# Dodatak A

# Uvod u Python za studente filozofije i ostalih ne-tehničkih grupa

## A.1 Zašto bi se filozof zanimao za programiranje?

Na prvi pogled, svijet filozofije i svijet programiranja mogu se činiti kao dva potpuno odvojena svemira. Jedan se bavi vječnim pitanjima o smislu, postojanju i vrijednostima, dok se drugi bavi preciznim uputama za strojeve. Međutim, ispod površine, ova dva svijeta dijele duboke i iznenađujuće veze. Logika, temeljni alat filozofske analize, ujedno je i srce svakog računalnog programa. Način na koji strukturiramo argumente, definiramo pojmove i izvodimo zaključke u filozofiji ima svoj odraz u načinu na koji pišemo kod.

Učenje programskog jezika `Python`, stoga, za studenta filozofije nije samo stjecanje tehničke vještine, već i prilika za istraživanje poznatih koncepata iz nove perspektive. Kroz `Python`, apstraktni pojmovi poput varijabli, uvjeta i petlji postaju konkretni alati s kojima možete raditi, eksperimentirati i stvarati.

Ovo poglavlje je osmišljeno kao blagi uvod u `Python`, posebno prilagođen studentima humanističkih i društvenih znanosti. Nećemo se baviti složenim matematičkim problemima niti dubokim tehničkim detaljima. Umjesto toga, fokusirat ćemo se na osnove jezika, koristeći primjere koji su vam bliski: analizu teksta, rad s riječima i rečenicama, te istraživanje ideja kroz kod. Koristit ćemo se **Jupyter bilježnicama**, interaktivnim okruženjem koje omogućuje pisanje koda, teksta i vizualizacija na jednom mjestu, čineći učenje intuitivnim i zabavnim.

Dok budete prolazili kroz ovo poglavlje, potičem vas da ne gledate na kod samo kao na niz naredbi, već kao na novi način izražavanja i strukturiranja misli. Možda ćete otkriti da vam učenje programiranja može pomoći da postanete precizniji u svom filozofskom promišljanju, jasniji u svom izražavanju i kreativniji u svom pristupu problemima. Dobrodošli u svijet `Python`a!

## A.2 Osnovni pojmovi: Varijable, tipovi podataka i izrazi

### A.2.1 Varijable: Imenovanje ideja

U filozofiji, često koristimo simbole ili nazive kako bismo predstavili složene ideje. Na primjer, u logici, slovo $P$ može predstavljati propoziciju "Svi ljudi su smrtni". Na sličan način, u Pythonu koristimo **varijable** kao imenovane spremnike za pohranu podataka.

**Definicija A.1. Varijabla** je imenovani prostor u memoriji koji služi za pohranu vrijednosti. Ime varijable (identifikator) koristimo kako bismo pristupili pohranjenoj vrijednosti.

Varijablu možete zamisliti kao oznaku koju pridružujete nekoj vrijednosti. Operator dodjele, znak jednakosti (=), koristi se za dodjeljivanje vrijednosti varijabli.

Dodjeljivanje vrijednosti varijablama.

```
pozdrav = "Zdravo, svijete!"
godinarodenjakanta = 1724
pipriblizno = 3.14159
```

U ovom primjeru, `pozdrav`, `godinarodenjakanta` i `pipriblizno` su nazivi varijabli. Jednom kada definiramo varijablu, možemo je koristiti u daljnjem kodu, na primjer za ispis njezine vrijednosti pomoću ugrađene funkcije `print()`.

```
print(pozdrav)
print(godinarodenjakanta)
```

Izlaz:

```
Zdravo, svijete!  1724
```

### A.2.2 Tipovi podataka: Različite vrste informacija

U filozofiji, razlikujemo različite vrste pojmova: konkretne, apstraktne, pojedinačne, opće. Slično tome, u Pythonu, svaka vrijednost pripada određenom **tipu podataka**. Osnovni tipovi podataka koje ćemo za početak koristiti su:

- **String (`str`):** Niz znakova, odnosno tekstualni podaci. Stringovi se uvijek pišu unutar navodnika (jednostrukih `'` ili dvostrukih `""`). Primjeri: `"Sokrat"`, `'Platonova Država'`.
- **Integer (`int`):** Cijeli brojevi, bez decimalnog dijela. Primjeri: `42`, `-399`, `2025`.
- **Float (`float`):** Brojevi s pomičnim zarezom (decimalni brojevi). Primjeri: `3.14`, `9.81`, `-0.5`.
- **Boolean (`bool`):** Logička ili istinitosna vrijednost. Može imati samo dvije vrijednosti: `True` (istina) ili `False` (laž).

Python je dinamički tipiziran jezik, što znači da ne moramo unaprijed deklarirati tip varijable. Interpretator automatski prepoznaje tip podatka kada dodijelimo vrijednost. Tip varijable možemo provjeriti pomoću ugrađene funkcije `type()`.

Provjera tipova podataka.

```
filozof = "Aristotel"
    godinarodenja = -384
    visinaumetrima = 1.7
    jeliziv = False

    print(type(filozof))
    print(type(godinarodenja))
    print(type(visinaumetrima))
    print(type(jeliziv))
```

**Izlaz:**

```
<class 'str'> <class 'int'> <class 'float'> <class 'bool'>
```

### A.2.3 Izrazi: Kombiniranje vrijednosti

U logici, kombiniramo propozicije pomoću veznika ($\wedge, \vee, \rightarrow$) kako bismo stvorili složenije izraze. U Pythonu, **izrazi** su kombinacije vrijednosti, varijabli i operatora koje se izračunavaju (evaluiraju) kako bi proizvele novu vrijednost.

- **Aritmetički izrazi:** Koriste standardne matematičke operatore (+, -, *, /).

```
a = 10
b = 5
zbroj = a + b
print(zbroj)
```

**Izlaz:**

```
15
```

- **String izrazi:** Operator + se može koristiti za spajanje (*konkatenaciju*) stringova.

```
ime = "Immanuel"
prezime = "Kant"
punoime = ime + " " + prezime
print(punoime)
```

**Izlaz:**

```
Immanuel Kant
```

## A.3 Strukture podataka: Organiziranje misli

Dok osnovni tipovi podataka predstavljaju pojedinačne vrijednosti, **strukture podataka** služe za organiziranje i pohranu više vrijednosti u jednoj varijabli.

### A.3.1 Liste: Uređeni nizovi argumenata

U filozofskim tekstovima, često nailazimo na nabrajanja ili nizove ideja, poput Aristotelovih četiriju uzroka. U Pythonu, **liste** su strukture podataka koje nam omogućuju pohranu uređenog niza elemenata. Elementi liste se navode unutar uglatih zagrada `[]`, odvojeni zarezima.

**Definicija A.2. Lista** (`list`) je promjenjiva, uređena kolekcija elemenata. "Uređena" znači da elementi zadržavaju redoslijed kojim su dodani. "Promjenjiva" znači da možemo dodavati, uklanjati ili mijenjati elemente nakon što je lista stvorena.

Kreiranje i pristupanje elementima liste.

```
aristoteloviuzroci = ["materijalni", "formalni", "djelatni", "svršni"] #
↪  Popiy Aristotelovih uzroka

# Pristupanje elementima pomoću indeksa
# Indeksiranje počinje od 0!
prviuzrok = aristoteloviuzroci[0]
treciuzrok = aristoteloviuzroci[2]

print("Prvi uzrok je:", prviuzrok)
print("Treći uzrok je:", treciuzrok)
```

Izlaz:

```
Prvi uzrok je:  materijalni Treći uzrok je:  djelatni
```

Liste su promjenjive. Možemo im dodavati nove elemente metodom `append()` ili uklanjati postojeće metodom `remove()`.

```
aristoteloviuzroci.append("imaginarni") # Dodavanje petog, "imaginarnog"
↪  uzroka
print(aristoteloviuzroci)

# Uklanjanje "imaginarnog" uzroka
aristoteloviuzroci.remove("imaginarni")
print(aristoteloviuzroci)
```

Izlaz:

```
['materijalni', 'formalni', 'djelatni', 'svršni', 'imaginarni'] ['materijalni',
'formalni', 'djelatni', 'svršni']
```

### A.3.2 Rječnici: Asocijativni parovi pojmova i definicija

U filozofiji, često definiramo pojmove tako da im pridružujemo njihove definicije. U Pythonu, **rječnici** (`dict`) omogućuju pohranu podataka u obliku parova **ključ-vrijednost**. Ključevi su jedinstveni i koriste se za pristup pripadajućim vrijednostima.

**Definicija A.3. Rječnik** (`dict`) je promjenjiva, neuređena kolekcija parova ključ-vrijednost. Svaki ključ mora biti jedinstven unutar rječnika.

Rječnici se definiraju unutar vitičastih zagrada , a parovi ključ-vrijednost odvojeni su dvotočkom.

Kreiranje i korištenje rječnika.

```
filozofskirjecnik = {
    "epistemologija": "grana filozofije koja se bavi znanjem",
    "metafizika": "grana filozofije koja se bavi prvim uzrocima i principima
    ↪  bića",
    "etika": "grana filozofije koja se bavi moralom"
}

# Pristupanje vrijednosti pomoću ključa
definicijaetike = filozofskirjecnik["etika"]
print(definicijaetike)

# Dodavanje novog para
filozofskirjecnik["logika"] = "znanost o metodama i principima ispravnog
↪  zaključivanja"
print(filozofskirjecnik["logika"])
```

**Izlaz:**

```
grana filozofije koja se bavi moralom znanost o metodama i principima ispravnog
zaključivanja
```

## A.4 Kontrola toka: Usmjeravanje argumentacije

Programi se ne izvršavaju uvijek linearno, od prve do zadnje naredbe. **Kontrola toka** odnosi se na naredbe koje nam omogućuju da usmjeravamo tijek izvršavanja programa, donosimo odluke i ponavljamo operacije.

### A.4.1 Uvjetno izvršavanje: `if`, `elif`, `else`

U filozofskoj argumentaciji, često koristimo uvjetne rečenice oblika "Ako $P$, onda $Q$". U Pythonu, **uvjetne naredbe** nam omogućuju da izvršimo određeni dio koda samo ako je zadovoljen neki uvjet. Uvjet je izraz koji se evaluira kao `True` ili `False`.

Korištenje `if-else` strukture.

```
tvrdnja = "Sokrat je smrtan"

if "Sokrat" in tvrdnja:
    print("Tvrdnja se odnosi na Sokrata.")
else:
```

```
    print("Tvrdnja se ne odnosi na Sokrata.")
```

Izlaz:

```
Tvrdnja se odnosi na Sokrata.
```

Možemo koristiti i `elif` (skraćeno od *else if*) za provjeru više uzastopnih uvjeta.

```
godina = 1804

if godina < 476:
    print("Antička filozofija")
elif 476 <= godina < 1500:
    print("Srednjovjekovna filozofija")
else:
    print("Moderna i suvremena filozofija")
```

Izlaz:

```
Moderna i suvremena filozofija
```

### A.4.2 Ponavljanje: `for` petlja

Često je potrebno ponoviti istu radnju više puta. Na primjer, analizirati svaku riječ u rečenici. U Pythonu, `for` **petlja** nam omogućuje da iteriramo (prolazimo) kroz elemente sekvence (poput liste ili stringa) i za svaki element izvršimo određeni blok koda.

Iteriranje kroz listu pomoću `for` petlje.

```
stoickevrline = ["mudrost", "pravednost", "hrabrost", "umjerenost"]

print("Prema stoicima, temeljne vrline su:")
for vrlina in stoickevrline:
    print("- " + vrlina)
```

Izlaz:

```
Prema stoicima, temeljne vrline su:  - mudrost - pravednost - hrabrost -
umjerenost
```

U ovom primjeru, varijabla `vrlina` se naziva *varijabla petlje*. U svakom prolasku (iteraciji) kroz petlju, ona poprima vrijednost sljedećeg elementa iz liste `stoickevrline`.

## A.5 Funkcije: Modularizacija i ponovna upotreba misli

U filozofiji, kompleksne ideje često razlažemo na manje, razumljivije dijelove. U Pythonu, **funkcije** nam omogućuju da grupiramo niz naredbi u logičku cjelinu koju možemo pozvati više puta. Time se izbjegava ponavljanje koda i programi postaju organiziraniji i lakši za čitanje.

**Definicija A.4. Funkcija** je imenovani blok koda koji izvršava određeni zadatak. Može primiti ulazne podatke (argumente) i vratiti izlaznu vrijednost.

Funkcije definiramo pomoću ključne riječi `def`.

Definiranje i pozivanje jednostavne funkcije.

```
def pozdravifilozofa(ime):
    """
    Ova funkcija ispisuje pozdrav filozofu čije je ime
    proslijeđeno kao argument.
    """
    print("Pozdrav, " + ime + "!")

# Pozivanje funkcije
pozdravifilozofa("Platon")
pozdravifilozofa("Nietzsche")
```

**Izlaz:**

```
Pozdrav, Platon!  Pozdrav, Nietzsche!
```

Tekst unutar trostrukih navodnika odmah nakon definicije funkcije naziva se *docstring* i služi kao dokumentacija funkcije.

Funkcije mogu vraćati vrijednost pomoću naredbe `return`. Vraćena vrijednost se tada može pohraniti u varijablu ili koristiti u daljnjim izrazima.

Funkcija koja vraća vrijednost.

```
def sastaviime(ime, prezime):
    """Sastavlja puno ime iz dva dijela."""
    return ime + " " + prezime

punoimefilozofa = sastaviime("Simone", "de Beauvoir")
print(punoimefilozofa)
```

**Izlaz:**

```
Simone de Beauvoir
```

## A.6 Primjer iz prakse: Analiza filozofskog teksta

Sada ćemo primijeniti sve što smo naučili na konkretnom primjeru: analizi kratkog filozofskog teksta. Cilj nam je prebrojati koliko se puta svaka riječ pojavljuje u poznatoj Descartesovoj izreci.

Brojanje riječi u tekstu.

```python
tekst = "Mislim, dakle jesam. Jesam, dakle postojim." # Korak 1: Definiramo tekst za analizu
print("Originalni tekst:", tekst)

# Korak 2: Priprema teksta
# Pretvaramo sva slova u mala slova kako 'Mislim' i 'mislim' ne bi bili različite riječi
tekstmali = tekst.lower()
# Uklanjamo interpunkcijske znakove
tekstbeztocke = tekstmali.replace('.', '')
tekstcisti = tekstbeztocke.replace(',', '')
print("Očišćeni tekst:", tekstcisti)

# Korak 3: Tokenizacija - razdvajanje teksta u listu riječi
rijeci = tekstcisti.split()
print("Lista riječi:", rijeci)

# Korak 4: Brojanje riječi pomoću rječnika
brojacrijeci = {}
for rijec in rijeci:
    if rijec in brojacrijeci:
        # Ako riječ već postoji u rječniku, povećaj brojač za 1
        brojacrijeci[rijec] = brojacrijeci[rijec] + 1
    else:
        # Ako je ovo prvo pojavljivanje riječi, dodaj je u rječnik s vrijednošću 1
        brojacrijeci[rijec] = 1

# Korak 5: Ispis rezultata
print("\nFrekvencija riječi:")
for rijec, broj in brojacrijeci.items():
    print(f"'{rijec}': {broj}")
```

**Izlaz:**

```
Originalni tekst: Mislim, dakle jesam. Jesam, dakle postojim. Očišćeni tekst:
mislim dakle jesam jesam dakle postojim Lista riječi: ['mislim', 'dakle',
'jesam', 'jesam', 'dakle', 'postojim']
Frekvencija riječi: 'mislim': 1 'dakle': 2 'jesam': 2 'postojim': 1
```

Ovaj primjer integrira varijable, stringove i njihove metode (`.lower()`, `.replace()`, `.split()`), liste, rječnike, `for` petlju i `if-else` uvjetnu logiku kako bi se riješio konkretan problem iz domene analize teksta.

## Vježbe za poglavlje 1

*Vježba* A.1. Kreirajte rječnik koji sadrži pet vaših omiljenih filozofa kao ključeve, a njihove glavne filozofske ideje ili djela kao vrijednosti. Zatim, koristeći `for` petlju, ispišite svakog filozofa i njegovu ideju u formatu: `Ime Filozofa: Glavna ideja`.

*Vježba* A.2. Napišite funkciju pod nazivom `brojrijeci` koja prima jedan argument (string) i vraća broj riječi u tom stringu. (Savjet: metoda `split()` bi mogla biti korisna). Testirajte funkciju s nekoliko rečenica.

*Vježba* A.3. Napišite program koji provjerava pripada li godina određenom filozofskom raz-

doblju.

1. Definirajte listu koja sadrži nekoliko filozofa egzistencijalizma, npr. `egzistencijalisti = ["Sartre", "Camus", "Kierkegaard"]`.
2. Pitajte korisnika da unese ime filozofa pomoću funkcije `input()`.
3. Koristeći `if` naredbu i operator `in`, provjerite nalazi li se uneseno ime u vašoj listi.
4. Ispišite odgovarajuću poruku, npr. `Sartre je egzistencijalist.` ili `Platon nije egzistencijalist.`.

## A.7 Zaključak: Sljedeći koraci

Ovo poglavlje pružilo je kratak pregled osnovnih elemenata programskog jezika Python. Vidjeli smo kako varijable i tipovi podataka predstavljaju osnovne gradivne blokove, kako strukture podataka poput lista i rječnika organiziraju informacije, kako kontrola toka usmjerava izvršavanje programa i kako funkcije omogućuju modularnost i ponovnu upotrebu koda.

Kao studenti filozofije, sada imate temelj za daljnje istraživanje. Sljedeći koraci mogli bi uključivati:

- **Rad s tekstom:** Python je izuzetno moćan za analizu teksta. Možete istražiti kako brojati riječi, analizirati sentiment, tražiti određene pojmove u velikim tekstualnim korpusima (npr. djelima pojedinih filozofa) i još mnogo toga. Biblioteke poput NLTK (Natural Language Toolkit) i spaCy otvaraju vrata svijeta *računalne lingvistike*.
- **Vizualizacija podataka:** Pomoću biblioteka kao što su Matplotlib i Seaborn, možete vizualizirati odnose između pojmova, učestalost riječi ili druge uvide koje dobijete analizom teksta, pretvarajući apstraktne podatke u jasne grafove.
- **Web scraping:** Možete naučiti kako automatski prikupljati tekstualne podatke s web stranica, na primjer, s filozofskih enciklopedija ili online arhiva.

Najvažnije je da se ne bojite eksperimentirati. Jupyter bilježnice su idealno okruženje za to. Pokušajte mijenjati primjere, postavljati si vlastite male probleme i tražiti rješenja. Programiranje, kao i filozofija, je vještina koja se razvija kroz praksu, znatiželju i upornost. Sretno s kodiranjem!