Katedra za računarstvo

Elektronski fakultet, Univerzitet u Nišu

Veštačka inteligencija

Python

Python - karakteristike

- Python je osmislio Guido van Rossum krajem osamdestih godina prošlog veka.
- To je open-source jezik široke namene;
- Radi na različitim platformama;
- Python je objektno-orijentisani jezik;
- Jako zastupljen u veštačkoj inteligenciji;
- Interpretatorski jezik;
- Jednostavna sintaksa;

Python - karakteristike

- Python je dizajniran da ima laku čitljivost koda.
- Naredbe se završavaju novim redom.
- Koristi se uvlačenje koda/razmaci za definisanje blokova koda poput opsega petlji, funkcija i klasa. Broj razmaka koji se koristi za definisanje bloka mora da bude minimum 1.

  npr.
```
if 2 > 1:
        print("Dva je veće od jedan!")
```
- Koristi # za označavanje komentara

  npr.
```
# Ovo je komentar.
```

Promenljive

- Promenljive se kreiraju onog trenutka kada im se dodeli vrednost.
- Promenljive ne moraju da se deklarišu određenim tipom podatka i mogu da promene tip nakon što su postavljene.
- Python razlikuje velika i mala slova u imenima promenljivih. Naziv promenljive može da sadrži slova, brojeve i donju crtu (“_”).
- Naziv promenljive mora početi slovom ili donjom crtom (“_”), ne može da počne brojem.

- Svi podaci se tretiraju kao objekti i imaju svoj tip i identifikator.
- Funkcija type(obj) vraća tip podatka.
- Funckija id(obj) vraća identifikator.

```
x = 10
print(type(x))
-> <class 'int'>
id(x)
-> 1491169575280
```

```
y = 'tekst'
print(type(y))
-> <class 'str'>
id(y)
-> 1491180423152
```

Osnovni tipovi podataka

- Logički: bool
- Numerički: int, float, complex
- Tekstualni: str
- list,
- tuple,
- dict,
- set

Logički tip - bool

- { False, True }

```
print(10 > 9)                  -> True
print(10 == 9)                 -> False
```

- Funkcija bool() omogućava evaluaciju vrednosti

```
bool("abc")                    -> True
bool(123)                      -> True
bool(["o1", "o2", "o3"])       -> True
bool(False)                    -> False
bool(None)                     -> False
bool(0)                        -> False
bool("")                       -> False
bool(())                       -> False
bool([])                       -> False
bool({})                       -> False
```

Numerički podaci

- Celobrojni – int

```
x = 5
```

- Realni brojevi – float

```
y = 3.5
```

- Kompleksni brojevi

```
z = 3+5j
```

- Konvertovanje iz jednog tipa u drugi

```
float(3)    -> 3.0
int(3.9)    -> 3
```

Aritmetičke operacije

- `i+j  - sabiranje`
- `i-j  - oduzimanje`
- `i*j  - množenje`
- `i/j  - deljenje, rezultat realan broj`
- `i%j  - ostatak pri deljenju`
- `i//j – deljenje sa zaokruživanjem razultata na najbliži ceo broj`
- `i**j – stepen broja i na j-ti stepen`

Aritmetičke operacije - primeri

- `print(5 / 2)` -> 2.5
- `print(5 % 2)` -> 1
- `print(5 // 2)` -> 2
- `print(2 + 3)` -> 5
- `print(3 ** 2)` -> 9
- `print(5 + 1.5)` -> 6.5
- `type(5 + 1.5)` -> <class 'float'>
- `type(2 + 3)` -> <class 'int'>
- `type(6/2)` -> <class 'float'>

Relacioni i logički operatori

- Python podržava standardne operatore poređenja vrednosti
- `i > j`
- `i >= j`
- `i < j`
- `i <= j`
- `i == j`
- `i != j`
- Logički operatori nad *bool* vrednostima
- `not a`
- `a and b`
- `a or b`

Tekstualni podaci - str

- Stringovi su nepromenljivi tip podatka.
- Tekstualni podatak se označava jednostrukim, dvostrukim i trostrukim znakom navoda.
- Uglaste zagrade se koriste za pristup nekom elementu stringa. Prvi karakter ima indeks 0.

```
x = 'ovo je string'
y = "Ovo je string"
z = '''I ovo je string'''

type(x)                 -> <class 'str'>
print(x)                -> ovo je string
a = "Hello, World!"
print(a[1])             -> e
```

Osnovne funkcije za rad sa tekstualnim podacima

```
a = "Hello, World!"
```

- len(str) – vraća dužinu stringa *str*

```
print(len(a))                    -> 13
```

- in – provera da li se neki karakter ili reč nalazi u stringu

```
print("hello" in a)              -> False
print("Hello" in a)              -> True
print("hello" not in a)          -> True
```

Osnovne funkcije za rad sa tekstualnim podacima

```
a = "Hello, World!"
```

- a[start : stop]– izvlačenje dela stringa od pozicije *start* do pozicije *stop*

```
print(a[2:5])    -> llo
```

- a[: stop]– izvlačenje dela stringa od početka stringa do pozicije *stop*

```
print(a[:5])     -> Hello
```

- a[start :]– izvlačenje dela stringa od pozicije *start* do kraja

```
print(a[7:])     -> World!
```

- Negativni indeksi se koriste za uzimanje dela stringa počevši od kraja

```
print(a[-5:-2]) -> orl -
```
na poziciji -2 nalazi se d i ne ulazi u rezultat

Osnovne funkcije za rad sa tekstualnim podacima

```
a = "Hello, World!"
```

- `upper()` – prevođenje u sva velika slova

```
print(a.upper())     -> HELLO, WORLD!
```

- `lower()` – prevođenje u sva mala slova

```
print(a.lower())     -> hello, world!
```

- `replace(a,b)` – menja string *a* stringom *b* u okviru jednog objekta

```
print(a.replace("W", "w")) ->Hello, world!
```

- `split(sep)` – kreira listu stringova na osnovu separatora *sep*

```
print(a.split(","))   -> ['Hello', ' World!']
```

Osnovne funkcije za rad sa tekstualnim podacima

- Konkatenacija stringova

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)                -> Hello World
print(a*2)              -> HelloHello
d = a + 2               -> vraća grešku
d = "ona ima {} godina".format(10)
print(d)                -> ona ima 10 godina
d = "%s %d" % (a,2)
print(d)                -> Hello 2
print(f'{"Ona"} ima {10} godina') -> Ona ima 10 godina
```

Liste

- Liste su promenljivi tip podatka.
- Elementi liste ne moraju biti istog tipa.

```
list1 = ["jabuka", "banana", "kruska"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
list4 = ["abc", 34, True, 40, [1, 5]]
```

- Pristup elementima liste po indeksu:

```
print(list1[1])        -> banana
print(list1[-1])       -> kruska
print(list4[1:4])      -> [34, True, 40]
```

Liste

```
list1 = ["jabuka", "banana", "kruska"]
```

- Provera da li je objekat u listi:

```
if "jabuka" in list1:
    print("Da")          -> Da
```

- Promena elementa u listi:

```
list1[1] = "mango"
print(list1)             -> ['jabuka', 'mango', 'kruska']
```

Liste

```
list1 = ["jabuka", "banana", "kruska"]
```

- list.append(obj) – dodaje element na kraj liste

```
list1.append("breskva")     -> ['jabuka', 'banana', 'kruska', 'breskva']
```

- list.insert(ind,obj) – dodaje element na određeno mesto u listi

```
list1.insert(1, "breskva") -> ['jabuka', 'breskva', 'banana', 'kruska']
```

- list.remove(obj) – briše element iz liste

```
list1.remove("breskva")     -> ['jabuka', 'banana', 'kruska']
```

- list.pop(ind) – briše element iz liste na poziciji *ind.* Ukoliko indeks nije naveden briše se poslednje element liste.

```
list1.pop(1)                -> ['jabuka', 'kruska']
```

- list.clear() – briše sve elemente iz liste

```
list1.clear()               -> []
```

Liste

```
list1 = ["jabuka", "mango", "banana", "kruska"]
```

- list.sort() – sortira elemente liste u rastući redosled

```
list1.sort()     -> ['banana', 'jabuka', 'kruska', 'mango']
list1.sort(reverse = True) -> ['mango', 'kruska', 'jabuka', 'banana']
```

- list.reverse() – okreće redosled elemenata u listi.

```
list1.reverse()  -> ['kruska', 'banana', 'mango', 'jabuka']
```

- print(lista) – štampa celu listu

```
print(list1)     -> ['jabuka', 'mango', 'banana', 'kruska']
```

- Kopiranje liste:

```
list5 = list1.copy()
list5 = list(list1)
List5 = list1 - Samo dodeljuje referencu na objekat, ne kopira listu
```

Tuple

- Tuple je uređeni nepromenljivi tip podatka. Nakon kreiranja ne može da se menja, ne mogu da mu se dodaju novi elementi ili da se brišu postojeći.
- Elementi su indeksirani. Prvi element ima indeks 0.
- Tuple može da sadrži podatke različitog tipa.
- Ukoliko tuple sadrži elemente promenljivog tipa, ti elementi mogu da se menjaju.

```
tuple1 = ("jabuka", "mango", "banana", "kruska")
tuple2 = ("jabuka", [1,2,3], "banana", "kruska")
tuple3 = (12, 13, 14)
tuple4 = ("jabuka",) – kreiranje tuple-a sa jednim elementom
nottuple = ("jabuka") – ovo je string
```

Tuple

```
tuple2 = ("jabuka", [1,2,3], "banana", "kruska")
```

- Pristup elementima:

```
print(tuple2[2])          -> banana
```

- Broj elemenata:

```
print(len(tuple2))        -> 4


tuple2[0]="breskva"       – vraća grešku – elementi ne mogu da se menjaju
tuple2[1][0] = 5
print(tuple2)             -> ('jabuka', [5, 2, 3], 'banana', 'kruska')
```

- Može da se radi spajanje 2 tuple-a:

```
y = ("breskva",)
tuple2 += y      -> ('jabuka', [5, 2, 3], 'banana', 'kruska', 'breskva')
```

Dictionary

- Dictionary je promenljivi tip podatka.
- Služi za pamćenje key-value podataka.
- Nije moguće dupliranje ključeva.
- primer:

```
dict1 = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964,
    "colors": ["red", "white", "blue"]
}
```

Dictionary

- Čitanje svih ključeva:

```
x = dict1.keys()
print(x) -> dict_keys(['brand', 'model', 'year', 'colors'])
```

- Čitanje svih vrednosti:

```
y = dict1.values()
print(y) -> dict_values(['Ford', Mustang, 1964, ['red', 'white', 'blue']])
```

- Prikaz svih elemenata:

```
z = dict1.items()
print(z) -> dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year',
1964)])
```

Dictionary

- Pristup određenom elementu na osnovu ključa:

```
x = dict1["model"]          -> Mustang
x = dict1.get("model")      -> Mustang
```

- Broj elemenata se dobija pozivom funkcije len(dict):

```
len(dict1)     -> 4
```

- Kopiranje se vrši pozivom funkcije copy():

```
dict1.copy()
```

- Brisanje svih podataka vrši se pozivom funkcije clear():

```
dict1.clear() -> {}
```

- Brisanje dictionary-a vrši se pozivom funkcije del(dict):

```
del(dict1)
```

Set

- Set-ovi su neuređeni tip podatka koji ne dozvoljavaju dupliranje vrednosti.
- Neuređeni znači da elemeti nemaju definisan redosled i elementima ne može da se pristupi na osnovu indeksa.
- Nakon kreiranja set-a elementi ne mogu da se menjaju ali mogu da se dodaju novi elementi i brišu postojeći.

```
set1 = {"jabuka", "banana", "kruska"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}
set4 = {"abc", 34, True, 40}
print(type(set1)) -> <class 'set'>
```

Set

- Pristup podacima u Set-u:

```
for x in set1:
    print(x)
```

```
jabuka
banana
kruska
```

- Provera da li je vrednost prisutna u Set-u:

```
print("banana" in set1)
```

```
True
```

Set

```
set1 = {"jabuka", "banana", "kruska"}
set2 = {1,2,3}
```

- Dodavanje elemenata u Set:

```
set1.add("breskva")
print(set1) -> {'kruska', 'breskva', 'banana', 'jabuka'}
```

- Dodavanje jednog Set-a u drugi Set:

```
set1.update(set2)
print(set1) -> {'kruska', 1, 'breskva', 2, 3, 'jabuka', 'banana'}
```

Set

```
set1 = {1,2,3,4,5}
```

- Brisanje elementa iz Set-a:

```
set1.remove(3)
print(set1) -> {1,2,4,5}
```

- Brisanje svih elemenata u Set-u:

```
set1.clear()
print(set1) -> set()
```

- Brisanje celog Set-a:

```
del set1
print(set1) -> NameError: name 'set1' is not defined
```

Grananje – If...else

- Uslovno grananje – logički izrazi

kreiraju se pomoću relacija poređenja i logičkih operacija and, or i not.

```
if uslov_1:
    blok_1
elif uslov_2:
    blok_2
elif uslov_3:
    blok_3
 ...
else:
    blok_n
```

- Primeri:

```
a = 5
b = 7
if b > a:
    print("b je veće od a")
elif a == b:
    print("a i b su jednaki")
else:
    print("a je veće od b")

a = 10
b = 7
c = 5
if a > b and b > c:
    print("Poredak je dobar")
```

Petlje

- Python podržava *for* i *while* petlje.
- *for* petlja
- Štampanje svih elemenata liste

```
lista= ["o1", "o2", "o3"]
for x in lista:
    print(x)
```

```
o1
o2
o3
```

For petlje

- Prolazak kroz string
  - Predstavlja prolazak kroz sekvencu karaktera

```
for x in "tekst":
    print(x)
```

```
t
e
k
s
t
```

For petlja

- Prekidanje izvršenja petlje
- *Break*

```
lista= ["o1","o2","o3"]
for x in lista:
    print(x)
    if x == "o2":
      break
```

```
o1
o2
```

- Prekid trenutne iteracije
- *Continue*

```
lista= ["o1","o2","o3"]
for x in lista:
    if x == "o2":
      continue
    print(x)
```

```
o1
o3
```

For petlja

- Za prolazak kroz petlju određeni broj puta koristi se range()
- range(num) - vraća sekvencu *num* brojeva. Sekvenca podrazumevano počinje od 0, povećava se za 1 i završava kada dođe do navedenog broja.
- range(start,kraj,korak) - vraća sekvencu brojeva počevši od *start* broja, do broja *kraj* (broj *kraj* nije uključen u sekvencu) sa korakom *korak*.

```
for x in range(4):
    print(x)
```

```
0
1
2
3
```

```
for x in range(3, 12, 3):
    print(x)
```

```
3
6
9
```

For petlja

- Korišćenje *else* u okviru *for* petlje
  - Predstavlja blok koji će biti izvršen kada se petlja završi.
  - Ukoliko je petlja prekinuta izvršenjem naredbe *break, else* blok se ne izvršava.

```
for x in range(4):
    print(x)
else:
    print("kraj!")
```

```
0
1
2
3
kraj!
```

```
for x in range(4):
    if x == 3:
        break
    print(x)
else:
    print("kraj!")
```

```
0
1
2
```

While petlja

- *while* petlja

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

```
1
2
3
4
5
```

While petlja

- Prekidanje izvršenja petlje
- *Break*

```
i = 1
while i < 4:
    print(i)
    if i == 2:
        break
    i += 1
```

```
1
2
```

- Prekid trenutne iteracije
- *Continue*

```
i = 0
while i < 4:
    i += 1
    if i == 2:
        continue
    print(i)
```

```
1
3
4
```

While petlja

- Korišćenje *else* u okviru *while* petlje
  - Predstavlja blok koji će biti izvršen kada se petlja završi
  - Ukoliko je petlja prekinuta izvršenjem naredbe *break, else* blok se ne izvršava.

```
i = 1
while i < 4:
    print(i)
    i += 1
else:
    print("i nije manje od 4")
```

```
1
2
3
i nije manje od 4
```

```
i = 0
while i < 4:
    if i == 2:
        break
    print(i)
    i += 1
else:
    print("i nije manje od 4")
```

```
0
1
```

Funkcije

- U Python-u funkcije se definišu korišćenjem ključne reči def.

```
def func1():
    print("Hello world!")

func1() -> Hello world!
```

```
def hello(param1='hello',param2='world'):
   print('%s, %s!' % (param1, param2))

hello()                    -> hello, world!
hello('Hi')                -> Hi, world!
hello('Nice','day')        -> Nice, day!
hello(param2='everyone') -> hello, everyone!
```

Funkcije

- Primer funkcije koja štampa sve elemente liste.

```
def stampa(lista):
    for x in lista:
        print(x)

fruits = ["jabuka", "banana", "kruska"]
stampa(fruits)
```

```
jabuka
banana
kruska
```

Funkcije

- Primer funkcije koja vraća proizvod dva broja uneta od strane korisnika.

```
def proizvod(x, y):
    return x * y

valX = input("Unesite vrednost za x: ") -> Unesite vrednost za x: 5
valY = input("Unesite vrednost za y: ") -> Unesite vrednost za y: 6

print(proizvod(int(valX),int(valY)))    -> 30
```