Python Mode for Processing: alat za razvoj digitalnih vještina u obrazovanju

Jonjić, Ivanka

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:410331

Rights / Prava: Attribution-NonCommercial 4.0 International/Imenovanje-Nekomercijalno 4.0 međunarodna

Download date / Datum preuzimanja: **2025-02-05**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivanka Jonjić

PYTHON MODE FOR PROCESSING: ALAT ZA RAZVOJ DIGITALNIH VJEŠTINA U OBRAZOVANJU

Diplomski rad

Voditelj rada:
dr.sc. Goran Igaly

Zagreb, 2024.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Svoj diplomski rad posvećujem svojoj majci Mariji. Zbog nje sam sve što imam i što jesam. Na poseban način hvala mom ocu koji mi je uvijek govorio da nema onog što ja ne mogu i potpunosti podržao svaku moju odluku. Hvala mom djedu koji je moj vječni navijač kroz dobro i zlo. Hvala i mom bratu na neprestanoj podršci. Također, hvala mojim dugogodišnjim prijateljicama koji su ovo dugo i izazovno poglavlje mog života prošli uz mene i učinile ga ljepšim i boljim. Hvala mojim mentorima prof. Brođancu i prof. Ćulav Markičević na neizmjernoj podršci i savjetima, kako za vrijeme prakse, tako i dan danas. Hvala dr. sc. Goranu Igalyju što mi je primjerom pokazao kako biti profesor koji se istinski trudi i voli ono što radi.

Sadržaj

Sadržaj **iv**

Uvod **1**

1 Uvod u Processing **3**
1.1 Instalacija 3
1.2 Izrada prvog programa 4
1.3 Funkcije setup() i draw() 6

2 Statičke aktivnosti **9**
2.1 Aktivnosti: crtanje oblika 9
2.2 Varijable i for petlja 18

3 Dinamičke aktivnosti **25**
3.1 While petlja i uvjetne naredbe 25
3.2 Primjeri aktivnosti 26

4 FMS Logo ili Python Mode for Processing **37**
4.1 FMS Logo 37
4.2 Primjeri aktivnosti 39
4.3 Prijelaz s FMS Loga na Python Mode for Processing 46

Bibliografija **51**

Uvod

Processing je programski jezik, tj. razvojno okruženje koje promovira softversku pismenost u sklopu vizualnih umjetnosti i vizualnu pismenost u sklopu tehnologije. Danas postoje deseci tisuća studenata, umjetnika, dizajnera i istraživača koji koriste Processing za učenje, modeliranje i proizvodnju.

Processing je prvotno predstavljen koristeći sintaksu baziranu na programskom jeziku Java sa skupom funkcija za grafiku i crtanje jednostavnih geometrijskih oblika koji su bili inspirirani OpenGLom, Postscriptom, Design by Numbersom i drugim izvorima. Postupnim dodavanjem alternativnih programskih sučelja uključujući JavaScript i Python postalo je sve jasnije da Processing nije samo jedan jezik, već umjetnički orijentiran pristup učenju, podučavanju i razvoju algoritama za rješavanje problema.

Python Mode for Processing prvenstveno je razvio Jonathan Feinberg, uz doprinose Jamesa Gillesa i Bena Alkova. Primjeri, reference i upute za Python način su preneseni i/ili kreirani od strane Jamesa Gillesa i ostatka njegovog tima.[4]

S obzirom na to da ne postoji velik broj materijala za Python Mode u Processingu, u ovom diplomskom radu posvetit će se upravo tome. Naglasak će biti na kreativnom i vizualnom uvođenju osnovnih koncepata programiranja.

Cilj ovog diplomskog rada je pružiti nastavnicima informatike još jedan način za ostvarivanje ciljeva iz domene Računalno razmišljanje i programiranje. Jedna od prednosti Processinga koju je bitno istaknuti je jednostavna instalacija. Naime, to je jako važno za nastavnike, ali i za učenike, kako bi im se olakšao proces učenja tako što mogu i sami na svojim prijenosnim računalima obaviti instalaciju i utvrditi naučeno.

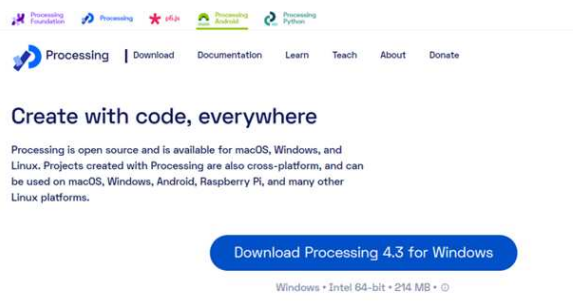
Poglavlje 1

Uvod u Processing

1.1 Instalacija

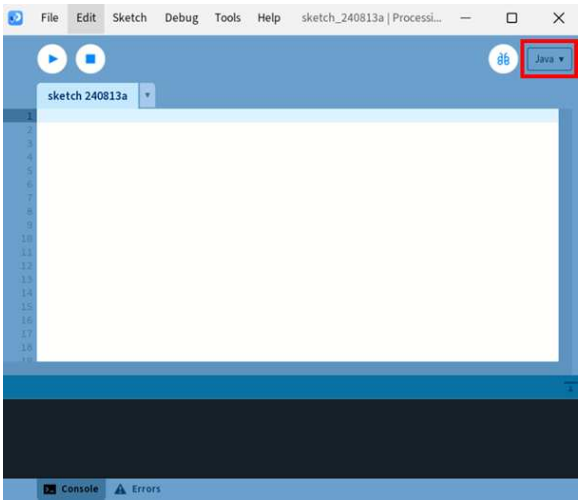
U ovom dijelu opisat ćemo ukratko instalaciju Processinga i to Python Modea. Postoji više izvora koje možemo koristiti, no u ovom radu ćemo koristiti upute sa službene web stranice koja se može pronaći na poveznici: https://py.processing.org/tutorials/gettingstarted/.

Instalacija je omogućena i objašnjena za operacijske sustave Mac, Linux i Windows. Za sva tri operacijska sustava početni korak je isti, treba posjetiti web stranicu: http://processing.org/download. Za operacijski sustav Windows idući korak je kliknuti na gumb koji će preuzeti .zip datoteku. Tu datoteku treba raspakirati na željeno mjesto na računalu. Processing pokrećemo dvostrukim klikom na datoteku processing.exe.



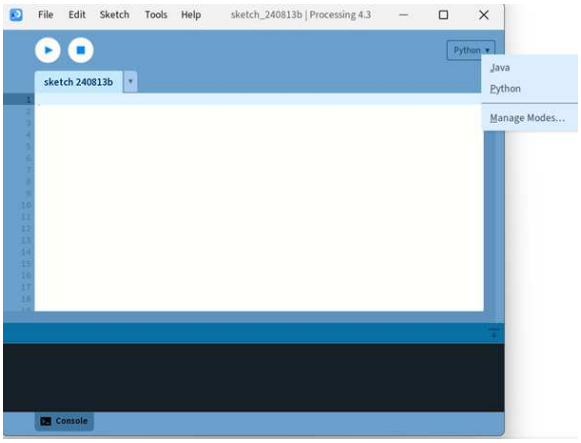
Slika 1.1: Instalacija za operacijski sustav Windows

Nakon pokretanja prikaže nam se zadani izgled aplikacije koji je postavljen u načinu za programski jezik Java.



Slika 1.2: Zadani izgled aplikacije

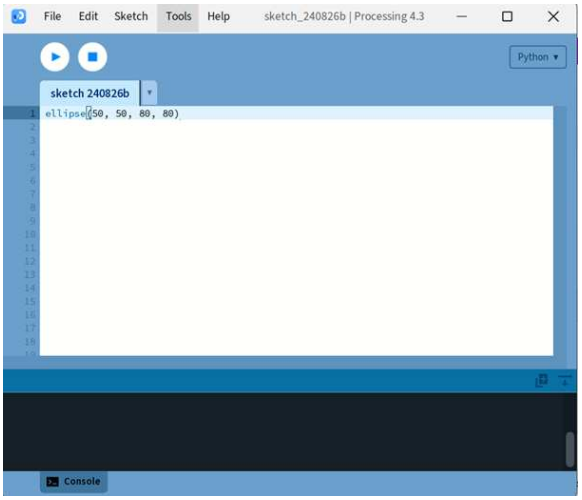
Klikom na padajući izbornik možemo odabrati programski jezik Python za željeni način.



Slika 1.3: Python Mode for Processing

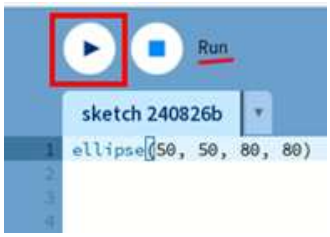
1.2 Izrada prvog programa

Kod prvog susreta s nepoznatim programskim jezikom uobičajena je praksa izrada jednostavnog programa. S obzirom na to da je riječ o programskom jeziku kojem je naglasak na vizualizaciji, za početni primjer može se npr. nacrtati elipsa. U uređivač teksta upiše se naredba: ellipse(50, 50, 80, 50).[2]



Slika 1.4: Prvi program

Idući korak je pritisak gumba koji je namijenjen za pokretanje (eng. *Run*) programa.



Slika 1.5: Gumb za pokretanje programa

Kao rezultat će se pojaviti prozor za crtanje (eng. *Display window*) u kojem bi trebala biti nacrtana elipsa, ako je naredba ispravno unesena.

Slika 1.6: Elipsa

1.3 Funkcije setup() i draw()

Dvije glavne funkcije čiji opis treba upoznati su setup() i draw(). Tek nakon upoznavanja s njima učenici mogu dalje pristupiti rješavanju raznoraznih primjera s razumijevanjem.

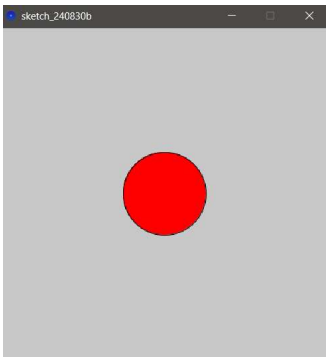
setup()

Funkcija setup() poziva se jednom prilikom pokretanja programa. Koristi se za definiranje početnih svojstava okruženja kao što su veličina zaslona koristeću naredbu i boja pozadine (background()) te za učitavanje medija kao što su slike i fontovi prilikom pokretanja programa. Može postojati samo jedna funkcija setup() za svaki program i ne bi se trebala ponovno pozivati nakon početnog izvođenja. Bitno je napomenuti da varijablama deklariranim unutar setup() funkcije nije moguće pristupiti unutar drugih funkcija, uključujući draw().[11]

Primjer jednostavnog programa koji koristi funkciju setup().

```
def setup():
    size(400, 400)  # Postavi veličinu prozora na 400x400 piksela
    background(200)  # Postavi boju pozadine na svijetlosivu
    fill(255, 0, 0)  # Postavi boju ispune na crvenu
    ellipse(200, 200, 100, 100)  # Nacrtaj crveni krug u sredini
```

Slika 1.7: setup() funkcija - primjer programa



Slika 1.8: Rješenje primjera koji koristi setup() funkciju

draw()

Nakon što je setup() pozvan, funkcija draw() se više puta poziva dok se program ne zaustavi ili dok se ne pozove naredba noLoop(). Funkcija draw() se poziva automatski i nikada se ne smije pozivati eksplicitno. Poziv bi se trebao kontrolirati s noLoop(), redraw() i loop(). Ako se noLoop() koristi za zaustavljanje izvršavanja koda u draw(), tada će redraw() uzrokovati jednokratno izvršavanje koda unutar draw(), a loop() će uzrokovati kontinuirano izvršavanje ostatka koda unutar draw() . Može postojati samo jedna funkcija draw() za svaki program, a ona mora postojati ako želimo da se kod kontinuirano izvodi ili da obrađuje događaje kao što je mousePressed() koji se koristi kada želimo da se neke naredbe izvrše kada je pritisnuta tipka na mišu.

Uobičajeno je pozvati background() blizu početka draw() funkcije za brisanje sadržaja prozora. Budući da se pikseli nacrtani u prozoru tu i zadržavaju, izostavljanje background() može rezultirati neželjenim rezultatima, posebno kada se crtaju oblici ili tekst sa zaglađenim rubovima.[13]

Poglavlje 2

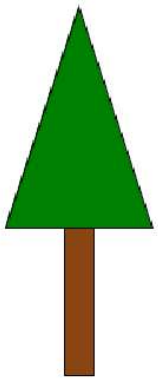
Statičke aktivnosti

U ovom poglavlju proučit ćemo nekoliko statičkih aktivnosti izrađenih u Python Modeu for Processing. Aktivnosti su navedene tako da su poredane po težini (od lakših prema težima), a u svakoj su opisane naredbe koje se koriste. Također, navedene su i najčešći problemi koje učenici mogu imati tijekom rješavanja i savjeti za nastavnika.

2.1 Aktivnosti: crtanje oblika

Aktivnost 1: Stablo

Tekst uz aktivnosti: Nacrtaj stablo kao na slici.

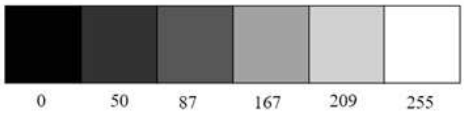


Slika 2.1: Stablo

Ova aktivnost može se iskoristiti kao početni primjer za uvođenje crtanja oblika u Python Modeu for Processing. Aktivnost je primjerena za učenike od petog razreda osnovne škole nadalje. Naime, učenici su tada već sposobni razumjeti i iskoristiti jednostavne naredbe i metodom pokušaja i pogreške doći do rješenja.
Očekivani ishod iz informatike[8] :
B.5.2 stvara algoritam za rješavanje jednostavnoga zadatka, provjerava ispravnost algoritma, otkriva i popravlja pogreške.
Potrebno je uvesti naredbe:

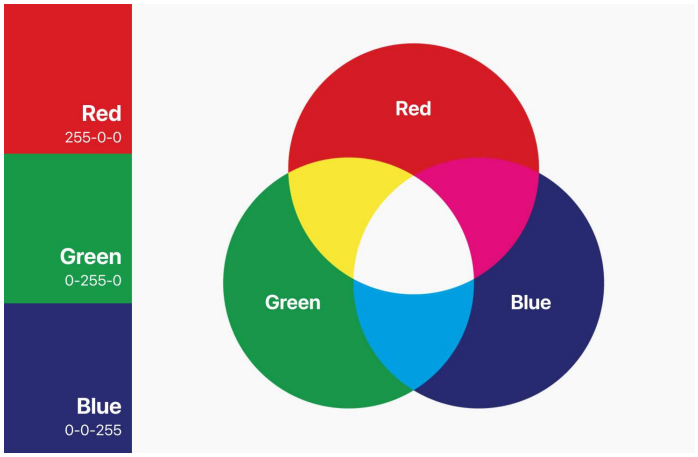
- **size(širina, visina)** - širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.

- **background(nijansa)** - postavlja boju pozadine. Nijansa može biti bilo koji cijeli broj od 0 = crno do 255 = bijelo, svaki drugi broj: 50, 87, 162, 209 i tako dalje, je nijansa sive u rasponu od crne do bijele.[12].



Slika 2.2: Nijansa pozadine ovisno o vrijednosti varijable nijansa[12]

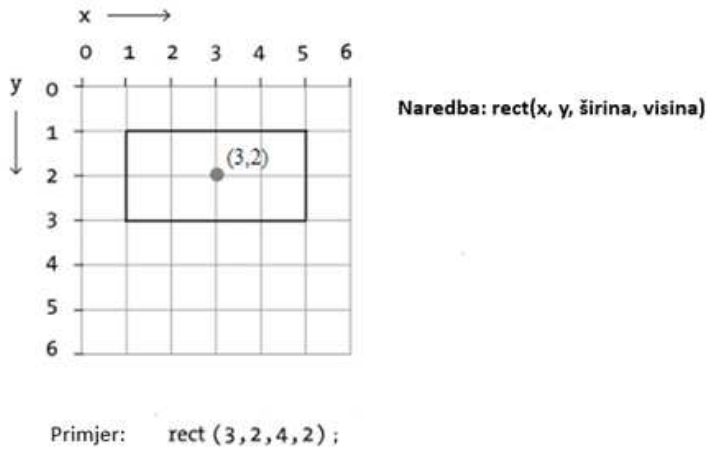
Bitno je napomenuti da naredba može imati i oblik **background(r, g, b)** Zadani skup boja je RGB (Red Green Blue), svaki parametar (r, g i b) poprima vrijednosti u rasponu 0 - 255.



Slika 2.3: RGB[18]

- **fill(r, g, b)** - Postavlja boju ispune za oblike koji će se crtati. Argumenti su vrijednosti crvene, zelene i plave komponente boje, što omogućuje specificiranje nijanse boje.

- Naredba **rect(x, y, širina, visina)** koristi se za crtanje pravokutnika na željenom mjestu na prozoru za prikaz.[9]



Slika 2.4: Objašnjenje naredbe za crtanje pravokutnika[3]

- **triangle(x1, y1, x2, y2, x3, y3)** - Crta trokut na temelju koordinata njegovih triju vrhova. Svaki par (xi, yi) za $i \in 1, 2, 3$ određuje jednu od tri točke koje definiraju trokut.

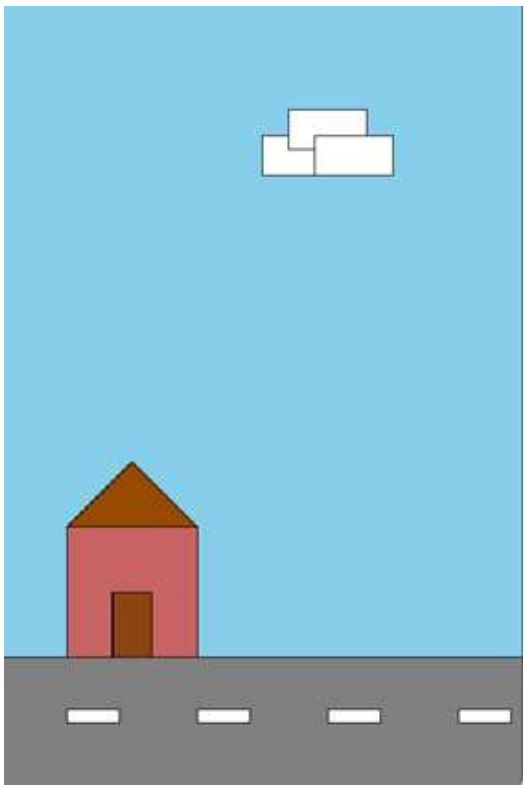
```
def setup():
    size(400, 600)  # Postavlja veličinu prozora na širinu 400 i visinu 600

def draw():
    background(255)  # Postavlja pozadinu na bijelu boju (255)

    # Deblo drveta
    fill(139, 69, 19)  # Postavlja boju za ispunu debla na smeđu
    rect(190, 400, 20, 100)  # Crta pravokutnik za deblo na poziciji (190, 400) s širinom 20 i visinom 100

    # Grane drveta
    fill(0, 128, 0)  # Postavlja boju za ispunu grana na zelenu
    triangle(150, 400, 200, 250, 250, 400)  # Crta trokut za grane na zadanim koordinatama (150, 400), (200, 250), (250, 400)
```

Slika 2.5: Kôd - aktivnost 1

Najčešći problemi: Učenicima je najveći izazov u ovom zadatku razumjeti kako odrediti koordinate vrhova geometrijskih oblika. Bitno je učenicima objasniti da se u Processingu za sustav koordinata namješten tako da je točka (0, 0) u gornjem lijevom kutu prozora. Kako se pomičemo dolje, vrijednosti za y koordinate rastu, a kako se pomičemo prema gore, vrijednosti y koordinate smanjuju. To znači da su manje y vrijednosti bliže vrhu prozora, a veće y vrijednosti bliže dnu prozora. Vrijednost x koordinata se povećavaju kako se pomičemo desno, a smanjuju kada se pomičemo ulijevo. Ovdje je dobra strategija crtanje geometrijskih oblika na ploči ili u GeoGebri.[1] Ovdje je preporučeno da nastavnik ne ulazi u strogo matematičke izračune koordinata, dovoljno je zaključiti koje koordinate koristiti na osnovu skice.

Aktivnost 1* - Kuća na cesti

Za učenike kojima ova aktivnost nije dovoljno izazovna, nastavnik može ponuditi zahtjevniju verziju ovog zadatka. Učenici i dalje koriste iste naredbe, no zadatak je vidno kompleksniji i može dati darovitim učenicima veći izazov, a nastavniku priliku da se posveti ostatku učenika.
Tekst uz aktivnost: Nacrtaj kuću na cesti kao na slici.



Slika 2.6: Aktivnost 1*

```
def setup():
    size(400, 600)  # Postavlja veličinu prozora na širinu 400 i visinu 600

def draw():
    background(135, 206, 235)  # Postavlja pozadinu na svijetloplavu boju (boja neba)

    # Cesta
    fill(128, 128, 128)  # Boja za cestu (siva)
    rect(0, 500, 400, 100)  # Crta cestu na dnu prozora

    # Bijele crte na cesti
    fill(255)  # Boja za crte (bijela)
    rect(50, 540, 40, 10)   # Prva crta
    rect(150, 540, 40, 10)  # Druga crta
    rect(250, 540, 40, 10)  # Treća crta
    rect(350, 540, 40, 10)  # Četvrta crta

    # Kuća na rubu ceste
    fill(200, 100, 100)  # Boja kuće (svijetlo crvena)
    rect(50, 400, 100, 100)  # Tijelo kuće (postavljeno iznad ceste, na rubu)
    fill(150, 75, 0)  # Boja krova (smeđa)
    triangle(50, 400, 100, 350, 150, 400)  # Krov kuće

    # Vrata kuće
    fill(139, 69, 19)  # Boja vrata (tamno smeđa)
    rect(85, 450, 30, 50)  # Crta vrata kuće

    # Oblak
    fill(255)  # Boja oblaka (bijela)
    rect(200, 100, 60, 30)  # Prvi dio oblaka
    rect(220, 80, 60, 30)   # Drugi dio oblaka
    rect(240, 100, 60, 30)  # Treći dio oblaka
```

Slika 2.7: Kôd-izazovnija verzija aktivnosti 1

Aktivnost 2: Sladoled

Tekst uz aktivnost: Nacrtaj sladoled kao na slici.

Slika 2.8: Aktivnost 2

Aktivnost može poslužiti za uvođenje naredbe za crtanje elipse. Od učenika se očekuje da su već upoznati s funkcijama **setup()** i **draw()** te s naredbom za veličinu prozora za crtanje i ispunu oblika. Aktivnost je primjerena za učenike od petog razreda osnovne škole nadalje. Naime, učenici su u toj dobi zasigurno upoznati s pojmom trokuta i koordinata točaka. Naravno, učenici nisu upoznati s definicijom elipse, ali ona nije ni potrebna za uspješno razumijevanje aktivnosti. Naime, učenici su sposobni iz primjera razumjeti sve što im je potrebno za crtanje elipse, također sposobni su metodom pokušaja i pogreške zaključiti kada je elipsa kružnica.
Očekivani ishod iz informatike[8]:

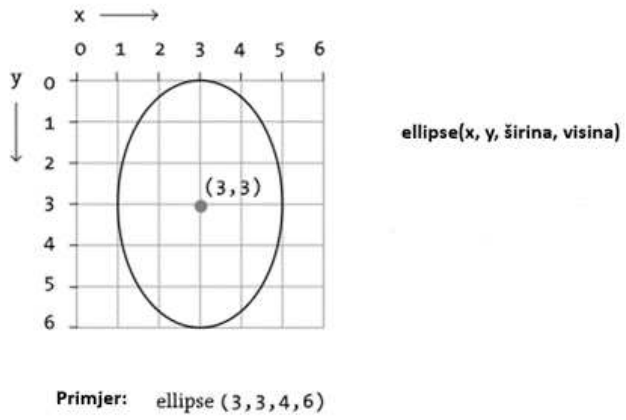
B.5.2 stvara algoritam za rješavanje jednostavnoga zadatka, provjerava ispravnost algoritma, otkriva i popravlja pogreške.
Potrebne naredbe:

- **size(širina, visina)** - širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.

- **background(boja)** postavlja pozadinsku boju prozora. Argument predstavlja boju u sivim tonovima (0-255) ili u RGB formatu (ako su navedene tri vrijednosti).

- **fill(r, g, b)** - Postavlja boju ispune za oblike koji će se crtati. Argumenti su vrijednosti crvene, zelene i plave komponente boje, što omogućuje specificiranje nijanse boje.

- **triangle(x1, y1, x2, y2, x3, y3)** crta trokut na temelju koordinata njegovih triju vrhova. Svaki par (xi, yi) za $i \in \{1, 2, 3\}$ određuje jednu od tri točke koje definiraju trokut.
- **ellipse(x, y, širina, visina)** crta elipsu na temelju zadanih koordinata njenog središta i dimenzija. Prvi par (x, y) predstavlja središte elipse, dok drugi par predstavlja širinu i visinu elipse.[22]



Slika 2.9: Primjer naredbe za crtanje elipse[3]

```
def setup():
    size(400, 600)  # Postavlja veličinu prozora na širinu 400 i visinu 600

def draw():
    background(255)  # Postavlja pozadinu na bijelu boju (255)

    # Kornet
    fill(210, 180, 140)  # Postavlja boju korneta na svijetlosmeđu
    triangle(200, 550, 160, 350, 240, 350)  # Crta kornet usmjeren prema dolje

    # Kuglica sladoleda (prva, vanilija)
    fill(255, 228, 196)  # Postavlja boju kuglice sladoleda na boju vanilije
    ellipse(175, 300, 100, 100)  # Crta prvu kuglicu sladoleda na poziciji (175, 300)

    # Kuglica sladoleda (druga, jagoda)
    fill(255, 105, 180)  # Postavlja boju kuglice sladoleda na ružičastu (jagoda)
    ellipse(225, 300, 100, 100)  # Crta drugu kuglicu sladoleda na poziciji (225, 300)

    # Kuglica sladoleda (treća, čokolada)
    fill(139, 69, 19)  # Postavlja boju kuglice sladoleda na smeđu (čokolada)
    ellipse(200, 250, 100, 100)  # Crta treću kuglicu sladoleda na poziciji (200, 250)
```

Slika 2.10: Kôd - aktivnost 2

Problem koji se može javiti je određivanje koordinata središta elipsa, odnosno kružnica i koordinata vrhova trokuta. Preporuka nastavniku je skica rješenja i dolazak do rješenja

metodom dijaloga i aktivnom raspravom s učenicima.
Izazovnija verzija aktivnosti koja su može provesti s učenicima nakon uvođenja naredbe za crtanje kružnice (elipse) navedena je u nastavku.

Aktivnost 2* - hrpa kuglica

Tekst uz aktivnost: Nacrtaj hrpu kuglica kao na slici.

Slika 2.11: Hrpa kuglica

```
def setup():
    size(400, 400)
    background(255)
    noStroke()

def draw():
    background(255)
    fill(0, 150, 200)  # Boja kružnica
    r = 50  # Polumjer kružnice (veći)
    razmak = r + 10  # Razmak između kružnica (povećano za veće kružnice)

    # Prvi red (1 kružnica)
    x1 = 200
    y1 = 100
    ellipse(x1, y1, r, r)

    # Drugi red (2 kružnice)
    x2_1 = 200 - razmak / 2
    x2_2 = 200 + razmak / 2
    y2 = 100 + razmak
    ellipse(x2_1, y2, r, r)
    ellipse(x2_2, y2, r, r)

    # Treći red (3 kružnice)
    x3_1 = 200 - razmak
    x3_2 = 200
    x3_3 = 200 + razmak
    y3 = 100 + 2 * razmak
    ellipse(x3_1, y3, r, r)
    ellipse(x3_2, y3, r, r)
    ellipse(x3_3, y3, r, r)
```

Slika 2.12: Kôd - aktivnost 2*

2.2 Varijable i for petlja

Varijabla u Pythonu može se definirati kao spremnik koji pohranjuje vrijednosti. Varijabli se ne dodjeljuje tip prije upotrebe. Naime, varijabla se kreira onog trenutka kada joj prvi put dodijelimo vrijednost. Python varijabla je zapravo naziv koji se daje memorijskoj lokaciji i ona je osnovna jedinica za pohranu u programu. Primjer varijable predstavlja naziv koji služi kao pokazivač na objekt. Nakon što je objekt dodijeljen varijabli, na njega se može pozivati tim imenom.[6] Postoje dvije vrste varijabli, s obzirom na njihov doseg unutar programa, a to su **globalne** i **lokalne** varijable. **Globalne varijable** su one koje nisu definirane unutar funkcije i imaju globalni doseg, dok su **lokalne varijable** one koje su definirane unutar funkcije i njihov je doseg ograničen samo na tu funkciju. Drugim riječima, možemo reći da su lokalne varijable dostupne samo unutar funkcije u kojoj su inicijalizirane, dok su globalne varijable dostupne u cijelom programu i unutar svake funkcije.
For petlja radi tako što pokreće naredbe unutar svog opsega sve dok navedeni uvjet više nije istinit. To omogućuje izvođenje zadataka kao što je prolazak kroz elemente neke ko-

lekcije podataka dok se ne dosegne kraj ili izvođenje neke radnje određeni broj puta.[15] Za potrebe crtanja u Python Modeu for Processing koristi se funkcija **range**. Funkcija **range** vraća niz brojeva, počevši od 0 prema zadanim postavkama i povećavajući se za 1 (prema zadanim postavkama) i zaustavlja se prije određenog broja. Najjednostavniji oblik **for** petlje prikazan je na slici ispod.

```
for i in range(n):
    naredbe
    ...
```

Slika 2.13: Najjednostavniji oblik **for** petlje

Riječima se može opisati: Neka je n je prirodan broj. Sve dok je varijabla i u rasponu od 0 do $n-1$, izvrši naredbe te nakon izvršavanja naredbi povećaj varijablu i za 1. Varijablu i nazivamo **kontrolna varijabla**.
Oblik **for** petlje u kojem funkcija **range** ima tri argumenta prikazan je na slici ispod.

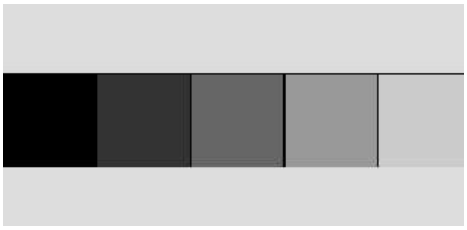
```
for i in range(m,n,k):
    naredbe
    ...
```

Slika 2.14: For petlja u kojoj range ima tri argumenta

Riječima: Neka su m, n i k prirodni brojevi. Sve dok je varijabla i u rasponu od m do $n-1$, izvrši naredbe te nakon izvršavanja povećaj vrijednost varijable i za k. Varijablu k nazivamo **korak**.

Aktivnost 3: Boje

Tekst aktivnosti: Nacrtaj niz kvadrata tako da se nijansa sive boje mijenja od crne do svijetlo sive.



Slika 2.15: Aktivnost 3

Potrebne naredbe:

- **size(širina, visina)** širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.

- **fill(r, g, b)** postavlja boju ispune za oblike koji će se crtati. Argumenti su vrijednosti crvene, zelene i plave komponente boje, što omogućuje specificiranje nijanse boje.

- **rect(x, y, širina, visina)** crta pravokutnik na ekranu s gornjim lijevim kutom na koordinatama (x, y) i dimenzijama definiranim širinom i visinom.

Aktivnost je primjerena za učenike od šestog razreda osnovne škole pa nadalje. Može poslužiti za uvođenje pojma for petlje (definicije i sintakse). Od učenika se očekuje da razumiju pojam varijable, da su upoznati s RGB sustavom za prikaz boja na zaslon i da znaju nacrtati kvadrat određenih dimenzija prije obrade ove aktivnosti. Ona može poslužiti kao jedna zanimljiva poveznica mogućnosti for petlje i boja ispune likova. Jedan od problema koji se može pojaviti je kako dobiti nijanse sive boje. Rješenje problema se može pretvoriti u jednu odvojenu aktivnost za koju može poslužiti RGB kalkulator koji se može pronaći na web stranici W3schools.[23]
Očekivani ishodi iz informatike[8]:

- **B.6.1.** Učenik stvara, prati i preuređuje programe koji sadrže strukture grananja i uvjetnoga ponavljanja te predviđa ponašanje jednostavnih algoritama koji mogu biti prikazani dijagramom, riječima govornoga jezika ili programskim jezikom.

- **B.6.2** Nakon šeste godine učenja predmeta Informatika u domeni Računalno razmišljanje i programiranje učenik razmatra i rješava složeniji problem rastavljajući ga na niz potproblema.

```
def setup():
    size(255, 51)  # Postavi veličinu prozora na 255x51 piksela
    y = 0  # Inicijalizacija y koordinata za pravokutnike
    for x in range(0, width, 51):  # Petlja koja prolazi kroz širinu prozora u koracima od 51
        r = x  # Postavi crvenu komponentu boje na vrijednost x
        g = x  # Postavi zelenu komponentu boje na vrijednost x
        b = x  # Postavi plavu komponentu boje na vrijednost x
        fill(r, g, b)  # Postavi boju ispune pravokutnika
        rect(x, y, 51, 51)  # Nacrtaj pravokutnik na poziciji (x, y) s veličinom 51x51 piksela
```

Slika 2.16: Aktivnost 3: kôd

Kao nadogradnja ove aktivnosti može se iskoristiti sljedeća aktivnost koja služi za uvođenje funkcije **random**.

Aktivnost 4: Šarena traka

Tekst uz aktivnost: Nacrtaj traku u kojoj dijelovi od koje je sastavljena (kvadrati) svakim pokretanjem mijenjaju boju.

Slika 2.17: Aktivnost 4

Aktivnost je primjerena za učenike najmanje prvog razreda srednje škole gimnazijskog programa. Od učenika se očekuje da su upoznati s pojmom varijable i **for** petlje te njezinom sintaksom, da su upoznati s RGB sustavom za prikaz boja na zaslon te da znaju nacrtati kvadrat određenih dimenzija.
Očekivani ishodi iz informatike:[8]

- **B.1.2** Nakon prve godine učenja predmeta Informatika u srednjoj školi u domeni Računalno razmišljanje i programiranje učenik primjenjuje jednostavne tipove podataka te argumentira njihov odabir, primjenjuje različite vrste izraza, operacija, relacija i standardnih funkcija za modeliranje jednostavnoga problema u odabranome programskom jeziku.

- **B.1.3** Nakon prve godine učenja predmeta Informatika u srednjoj školi u domeni Računalno razmišljanje i programiranje učenik razvija algoritam i stvara program u odabranome programskom jeziku rješavajući problem uporabom strukture grananja i ponavljanja.

Potrebne naredbe:

- **size(širina, visina)** gdje su širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.
- **int(a)** pretvara vrijednost unutar zagrade (a) u cijeli broj (*integer*). Na primjer, int(3.7) bi se pretvorilo u 3.
- **random(a, b)** vraća nasumični decimalni broj (*floating-point number*) između vrijednosti *a* (uključivo) i *b* (isključivo).
- **fill(r, g, b)** postavlja boju ispune za oblike koji će se crtati. Argumenti su vrijednosti crvene, zelene i plave komponente boje, što omogućuje specificiranje nijanse boje.
- **rect(x, y, širina, visina)** crta pravokutnik na ekranu s gornjim lijevim kutom na koordinatama (x, y) i dimenzijama definiranim **širinom** i **visinom**.

```
def setup():
    size(300, 30)

    y = 0  # Postavljamo y koordinatu na 0 jer želimo jedan redak
    for x in range(0, width, 30):
        r = int(random(0, 255))
        g = int(random(0, 255))
        b = int(random(0, 255))
        fill(r, g, b)
        rect(x, y, 30, 30)
```

Slika 2.18: Aktivnost 4: kôd

Nadovezivanjem na prethodnu nastavnik može uvesti pojam ugniježđene petlje i to pomoću iduće aktivnosti.

Aktivnost 4* - Šarena mreža[21]

Tekst uz aktivnost: Nacrtaj šarenu mrežu u kojoj se kvadratima svakim pokretanjem mijenja boja ispune.

Slika 2.19: Aktivnost 4

```
def setup():
    size(300, 300)

    for y in range(0, height, 30):
        for x in range(0, width, 30):
            r = int(random(0, 255))
            g = int(random(0, 255))
            b = int(random(0, 255))
            fill(r, g, b)
            rect(x, y, 30, 30)
```

Slika 2.20: Aktivnost 4 kôd

Dobro je s učenicima komentirati zašto se u ovoj aktivnosti ne koristi funkcija draw(). Bitno je doći do zaključka da se funkcija draw() izvodi beskonačno puta dok je u slučaju ove aktivnosti potrebno samo nacrtati mrežu i onda je aktivnost završena.

Poglavlje 3

Dinamičke aktivnosti

U ovom poglavlju bit će opisane dinamičke aktivnosti prikladne za učenike. Ovakve aktivnosti učenicima su već na prvu zanimljivije nego statičke. Jedan od razloga je taj što su dinamičke aktivnosti same po sebi zahtijevaju više znanja nego statičke, a ih onda je učenicima veći osjećaj postignuća uspješno odraditi. Također, vizualno su privlačnije jer su uglavnom "pomične" ili interaktivne (ili oboje).

3.1 While petlja i uvjetne naredbe

While petlja

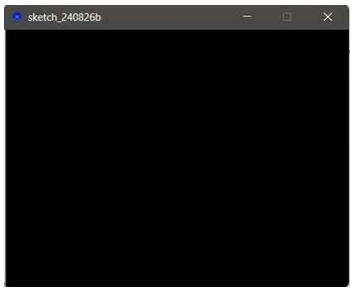
Slika 3.1: Sintaksa while petlje

While petlja procjenjuje **Uvjet**, koji je Booleov izraz. Ako je uvjet istinit (Booleov izraz ***True***), izvršava se tijelo while petlje (naredbe). Stanje se ponovno procjenjuje te se taj proces nastavlja sve dok uvjet nije lažan (Booleov izraz ***False***). Nakon što se uvjet ocijeni kao lažan, petlja se prekida. Bitno je napomenuti da bi se varijable koje se koriste u uvjetu unutar petlje trebale ažurirati tako da na kraju dobije vrijednost ***False***. Inače, petlja nastavlja raditi, stvarajući beskonačnu petlju.[14]

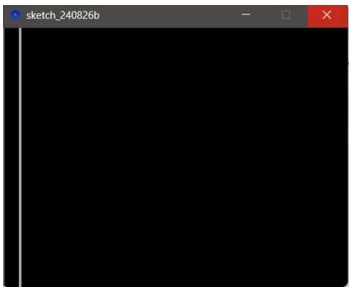
3.2 Primjeri aktivnosti

Aktivnost 5 - Putujuća dužina ostavlja trag[20]

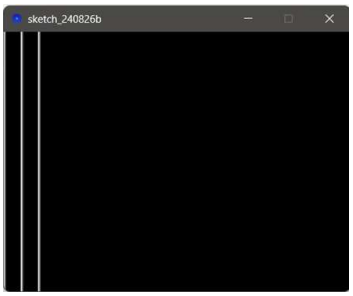
Tekst aktivnosti: Prikaži putanju dužine od jednog do drugog ruba ekrana koja za sobom ostavlja trag.



Slika 3.2: Putujuća dužina - Korak 1



Slika 3.3: Putujuća dužina - Korak 2



Slika 3.4: Putujuća dužina - Korak 3

U ovoj aktivnosti bit će prikazano je kako se može animirati putovanje dužine preko ekrana zadanih dimenzija koristeći Processing. Može se iskoristiti za početni primjer načina izrade animacije. Ono što se od učenika očekuje da znaju prije provođenja aktivnosti je razumijevanje razlike između funkcije draw() koja je sama po sebi beskonačna petlja i petlje while koja se koristi unutar nje. Također, bitno je da učenici znaju pojam globalne varijable, kako nacrtati dužinu i promijeniti njezina svojstva.
Očekivani ishodi iz informatike[8]:

- **B.1.1** analizira problem, definira ulazne i izlazne vrijednosti te uočava korake za rješavanje problema.
- **B.1.2** primjenjuje jednostavne tipove podataka te argumentira njihov odabir, primjenjuje različite vrste izraza, operacija, relacija i standardnih funkcija za modeliranje jednostavnoga problema u odabranome programskom jeziku.
- **B.1.3** razvija algoritam i stvara program u odabranome programskom jeziku rješavajući problem uporabom strukture grananja i ponavljanja.

Potrebne naredbe:

- **size(širina, visina)** - širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.

- **background(boja)** postavlja pozadinsku boju prozora. Argument predstavlja boju u sivim tonovima (0-255) ili u RGB formatu (ako su navedene tri vrijednosti).
- **strokeWeight(broj)** postavlja debljinu obruba na vrijednost broj.
- **stroke(nijansa)** postavlja boju obruba za crtanje oblika. **Nijansa** može biti bilo koji cijeli broj od 0 = crno do 255 = bijelo, svaki drugi broj: 50, 87, 162, 209 i tako dalje, je nijansa sive u rasponu od crne do bijele. Bitno je napomenuti da naredba može imati i oblik **stroke(r, g, b)**. Zadani skup boja je RGB (Red Green Blue), svaki parametar (r, g i b) poprima vrijednosti u rasponu 0-255.
- **line(x1, y1, x2, y2)** crta dužinu između dviju točaka s danim koordinatama.

```
maxX=0
def setup():
    size(400, 300)  # Postavi veličinu prozora

def draw():
    global maxX
    background(0)  # Postavi crnu pozadinu
    strokeWeight(2)  # Debljina linije
    stroke(255)  # Postavi bijelu boju linija

    x = 0
    while x < maxX:  # Petlja koja crta vertikalne linije
        line(x, 0, x, height)
        x=x+20
    if maxX < width:  # Prestani povećavati maxX kad dosegne vrijednost jednaku kao width
        maxX = maxX + 1
```

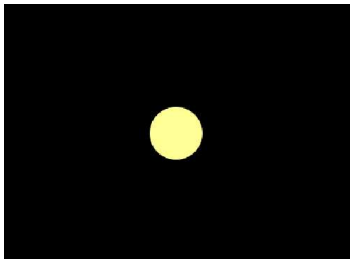
Slika 3.5: Putujuća dužina koja ostavlja trag-kôd

Za učenike kojima ova aktivnost nije dovoljno izazovna, nastavnik može ponuditi zahtjevniju verziju ove aktivnosti.

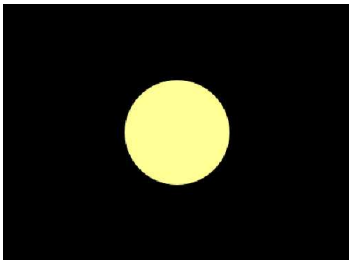
Aktivnost 5* - Krugovi u pokretu

Tekst uz aktivnost: Izradi animaciju koja prikazuje izlazak sunca.
Dodatne naredbe potrebne:

- **noStroke()** isključuje crtanje obruba oblika.
- **break** završava izvođenje switch, for ili while prelazi na sljedeću naredbu nakon petlje.



Slika 3.6: Krugovi u pokretu - Korak 1



Slika 3.7: Krugovi u pokretu - Korak 2

Slika 3.8: Krugovi u pokretu - Korak 3

```
r = 0  # početni radijus

def setup():
    size(400, 400)  # veličina prozora
    background(0)  # crna pozadina

def draw():
    global r
    background(0)  # osvježi pozadinu svaki frame
    # Crtanje kruga dok radijus ne dosegne širinu prozora
    while r < width:
        fill(255, 255, 0)  # žuta boja
        noStroke()  # bez rubova
        ellipse(width / 2, height / 2, r*2 , r*2 )  # crta krug
        r += 5  # povećaj radijus
        break
```

Slika 3.9: Krugovi u pokretu - kôd

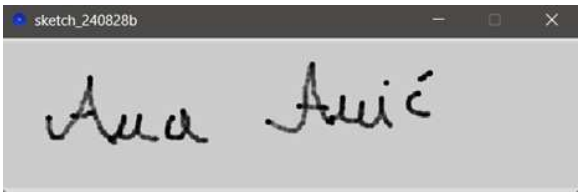
Problem koji se može javiti u ovoj aktivnosti je da učenici ne mogu razumjeti kako funkcionira while petlja unutar funkcije draw() i zašto i kako koristimo naredbu break. Ovaj zadatak se može riješiti i bez while petlje i dobro je s učenicima raspraviti i to rješenje. Ovdje je dobra prilika komentirati prednosti i mane oba načina i koji je način ”bolji” te što ga točno čini boljim.

```
r = 0  # početni radijus

def setup():
    size(400, 400)  # veličina prozora
    background(0)  # crna pozadina

def draw():
    global r
    background(0)  # osvježi pozadinu svaki frame

    # Crtanje kruga dok radijus ne dosegne polovicu širine prozora
    if r < width:
        fill(255, 255, 0)  # žuta boja
        noStroke()  # bez rubova
        ellipse(width / 2, height / 2, r * 2, r * 2)  # crta krug
        r += 5  # povećaj radijus
```

Slika 3.10: Krugovi u pokretu - alternativno rješenje

Aktivnost 6: Digitalni potpis[2]

Tekst uz aktivnost: Izradi program koji će korisniku omogućiti napisati i spremiti svoj potpis za daljnju uporabu.



Slika 3.11: Digitalni popis-primjer

Ova aktivnost primjerena je za učenike najmanje 1. razreda srednje škole. Ona omogućuje učenicima učenje osnovnih koncepata programiranja kroz grafičko programiranje i interaktivnost. Učenici mogu dobiti uvid kako funkcije poput *mousePressed* i *keyPressed* koriste položaj miša te tipki na tipkovnici. Osim toga mogu razvijati digitalne crteže koje onda spremaju kao fotografije i time steknu osnovne vještine digitalne pismenosti. Također, još jedna bitna stvar na koju nastavnik može skrenuti pozornost je korištenje digitalnog potpisa u svakodnevnom životu. Naime, modernizacijom tehnologije i povećanjem užurbanosti života stvorila se potreba za digitalizacijom dokumenata i potpisa.
Očekivani ishodi iz informatike[8]:

- **B.1.1** analizira problem, definira ulazne i izlazne vrijednosti te uočava korake za rješavanje problema

- **B.1.2** primjenjuje jednostavne tipove podataka te argumentira njihov odabir, primjenjuje različite vrste izraza, operacija, relacija i standardnih funkcija za modeliranje jednostavnoga problema u odabranome programskom jeziku

- **B.1.3** razvija algoritam i stvara program u odabranome programskom jeziku rješavajući problem uporabom strukture grananja i ponavljanja.

- **C.1.1** pronalazi podatke i informacije, odabire prikladne izvore informacija te uređuje, stvara i objavljuje/dijeli svoje digitalne sadržaje

- **D.1.1** u suradničkome online okruženju na zajedničkom projektu analizira etička pitanja koja proizlaze iz korištenja računalnom tehnologijom

Potrebne naredbe:

- **size(širina, visina)** - širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.

- **background(boja)** postavlja pozadinsku boju prozora. Argument predstavlja boju u sivim tonovima (0-255) ili u RGB formatu (ako su navedene tri vrijednosti).

- **strokeWeight(broj)** postavlja debljinu obruba na vrijednost broj.

- **stroke(nijansa)** postavlja boju obruba za crtanje oblika. **Nijansa** može biti bilo koji cijeli broj od 0 = crno do 255 = bijelo, svaki drugi broj: 50, 87, 162, 209 i tako dalje, je nijansa sive u rasponu od crne do bijele. Bitno je napomenuti da naredba može imati i oblik **stroke(r, g, b)**. Zadani skup boja je RGB (Red Green Blue), svaki parametar (r, g i b) poprima vrijednosti u rasponu 0 - 255.

- **line(x1, y1, x2, y2)** crta dužinu između dviju točaka s danim koordinatama.

- **save(”nazivdatoteke”)** sprema sliku iz prozora za crtanje. Datotečni nastavak se dodaje nazivu datoteke kako bi se označio format datoteke koji će se koristiti: TIFF (.tif), TARGA (.tga), JPEG (.jpg) ili PNG (.png). Ako u nazivu datoteke nije uključena ekstenzija, slika će se spremiti u TIFF formatu, a nazivu će se dodati .tif. Te se datoteke spremaju u mapu programa, koja se može otvoriti odabirom ”Prikaži mapu skice” iz izbornika ”Skica”. Alternativno, datoteke se mogu spremiti na bilo koje mjesto na računalu korištenjem apsolutne staze.[4]

- **print("tekst")** ispisuje tekst u područje konzole što je prozor na dnu okruženja obrade. Ova je funkcija često korisna za pregled podataka koje program vraća korisniku.

- **keyPressed()** poziva se jednom svaki put kada se pritisne tipka. Tipka koja je pritisnuta pohranjuje se u varijabli *key*.

Također, potrebno je poznavati i varijable koje daju koordinate trenutne pozicije miša: **mouseX, mouseY** i koordinate prethodne pozicije miša: **pmouseX, pmouseY**.

```
def setup():
    size(480, 120)
    strokeWeight(4)  # Postavi debljinu linije na 4 piksela
    stroke(0, 102)  # Postavi boju linije (tamno plava)
    background(255)  # Bijela pozadina

def draw():
    # Ako je miš pritisnut, crtaj liniju između trenutne i prethodne pozicije miša
    if mousePressed:
        line(mouseX, mouseY, pmouseX, pmouseY)

def keyPressed():
    # Ako pritisneš tipku 's' ili 'S'
    if key == 's' or key == 'S':
        save("potpis.png")  # Spremi crtež kao sliku pod nazivom "potpis.png"
        print("Slika spremljena.")  # Ispiši poruku u konzoli
```

Slika 3.12: Digitalni potpis-kôd

Aktivnost 8 - Vjerojatnost u boji[16]

Tekst uz aktivnost: Napravi skicu u koja nasumično crta kružnice u tri boje: crvena, zelena i plava, s različitim šansama pojavljivanja na ekranu. Crvena boja neka dominira s 60% vjerojatnosti, dok plava i zelena imaju redom 30% i 10%.

Slika 3.13: Vjerojatnost u boji

Ova aktivnost primjerena je za učenike 4. razreda srednje škole. Naime, osim osnovnih koncepata programiranja učenici moraju poznavati pojam vjerojatnosti slučajnog događaja, tj. što on označava i kako se određuje. Aktivnost je idealna za povezivanje matematike i informatike i može se koristiti kao međupredmetna aktivnost.
Očekivani ishodi iz informatike:[8]

- **B.1.1** analizira problem, definira ulazne i izlazne vrijednosti te uočava korake za rješavanje problema
- **B.1.2** primjenjuje jednostavne tipove podataka te argumentira njihov odabir, primjenjuje različite vrste izraza, operacija, relacija i standardnih funkcija za modeliranje jednostavnoga problema u odabranome programskom jeziku
- **B.1.3** razvija algoritam i stvara program u odabranome programskom jeziku rješavajući problem uporabom strukture grananja i ponavljanja.

Postoji i poveznica s matematikom, a ishod koji se postiže je[10]:

- **E.4.1.** Argumentirano računa vjerojatnost.

Potrebne naredbe:

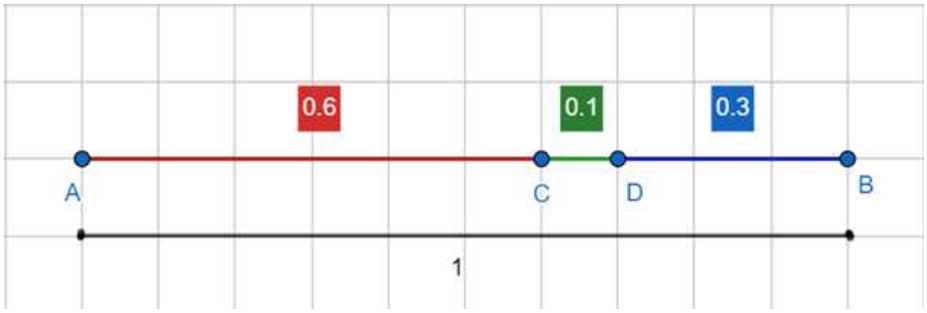
- **size(širina, visina)** - širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.
- **background(boja)** postavlja pozadinsku boju prozora. Argument predstavlja boju u sivim tonovima (0 - 255) ili u RGB formatu (ako su navedene tri vrijednosti).

- **fill(r, g, b)** postavlja boju ispune za oblike koji će se crtati. Argumenti su vrijednosti crvene, zelene i plave komponente boje, što omogućuje specificiranje nijanse boje.
- **ellipse(x, y, širina, visina)** crta elipsu na temelju zadanih koordinata njenog središta i dimenzija. Prvi par (x, y) predstavlja središte elipse, dok drugi par predstavlja širinu i visinu elipse.[22]
- **noStroke()** isključuje crtanje obruba oblika.
- **random(a, b)** vraća nasumični decimalni broj (floating-point number) između vrijednosti a (uključivo) i b (isključivo).

```
def setup():
    size(480, 270)  # Postavi veličinu platna
    background(255)  # Postavi boju pozadine na bijelu
    noStroke()  # Onemogući obrise (konture) za oblike

def draw():
    # Vjerojatnosti za 3 različita slučaja
    # Ove vjerojatnosti moraju biti ukupno 100%!
    vjerojatnost_crvena = 0.60  # 60% šanse za crvenu boju
    vjerojatnost_zelena = 0.10  # 10% šanse za zelenu boju
    vjerojatnost_plava = 0.30   # 30% šanse za plavu boju

    # Odaberi slučajan broj između 0 i 1
    broj = random(1)

    # Ako je slučajan broj manji od 0.60
    if broj < vjerojatnost_crvena:
        fill(255, 53, 2, 150)  # Postavi boju ispune na poluprozirnu crvenu
    # Ako je slučajan broj između 0.60 i 0.70
    elif broj < vjerojatnost_zelena + vjerojatnost_crvena:
        fill(156, 255, 28, 150)  # Postavi boju ispune na poluprozirnu zelenu
    # Svi ostali slučajevi (tj. između 0.70 i 1.0)
    else:
        fill(10, 52, 178, 150)  # Postavi boju ispune na poluprozirnu plavu

    # Nacrtaj elipsu na nasumičnoj poziciji s fiksnom veličinom
    ellipse(random(width), random(height), 64, 64)
```

Slika 3.14: Vjerojatnost u boji - kôd

Ono što će učenicima uglavnom predstavljati problem s ovim zadatkom je razumjeti kako je vjerojatnost odabira slučajnog broja manjeg od 0.6 u rasponu od 0 do 1 jednaka 60%, kako je vjerojatnost da se odabere broj veći od 0.6, a manji od 0.7 jednaka 10% i da je vjerojatnost odabira broja većeg od 0.7 i manjeg od 1 jednaka 30%. Ovdje bi bilo dobro riješiti problem koristeći geometrijsku vjerojatnost. Npr. ako imamo duljinu dužine 1 na kojoj se nalaze točke A, B i C zadane na sljedeći način:



Slika 3.15: Vjerojatnost u boji

Naš problem je analogan kao i problem određivanja koja je vjerojatnost da slučajno odabrana točka na dužini $\overline{AB}$ bude na crvenom dijelu dužine (dio dužine $\overline{AC}$), zelenom dijelu dužine (dio dužine $\overline{CD}$) i plavom dijelu dužine (dio dužine $\overline{DB}$). Npr.:

$$p(\text{odabrana točka je na dužini } \overline{AC}) = \frac{|AC|}{|AB|} = 0.6 = 60\%,$$

Analogno vrijedi:

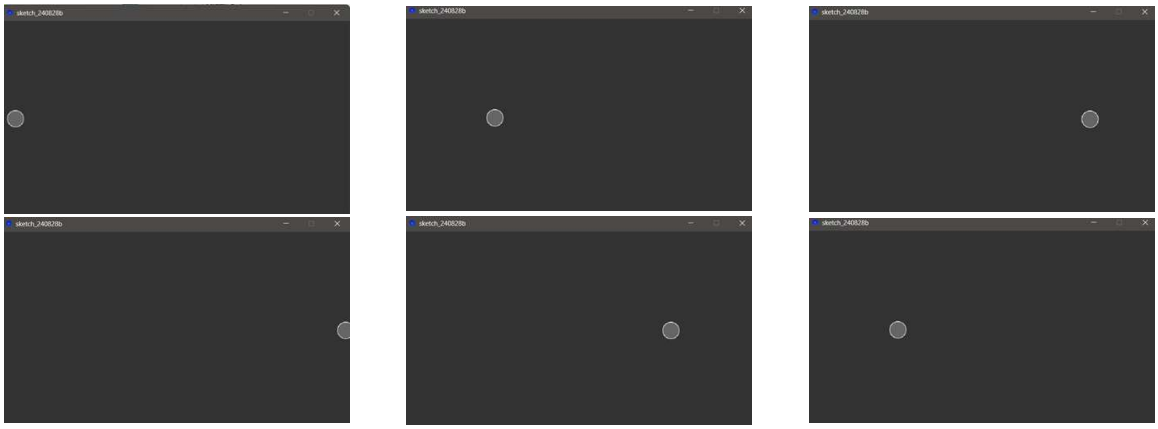
$$p(\text{odabrana točka je na dužini } \overline{CD}) = \frac{|CD|}{|AB|} = 0.1 = 10\%,$$

$$p(\text{odabrana točka je na dužini } \overline{CB}) = \frac{|CB|}{|AB|} = 0.3 = 30\%.$$

Slična aktivnost u kojoj se također koriste animacije i "kretanje kružnice" po ekranu, ali bez poveznice s vjerojatnosti prikazana je u nastavku.

Aktivnost 8*: Odbijajuća loptica[17]

Tekst uz aktivnost: Napravi animaciju u kojoj se loptica kreće od jednog do drugog ruba ekrana te se odbija od njih.



Slika 3.16: Odbijajuća loptica - putanja

```
# Definiraj globalne varijable
xbrzina = 5
lopticaX = 0

def setup():
    size(640, 360)  # Postavi veličinu prozora

def draw():
    global lopticaX, xbrzina  # Omogućava promjene globalnih varijabli
    background(51)  # Postavi pozadinu na tamno sivu
    fill(102)  # Postavi boju ispunjavanja loptice
    stroke(255)  # Postavi boju obruba loptice
    ellipse(lopticaX, height / 2, 30, 30)  # Crtaj krug na trenutnoj poziciji

    lopticaX += xbrzina  # Ažuriraj poziciju loptice

    # Promjeni smjer kretanja loptice ako dotakne rubove prozora
    if lopticaX > width or lopticaX < 0:
        xbrzina *= -1
```

Slika 3.17: Odbijajuća loptica - kôd

Poglavlje 4

FMS Logo ili Python Mode for Processing

4.1 FMS Logo

FMS Logo je stvoren 1967. i bio je jednako moćan kao i drugi programi koji su nastali u to vrijeme. Svijet računarstva se od tada dosta promijenio i zahtjevi industrije doveli su do mnogih novih jezika koji nude nove moćnije značajke. Za to vrijeme FMS Logo se nije puno promijenio. Danas mu nedostaju mnoge značajke koje su potrebne u industriji, pa ga programeri ne koriste ni za što drugo osim za male projekte. Dakle, postavlja se pitanje: ”Ako FMS Logo ne koriste programeri, zašto ga uopće koristiti?” Kao prvo, Logo nije dizajniran za stvaranje industrijskog softvera, dizajniran je kao alat koji pomaže djeci da nauče važnu vještinu, onu koja je jednako važna danas kao što je bila 1967.: vještina učenja kako učiti. Dok su drugi jezici dizajnirani oko apstraktnih pojmova u informatici, Logo je dizajniran oko toga kako ljudi uče. Osnovna filozofija FMS Loga je da se obrazovanje treba usredotočiti na učenje, a ne na poučavanje, i da se učenje najbolje događa kada je učenik mentalno angažiran u konstruktivnom procesu. Programer preuzima ulogu ”učitelja” i uči ”podučavajući” računalo da nešto napravi. FMS Logo nije, sam po sebi, predmet učenja, već je alat za istraživanje ideja u područjima kao što su umjetnost, glazba, matematika, inteligencija i jezik.

Još jedna prednost FMS Loga je podrška za ”Turtle Graphics” (hrv. ”kornjačinu grafiku”). Kornjačina grafika jednostavan je i moćan skup naredbi koji se koristi za upravljanje zaslonskim objektom koji se zove kornjača. Ideja koja stoji iza kornjačine grafike je da se programira davanjem uputa ”kornjači” (predstavljenoj u FMS Logu kao trokut) i te upute omogućuju da kornjača hoda uokolo po ekranu, povlačeći crtu od prethodne do iduće pozicije. Kornjačina grafika donosi računalno programiranje u svijet koji je učenicima poznat: svijet boja, oblika i umjetnosti. Omogućuje im da počnu programirati u fazi svog kognitiv-

nog razvoja prije nego što razumiju simboličko zaključivanje.

Logo ima jednostavnu sintaksu koja zahtijeva manje kôda nego drugi jezici, što je posebno važno za djecu koja nisu u potpunosti razvila sposobnost brzog i sigurnog pisanja koristeći tipkovnicu. Ravna crta povlači se naredbom *FD 100*. U drugim programskim jezicima, ista naredba može biti *Turtle.Forward(100);*, a ako se zaboravi jedan od sintaktičkih elemenata, kao što je točka-zarez na kraju, javit će se poruka o pogrešci koju je teško razumjeti, poput: ”sintaktička pogreška: nezavršena izjava”, što je teško razumljivo učeniku i otežava mu proces učenja.[19]

Prije nego prijeđemo na konkretne aktivnosti uvest ćemo i pojam funkcije te rekurzivne fukcije.

Funkcija

Funkcija je blok naredbi koje izvršavaju određeni zadatak. Ideja je okupiti neke uobičajene ili često ponavljane dijelove kôda i smjestiti ih u funkciju, kako bismo, umjesto da pišemo isti kod iznova za različite ulazne vrijednosti, mogli pozivati funkciju i ponovo koristiti kod koji se u njoj nalazi.
Prednosti korištenja funkcija su:

- povećavanje čitljivosti kôda,
- povećavanje ponovne iskoristivosti kôda.

Sintaksa funkcije u Pythonu prikazana je na slici ispod:

```
def ime_funkcije(parametri):
    tijelo_fukcije
    return izraz
```

Slika 4.1: Sintaksa funkcije u Pythonu

Objašnjenje dijelova sintakse:

- **def** je ključna riječ u Pythonu koja označava početak definicije funkcije.
- **ime_funkcije** je naziv funkcije koji sami odabiremo. Naziv bi trebao biti smislen, kako bi odražavao svrhu funkcije, i slijediti pravila imenovanja u Pythonu (bez razmaka, počinje slovom ili donjom crtom itd.).

- **parametri** su varijable koje funkcija prima prilikom poziva. Unutar zagrada pišemo jedan ili više parametara, odvojenih zarezom. Ako za rad funkcije nisu potrebne dodatne informacije, zagrade mogu ostati i prazne.

- **: (dvotočka)** - na kraju prve linije definicije funkcije koristimo dvotočku kako bismo označili početak tijela funkcije.

- **tijelo_funkcije** je glavni dio funkcije. Tu pišemo naredbe koje funkcija izvršava. Sve naredbe unutar tijela funkcije moraju biti uvučene (koristeći tabulator), što Python koristi za prepoznavanje gdje tijelo počinje i završava.

- **return izraz** - ključna riječ **return** označava vrijednost koju funkcija vraća kada završi izvođenje. **Izraz** iza return može biti bilo koji tip podatka. Ako **return** nije naveden, funkcija automatski vraća *None*, što znači da nema povratne vrijednosti.[5]

Rekurzivna funkcija

Rekurzija se definira kao proces koji poziva sam sebe (izravno ili neizravno), a odgovarajuća funkcija naziva se rekurzivna funkcija. Rekurzija omogućuje rješavanje problema razbijanjem na manje i lakše potprobleme (koji se rješavaju na isti način), ali zahtijeva i posebni tzv. **osnovni slučaj** kako bi se izbjegao beskonačan posao. Rekurzivne funkcije u nekim slučajevima nemaju smisla jer se za svaki poziv moraju neprestano postavljati novi podaci na stog. Prednosti rekurzije su uglavnom pojednostavljivanje problema i veća preglednost i čitljivost koda. Rekurzija se često koristi u algoritmima pretraživanja i sortiranja te u grafici za generiranje uzoraka.[7]

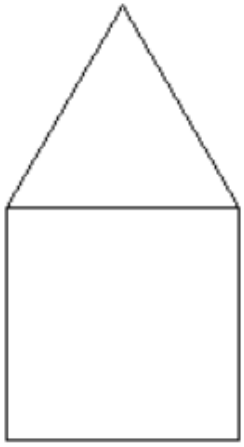
Općenito, rekurzivne funkcije kod učenika izazivaju strah jer im je teško zamisliti što bi značilo da funkcija poziva samu sebe, gdje su zapisani ti pozivi i kojim redom se obrađuju. Također, teško im je odrediti osnovni slučaj koji zaustavlja pozivanje rekurzije. Postavlja se i pitanje zašto je nužan i što bi značilo da se nešto beskonačno puta izvodi. Iako su učenicima možda nekad teško shvatljive te ih se možda plaše koristiti, treba im pokazati njihovu pozitivnu stranu koja je značajno skraćivanje kôda, tj. njegove čitljivosti.

4.2 Primjeri aktivnosti

Prikazat ćemo na jednostavnom primjeru crtanja kuće koje su prednosti i/ili nedostaci odabira FMS Loga i Python Mode for Processinga za crtanje jednostavnih crteža koristeći geometrijske oblike.

Aktivnost 9 - Kuća

Tekst uz aktivnost: Nacrtaj kuću po uzoru na sliku ispod.



Slika 4.2: Kuća

Najprije proučimo rješenje u FMS Logu.

```
TO KUCA :A
REPEAT 4[FD :A RT 90]
FD :A
RT 90
REPEAT 3[FD :A LT 120]
END
```

Slika 4.3: Kuća - kôd FMS Logo

Kuću ovisno o tome koja je duljina stranice kvadrata i trokuta (koji zajedno čine kuću) crtamo upisujući u editor teksta npr. naredbu oblika: **KUCA 100** (za duljinu stranice jednaku 100).
Potrebne naredbe:

- **FD :A** - pomiče kornjaču naprijed za :A jedinica.
- **RT :KUT** - zakreće kornjaču za vrijednost varijable :KUT udesno.
- **LT :KUT** - zakreće kornjaču za vrijednost varijable :KUT ulijevo.

- **REPEAT :N [NAREDBE]** - koristi se za ponavljanje skupa naredbi navedenih u uglatim zagradama :N puta.

Aktivnost je primjerena za učenike petog razreda osnovne škole nadalje. Naime, koriste se samo osnovne naredbe kretanja kornjače: naprijed, lijevo i desno.
Očekivani ishod iz informatike[8] :
B.5.1 koristi se programskim alatom za stvaranje programa u kojemu se koristi ulaznim i izlaznim vrijednostima te ponavljanjem.
Proučimo rješenje i u Python Modeu for Processing.

```
def setup():
    size(400, 400)
    background(255)
    stroke(0)

def draw():
    translate(width / 2, height / 2)  # Pomičemo početnu poziciju u središte prozora za crtanje
    for i in range(4):
        line(0, 0, 100, 0)  # Crtamo liniju od trenutne pozicije do (a, 0)
        translate(100, 0)   # Pomičemo koordinatni sustav desno za 'a'
        rotate(radians(90))  #Okrećemo se za 90 stupnjeva udesno
    for i in range(3):
        line(0, 0, 100, 0)  # Crtamo liniju od trenutne pozicije do (a, 0)
        translate(100, 0)   # Pomičemo koordinatni sustav desno za 'a'
        rotate(-radians(120))

    noLoop()  # Zaustavlja daljnje pozivanje funkcije draw()
```

Slika 4.4: Kuća - kôd Python Mode for Processingu

Naredbe koje su potrebne:

- **size(širina, visina)** - Širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.

- **stroke(nijansa)** postavlja boju obruba za crtanje oblika. Nijansa može biti bilo koji cijeli broj od 0 = crno do 255 = bijelo.

- **background(nijansa)** - Naredba postavlja boju pozadine. Nijansa može biti bilo koji cijeli broj od 0 = crno do 255 = bijelo, svaki drugi broj: 50, 87, 162, 209 i tako dalje, je nijansa sive u rasponu od crne do bijele.[12].

- **translate(x, y)** određuje položaj objekata unutar prozora za prikaz. Parametar x određuje pomak lijevo/desno, a parametar y određuje pomak gore/dolje.

- **line(x1, y1, x2, y2)** crta dužinu između dviju točaka s danim koordinatama.

- **rotate(kut)** rotira oblik za iznos određen parametrom **kut**. Kutovi bi trebali biti navedeni u radijanima (vrijednosti od 0 do TWO_PI) ili pretvoreni u radijane pomoću funkcije **radians()**.

- **noloop()** zaustavlja obradu od neprekidnog izvršavanja koda unutar draw().

Ova aktivnost primjerena je za učenike najmanje šestog razreda osnovne škole pa nadalje. Naime, ovdje nastavnik može izbjeći detaljno objašnjavanje pojmova kao što su translacija i rotacija te ih povezivati s odgovarajućim konceptima transformacija ravnine. Ako se aktivnost provodi u sedmom ili nekom višem razredu dobro je spomenuti te poveznice. U šestom razredu dovoljno je naredbe objasniti samo prijevodom na hrvatski jezik *translate* možemo prevesti kao "pomicanje", a *rotate* kao "zakretanje".
Očekivani ishodi iz informatike[8]:

- **B.6.1.** Učenik stvara, prati i preuređuje programe koji sadrže strukture grananja i uvjetnoga ponavljanja te predviđa ponašanje jednostavnih algoritama koji mogu biti prikazani dijagramom, riječima govornoga jezika ili programskim jezikom.

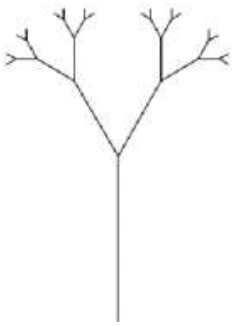
- **B.6.2** Nakon šeste godine učenja predmeta Informatika u domeni Računalno razmišljanje i programiranje učenik razmatra i rješava složeniji problem rastavljajući ga na niz potproblema.

Ono što zasigurno možemo primijetiti je da je rješenje jednostavnije u FMS Logu. Potrebne su samo najosnovnije naredbe te se kôd sastoji samo od nekoliko naredbi. Za rješenje u Python Mode for Processingu potrebno je poznavati pojam **for** petlje te beskonačne petlje i potrebe njezinog zaustavljanja (kao koncept bitan za rad funkcije **draw()**).

U nastavku ćemo na primjeru zadatka koji koristi rekurzivnu funkciju u rješenju raspraviti razlike između programiranja u FMS Logu i Python Mode for Processingu.

Aktivnost 10 - Fraktalno stablo

Tekst uz aktivnost: Napišite program koji crta stablo koristeći rekurzivan poziv funkcije. Stablo treba biti sastavljeno od grana koje se granaju u dva smjera.
Očekivani ishod iz informatike[8]:
B.3.3 rješava problem primjenjujući rekurzivnu funkciju



Slika 4.5: Fraktalno stablo - primjer

Potrebne naredbe:

- **size(širina, visina)** - širina i visina su prirodni brojevi, a naredba postavlja veličinu prozora za prikaz.
- **background(boja)** postavlja pozadinsku boju prozora. Argument predstavlja boju u sivim tonovima (0 - 255) ili u RGB formatu (ako su navedene tri vrijednosti).
- **translate(x, y)** određuje položaj objekata unutar prozora za prikaz. Parametar x određuje pomak lijevo/desno, a parametar y određuje pomak gore/dolje.
- **line(x1, y1, x2, y2)** crta dužinu između dviju točaka s danim koordinatama.
- **pushMatrix() i popMatrix()**: Funkcija **pushMatrix()** sprema trenutni koordinatni sustav na stog, a **popMatrix()** vraća prethodni koordinatni sustav.
  pushMatrix() i popMatrix() koriste se zajedno s drugim transformacijskim funkcijama i mogu se ugraditi za kontrolu opsega transformacija.
- **rotate(kut)** rotira oblik za iznos određen parametrom **kut**. Kutovi bi trebali biti navedeni u radijanima (vrijednosti od 0 do TWO_PI) ili pretvoreni u radijane pomoću funkcije **radians()**.

```
def setup():
    size(600, 600)              # Postavljanje veličine prozora
    background(255)             # Bijela pozadina
    translate(width / 2, height) # Pomak na sredinu donjeg dijela ekrana
    draw_tree(150)              # Crtanje stabla s veličinom grane 150

def draw_tree(size):
    if size < 5:                # Ako je veličina grane manja od 5, zaustavi se
        return

    line(0, 0, 0, -size)        # Crtaj granu prema gore
    translate(0, -size)         # Pomiči se na vrh grane

    # Crtanje desne grane
    pushMatrix()                 # Spremi trenutni položaj
    rotate(radians(30))         # Rotiraj za 30 stupnjeva
    draw_tree(size / 2)         # Crtaj manju granu
    popMatrix()                  # Vrati se na prethodni položaj

    # Crtanje lijeve grane
    pushMatrix()                 # Spremi trenutni položaj
    rotate(radians(-30))        # Rotiraj za -30 stupnjeva
    draw_tree(size / 2)         # Crtaj manju granu
    popMatrix()                  # Vrati se na prethodni položaj

    translate(0, size)          # Vraća se na početni položaj
```

Slika 4.6: Fraktalno stablo - kôd

Ono što se svakako može primijetiti je složenost ovog zadatka. Izuzevši kompleksni pojam rekurzivne funkcije učenicima predstavljaju izazov i naredbe za spremanje i ponovno vraćanje položaja koordinatnog sustava. Pogledajmo rješenje u FMS Logu.
Naredbe koje su potrebne:

- **FD :A** - pomiče kornjaču naprijed za :A jedinica.
- **BK :A** - pomiče kornjaču natrag za :A jedinica.
- **RT :KUT** - zakreće kornjaču za vrijednost varijable :KUT ulijevo.
- **LT :KUT** - zakreće kornjaču za vrijednost varijable :KUT udesno.

```
TO TREE :SIZE
  IF :SIZE < 5 [STOP] ; Osnovni slučaj: zaustavlja rekurziju kada je veličina grane premala

  ; Nacrtaj glavnu granu
  FORWARD :SIZE

  ; Stvori desnu granu
  RIGHT 30
  TREE :SIZE / 2

  ; Vrati se u početni kut i stvori lijevu granu
  LEFT 60
  TREE :SIZE / 2

  ; Vrati se u početni položaj i kut
  RIGHT 30
  BACK :SIZE
END
```

Slika 4.7: Fraktalno stablo - kôd

Očekivani ishod iz informatike[8] :
B.8.3 Učenik prepoznaje i opisuje mogućnost primjene rekurzivnih postupaka pri rješavanju odabranih problema te istražuje daljnje mogućnosti primjene rekurzije.

Iz ovih primjera rješenja možemo zaključiti je da je kôd puno jednostavniji u programskom jeziku FMS Logo. Koriste se samo osnovne naredbe koje učenici uče već u šestom razredu osnovne škole. Ipak pojam rekurzivne funkcije uvodi se tek u osmom razredu osnovne škole pa je ovaj zadatak primjeren tek za kraj osnovnoškolskog obrazovanja. Isti zadatak u Python Modeu for Processing primjeren je tek za srednju školu. To je prvenstveno zbog većeg broja kompleksnijih naredbi, ali i pojmova (mjere kuta: stupnjevi i radijani, transformacije: rotacija, translacija, matrica, stog) koje je ključno razumjeti kako bi se zadatak točno riješio i to s razumijevanjem.

Zaključak

FMS Logo je poprilično jednostavan programski jezik. U potpunosti je primjeren učenicima koji se prvi put susreću s programiranjem i u njemu se može koristeći nekoliko jednostavnih naredbi steći osnovno znanje o varijablama, grananju i naredbama ponavljanja. Ipak, njegove mogućnosti su dosta sužene. Već za neke složenije probleme izrade animacija ili igrica, FMS Logo nam nije poželjan niti koristan alat. S druge strane, Python Mode for Processing pruža mnogo više mogućnosti. Za početak, učenici mogu s jedne strane koristiti gotove funkcije za crtanje oblika. Također, ne trebaju toliko posvećivati pažnju na pisanja svakog detalja kojeg koriste u složenom problemu već postoje razne biblioteke koje sadrže već gotove funkcije. Samim tim učenici imaju dobru podlogu za istraživanje i rješavanje kompleksnih problema.

Što se tiče primjerenosti dobi, FMS Logo je odličan alat za učenike u nižim razredima osnovne škole. Već u višim razredima osnovne škole učenike bi trebalo upoznati s programskim jezikom koji ima više mogućnosti i koji se uistinu koristi u velikom broju područja rada danas. Takav je upravo programski jezik Python. Koristi se za razvoj softvera u brojnim tvrtkama koje se bave: razvojem web i mobilnih aplikacija, analizom podataka, strojnim učenjem, financijskom tehnologijom, automatizacijom itd. Jedna od njegovih velikih prednosti je da je to programski jezik s iznimno jednostavnom sintaksom. Naime, Python se može uvesti i korištenjem Python Mode for Processinga koristeći neke od aktivnosti iz poglavlja 2 ili 3. Kako se prijelaz s FMS Loga na Python Mode for Processing može napraviti pokazat ćemo u sljedećem potpoglavlju.

4.3 Prijelaz s FMS Loga na Python Mode for Processing

Prvi korak prijelaza iz FMS Loga u Python Mode for Processing bio bi osnovne funkcije iz FMS Loga napisati u biblioteku u Pythonu pa ju onda koristiti Python Modeu for Processing programima. Kako bi prijelaz imao smisla te funkcije ćemo nazvati isto kao u FMS Logu. Biblioteka bi trebala sadržavati funkcije s nazivima: fd, bk, cs, rt, lt, pu, pd, setpensize i setpencolor. Možemo ih spremiti kao Python datoteku s nazivom npr. *logo_funkcije.py* te ju kao takvu možemo koristiti. Jedino na što trebamo pripaziti jest da su naša biblioteka i program u kojem je koristimo spremljeni u istoj mapi.
Jedan od načina implementacije funkcija je:

```
1  crtati = 1  # Varijabla za zapis stanja "crtanja" (1 za crtanje,
       0 za pauzu)
2
3  def fd(x):
4      # Funkcija za pomicanje naprijed
5      # Ako je "olovka" spuštena, tj. crtanje dozvoljeno (crtati ==
           1), crtamo dužinu
6      if crtati == 1:
7          line(0, 0, x, 0)  # Crta dužinu duljine x prema naprijed
8      translate(x, 0)  # Pomak u istom smjeru za x
9
10 def bk(x):
11     # Funkcija za pomicanje unatrag
12     # Ako je "olovka" spuštena, tj. crtanje dozvoljeno, crta
           liniju prema unatrag
13     if crtati == 1:
14         line(0, 0, -x, 0)  # Crtanje linije duljine x prema nazad
15     translate(-x, 0)  # Pomak u suprotnom smjeru za x
16
```

```
17 def cs():
18     # Funkcija za vraćanje prozora za crtanje na početne postavke
19     resetMatrix()  #vraćanje sve transformacije prozora za
           crtanje na početne postavke  (rotacije i translacije)
20     background(255)  # Postavlja bijelu pozadinu
21     translate(width / 2, height / 2)  # Prelazak na sredinu
           prozora za crtanje
22
23 def rt(kut):
24     # Funkcija za rotaciju udesno za zadani kut
25     rotate(-radians(kut))  # Rotira za negativan kut kako bi se
           okrenuli udesno
26
27 def lt(kut):
28     # Funkcija za rotaciju ulijevo za zadani kut
29     rotate(radians(kut))  # Rotira za pozitivan kut kako bi se
           okrenuli ulijevo
30
31 def pu():
32     # Funkcija za postavljanje "olovke" u stanje u kojem ne
           ostavlja trag - olovka je podignuta
33     # Postavlja crtati na 0, tako da "olovka" ne ostavlja trag
34     global crtati
35     crtati = 0
36
37 def pd():
38     #Funkcija za postavljanje "olovke" u stanje crtanja - spušta
           olovku
39     global crtati
40     crtati = 1 # Postavlja crtati na 1, dozvoljava crtanje
41
42 def setpensize(x):
43     # Funkcija za postavljanje debljine "olovke"
44     strokeWeight(x)  # Postavlja debljinu "olovke" na zadanu
           vrijednost x
45
46 def setpencolor(r, g=-1, b=-1):
47     # Funkcija za postavljanje boje "olovke"
48     # Ako je samo crvena (r) zadana, boja je neka nijansa sive
49     if g == -1:
50         stroke(r)
51     else:
```

```
52          stroke(r, g, b)  # Postavlja boju "olovke"
```

Na početak programa u kojem koristimo ove funkcije trebe dodati naredbu:

```
1 from logo_funkcije import *
```

S pomoću ove jednostavne biblioteke sa samo devet funkcija dobili smo sve osnovne funkcionalnosti FMS Loga u Python Modeu for Processing.
Sljedećom aktivnošću pokazat ćemo kako polazeći od jednostavnog crteža šesterokuta, nastavnik dodajući aspekt interaktivnosti koristeći Python Mode for Processing može pokazati jednu od prednosti nad FMS Logom. Naime, od jednostavnih crteža koristeći nekolicinu jednostavnih funkcija idemo korak dalje do interakcije s tim crtežima.

Aktivnost 12 - Šesterokut prati položaj miša

Tekst uz aktivnost: Napiši program u kojem šesterokut prati položaj miša na ekranu.

Slika 4.8: Šesterokut prati položaj miša - primjer 1

Slika 4.9: Šesterokut prati položaj miša - primjer 2

Bitan korak izrade ove aktivnosti je u program dodati naredbu kojom uključujemo biblioteku: *logo_funkcije*. Pomoću nje možemo koristit funkcije **fd**, **lt**, **setpensize**, **setpencolor** i **cs** na isti način kao u FMS Logu.

```
# Uvoz svih funkcija iz biblioteke logo_funkcije
from logo_funkcije import *

# Funkcija za crtanje šesterokuta
def sesterokut():
    setpensize(12)          # Postavlja debljinu olovke na 12
    setpencolor(155)        # Postavlja boju olovke na nijansu sive (vrijednost 155)
    for x in range(6):      # Petlja za crtanje 6 stranica šesterokuta
        fd(50)              # Pomak naprijed za 50
        lt(360 / 6)         # Rotacija ulijevo za 60 stupnjeva (360 / 6)

def setup():
    size(600, 400)          # Postavlja veličinu prozora za crtanje na 600x400 piksela
    cs()                    # Čisti platno i postavlja početnu poziciju na sredinu prozora

# Funkcija koja se beskonačno puta ponavlja
def draw():
    background(255)         # Postavlja bijelu pozadinu za svaki frame
    resetMatrix()           # Resetira sve transformacije (rotacije, translacije)
    translate(mouseX, mouseY)  # Postavlja poziciju šesterokuta na položaj miša
    sesterokut()            # Poziva funkciju za crtanje šesterokuta
```

Slika 4.10: Šesterokut prati položaj miša - kôd

Aktivnost je primjerena za učenike najmanje šestog razreda osnovne škole. Učenici koji su do tada programirali u FMS Logu upoznati su sa svim korištenim funkcijama potrebnim za izradu aktivnosti te s pojmom petlje (for i while).
Očekivani ishodi iz informatike[8]:

- **B.6.1** stvara, prati i preuređuje programe koji sadrže strukture grananja i uvjetnoga ponavljanja te predviđa ponašanje jednostavnih algoritama koji mogu biti prikazani dijagramom, riječima govornoga jezika ili programskim jezikom.
- **B.6.2** razmatra i rješava složeniji problem rastavljajući ga na niz potproblema.

Zaključno, iako je FMS Logo doista koristan i jednostavan alat za uvođenje programiranja u osnovnoj školi, sve njegove njegove osnovne fukcionalnosti, a i puno više možemo vrlo jednostavno ostvariti koristeći Python Mode for Processing na zanimljiv i kreativan način.

Bibliografija

[1] *GeoGebra Calculator*, https://www.geogebra.org/calculator, pristupljeno: 12. 10. 2024.

[2] Ben Fry Casey Reas, *Make: Getting Started with Processing: A Quick, Hands-on Introduction*, 2nd., Make: Community, 2010.

[3] Processing Foundation, *Coordinate System and Shapes Tutorial*, https://processing.org/tutorials/coordinatesystemandshapes, pristupljeno: 30. 8. 2024.

[4] Processing Foundation, *Processing.py*, https://py.processing.org/, pristupljeno: 31. 7. 2024.

[5] GeeksforGeeks, *Python Functions*, https://www.geeksforgeeks.org/python-functions/, pristupljeno 25.10.2024.

[6] GeeksforGeeks, *Python Variables*, https://www.geeksforgeeks.org/python-variables/, pristupljeno: 26. 8. 2024.

[7] GeeksforGeeks, *What is Recursion?*, https://www.geeksforgeeks.org/what-is-recursion/, pristupljeno: 29. 9. 2024.

[8] Narodne novine, *Odluka o donošenju kurikuluma za nastavni predmet Informatike za osnovne škole i gimnazije u Republici Hrvatskoj*, 2018., https://narodne-novine.nn.hr/clanci/sluzbeni/2018_03_22_436.html, pristupljeno: 2. 9. 2024.

[9] Long Bao Nguyen, *Introduction to Processing for AP Computer Science A*, https://longbaonguyen.github.io/courses/apcsa/processing/processing1.pdf, pristupljeno: 30. 8. 2024.

[10] Narodne novine, *Odluka o donošenju kurikuluma za nastavni predmet Matematike za osnovne škole i gimnazije u Republici Hrvatskoj*, 2019., https:

//narodne-novine.nn.hr/clanci/sluzbeni/2019_01_7_146.html, pristupljeno:10. 10. 2024.

[11] Processing Organization, *setup()*, https://processing.org/reference/setup_.html, pristupljeno: 30. 8. 2024.

[12] Processing Foundation, *Color Tutorial*, https://processing.org/tutorials/color, pristupljeno: 30. 8. 2024.

[13] Processing Organization, *draw() - Reference*, https://py.processing.org/reference/draw, pristupljeno: 30. 8. 2024.

[14] Programiz, *Python While Loop - Programiz*, https://www.programiz.com/python-programming/while-loop, pristupljeno: 5. 9. 2024.

[15] Python Software Foundation, *For Loop - Python Wiki*, https://wiki.python.org/moin/ForLoop, pristupljeno: 2. 9. 2024.

[16] Daniel Shiffman, *Learning processing: The nature of code: Example 13-03: Probability*, http://learningprocessing.com/examples/chp13/example-13-03-probability, pristupljeno: 29. 8. 2024.

[17] The Coding Train Daniel Shiffman, *The Bouncing Ball - Processing Tutorial*, https://www.youtube.com/watch?v=YIKRXl3wH8Y, pristupljeno: 26. 8. 2024.

[18] Amadine Team, *RGB vs. CMYK: What's the Difference and Which One Should You Use?*, https://amadine.com/useful-articles/rgb-vs-cmyk, pristupljeno: 30. 8. 2024.

[19] FMSLogo Development Team, *FMS Logo Tutorial*, https://fmslogo.sourceforge.io/manual/tutorial.html, 2023, pristupljeno: 5. 10. 2024.

[20] Daniel Shiffman The Coding Train, *Loop vs. Draw - Processing Tutorial*, https://www.youtube.com/watch?v=Z8s-7beNP1c, pristupljeno: 26. 8. 2024.

[21] Daniel Shiffman The Coding Train, *Random Color Grid with P5.js*, https://www.youtube.com/watch?v=H7frvcAHXps, pristupljeno: 26. 8. 2024.

[22] The Processing Foundation, *ellipse() - Processing Reference*, https://processing.org/reference/ellipse_.html, pristupljeno: 2. 9. 2024.

[23] W3Schools.com, *RGB Colors - W3Schools*, https://www.w3schools.com/colors/colors_rgb.asp, pristupljeno: 2. 9. 2024.

Sažetak

U ovom diplomskom radu opisan je alternativan način pristupa programiranju u osnovnoj i srednjoj školi. Naime, Python Mode for Processing idealan je alat za nastavnika koji želi spojiti ono najbolje što nudi FMS Logo i Python te nastavu informatike učiniti kreativnom, dinamičnom i korisnom za učenika. Nude se i brojne mogućnosti međupredmetnih tema vezanih za matematiku i za digitalne sadržaje. Opisane su brojne aktivnosti, njihovi očekivani ishodi, potencijalne učeničke poteškoće te je za svaku navedena izazovnija verzija za učenike koji početne brzo savladaju.

Summary

This thesis describes an alternative approach to programming in primary and secondary schools. Python Mode for Processing is an ideal tool for a teacher who wants to combine the best of what FMS Logo and Python have to offer, and we can use it to make informatics classes creative, dynamic and useful for students. Numerous opportunities for cross-curricular topics related to mathematics and digital content are also offered. Many activities, their expected outcomes and potential student difficulties are described, and a more challenging version is listed for each one, for students who master the initial ones quickly.

Životopis

Rođena sam 18. siječnja 2001. godine u Splitu. Završila sam prirodoslovno-matematički smjer Gimnazije dr. Mate Ujevića u Imotskom, nakon čega sam upisala Prirodoslovno-matematički fakultet u Zagrebu. Po završetku preddiplomskog smjera 2022. godine i stjecanja titule prvostupnika edukacije matematike, upisala sam diplomski studij Matematika i informatika; nastavnički smjer. 2024. godine započela sam rad u dvije srednje škole: V. gimnaziji i Obrtničkoj školi za osobne usluge gdje predajem matematiku i informatiku.