Python Development

Recap, Operacije nad listama i stringovima, Stack and Queue

POWERED BY COMTRADE

Recap

- Python 2 vs. Python 3
- Šta je PATH promenljiva?
- Python 3.7.2 (Major, Minor, Build)
- Kada koristiti Python IDLE?
- Koji karakter služi za komentarisanje koda?
- Kojom funkcijom se čita sa standardnog ulaza?
- Kompajleri vs. Interpretatori
- Naming konvencija promenljivih
- Kako se zove Python style guide?
- Rezervisane reči u Pythonu i promenljive?

Recap

- Da li je Python case sensitive jezik?
- Na koje načine je moguće štampati promenljive uz pomoć print-a?
- Tipovi podataka u Python-u
- Kako se zove promena tipa promenljive? Koje funkcije rade promenu tipova?
- Koje aritmetičke operacije Python podržava?
- Koje logičke operacije Python podržava?
- Koji operatori poređenja postoje u Python-u?
- Kontrola toka uz pomoć if, else i elif
- Liste i operacije/funkcije/metodi nad listama
- Zašto se kao prvi element liste uzima element sa indeksom 0?
- Kako napraviti listu koja ima 100 elemenata gde su svi elementi jedinice?

Recap

- Slajsovanje listi?
- Petlje, razlika između for i while petlji u Pythonu
- Funkcije, koje smo ugrađene funkcije radili?
- Parametri (argumenti) funkcija, obavezni i neobavezni (opcioni)
- Razlika između list.sort() i sorted(list)
- Šta znači inplace metod?
- Numeričko vs. Leksikografsko sortiranje
- Kodne šeme: ASCII, Unicode, UTF-8
- Koja funkcija štampa Unicode vrednost karaktera?
- Kako radi else za for petlju?
- Kako rade list comprehension-i?
- Da li je u list comprehension-ima moguće imati if-ove?
- Šta je rekurzija?

Operacije nad listama

- Do sada smo prešli:
  - Pristupanje elementima liste
  - Slajsovanje listi
  - len() nad listama
  - .append() metod
  - list.sort() i sorted(list)
  - Konkatenacija listi: lista1 + lista2
  - Inicijalizacija listi određene dužine uz pomoć: [] * x

Operacije nad listama

- Šta još možemo raditi sa listama?
  - Kompletna dokumentacija - https://docs.python.org/3/tutorial/datastructures.html
  - `list.`**`remove`**(*x*): *Remove the first item from the list whose value is equal to x. It raises a* `ValueError` *if there is no such item.*
  - `list.`**`pop`**([*i*]): *Remove the item at the given position in the list, and return it. If no index is specified,* `a.pop()` *removes and returns the last item in the list. (The square brackets around the i in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)*
  - **`del`** `list[i]`: *Removes i-th item of the list*
  - `list.`**`clear`**(): *Remove all items from the list. Equivalent to* `del a[:]`
  - `list.`**`count`**(*x*): *Return the number of times x appears in the list*
  - `list.`**`reverse`**(): *Reverse the elements of the list in place.* *** **[::-1]** ***
  - `list.`**`copy`**(): *Return a shallow copy of the list. Equivalent to* `a[:]`.

Operacije nad listama

- **Normal assignment** vs. Shallow copy vs. Deep copy

```
lista_brojeva_2 = lista_brojeva_1

lista_brojeva_2[0] = “Neki string”

print(lista_brojeva_1)

print(lista_brojeva_2)
```

- Šta se ovde desilo?
- Implementacija Python liste radi se preko C-ovih povezanih listi i pokazivača
- len() funkcija radi momentalno jer je upisana u čvor liste

Operacije nad listama

- Normal assignment vs. **Shallow copy** vs. Deep copy

```
lista_brojeva_2 = lista_brojeva_1.copy()

lista_brojeva_2[0] = “Neki string”

print(lista_brojeva_1)

print(lista_brojeva_2)
```

- Shallow copy konstruiše nov objekat a onda napravi reference koje pokazuju ka objektima koje su pronađene u originalnoj listi

Operacije nad listama

- Normal assignment vs. **Shallow copy** vs. Deep copy

```
lista1 = [1,2,[3,4]]

lista2 = lista1.copy()

lista2[2] = “Umesto liste [3,4] stavljam nekakav string”

print(lista1)

print(lista2)
```

- Šta se sada dešava?

Operacije nad listama

- Normal assignment vs. **Shallow copy** vs. Deep copy

```
lista1 = [1,2,[3,4]]

lista2 = lista1.copy()

lista2[2][0] = 55555

print(lista1)

print(lista2)
```

- Šta se sada dešava?

Operacije nad listama

- Normal assignment vs. Shallow copy vs. **Deep copy**

```
from copy import deepcopy

lista1 = [1,2,[3,4]]

lista2 = deepcopy(lista1)

lista2[2][0] = 55555

print(lista1)

print(lista2)
```

- Šta se sada dešava?
- https://stackoverflow.com/questions/17246693/what-is-the-difference-between-shallow-copy-deepcopy-and-normal-assignment-oper

Operacije nad stringovima

- Do sada smo prešli:
  - Format metod
  - upper() metod
  - lower() metod
  - F stringove
  - Konkatenacija stringova
  - Umnožavanje stringova
  - Join metod
  - Slajsovanje stringova
  - len() funkcija nad stringovima
  - Split() metod
  - ord() funkcija

Operacije nad stringovima

- Šta još možemo raditi sa stringovima?
  - `str.` **capitalize()** metod - Stavlja veliko slovo na početak stringa
  - `str.` **count(x)** metod - Pronalazi koliko se puta podstring x nalazi u okviru stringa
  - `str.` **startswith(x)** i str**.endswith(x)** metodi - Proverava da li string počinje/se završava sa podstringom x
  - `str.` **find(x)** metod - Pronalazi indeks prvog pojavljivanja podstringa x u okviru stringa
  - `str.` **rfind(x)** metod - Pronalazi indeks prvog pojavljivanja podstringa x u okviru stringa ali tražeći s desna na levo
  - `str.` **lstrip()** metod - Uklanjanje belina sa leve strane stringa
  - `str.` **rstrip()** metod - Uklanjanje belina sa desne strane stringa
  - `str.` **strip()** metod - Uklananje belina sa obe strane stringa
- Primeri: *stringovi.py*

List comprehension - Podsetnik

```
>>> symbols = '$¢£¥€¤'
>>> codes = [ord(symbol) for symbol in symbols]
>>> codes
[36, 162, 163, 165, 8364, 164]
```

List comprehension - Doseg promenljivih

Python 2

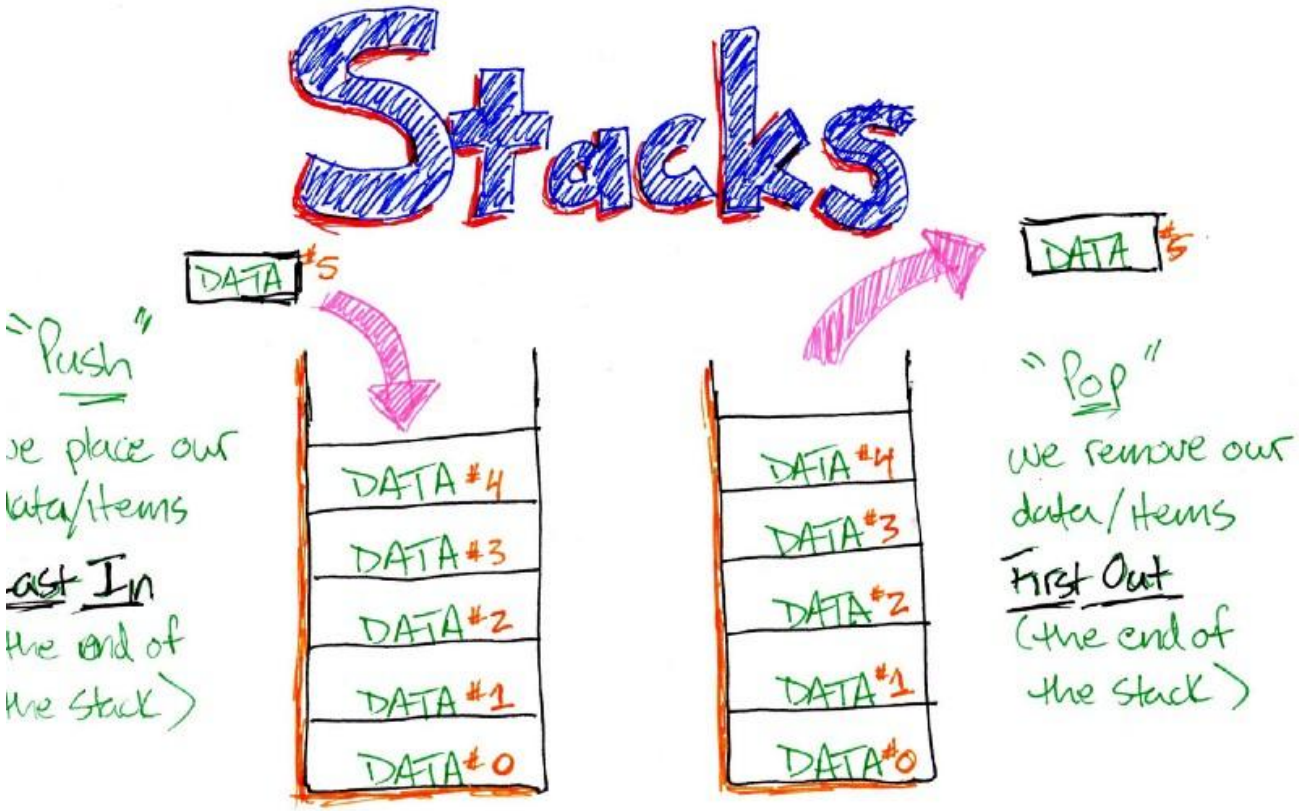
```
Python 2.7.6 (default, Mar 22 2014, 22:59:38)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 'my precious'
>>> dummy = [x for x in 'ABC']
>>> x
'C'
```

List comprehension - Doseg promenljivih

Python 3

```
>>> x = 'ABC'
>>> dummy = [ord(x) for x in x]
>>> x
'ABC'
>>> dummy
[65, 66, 67]
```

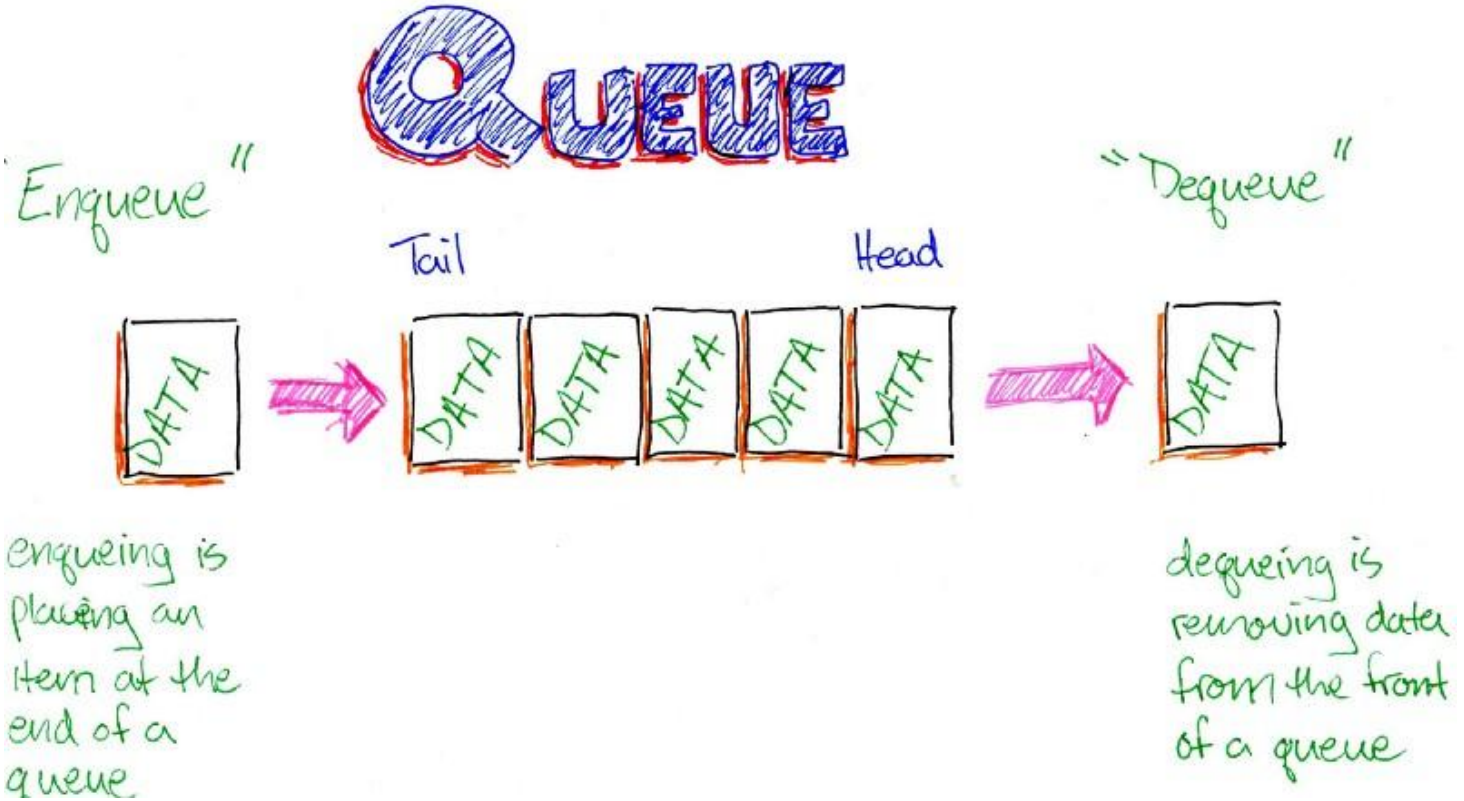
Strukture podataka: Stek



Strukture podataka: Stek

- Stack je LIFO struktura: **L**ast **I**n **F**irst **O**ut
- Koncept Stack overflow
- Koncept Stack underflow

Strukture podataka: Red (FIFO)



Zadaci za vežbanje (1):

Sa standardnog ulaza unosi se aritmetički izraz u obliku stringa: npr. *“2+3*(5 - 2)”*, napisati program koji provarava da li su zagrade pravilno uparene. Koristiti stack implementiran uz pomoć liste. Napisati pomoćne funkcije koje primaju listu kao argument i znaju da implementiraju push i pop operacije.

Primer nepravilno uparenih zagrada:

“2+3(5 - 2))”,*

“2+3(5 - 2)(”*