Programiranje za umetnike 1

~ 8 ~

Staša Vujičić Stanković

Kontrolne strukture

- Kontrolne strukture se koriste za kontrolu toka programa.

  Ako su ispunjeni određeni uslovi onda se izvršavaju njima odgovarajući delovi programa.

Kontrolne strukture

Uslovna grananja

- Omogućavaju izbor jednog od većeg broja ishoda.

Petlje (ciklusi)

- Omogućavaju ponavljanje zadatih instrukcija
  - Određen broj puta (brojačke petlje)
  - Dok je određeni uslov ispunjen

Logički uslovi

Uslovi se često definišu korišćenjem logičkih tipova (`bool` – `True`, `False`)

Logička neistina	Logička istina
• Vrednosti `False` i `None` • Vrednost `0` bilo kog numeričkog tipa: `0`, `0.0`, `0j` • Prazne sekvence i kolekcije	• Sve druge vrednosti se smatraju logičkom istinom

Uslovna grananja

- Uslovna grananja omogućavaju izvršavanje bloka koda pod uslovom da je neki logički uslov ispunjen

```
if uslov:
    telo_then_grane # sekvence naredbi koje predstavljaju blok koda
                    # koji se izvršava ako je uslov ispunjen
else:
    telo_else_grane # sekvence naredbi koje predstavljaju blok koda
                    # koji se izvršava ako uslov nije ispunjen
```

- Navođenje `else` grane nije obavezno

```
if a > b: print("a je vece od b")
```

Uslovna grananja

```
if uslov:
    telo_then_grane # sekvence naredbi koje predstavljaju blok koda
                    # koji se izvršava ako je uslov ispunjen
else:
    telo_else_grane # sekvence naredbi koje predstavljaju blok koda
                    # koji se izvršava ako uslov nije ispunjen
```

- Mora biti izraz logičkog tipa!
  - Mora se evaluirati u vrednost `True` ili `False`
  - Formira se korišćenjem relacionih i logičkih operatora,
    ili kao rezultat poziva funkcije
- Obavezno je navođenje dvotačke. Njome se zadaje početak novog bloka koda!

Blok koda

Ako se sastoji od samo jedne naredbe

ona se može navesti u produžetku iza dvotačke

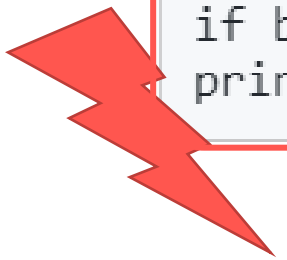
Ako ima više naredbi u bloku

svaka naredba mora da bude u novom redu
pri čemu se mora voditi računa da svaka bude
podjednako uvučena u odnosu na if else
da bi se smatrala delom istog bloka

Uslovna grananja

```
a = 33
b = 200
if b > a:
  print("b je vece od a")
```

```
a = 33
b = 200
if b > a:
print("b je vece od a")
```



Višestruko uslovno grananje

ključna reč `elif`

ako prethodni uslovi nisu ispunjeni,
pokušaj sa uslovom navedenim uz elif

ključna reč `else`

ako se navede bez uslova iza nje,
onda se (ukoliko ni jedan uslov pre nje nije izvršen)
izvršava blok koda koji sledi za njom

Višestruko uslovno grananje

```
a = 33
b = 33
if b > a:
  print("b je vece od a")
elif a == b:
  print("a i b su jednaki")
```

```
a = 233
b = 33
if b > a:
  print("b je vece od a")
elif a == b:
  print("a i b su jednaki")
else:
  print("a je vece od b")
```

Uslovni izrazi

- Skraćeno pisanje uslovnih grananja
- Omogućavaju dodelu vrednosti promenljivoj ili ispis u zavisnosti od nekog uslova
- Ima najmanji prioritet.

```
A if USLOV else B
```

- Evaluira se USLOV
  - Ako je USLOV tačan izračunava se i vraća vrednost izraza A
  - Inače se izračunava se i vraća vrednost izraza B

Uslovni izrazi

```
a = 233
b = 33
print("A je veće") if a > b else print("B je veće")
```

```
a = 33
b = 33
print("A je veće") if a > b else print("A je jednako B") if a == b else print("B je veće")
```

Višestruko uslovno grananje

ugnježdeni if - else

- Može se koristiti i sintaksa u kojoj se navodi više if izraza u okviru jednog – tzv. ugnježdavanje.

```
x = 50

if x % 2 == 0:
  print("Deljiv sa 2")
  if x % 5 == 0:
    print("i sa 5!")
  else:
    print("ali nije sa 5.")
```

Iterativne kontrolne strukture

- Iterativne kontrolne strukture (petlje ili ciklusi) se koriste kada je potrebno neku obradu izvršiti veći broj puta
- Ponavljanja mogu da se vrše:
  - unapred zadati broj puta
  - dok ne bude zadovoljen određeni uslov
  - za svaki element kolekcije (niza ili liste)

Iterativne kontrolne strukture

- tipovi -

WHILE PETLJA

- Za iteriranje do zadovoljenja nekog opšteg uslova

FOR PETLJA

- Za iteriranje u određenom opsegu
- Za iteriranje kroz kolekcije

WHILE petlja

- Blok naredbi se ponavlja dok je ispunjen zadati logički uslov

1. Provera uslova se radi pre izvršavanja tela petlje
2. Ciklus koji se izvršava 0 ili više puta (sa izlaskom na vrhu)

WHILE petlja

```
while uslov:
    telo_petlje # sekvence naredbi koje predstavljaju blok koda
                # koji se izvršava dok je uslov ispunjen
# deo koda nakon while petlje
```

```
i = 2
while i < 6:
  print(i)
  i += 1
print("Kraj")
```

Šta se dogodi ukoliko zaboravimo da povećamo brojač `i`?

Kontrola izvršavanja petlje

- Koristi se kada postoji potreba da petlja ranije završi svoje izvršavanje

  Npr. u slučaju zadovoljavanja nekog spoljašnjeg uslova,
  moguće je izvršiti izlazak iz petlje nezavisno od ispunjenja uslova

break naredba

- Naredbom `break` se prekida petlja iako je uslov i dalje tačan:

```
i = 1
while i < 10:
  print(i)
  if i == 5:
    break
  i += 1

print("Kraj")
```

- Izvršavanje se nastavlja prvom sledećom naredbom iza while petlje (ako postoji).

continue naredba

- Naredbom `continue` se zaustavlja trenutna iteracija i nastavlja se sa sledećom:

```
i = 1
while i < 10:
  i += 1
  if i == 5:
    continue
  print(i)

print("Kraj")
```

Preskaču se sve naredbe do kraja bloka u okviru koga je pozvana i nastavlja izvršavanje naredne iteracije petlje

else naredba

- Naredbom `else` se izvršava deo koda u slučaju kada je uslov neistinit:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("Kraj")
```

```
i = 1
while i < 6:
  print(i)
  i += 1
  if i==5:
    break

else:
  print("Kraj")

print("Posle kraja")
```

- else se neće izvršiti ukoliko je petlja prekinuta upotrebom naredbe break.

WHILE petlja

sa izlazom na dnu

```
while True
  # telo_petlje
  if uslov:
    break
# naredbe posle petlje
```

- Obavezno se izvrši jednom
- Nakon izvršavanja tela petlje proverava se uslov izlaska ili ostanka u petlji

WHILE petlja

sa izlazom na dnu

```
# Sabrati sve brojeve deljive sa 5
# unete sa standardnog ulaza dok korisnik ne unese 0

zbir = 0
while True:
  broj = int(input("Unesite broj: "))
  if broj == 0:
    break
  elif broj % 5 != 0:
    continue
  zbir += broj
print("zbir = ",  zbir)
```

pass naredba

- Kreira se „prazno mesto“ za budući kod.
- Najčešće se koristi za deo koda koji će biti implementiran kasnije
- Izvršava se, ali se ništa ne dešava
- Koristi se u slučajevima kada je potrebno staviti praznu naredbu, a to nije sintaksno ispravno.

neki primeri

FOR petlja

- Ponavlja blok naredbi
  - Određen, unapred zadati, broj puta (koji se nalazi u nekom opsegu)
  - Za svaki element kolekcije

```
for iterator in opseg/kolekcija
  telo_for_petlje
```

- Promenljiva `iterator` uzima redom vrednosti iz opsega ili kolekcije
- Za svaku vrednost se obavlja jedna iteracija

FOR petlja

- Generisanje niza brojeva za iteriranje se vrši funkcijom `range()`

```
# range generise niz u opsegu [start, stop)
# Donja granica se ukljucuje, gornja ne!

# Ako se ne zada start, smatra se da se krece od 0
range(stop)

# step je opciono.
# Njime se definise korak koji mora da bude razlicit od 0
# Ako je step negativno iterira se unazad
range(start, stop, [step])
```

FOR petlja

```
for x in range(5):
  print(x)
```

```
for x in range(10):
  print(x)
```

```
for x in range(1, 30, 5):
  print(x)
```

FOR petlja

```
# Ispisati sumu svih brojeva deljivih sa 5
# u zadatom opsegu od 1 do n kojeg korisnik unosi sa standardnog ulaza

n = int(input("Unesite ceo broj n veci od 1: "))
zbir = 0

for i in range(1, n+1):
  zbir += i if i % 5 == 0 else 0

print (zbir)
```

FOR petlja

```
# Ispisati sve parne brojeve
# u zadatom opsegu od 1 do n kojeg korisnik unosi sa standardnog ulaza

n = int(input("Unesite ceo broj n veci od 1: "))

for i in range(2, n+1, 2):
  print (i, end= ' ')
```

Literatura

- Python 3.10.0 documentation
- Wentworth, Peter, Elkner, Jeffrey, Downey, Allen B. and Meyers, Chris. How to Think Like a Computer Scientist: Learning with Python 3. free online book
- Lutz, Mark. Learning python: Powerful object-oriented programming. O'Reilly Media, Inc., 2013.
- Beazley, David, and Jones, Brian. Python Cookbook: Recipes for Mastering Python 3. O'Reilly Media, Inc., 2013.
- Python Cheatsheet
- Website Setup Python cheat sheet
- Learn Python, basic tutorial

Hvala

Staša Vujičić Stanković

stasa.vujicic.stankovic@math.rs

www.matf.bg.ac.rs/~stasa.vujicic.stankovic