Elementi Python programa

Slajdovi za predmet Osnove programiranja

Katedra za informatiku, Fakultet tehničkih nauka, Novi Sad

2020.

Ciljevi

- razumevanje i pisanje Python naredbi za ispis podataka
- dodela vrednosti promenljivama
- unos numeričkih podataka sa tastature
- pisanje konačnih petlji

Proces pisanja programa

- proces pisanja programa je često podeljen u faze/delove prema podacima koji se proizvode u svakoj fazi

Proces pisanja programa 2

- **analiza problema**
  tačno ustanoviti problem koji treba rešiti; razumeti ga što je bolje moguće

Proces pisanja programa 3

- **pravljenje specifikacije**
  tačno opisati ono što program treba da radi
  - fokus nije na tome kako program radi, nego šta treba da radi
  - uključuje opis ulaznih i izlaznih podataka i veza između njih

Proces pisanja programa 4

- **pravljenje dizajna**
  - formulisanje globalne strukture programa
  - ovde se određuje kako program radi
  - izbor postojećeg ili pravljenje novog algoritma koji odgovara specifikaciji

Proces pisanja programa 5

- **pisanje programa** (implementacija dizajna)
  - pretakanje dizajna u program pisan u nekom programskom jeziku
  - u ovom predmetu koristićemo Python

Proces pisanja programa 6

- **testiranje programa i ispravljanje grešaka** (debugging)
  - pokretanje programa sa ciljem provere ispravnosti
  - ako ima grešaka (bug), potrebno ih je pronaći i ispraviti
  - cilj je pronaći greške → treba probati sve što može „pokvariti“ rad programa

Proces pisanja programa 7

- **održavanje programa**
  - nastavak razvoja programa prema novim potrebama korisnika
  - u praksi, većina programa nikad nije završena – oni se menjaju (evoluiraju) vremenom

Primer programa: konverzija temperature

- analiza
  - za temperaturu datu u stepenima Celzijusa, izračunati je u stepenima Farenhajta
- specifikacija
  - ulaz: temperatura u stepenima Celzijusa
  - izlaz: temperatura u stepenima Farenhajta
  - $F = \frac{9}{5}C + 32$

Primer programa: konverzija temperature $_2$

- dizajn: ulaz, obrada, izlaz
  - zatražiti od korisnika ulazne podatke (temperaturu u °C)
  - izračunavanje temperature u °F
  - ispis rezultata na ekran

Primer programa: konverzija temperature $_3$

- pre kodiranja, napišimo skicu programa u pseudokodu
- pseudokod je precizan tekst (prirodni jezik) koji opisuje šta program radi, korak-po-korak
- pomoću pseudokoda možemo se koncentrisati na algoritam, a ne na konkretan programski jezik

Primer programa: konverzija temperature 4

- pseudokod:
  1. unos `celsius`
  2. izračunaj `fahrenheit` kao `(9/5)*celsius+32`
  3. ispis `fahrenheit`
- sada treba napisati ovo u Pythonu

Primer programa: konverzija temperature 5

```
# convert.py
# Konvertuje temperaturu Celzijus -> Farenhajt

celsius = eval(input("Unesite temperaturu C >> "))
fahrenheit = 9.0/5 * celsius + 32
print("Temperatura je", fahrenheit, "stepeni Farenhajta.")
```

Primer programa: konverzija temperature $_6$

- kada napišemo program, treba ga testirati

```
$ python3 convert.py
Unesite temperaturu u C >> 0
Temperatura je  32.0  stepeni Farenhajta.
$ python3 convert.py
Unesite temperaturu u C >> 100
Temperatura je  212.0  stepeni Farenhajta.
$ python3 convert.py
Unesite temperaturu u C >> -40
Temperatura je  -40.0  stepeni Farenhajta.
$
```

Imena

- imena se daju promenljivama (`celsius`, `fahrenheit`), funkcijama (`main`), modulima (`convert`), itd.
- ova imena se zovu identifikatori
- svaki identifikator počinje slovom ili donjom crtom („_“), i nastavlja se bilo kojim nizom slova, cifara i donjih crta
- razlikuju se velika i mala slova (case-sensitive)

Primeri imena

- ovo su različita ispravna imena
  - `X`
  - `Celsius`
  - `Spam`
  - `spam`
  - `spAm`
  - `Spam_And_Eggs`
  - `Spam_and_Eggs`

Rezervisane reči

- neki identifikatori su već deo Pythona – rezervisane reči
- ne možemo ih koristiti kao imena za svoje promenljive
- `and`, `del`, `for`, `raise`, `assert`, `print`, itd.

Izrazi

- delovi koda koji izračunavaju nove vrednosti zovu se izrazi
- literali predstavljaju pojedine konkretne vrednosti, npr. `3.9`, `1`, `1.0`
- identifikatori se tretiraju kao prosti izrazi

NameError

```
>>> x = 5
>>> x
5
>>> print(x)
5
>>> print(spam)

Traceback (most recent call last):
  File "<pyshell#15>", line 1, in -toplevel-
    print spam
NameError: name 'spam' is not defined
>>>
```

- `NameError` je greška koju pravimo kada koristimo promenljivu kojoj nismo prethodno dodelili vrednost

Složeni izrazi

- složene izraze pravimo kombinovanjem drugih izraza pomoću operatora
- npr. aritmetički operatori +, -, *, /, **
- razmaci u izrazu se ignorišu
- prioritet operacija kao u matematici
- npr. `((x1 - x2)/2*n)+(spam/k**3)` $\Leftrightarrow \frac{x1-x2}{2}n + \frac{spam}{k^3}$

Naredbe ispisa

- naredba `print` može da ispiše više izraza odjednom
- više uzastopnih `print` naredbi će ispisivati podatke u više redova teksta
- `print` bez parametara će ispisati prazan red

Primeri print-a

program	ispis
`print(3+4)`	`7`
`print(3,4,3+4)`	`3 4 7`
`print()`	
`print(3,4, end='')`	`3 4 7`
`print(3 + 4)`	`Rezultat je 7`
`print('Rezultat je', 3+4)`	

Dodela vrednosti

- prosta dodela: `<variable> = <expr>`
- `variable` je identifikator, `expr` je izraz
- izraz na desnoj strani znaka = se izračunava i dobijena vrednost se dodeljuje promenljivoj sa imenom na levoj strani znaka =

Dodela vrednosti 2

- `x = 3.9 * x * (1-x)`
- `fahrenheit = 9.0/5 * celsius + 32`
- `x = 5`

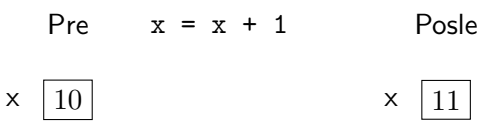
Dodela vrednosti 3

- promenljivoj možemo dodeliti vrednost više puta!

```
>>> myVar = 0
>>> myVar
0
>>> myVar = 7
>>> myVar
7
>>> myVar = myVar + 1
>>> myVar
8
>>>
```

Dodela vrednosti 4

- promenljiva je skadište („kutija“) u koju smeštamo podatak
- kada se promenljiva menja, stara vrednost se briše a nova se upisuje



Dodela vrednosti 5

- zapravo, ovo je pojednostavljeno gledanje
- Python ne briše stare podatke
- dodela vrednosti je više kao stavljanje oznake na neku vrednost, govoreći „ovo je x“

Pre `x = x + 1` Posle

Čitanje ulaza

- svrha `input` funkcije je da očita podatak koji unosi korisnik i smesti ga u promenljivu
- `<variable> = eval(input(<prompt>))`

Čitanje ulaza 2

- prvo se ispisuje `prompt`
- čeka se na korisnika da unese vrednost i pritisne enter
- izraz koji je unet se `eval`-uira iz niza znakova u Python vrednost (broj)
- taj broj se dodeljuje promenljivoj

Višestruka dodela

- više vrednosti se može izračunati istovremeno
- `<var>, <var>, ... = <expr>, <expr>, ...`
- izračunaj izraze na desnoj strani i dodeli ih promenljivama na levoj strani

Višestruka dodela $_2$

- `zbir, razlika = x+y, x-y`
- kako možemo zameniti vrednosti promenljivama `x` i `y`?
  - zašto ovo ne radi?

    ```
    x = y
    y = x
    ```
- možemo upotrebiti treću (pomoćnu) promenljivu

Višestruka dodela $_3$

- zamena vrednosti pomoću višestruke dodele je jednostavna u Pythonu:

```
x,y = y,x
```

```
>>> x = 3
>>> y = 4
>>> print(x, y)
3 4
>>> x, y = y, x
>>> print(x, y)
4 3
```

Višestruka dodela 4

- možemo koristiti istu ideju za unos više promenljivih pomoću jednog `input`-a
- vrednosti koje unosimo razdvajamo zarezom

```
def babezabe():
    babe, zabe = eval(input(
        'Unesite broj baba i broj zaba: '))
    print('Narucili ste', babe, 'baba i', zabe, 'zaba.')
```

```
>>> babezabe()
Unesite broj baba i žaba: 3, 2
Narucili ste 3 baba i 2 zaba.
```

Konačna petlja

- konačna petlja izvršava svoje telo određeni broj puta
- kada petlja počne zna se tačno koliko će iteracija biti
- `for <var> in <sequence>:`
  `  <body>`
- telo petlje se određuje uvlačenjem redova teksta u programu

Konačna petlja $_2$

```
for <var> in <sequence>:
  <body>
```

- promenljiva `var` zove se indeks petlje
- u svakom prolazu petlje uzima narednu vrednost iz date sekvence

Konačna petlja $_3$

```
>>> for i in [0,1,2,3]:
      print(i)

0
1
2
3
>>> for odd in [1, 3, 5, 7]:
      print(odd*odd)

1
9
25
49
>>>
```

Konačna petlja 4

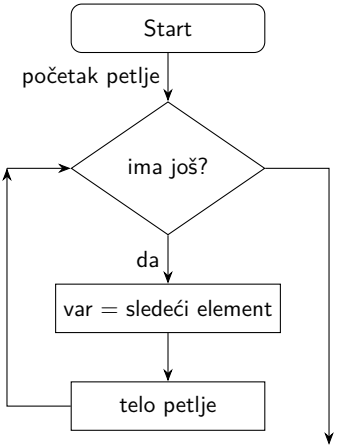
- u `chaos.py` šta predstavlja `range(10)`?

```
>>> range(10)
(0,1,2,3,4,5,6,7,8,9)
>>> list(range(10))
[0,1,2,3,4,5,6,7,8,9]
```

- `range` je Python funkcija koja generiše niz brojeva počevši od 0
- `list` je Python funkcija koja sekvencu konvertuje u listu
- telo petlje se izvršava 10 puta (ima 10 elemenata u sekvenci)

Konačna petlja 5

- `for` petlja menja tok izvršavanja programa
- predstavlja deo struktura za kontrolu toka programa



Primer programa: obračun kamata

- analiza
  - novac položen na račun u banci donosi kamatu
  - koliko će biti novca na računu za 10 godina?
  - ulazi: početno stanje, kamata
  - izlaz: stanje na računu za 10 godina

Primer programa: obračun kamata $_2$

- specifikacija
  - ulazi:
    `principal` – početno stanje na računu
    `apr` – godišnja kamata izražena kao decimalni broj
  - izlaz: stanje na računu za 10 godina
  - veza: stanje na računu posle jedne godine iznosi `principal*(1+apr)`; ovo treba izračunati 10 puta

Primer programa: obračun kamata $_3$

- dizajn
  - ispiši uvodnu poruku
  - unesi početno stanje računa (`principal`)
  - unesi godišnju kamatu (`apr`)
  - ponavljaj 10 puta: `principal = principal*(1+apr)`
  - ispiši stanje računa

Primer programa: obračun kamata 4

- implementacija
  - svaki red specifikacije se preslikava na jedan red u Pythonu (u ovom primeru!)
  - ispiši uvodnu poruku
    ```
    print('Vrednost 10-godišnjeg ulaganja.')
    ```
  - unesi početno stanje računa
    ```
    principal = eval(input('Početni ulog: '))
    ```
  - unesi godišnju kamatu
    ```
    apr = eval(input('Unesite kamatu: '))
    ```
  - ponavljaj 10 puta
    ```
    for i in range(10):
    ```
  - izračunaj principal
    ```
    principal = principal*(1+apr)
    ```
  - ispiši vrednost principal-a nakon 10 godina
    ```
    print('Vrednost iznosi:', principal)
    ```

Primer programa: obračun kamata $_5$

```
# futval.py
# Program racuna vrednost investicije
# 10 godina nakon ulaganja

print("Vrednost 10-godišnjeg ulaganja.")
principal = eval(input("Početni ulog: "))
apr = eval(input("Unesite kamatu: "))
for i in range(10):
    principal = principal * (1 + apr)
print("Vrednost iznosi:", principal)
```

Primer programa: obračun kamata $_6$

```
$ python futval.py
Vrednost 10-godišnjeg ulaganja.
Početni ulog: 100
Unesite kamatu: .03
Vrednost iznosi: 134.391637934
$ python futval.py
Vrednost 10-godišnjeg ulaganja.
Početni ulog: 100
Unesite kamatu: .1
Vrednost iznosi: 259.37424601
```