMoves = []

    q = Query(currentlyAvailableSkills(character, Ability),
            usableFromPosition(Ability, LowFrom, HighFrom, LowTarget, HighTarget),
            characterAbility(character, Ability, AbilityType))
    while q.nextSolution():
        fromWhereLow = int(LowFrom.get_value())
        fromWhereHigh = int (HighFrom.get_value())
        if fromWhereLow <= position and fromWhereHigh >= position:
            usableMoves.append([str(Ability.value), str(AbilityType.value)])
    q.closeQuery()

    return usableMoves

def findCurrentCharacterFromExisting():
    global allCharacters
    for dude in allCharacters:
        result =
imagesearch("./imagesForTheProgram/CharacterStuff/Portraits/"+dude[1]+"_pic.png", 0.8)
        if result[0] != -1:
            return dude[1]


def useAbilities(thingsToTry):
    global currentCharacter
    somethingHappened = False
    oldTarget = [0,0]
    character = findCurrentCharacterFromExisting()

    for thing in thingsToTry:
        if somethingHappened == False and thing[0] != "":
            if thing[0] != oldTarget:
                clickOnAbility(character, thing[0])
                clickOnTarget(thing[1])
                oldTarget = thing[0]
            time.sleep(0.3)
            lookForCharacterMarker()
            someGuy = findCurrentCharacterFromExisting()
            print(somethingHappened)
            if character != someGuy:
                somethingHappened = True
                print("Something did in fact happen")
                time.sleep(1)
        time.sleep(1)
    if somethingHappened == False:
        pressMove()

def countEnemiesAndStats(positions):
    enemies = []
    currentPositionOfSkull = [0,0]
```

```
    oldPositionOfSkull = 0
    number = 0
    canStress = ""
    hasProt = ""
    isCorpse = ""

    for position in positions:
        moveTo(position[1], position[2])
        currentPositionOfSkull =
imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Enemy_Targetting.png",
0.8)
        protection =
imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Prot.png", 0.8)
        stress = imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Stress.png",
0.8)
        corpse =
imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Corpse.png", 0.8)
        if protection[0] != -1 and protection[0] > 945:
            hasProt = "prot"
        else:
            hasProt = "no_prot"
        if stress[0] != -1:
            canStress = "stress"
        else:
            canStress = "no_stress"
        if corpse[0] == -1:
            isCorpse = "notCorpse"
        else:
            isCorpse = "isCorpse"
        if currentPositionOfSkull[0] != oldPositionOfSkull and currentPositionOfSkull[0] != -1:
            number = number + 1
            oldPositionOfSkull = currentPositionOfSkull[0]

            enemies.append([number, position[1], position[2], hasProt, canStress, isCorpse])
        else:
            pass

    return enemies

def findTargets():
    global p
    removeAllEnemies()
    positions = getAllEnemyPositions()
    numberOfEnemies = countEnemiesAndStats(positions)

    for enemy in numberOfEnemies:
        forAdding = "currentEnemies("+str(enemy[0])+", "+str(enemy[1])+", "+str(enemy[2])+",
"+enemy[3]+", "+enemy[4]+","+ enemy[5]+")"
        p.assertz(forAdding)

    showAllEnemies()

def showAllEnemies():
    currentEnemies = Functor("currentEnemies", 6)
```

```
    X = Variable()
    Y = Variable()
    Z = Variable()
    A = Variable()
    B = Variable()
    C = Variable()

    q = Query(currentEnemies(X,Y,Z,A,B,C))
    while q.nextSolution():
        print(str(X.value),str(Y.value),str(Z.value),str(A.value),str(B.value),str(C.value))
    q.closeQuery()

def getAllEnemyPositions():
    positions = []
    enemyPosition = Functor("enemyPosition", 3)

    X = Variable()
    Y = Variable()
    Z = Variable()

    q = Query(enemyPosition(X,Y,Z))
    while q.nextSolution():
        positions.append([int(X.value), int(Y.value), int(Z.value)])
    q.closeQuery()

    return positions

def pressMove():
    results = imagesearch("./imagesForTheProgram/Utility/Utility_Move.png", 0.8)
    clickMouse(results[0]+5,results[1]+5)

    position, coordinates = findOutPosition()

    if position == 1:
        coordinateeees = findOutPositionCoordinates(4)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(3)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(2)
        clickMouse(coordinateeees[0], coordinateeees[1])
    elif position == 2:
        coordinateeees = findOutPositionCoordinates(4)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(3)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(1)
        clickMouse(coordinateeees[0], coordinateeees[1])
    elif position == 3:
        coordinateeees = findOutPositionCoordinates(1)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(2)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(4)
        clickMouse(coordinateeees[0], coordinateeees[1])
    elif position == 3:
```

```
        coordinateeees = findOutPositionCoordinates(1)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(2)
        clickMouse(coordinateeees[0], coordinateeees[1])
        coordinateeees = findOutPositionCoordinates(3)
        clickMouse(coordinateeees[0], coordinateeees[1])


def findOutPositionCoordinatesEnemy(position):
    enemyPosition = Functor("enemyPosition", 3)

    X = Variable()
    Y = Variable()

    toReturn = []
    q = Query(enemyPosition(position, X,Y))
    while q.nextSolution():
        toReturn = [int(X.value), int(Y.value)]
    q.closeQuery

    return toReturn

def findOutPositionCoordinates(position):
    partyPosition = Functor("partyPosition", 3)

    X = Variable()
    Y = Variable()

    toReturn = []
    q = Query(partyPosition(position, X,Y))
    while q.nextSolution():
        toReturn = [int(X.value), int(Y.value)]
    q.closeQuery

    return toReturn

def checkEveryonesHealth():
    positions = [[262, 614],[385, 614],[515, 614],[641, 614]]
    bboxes = [[192, 540, 335, 596],[321, 540, 463, 596],[452,540,590,596],[581,540,723,596]]

    whoToHealPositions = []

    for i in range(0, 4):
        moveMouse(positions[i][0],positions[i][1])
        healthImage = pyscreenshot.grab(bboxes[i])
        healthImage.save("checkThis2.png")

        image = PIL.Image.open("checkThis2.png")
        health = pytesseract.image_to_string(image)
        try:
            maxHealth = health.split("/")[1].split("\n")[0]
        except Exception as ex:
            maxHealth = str(40)
        except ValueError as vex:
            maxHealth = str(40)
```

```
try:
    currHealth = health.split("/")[0]
except Exception as ex:
    currHealth = str(30)
except ValueError as vex:
    currHealth = str(30)

try:
    currStress = health.split("\n")[1].split("/")[0]
except Exception as ex:
    currStress = str(50)
except ValueError as vex:
    currStress = str(50)

try:
    maxHealth = int(maxHealth)
    maxHealth = str(maxHealth)
except Exception as ex:
    maxHealth = str(40)
except ValueError as vex:
    maxHealth = str(40)

try:
    currHealth = int(currHealth)
    currHealth = str(currHealth)
except Exception as ex:
    currHealth = str(30)
except ValueError as vex:
    currHealth = str(30)

try:
    currStress = int(currStress)
    currStress = str(currStress)
except Exception as ex:
    currStress = str(50)
except ValueError as vex:
    currStress = str(50)

ratio = 1
try:
    ratio = float(currHealth)/(float(maxHealth))
except Exception as ex:
    ratio = 0.5
except ValueError as vex:
    ratio = 0.5

stress = 30
try:
    stress = int(currStress)
except Exception as ex:
    stress = 50
except ValueError as vex:
    stress = 50
```

```
        whoToHealPositions.append([4-i, ratio, stress])

    return whoToHealPositions

def clickOnAbility(character, ability):
    results =
imagesearch("./imagesForTheProgram/CharacterStuff/"+character+"/"+character+"_"+ability
+".png", 0.8)
    clickMouse(results[0]+5,results[1]+5)

def clickOnTarget(target):
    clickMouse(target[0],target[1])

def combatCheckCharacter(position):
    searchForAilmentsOnOnePerson(position)

def findOutPosition():
    position =
imagesearch("./imagesForTheProgram/CombatStuff/CombatStuff_Current_Character.png")
    print(position)

    positionToSend = 0
    coordinates = [0,0]

    if position[0] < 268:
        positionToSend = 4
        coordinates = [268, 431]
    elif position[0] > 268 and position[0] < 392:
        positionToSend = 3
        coordinates = [392, 431]
    elif position[0] > 392 and position[0] < 525:
        positionToSend = 2
        coordinates = [525, 431]
    elif position[0] > 525 and position[0] < 663:
        positionToSend = 1
        coordinates = [663, 431]

    return positionToSend, coordinates

def isThePartyInRoom():

    results = imagesearch("./imagesForTheProgram/Utility/Utility_Fully_Zoomed_Frame.png",
0.8)
    if results[0] != -1 and results[0] > 1015 and results[1] > 712 and results[0] < 1041 and
results[1] < 732:
        print("Party in room")
        placeIntoRoom()
        return True
    else:
        print("Party not in room")
        return False

def didSomethingHappen():
    enteredCombat = isThePartyInCombat()
    if enteredCombat == False:
```

```
        selectMap()
        selectCenteringOnParty()
        enteredRoom = isThePartyInRoom()
        checkForHunger()
        checkForCurio()

    if enteredCombat == False and enteredRoom == False:
        return False
    else:
        return True

def makeAnExplorationMove():

    checkOnParty()
    pressW()
    checkForCurio()
    whereThePartyIs = currentPartyLocation()

    if str(whereThePartyIs) == "hallway":
        somethingHappened = False
        moveMouse(400,400)
        popUp = []
        fullyZoomMap()
        while somethingHappened == False:
            while len(popUp) == 0:
                holdDAndCheckForTrap()
                fullyZoomMap()
                popUp = imagesearch_from_folder("./imagesForTheProgram/PoppedUp/", 0.8)
            somethingHappened = didSomethingHappen()
            popUp = []

    elif str(whereThePartyIs) == "room":
        selectMap()
        selectCenteringOnParty()
        didWeMove = selectWhereToGo()
        if didWeMove ==False:
            pass
        else:
            checkForTrap()
            moveForABit()
            placeIntoHallway()
    else:
        print("Don't know where the party is")

def checkForTrap():
    clickMouse(1073, 557)
    time.sleep(1)

def checkForHunger():
    results = imagesearch("./imagesForTheProgram/Utility/Utility_Eat_Food.png", 0.8)
    results2 = imagesearch("./imagesForTheProgram/Utility/Utility_Dont_Eat_Food.png", 0.8)
    if results[0] != -1:
        clickMouse(results[0]+5,results[1]+5)
    elif results[0] == -1 and results2[0] != -1:
        clickMouse(results2[0]+5,results2[1]+5)
```

```
    else:
        print("No one is hungry")

def checkForCurio():

    resultOfHand = imagesearch("./imagesForTheProgram/Utility/Utility_Take_All.png", 0.8)
    if resultOfHand[0] == -1:
        return

    results = str(imagesearch_from_folder("./imagesForTheProgram/CurioStuff/",0.8))
    results = results[2:]
    results = results[:len(results)-1]
    listOfResults = results.split(", '")

    listOfResults = [i.replace("./imagesForTheProgram/CurioStuff/", "") for i in listOfResults]
    listOfResults = [i.replace(".png'", "") for i in listOfResults]
    time.sleep(0.1)
    print()

    filteredList = []

    for j in listOfResults:
        #print(j.split(": ")[1] == "[-1, -1]")
        #print(j.split(": ")[1])
        if j.split(": ")[1] != "[-1, -1]":
            filteredList += [j]
    itemNeeded = ""

    curio = ""

    if len(filteredList) != 0:
        curio = filteredList[0].split(":")[0]
        itemNeeded = checkIfYouHaveWhatYouNeedForCurio(curio)
        time.sleep(0.2)

    if itemNeeded != "":
        useItemOnCurio(itemNeeded)
        collectTreasure()
    else:
        if curio != "":
            isItDangerousWithoutAnything(curio)
            collectTreasure()


    results = imagesearch("./imagesForTheProgram/Utility/Utility_Take_All.png", 0.8)
    clickMouse(results[0]+5,results[1]+5)


def isItDangerousWithoutAnything(curio):
    result = checkIfCurioPositiveWithoutAnything(curio)
    if result != "negative":
        collectTreasure()
    else:
        results = imagesearch("./imagesForTheProgram/Utility/Utility_Close.png", 0.8)
        time.sleep(0.5)
```

```
        if results[0] == -1:
            pass
        else:
            clickMouse(results[0],results[1])

def collectTreasure():
    time.sleep(1)
    ############
    pressHand()

    time.sleep(0.3)
    nothingLeftToKick = False
    allSorted = False

    print("Trinyg to collect")

    results = imagesearch("./imagesForTheProgram/Utility/Utility_Ok.png", 0.8)
    if results[0] == -1:
        allSorted = True

    while allSorted == False and nothingLeftToKick == False:
        pressHand()
        time.sleep(1)
        results = imagesearch("./imagesForTheProgram/Utility/Utility_Ok.png", 0.8)
        if results[0] == -1:
            allSorted = True
        else:
            clickMouse(results[0],results[1])
            whatsInTheInventory()
            nothingLeftToKick = kickLeastValuableThing()

    results = imagesearch("./imagesForTheProgram/Utility/Utility_Close.png", 0.8)
    if results[0] == -1:
        pass
    else:
        clickMouse(results[0],results[1])
        whatsInTheInventory()
        moveForABit


def kickLeastValuableThing():
    somethingKicked = False

    valuableItem = Functor("valuableItem", 2)
    heirloom = Functor("heirloom", 2)
    playerHas = Functor("playerHas", 1)
    valueAndHowManyStacks = Functor("valueAndHowManyStacks", 2)

    genre = ""
    bestToKick = ""
    worthOfMostKickable = 10000

    X = Variable()
    Y = Variable()
    Z = Variable()
```

```
    q = Query(playerHas(X), valuableItem(X, valueAndHowManyStacks(Y, Z)))
    while q.nextSolution():
        if int(Y.value)*int(Z.value) < worthOfMostKickable:
            worthOfMostKickable = int(Y.value)*int(Z.value)
            bestToKick = str(X.value)
            genre = "valuable"
    q.closeQuery()

    if bestToKick == "":
        worthOfMostKickable = 5
        q = Query(playerHas(X), heirloom(X,Y))
        while q.nextSolution():
            if int(Y.value) < worthOfMostKickable:
                worthOfMostKickable = int(Y.value)
                bestToKick = str(X.value)
                genre = "heirloom"
        q.closeQuery()

    if bestToKick == "":
        q = Query(playerHas(X), item(X))
        while q.nextSolution():
            bestToKick = str(X.value)
            genre = "item"
        q.closeQuery()

    print(bestToKick)

    if bestToKick != "":
        if genre == "valuable":
            results =
imagesearch("./imagesForTheProgram/ValuableStuff/ValuableStuff_"+bestToKick+".png")
            shiftClick(results[0],results[1])
        elif genre == "heirloom":
            results =
imagesearch("./imagesForTheProgram/HeirloomStuff/HeirloomStuff_"+bestToKick+".png")
            shiftClick(results[0],results[1])
        elif genre == "item":
            results =
imagesearch("./imagesForTheProgram/HeirloomStuff/HeirloomStuff_"+bestToKick+".png")
            shiftClick(results[0],results[1])
        somethingKicked = True
        removeFromInventory(bestToKick)

    if somethingKicked == True:
        return False
    else:
        return True

def pressHand():
    results = imagesearch("./imagesForTheProgram/Utility/Utility_Take_All.png", 0.8)
    clickMouse(results[0]+180, results[1]+5)
    clickMouse(results[0]+5, results[1]+5)

def useItemOnCurio(item):
```

```
    results =
imagesearch("./imagesForTheProgram/InventoryStuff/InventoryStuff_"+item+".png")
    if results[0] != -1:
        rightClickMouse(results[0]+5, results[1]+5)
    checkFor(item)

def checkIfYouHaveWhatYouNeedForCurio(investigatedCurio):
    curio = Functor("curio", 1)
    curioAndWhatWorksOnIt = Functor("curioAndWhatWorksOnIt", 3)
    playerHas = Functor("playerHas",1)

    X = Variable()

    itemNeeded = ""
    q = Query(curio(investigatedCurio),
curioAndWhatWorksOnIt(investigatedCurio,X,"positive"), playerHas(X))

    while q.nextSolution():
        itemNeeded = str(X.value)
    q.closeQuery()

    return itemNeeded

def checkIfCurioPositiveWithoutAnything(investigatedCurio):
    curio = Functor("curio", 1)
    curioAndWhatWorksOnIt = Functor("curioAndWhatWorksOnIt", 3)

    X = Variable()
    Y = Variable()

    result = ""
    q = Query(curio(investigatedCurio),
curioAndWhatWorksOnIt(investigatedCurio,"nothing",X))

    while q.nextSolution():
        result = str(X.value)
    q.closeQuery()

    print(result)
    return result

def makeAMove():
    global dungeonCompleted

    partyIsInCombat = isThePartyInCombat()

    if partyIsInCombat:

        makeACombatMove()

    else:

        makeAnExplorationMove()

    isTheObjectiveReached()
```

```
def completingTheCurrentDungeon():
    global dungeonCompleted

    initialSetupForDungeon()

    while dungeonCompleted == False:
        makeAMove()

    print("Current dungeon is complete")

def main():
    print("Darkest Dungeon AI, please tab into the game")

    prepareProlog()

    global keepDoingIt
    currentlyInDungeon = True

    typeYIfYouWantToContinue = "y"
    while keepDoingIt == True:
        firstCountdown()

        completingTheCurrentDungeon()

        shouldIContinue()
    goodbyeMessage()

p = Prolog()

abilityCounter= 0
transformed = False
recentlyLooked = True
comingFrom = ""
currentCharacter = ""
currentCharacterAbilities = []
allCharacters = []

wasInFight = False
keepDoingIt = True
dungeonCompleted = False

main()
```

Prilog 2

Primjer Prolog koda

Napomene:

- Sve vezano uz Prolog je napisano kroz Python te se baza znanja definira prilikom pokretanja programa
- varijabla "p" je instanca Prolog() objekta iz PySwip knjižice koja je deklarirana na globalnoj razini

Definiranje činjenica:

```
def defineAbilities():
    global p
    characterAbility = Functor("characterAbility", 3)
    p.assertz("characterAbility(abomination, absolution, selfheal)")
    p.assertz("characterAbility(abomination, absolution, selfdestress)")
    p.assertz("characterAbility(abomination, manacles, stun)")
    p.assertz("characterAbility(abomination, manacles, attack)")
    p.assertz("characterAbility(abomination, beasts_Bile, blight)")
    p.assertz("characterAbility(abomination, beasts_Bile, attack)")
    p.assertz("characterAbility(abomination, transform, buff)")
    p.assertz("characterAbility(abomination, rage, attack)")
    p.assertz("characterAbility(abomination, rake, attack)")
    p.assertz("characterAbility(abomination, slam, movement)")
    p.assertz("characterAbility(abomination, slam, attack)")

    p.assertz("characterAbility(antiquarian, fortifying_Vapours, heal)")
    p.assertz("characterAbility(antiquarian, invigorating_Vapours, buff)")
    p.assertz("characterAbility(antiquarian, nervous_Stab, attack)")
    p.assertz("characterAbility(antiquarian, festering_Vapours, blight)")
    p.assertz("characterAbility(antiquarian, get_Down, buff)")
    p.assertz("characterAbility(antiquarian, get_Down, movement)")
    p.assertz("characterAbility(antiquarian, flashpowder, debuff)")
    p.assertz("characterAbility(antiquarian, flashpowder, reveal)")
    p.assertz("characterAbility(antiquarian, protect_Me, buff)")
```

```
p.assertz("characterAbility(arbalest, battlefield_Bandage, heal)")
p.assertz("characterAbility(arbalest, sniper_Shot, markable)")
p.assertz("characterAbility(arbalest, sniper_Shot, attack)")
p.assertz("characterAbility(arbalest, snipers_Mark, mark)")
p.assertz("characterAbility(arbalest, suppressing_Fire, attack)")
p.assertz("characterAbility(arbalest, rallying_Flare, buff)")
p.assertz("characterAbility(arbalest, rallying_Flare, reveal)")
p.assertz("characterAbility(arbalest, bola, movement)")
p.assertz("characterAbility(arbalest, bola, attack)")
p.assertz("characterAbility(arbalest, blindfire, attack)")

p.assertz("characterAbility(bountyHunter, collect_Bounty, markable)")
p.assertz("characterAbility(bountyHunter, collect_Bounty, attack)")
p.assertz("characterAbility(bountyHunter, mark_For_Death, mark)")
p.assertz("characterAbility(bountyHunter, mark_For_Death, debuff)")
p.assertz("characterAbility(bountyHunter, come_Hither, mark)")
p.assertz("characterAbility(bountyHunter, come_Hither, movement)")
p.assertz("characterAbility(bountyHunter, uppercut, stun)")
p.assertz("characterAbility(bountyHunter, uppercut, movement)")
p.assertz("characterAbility(bountyHunter, flashbang, stun)")
p.assertz("characterAbility(bountyHunter, flashbang, movement)")
p.assertz("characterAbility(bountyHunter, finish_Him, attack)")
p.assertz("characterAbility(bountyHunter, caltrops, bleed)")

p.assertz("characterAbility(crusader, battle_Heal, heal)")
p.assertz("characterAbility(crusader, inspiring_Cry, heal)")
p.assertz("characterAbility(crusader, inspiring_Cry, destress)")
p.assertz("characterAbility(crusader, smite, attack)")
p.assertz("characterAbility(crusader, zealous_Accusation, attack)")
p.assertz("characterAbility(crusader, stunning_Blow, stun)")
p.assertz("characterAbility(crusader, stunning_Blow, attack)")
p.assertz("characterAbility(crusader, holy_Lance, movement)")
p.assertz("characterAbility(crusader, holy_Lance, attack)")
p.assertz("characterAbility(crusader, bulwark_Of_Faith, buff)")

p.assertz("characterAbility(flagellant, punish, attack)")
```

```
p.assertz("characterAbility(flagellant, punish, bleed)")
p.assertz("characterAbility(flagellant, rain_Of_Sorrows, attack)")
p.assertz("characterAbility(flagellant, rain_Of_Sorrows, bleed)")
p.assertz("characterAbility(flagellant, exsanguinate, selfheal)")
p.assertz("characterAbility(flagellant, exsanguinate, attack)")
p.assertz("characterAbility(flagellant, exsanguinate, bleed)")
p.assertz("characterAbility(flagellant, reclaim, heal)")
p.assertz("characterAbility(flagellant, redeem, heal)")
p.assertz("characterAbility(flagellant, endure, destress)")
p.assertz("characterAbility(flagellant, suffer, heal)")
p.assertz("characterAbility(flagellant, suffer, cleanse)")

p.assertz("characterAbility(graverobber, lunge, attack)")
p.assertz("characterAbility(graverobber, lunge, movement)")
p.assertz("characterAbility(graverobber, thrown_Dagger, attack)")
p.assertz("characterAbility(graverobber, thrown_Dagger, markable)")
p.assertz("characterAbility(graverobber, toxin_Trickery, buff)")
p.assertz("characterAbility(graverobber, flashing_Daggers, attack)")
p.assertz("characterAbility(graverobber, flashing_Daggers, debuff)")
p.assertz("characterAbility(graverobber, pick_To_The_Face, attack)")
p.assertz("characterAbility(graverobber, pick_To_The_Face, piercing)")
p.assertz("characterAbility(graverobber, shadow_Fade, movement)")
p.assertz("characterAbility(graverobber, shadow_Fade, buff)")
p.assertz("characterAbility(graverobber, poison_Darts, blight)")
p.assertz("characterAbility(graverobber, poison_Darts, debuff)")

p.assertz("characterAbility(hellion, wicked_Hack, attack)")
p.assertz("characterAbility(hellion, iron_Swan, attack)")
p.assertz("characterAbility(hellion, adrenaline_Rush, selfheal)")
p.assertz("characterAbility(hellion, adrenaline_Rush, buff)")
p.assertz("characterAbility(hellion, barbaric_Yawp, stun)")
p.assertz("characterAbility(hellion, if_It_Bleeds, bleed)")
p.assertz("characterAbility(hellion, if_It_Bleeds, attack)")
p.assertz("characterAbility(hellion, breakthrough, attack)")
p.assertz("characterAbility(hellion, breakthrough, movement)")
p.assertz("characterAbility(hellion, bleed_Out, attack)")
p.assertz("characterAbility(hellion, bleed_Out, bleed)")
```

```
p.assertz("characterAbility(highwayman, point_Blank_Shot, attack)")
p.assertz("characterAbility(highwayman, point_Blank_Shot, movement)")
p.assertz("characterAbility(highwayman, tracking_Shot, attack)")
p.assertz("characterAbility(highwayman, tracking_Shot, buff)")
p.assertz("characterAbility(highwayman, tracking_Shot, reveal)")
p.assertz("characterAbility(highwayman, pistol_Shot, attack)")
p.assertz("characterAbility(highwayman, pistol_Shot, markable)")
p.assertz("characterAbility(highwayman, wicked_Slice, attack)")
p.assertz("characterAbility(highwayman, grapeshot_Blast, attack)")
p.assertz("characterAbility(highwayman, grapeshot_Blast, debuff)")
p.assertz("characterAbility(highwayman, duelists_Advance, attack)")
p.assertz("characterAbility(highwayman, duelists_Advance, movement)")
p.assertz("characterAbility(highwayman, open_Vein, attack)")
p.assertz("characterAbility(highwayman, open_Vein, bleed)")

p.assertz("characterAbility(houndmaster, lick_Wound, selfheal)")
p.assertz("characterAbility(houndmaster, blackjack, stun)")
p.assertz("characterAbility(houndmaster, blackjack, stun)")
p.assertz("characterAbility(houndmaster, hounds_Rush, attack)")
p.assertz("characterAbility(houndmaster, hounds_Rush, bleed)")
p.assertz("characterAbility(houndmaster, hounds_Rush, markable)")
p.assertz("characterAbility(houndmaster, hounds_Harry, attack)")
p.assertz("characterAbility(houndmaster, hounds_Harry, bleed)")
p.assertz("characterAbility(houndmaster, target_Whistle, mark)")
p.assertz("characterAbility(houndmaster, target_Whistle, debuff)")
p.assertz("characterAbility(houndmaster, guard_Dog, buff)")
p.assertz("characterAbility(houndmaster, cry_Havoc, destress)")

p.assertz("characterAbility(jester, inspiring_Tune, destress)")
p.assertz("characterAbility(jester, battle_Ballad, buff)")
p.assertz("characterAbility(jester, dirk_Stab, attack)")
p.assertz("characterAbility(jester, dirk_Stab, movement)")
p.assertz("characterAbility(jester, harvest, bleed)")
p.assertz("characterAbility(jester, harvest, attack)")
p.assertz("characterAbility(jester, slice_Off, bleed)")
p.assertz("characterAbility(jester, slice_Off, attack)")
```

```
p.assertz("characterAbility(jester, finale, attack)")
p.assertz("characterAbility(jester, finale, movement)")
p.assertz("characterAbility(jester, solo, movement)")
p.assertz("characterAbility(jester, solo, buff)")

p.assertz("characterAbility(leper, solemnity, selfheal)")
p.assertz("characterAbility(leper, solemnity, selfdestress)")
p.assertz("characterAbility(leper, chop, attack)")
p.assertz("characterAbility(leper, hew, attack)")
p.assertz("characterAbility(leper, purge, movement)")
p.assertz("characterAbility(leper, purge, attack)")
p.assertz("characterAbility(leper, revenge, buff)")
p.assertz("characterAbility(leper, withstand, buff)")
p.assertz("characterAbility(leper, intimidate, attack)")
p.assertz("characterAbility(leper, intimidate, reveal)")

p.assertz("characterAbility(manAtArms, rampart, stun)")
p.assertz("characterAbility(manAtArms, rampart, attack)")
p.assertz("characterAbility(manAtArms, rampart, movement)")
p.assertz("characterAbility(manAtArms, command, buff)")
p.assertz("characterAbility(manAtArms, bolster, buff)")
p.assertz("characterAbility(manAtArms, crush, attack)")
p.assertz("characterAbility(manAtArms, bellow, debuff)")
p.assertz("characterAbility(manAtArms, defender, buff)")
p.assertz("characterAbility(manAtArms, retribution, buff)")
p.assertz("characterAbility(manAtArms, retribution, attack)")

p.assertz("characterAbility(musketeer, patch_Up, heal)")
p.assertz("characterAbility(musketeer, aimed_Shot, markable)")
p.assertz("characterAbility(musketeer, aimed_Shot, attack)")
p.assertz("characterAbility(musketeer, call_The_Shot, mark)")
p.assertz("characterAbility(musketeer, smokescreen, debuff)")
p.assertz("characterAbility(musketeer, smokescreen, attack)")
p.assertz("characterAbility(musketeer, buckshot, movement)")
p.assertz("characterAbility(musketeer, buckshot, attack)")
p.assertz("characterAbility(musketeer, sidearm, attack)")
p.assertz("characterAbility(musketeer, skeet_Shot, buff)")
```

```
p.assertz("characterAbility(musketeer, skeet_Shot, reveal)")

p.assertz("characterAbility(occultist, wyrd_Reconstruction, heal)")
p.assertz("characterAbility(occultist, weakening_Curse, debuff)")
p.assertz("characterAbility(occultist, vulnerability_Hex, mark)")
p.assertz("characterAbility(occultist, abyssal_Artillery, attack)")
p.assertz("characterAbility(occultist, daemons_Pull, attack)")
p.assertz("characterAbility(occultist, daemons_Pull, movement)")
p.assertz("characterAbility(occultist, hands_From_The_Abyss, stun)")
p.assertz("characterAbility(occultist, hands_From_The_Abyss, attack)")
p.assertz("characterAbility(occultist, sacrificial_Stab, attack)")

p.assertz("characterAbility(plagueDoctor, noxious_Blast, attack)")
p.assertz("characterAbility(plagueDoctor, noxious_Blast, blight)")
p.assertz("characterAbility(plagueDoctor, disorienting_Blast, stun)")
p.assertz("characterAbility(plagueDoctor, disorienting_Blast, movement)")
p.assertz("characterAbility(plagueDoctor, plague_Grenade, attack)")
p.assertz("characterAbility(plagueDoctor, plague_Grenade, blight)")
p.assertz("characterAbility(plagueDoctor, blinding_Gas, stun)")
p.assertz("characterAbility(plagueDoctor, incision, attack)")
p.assertz("characterAbility(plagueDoctor, incision, bleed)")
p.assertz("characterAbility(plagueDoctor, battlefield_Medicine, heal)")
p.assertz("characterAbility(plagueDoctor, emboldening_Vapours, buff)")

p.assertz("characterAbility(shieldbreaker, pierce, attack)")
p.assertz("characterAbility(shieldbreaker, pierce, piercing)")
p.assertz("characterAbility(shieldbreaker, pierce, movement)")
p.assertz("characterAbility(shieldbreaker, puncture, attack)")
p.assertz("characterAbility(shieldbreaker, puncture, movement)")
p.assertz("characterAbility(shieldbreaker, adders_Kiss, blight)")
p.assertz("characterAbility(shieldbreaker, impale, blight)")
p.assertz("characterAbility(shieldbreaker, impale, movement)")
p.assertz("characterAbility(shieldbreaker, expose, reveal)")
p.assertz("characterAbility(shieldbreaker, expose, attack)")
p.assertz("characterAbility(shieldbreaker, captivate, attack)")
p.assertz("characterAbility(shieldbreaker, captivate, blight)")
p.assertz("characterAbility(shieldbreaker, captivate, markable)")
```

```
    p.assertz("characterAbility(shieldbreaker, serpent_Sway, movement)")

    p.assertz("characterAbility(vestal, divine_Grace, heal)")
    p.assertz("characterAbility(vestal, divine_Comfort, heal)")
    p.assertz("characterAbility(vestal, dazzling_Light, stun)")
    p.assertz("characterAbility(vestal, dazzling_Light, attack)")
    p.assertz("characterAbility(vestal, mace_Bash, attack)")
    p.assertz("characterAbility(vestal, judgement, attack)")
    p.assertz("characterAbility(vestal, illumination, reveal)")
    p.assertz("characterAbility(vestal, illumination, debuff)")
    p.assertz("characterAbility(vestal, hand_Of_Light, attack)")
    p.assertz("characterAbility(vestal, hand_Of_Light, buff)")

    print("Abilities defined")

##############

def defineClasses():
    global p
    characterClass = Functor("characterClass", 1)
    p.assertz("characterClass(abomination)")
    p.assertz("characterClass(antiquarian)")
    p.assertz("characterClass(arbalest)")
    p.assertz("characterClass(bountyHunter)")
    p.assertz("characterClass(crusader)")
    p.assertz("characterClass(flagellant)")
    p.assertz("characterClass(graverobber)")
    p.assertz("characterClass(hellion)")
    p.assertz("characterClass(highwayman)")
    p.assertz("characterClass(houndmaster)")
    p.assertz("characterClass(jester)")
    p.assertz("characterClass(leper)")
    p.assertz("characterClass(manAtArms)")
    p.assertz("characterClass(musketeer)")
    p.assertz("characterClass(occultist)")
    p.assertz("characterClass(plagueDoctor)")
    p.assertz("characterClass(shieldbreaker)")
```

```
    p.assertz("characterClass(vestal)")

    print("Classes defined")
```

Povlačenje činjenica:

```
def removeCurrentCharacters():
    global p
    p.retractall("currentlyAvailableCharacters(_,_,_,_,_)")

############

def removeCurrentSkills():
    global p
    p.retractall("currentlyAvailableSkills(_,_)")

############

def AbominationBeastForm():
    global p
    p.retract("currentlyAvailableSkills(abomination, absolution)")
    p.retract("currentlyAvailableSkills(abomination, manacles)")
    p.retract("currentlyAvailableSkills(abomination, beasts_Bile)")

    p.assertz("currentlyAvailableSkills(abomination, rage)")
    p.assertz("currentlyAvailableSkills(abomination, rake)")
    p.assertz("currentlyAvailableSkills(abomination, slam)")
```

Postavljanje upita:

```
def whatAmIPreparedToHeal():
    stuffToCure = []

    cures = Functor("cures", 2)
    playerHas = Functor("playerHas", 1)
```

```
    X = Variable()
    Y = Variable()

    q = Query(cures(X,Y), playerHas(X))
    while q.nextSolution():
        stuffToCure.append(str(Y.value))
    q.closeQuery()

    print(stuffToCure)

    return stuffToCure

################

def whatSkillsCanTheyUse(character, position):
    currentlyAvailableCharacters = Functor("currentlyAvailableCharacters", 5)
    characterAbility = Functor("characterAbility", 3)
    usableFromPosition = Functor("usableFromPosition", 5)
    currentlyAvailableSkills = Functor("currentlyAvailableSkills", 2)

    print(character)

    Ability = Variable()
    MaxHealth = Variable()
    CurrHealth = Variable()
    Stress = Variable()
    AbilityType = Variable()
    LowFrom = Variable()
    HighFrom = Variable()
    LowTarget = Variable()
    HighTarget = Variable()

    usableMoves = []

    q = Query(currentlyAvailableSkills(character, Ability),
            usableFromPosition(Ability, LowFrom, HighFrom, LowTarget, HighTarget),
```

```
        characterAbility(character, Ability, AbilityType))
while q.nextSolution():
    fromWhereLow = int(LowFrom.get_value())
    fromWhereHigh = int (HighFrom.get_value())
    if fromWhereLow <= position and fromWhereHigh >= position:
        usableMoves.append([str(Ability.value), str(AbilityType.value)])
q.closeQuery()

return usableMoves
```