Sekvence, liste i torke

U prethodnim nastavnim jedinicama je bilo reči o sekvencama u Pythonu, ali ćemo ih u ovoj nastavnoj jedinici detaljnije obraditi. Same strukture podataka u Pythonu se mogu podeliti na sekvence i kolekcije. Razlika je u redosledu elemenata. Elementi u sekvencama zadržavaju redosled po kojim ulaze, dok to u koListe su promlekcijama nije slučaj. Kod kolekcija se redosled menja pri svakom ubacivanju novog elementa, jer je kod njih akcenat na brzini pristupanja traženom elementu, a ne na redosledu.

U Pythonu, sekvence su generički niz elemenata. Python podržava šest vrsti sekvenci; to su:

- stringovi
- liste
- n-torke (engl. n-tuples)
- bajt sekvence
- bajt nizovi
- range objekti

Ponovićemo osnove ovih sekvenci i nastaviti dalje sa operacijama i funkcijama nad njima (range objekti su detaljno obrađeni u prethodnoj nastavnoj jedinici).

Stringovi

Stringovi su grupa karaktera koja je ispisana unutar navodnika. U nekim programskim jezicima postoji podrška i za tip char (karakter), dok kod Pythona to nije slučaj, pa se čak i jedan karakter unutar navodnika računa kao string.

Primer stringa

Radno okruženje

```
name = "Marco"

print(type(name))

print(name[2])

print(name[2:4])

#Na ovaj nacin rotiramo string jer prolaziamo kroz ceo string ali od kraja za jedan karakter
rotiran_name=name[::-1]
print(rotiran_name)

#Ovaj red će izazvati grešku pošto string ne možemo da menjamo
name[2] = 'a'
print(name)
```

Copyright © Link group

Objašnjenje:

Prvo smo definisali i inicirali promenljivu ime na vrednost 'Marco'. Korišćenjem uglastih zagrada možemo pristupati isečcima svake sekvence pomoću indeksa elemenata, bilo da je reč o jednom elementu tog isečka (name[2]) ili da želimo više elemenata u tom isečku (name[2:4]). S obzirom na to da su stringovi, jednom kada se definišu – nepromenljivi, na kraju smo dobili grešku, kada smo pokušali da u tom stringu jedan element zamenimo drugim.

Liste

Liste nam omogućavaju da kreiramo heterogene kolekcije elemenata. Lista može sadržati različite tipove podataka, pa tako možemo imati listu čiji su elementi stringovi, n-torke, rečnici i drugi namenski objekti. Liste se deklarišu pomoću uglastih zagrada, a elementi unutar liste su odvojeni zarezom.

Primer liste

```
lst1 = [1,2,3,4]

lst2 = ['New York', 'Moscow', 'London']

lst3 = ['Marco', 3, (1993, 12), {"gender":"male"}]
```

Liste su promenljiv tip podataka, pa tako elemente unutar njih možemo brisati, dodavati i menjati:

Primer mogućnosti izmene liste

```
lst3[0]=1
print(lst3)

lst2[1]= 'Belgrade'
print(lst2)
```

Takođe se i u slučaju listi isečci prave zadavanjem opsega od kog do kog indeksa želimo elemente.

Objašnjenje:

Ako napišemo lst3[2] izdvajamo element na poziciji 2 liste lst3 odn. (1993, 12). Možemo izdvojiti i više elemenata odn. isečak iz liste(dobijamo podlistu) ako napišemo lst[1:4]. Rezultat je [3, (1993, 12), {'gender': 'male'}] jer izdvajamo elemente od pozicije 1 do pozicije 4 ali bez tog elementa na poziciji 4.

N-torke

N-torke su nepromenljivi tip sekvenci, što znači da, kada su vrednosti jednom dodeljene n-torki, one se ne mogu više menjati niti se mogu dodavati nove. One se u Pythonu kreiraju

Copyright © Link group

pomoću običnih zagrada sa zarezom kao separatorom elemenata. Ako definišemo samo jedan element u n-torki, onda stavljamo zarez nakon njega i zatvaramo zagradu (`tup = (2,)`).

Primer ntorke

Radno okruženje

```
tup = ()
print(type(tup))

tup = ('Marco', 3, (1993, 12), {"gender":"male"})

print(tup[1:4])

tup[0] = 'John'
```

Objašnjenje:

Kako su torke nepromenljiv tip podataka kao i stringovi, kada smo pokušali da promenimo element, dobili smo grešku. Odsečci se dobijaju na isti način kao i kod stringova i listi.

Bajt sekvence i nizovi

Oba tipa podataka se koriste za rukovanje objektom byte, s tim što su bajt sekvence nepromenljivi tip podatka, a bajt niz – promenljivi niz podatka.

Od datog podatka, bajt sekvence se generiše pomoću `bytes()` funkcije, a bajt nizovi pomoću `bytearray()`.

Primer byte sekvence

```
byte = 7
print(bytes(byte))


print(type(bytes(byte)))
print(bytearray([1, 2, 3, 4]))
```

Operacije nad Python sekvencama

Kako smo tipove podataka u Pythonu klasifikovali kao sekvence i kolekcije, tako možemo obraditi i operacije koje se mogu vršiti nad njima, a to su:

- konkatenacija;
- multiplikacija;
- članstvo elemenata;
- isečci.

Copyright © Link group

Konkatenacija

Ovom operacijom se prvoj sekvenci dodaje drugi operand.

Primer konkatenacije

```
print('Mar' + 'Co')

print([1,2,3] + list(range(4,6)))
```

Objašnjenje:

Ako spojimo 2 stringa dobijamo string a ako spojimo 2 liste dobijamo listu. Range vraća sekvencu a pomoću funkcije list je pretvaramo u listu pa je možemo spojiti sa listom [1,2,3].

Multiplikacija

Ovom operacijom možemo n puta ponoviti datu sekvencu:

Primer multiplikacije

```
print('Mar' * 3)

print([1,2,3] + list(range(4,6)) * 2)
```

Objašnjenje:

Ako pomnožimo string sa celim brojem dobićemo string čiji se niz karaktera ponavlja toliko puta koliki je broj. Ako pomnožimo listu dobićemo listu čiji se elementi ponavljaju toliko puta koliki je taj broj.

Članstvo elemenata

Članstvo elemenata se proverava sintaksom x in s, gde je x bilo koji podatak, a s sekvenca. Ta naredba će vratiti True vrednost ako vrednost x postoji u sintaksi sekvence s. Naredba `x not in s` je ista kao i `not (x in s)` i vratiće True ako x ne postoji ni u jednom podskupu sekvence s.

Primer	
Kod	**Rezultat**
`s = 'New York'` `print('ew' in s)`	`True`
`s = 'New York'` `print('new' in s)`	`False`
`lista= list(range(0,5))` `print(5 not in lista)`	`True`

Tabela 11.1. Članstvo elemenata

Copyright © Link group

Isečci

Nekada nam je potreban samo deo date sekvence, podskup ili isečak i za to koristimo operator (`:`). Za biranje samo jednog elementa iz sekvence dovoljno je u uglastim zagradama proslediti njihov indeks (`s[3]`). Takođe, moguće je koristiti i negativne vrednost za indekse, ali će to onda biti redosled elemenata zdesna nalevo, pa tako indeks od -1 daje poslednji element, a -2 indeks daje pretposlednji element.

Pravljenje isečka sekvence funkcioniše po sličnom principu kao i parametri `range()` funkcije. Zadajemo početne i krajnje indekse isečka koji želimo u formatu `s[1:3]`. Pravilo negativnih indeksa važi i ovde, pa tako izraz `'New York'[-3:]` daje rezultat 'ork'.

Takođe, prilikom generisanja isečaka, kao i u funkciji `range()`, možemo odrediti korak i to na sledeći način: `s[i:j:k]`, što je ekvivalent izrazu `s[::2]`. U sledećem primeru ćemo videti načine generisanja isečaka na tipu podatka string, ali se isti princip primenjuje i na ostale sekvence:

Primer	
Kod	**Rezultat**
`print('New York'[-1:])`	`'k'`
`print('New York'[1:])`	`'ew York'`
`print('New York'[-3])`	`'o'`
`print('New York'[:-3])`	`'New Y'`
`print('New York'[-3:])`	`'ork'`
`print('New York'[0:-1])`	`'New Yor'`
`print('New York'[::2])`	`'NwYr'`
`print('New York'[::-1])`	`'kroY weN'`

Tabela 11.2. Isečci

Isprobajte primere sa stringovima:

Radno okruženje

Pitanje

Naredba print(`True in [1,2,3,4]`) će ispisati:

- **True**
- False
- SyntaxError

Objašnjenje:

Tačan odgovor je True, jer svaka sekvenca koja nije prazna je po sintaksi Pythona tačna, dok su prazne sekvence False.

Copyright © Link group

Vežba 1

Iz datog stringa "Hello World" ispišite svako drugo slovo.

Radno okruženje

Rešenje možete naći na kraju lekcije.

U narednoj tabeli se nalazi rezimirani, uporedni prikaz prethodno obrađenih tipova sekvence i njihovih svojstava:

Tipovi sekvenci			
Ime	**Tip**	**Promenljivi/Nepromenljivi**	**Sintaksa**
Stringovi	str	Nepromenljivi	Kreiraju se pomoću jednostrukih, dvostrukih ili trostrukih navodnika. Elementi stringova su jedino karakteri.
Liste	list	Promenljivi	Kreiraju se pomoću uglastih zagrada []. Bilo koji tip podatka može biti element liste.
N-torke	tuple	Nepromenljivi	Kreiraju se pomoću običnih zagrada (). Bilo koji tip podatka može biti element n-torke.
Bajt sekvence	bytes	Nepromenljivi	Stringovi u bajt reprezentaciji. Kreiraju se pomoću bytes() funkcije.
Bajt nizovi	bytearray	Promenljivi	Lista bajtova. Kreira se pomoću bytearray() funkcije. Bilo koji tip podatka može biti element ove liste.
Range	range	Nepromenljivi	Kreira se pomoću range() funkcije sa makar jednim (stop) argumentom. Elementi su celi brojevi (int).

Tabela 11.3. Rezimirani prikaz kolekcija

Funkcije nad sekvencama

Kako sekvence funkcionišu po sličnom principu, postoji set funkcija koji se može primeniti na svakoj od njih. U našim primerima ćemo ih primenjivati nad tipom podatka string, ali se mogu primeniti i na ostale sekvence.

Funkcija len()

Često korišćena funkcija koja vraća veličinu prosleđene sekvence. Takođe, rezultat ove funkcije se može koristiti i kao argument prilikom generisanja isečaka.

Copyright © Link group

Primer

```
print(len("New York"))
```

Objašnjenje:

Funkcijom len dobijamo dužinu sekvence, odn. broj karaktera stringa.

Funkcije min() i max()

Ove funkcije vraćaju najveću ili najmanju vrednost sekvence. Ako je tip sekvence koji se prosleđuje string, onda vraća najmanju ili najveću vrednost ugledajući se na ASCII tabelu[1]. Takođe, ako data sekvenca sadrži mešovite tipove podataka, poziv ove dve funkcije vratiće grešku.

Primer	
Kod	**Rezultat**
`print(min("London"))`	`'L'`
`print(max("London"))`	`'o'`

Kao rezultat funkcije min() dobijamo slovo L, jer u stringu London veliko slovo L ima najmanju vrednost u ASCII tabeli (76), dok malo slovo o (111) ima najvišu vrednost u ASCII tabeli.

Metode nad sekvencama

Neke od najkorišćenijih metoda nad sekvencama su index() i count().

Metoda **index("string")** vraća index prvog pojavljivanja prosleđenog elementa:

Primer

```
print("New York".index("Y"))
```

Objašnjenje:

Prvo i jedino pojavljivanje karaktera **"** Y **"** u stringu **"** New York**"** je na poziciji 4.

Ako prosledimo element koji ne postoji u sekvenci, dobićemo ValueError grešku.

Metoda **count("string")** vraća broj ponavljanja datog elementa. Ako se dati element ne pronađe, vratiće 0.

Primer

```
print('Moscow'.count('o'))
```

[1] https://www.ascii-code.com/

Copyright © Link group

Vežba 2

Date su dve liste a, i b, iste veličine :

```
a = ['H','l','o','W','r','d']
b = ['e','l',' ','o','l','']
```

Ispišite na ekran string "Hello World" koristeći elemente ove dve liste.

Radno okruženje

Rešenje možete naći na kraju lekcije.

Metoda **split**("welcome to the jungle") vraća listu reči u stringu. Metod pretpostavlja da su reči razdvojene blanko znacima(razmakom).

Primer

```
print("New York".split())
```

Metod **split** ima jedan opcioni argument kojim se omogućava da se razbijanje stringa vrši po nekom drugom znaku a ne na blanko znaku.

Primer

```
print("apple#banana#orange".split('#'))
```

Metoda **join** je na neki način suprotno od metode split. To je string metod koji uzima listu stringova i spaja je u jedan string. Primena metode join nad listom L = ['a', 'b', 'c'] bi izgledala ovako:

Primer	
Kod	**Rezultat**
`print(" ".join(L))`	`a b c`
`print("".join(L))`	`abc`
`print(",".join(L))`	`a,b,c`
`print("Y".join(L))`	`aYbYc`

Metoda **eval** analizira izraz prosleđen ovoj metodi i pokreće taj Python izraz (kod) unutar programa. Izraz koji se prosleđuje je string.Metoda eval() vraća rezultat izraza.

Primer

```
x = 1
print(eval('x+1'))

x = '5/3'
print(eval(x))
```

Copyright © Link group

```
a='5+7'
print(eval(a))
```

Rešenje vežbe 1

Rešenje

```
s = "Hello World"
print(s[::2])
```

Ispis:

```
'HloWrd'
```

Objašnjenje:
U rešavanju ove vežbe smo koristili string isečke, ali nismo zadali početne i krajnje vrednosti isečku, već samo korak – 2, jer nam je svako drugo slovo i potrebno i dobili smo string `'HloWrd'`.

Rešenje vežbe 2

Rešenje

```
a = ['h','l','o','w','r','d']
b = ['e','l',' ','o','l','']

for i in range(len(a)):
    print("{}{}".format(a[i], b[i]), end = '')
```

Ispis:

```
Hello World
```

Objašnjenje:

U rešavanju ove vežbe smo koristili funkciju len() zajedno sa range() funkcijom. Koristeći len() funkciju, prvo smo otkrili veličinu liste a, a potom tu vrednost prosledili range funkciji kako bismo generisali brojeve od nule do len(a) koje ćemo iskoristii za indeksno pristupanje elementima listi a i b. Ovaj pristup možemo koristiti jer znamo unapred da su liste iste veličine. Svaka {} se zamenjuje argumentima format metode. Prva zagrada se zamenjuje sa vrednoscu promenljive a[i] odn. trenutnim elementom niza a u zavisnosti od vrednosti promenljive i u for petlji koja predstavlja indeks, druga sa vrednoscu promenljive b[i] odn. trenutnim elementom niza b u zavisnosti od vrednosti promenljive i.

Copyright © Link group

Rezime

- Elementi u sekvencama zadržavaju originalni redosled unetih elemenata.
- Sekvence u Pythonu su: stringovi, liste, n-torke (tuple), bajt sekvence, bajt nizovi i range objekti.
- Stringovi su grupa karaktera ispisana unutar jednostrukih, dvostrukih ili trostrukih navodnika ('','',"""""").
- Stringovi su nepromenljiv tip podatka. Kada se string jednom definiše, ne može se promeniti.
- Liste nam omogućavaju da kreiramo sekvence heterogenih elemenata (elementi mogu biti različitih tipova).
- Liste se deklarišu uglastim zagradama ([]), dok su elementi unutar njih odvojeni zarezom.
- Liste su promenljiv tip podataka, što nam omogućava izmenu, dodavanje i brisanje elemenata.
- N-torke su nepromenljiv tip sekvenci. Kada se n-torka jednom definiše, ne može se promeniti;
- Torke nam omogućavaju da kreiramo sekvence heterogenih elemenata (elementi mogu biti različitih tipova).
- Torke se deklarišu običnim zagradama (()), dok su elementi unutar njih odvojeni zarezom.
- Bajt sekvence se generišu pomoću bytes() funkcije i nepromenljiv su tip podatka.
- Bajt nizovi se generišu uz pomoć bytearray() funkcije i promenljiv su tip podatka.
- Operacije nad sekvencama su: konkatenacija, multiplikacija, članstvo i isečci.
- Konkatenacija dve sekvence se vrši pomoću operatora (+).
- Multiplikacija sekvenci se vrši korišćenjem operatora (*).
- Članstvo elementa se proverava pomoću ključne reči in, i to ovako: element in sekvenca. Povratna vrednost je True ili False, u zavisnosti od rezultata provere.
- Isečci se koriste kada nam je potreban samo jedan deo date sekvence. Isečak može, ali i ne mora imati početni indeks i krajnji indeks, ali treba da ima dve tačke kao separator ove dve vrednosti: l[start:stop].
- Neke od ugrađenih funkcija za rad sa sekvencama su min(), max() i len().
- Neke od uobičajenih metoda nad sekvencama su index() i count().
- Sve ove sekvence podržavaju i indeksiranje, dobavljanje elementa po zadatom indeksu.

Copyright © Link group