UNIVERZITET METROPOLITAN BEOGRAD

CS324 - SKRIPTING JEZICI

Pregled predmeta

Lekcija 15

PRIRUČNIK ZA STUDENTE

CS324 - SKRIPTING JEZICI

Lekcija 15

PREGLED PREDMETA

UVOD

Uvod u lekciju #15

U petnaestoj lekciji najpre je predstavljen pojam ***razumevanja listi*** kao napredni način kreacije listi i imenika.

Zatim su predstavljene dobre prakse u Python programiranju, kao i neke česte greške koje se javljaju, najviše jer Python ne predstavlja prvi naučeni programski jezik.

Tokom predmeta predstavljeni su najpopularniji paketi u Python jeziku za razne primene. Naravno, pregršt biblioteka, paketa i modula jeste ono što čini Python popularnim. Upravo zbog velikog broja paketa neki od njih nisu obrađeni, ali su nabrojeni, zajedno sa linkovima i dokumentacijom.

Konačno, dat je pregled predmeta sa Google formom koju je poželjno popuniti zbog unapređenja predmeta.

U pokaznim vežbama dati su opisi pitanja kakva studenti mogu očekivati na ispitu.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 1

Python i razumevanje listi

UVOD U RAZUMEVANJE LISTI

Razumevanje listi jeste jedan sintaksni oblik pisanja listi i sličnih iterabilnih promenljivih.

Razumevanje listi (en. list comprehension) jeste metoda pisanja listi i sličnih iterabilnih promenljiva koja je bliža matematičkoj notaciji konstrukciji skupova.

Primer:

Matematički formulisati niz koji je skup svih brojeva ***2 puta X***, gde je ***X*** element skupa ***prirodnih brojeva***, i X^2 je veće od 3.

$$S = \{2 \cdot x | x \in \mathbb{N}, x^2 > 3\}$$

U pojedinim programskim jezicima, razumevanje listi jeste sintaksni oblik pisanja listi i sličnih iterabilnih promenljivih koje pojednostavljuju pisanje koda, često izbegavajući više linija koda i for petlje.

U Python jeziku, ovakva notacija iterabilnih promenljiva olakšava pisanje listi koja je ujedno bliža i ljudskom govoru, što čini Python sintaksu još lakšom.

Primer #1

Lista ***nums*** sadrži sve prirodne brojeve od 1 do 10. Popuniti promenljivu ***my_list*** ovim brojevima, bez i sa korišćenjem razumevanja listi.

```
nums = [1,2,3,4,5,6,7,8,9,10]

# zadatak:
# Popuniti my_list sa n, za svako n u nums

# bez razumevanja listi
my_list = []
for num in nums:
    my_list.append(num)
print(my_list)

# sa razumevanjem listi
my_list = [n for n in nums]
print(my_list)
```

Izlaz:

```
>>>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Može se primetiti da je kod razumevanja listi ***for*** petlja sadržana unutar jedne komande, kao i da nema inicijalne kreacije prazne liste.

Takođe se može primetiti da je sama sintaksa sličnija govornom jeziku zadakta: "Popuniti ***my_list*** sa ***n***, za svako ***n*** u ***nums***"

PRIMERI ZA RAZUMEVANJE LISTI BEZ DODATNIH USLOVA

Nekada je moguće koristiti map funkciju umesto for funkcije pri sintaksi razumevanja listi.

Primer #2

Lista ***nums*** sadrži sve prirodne brojeve od 1 do 10. Popuniti promenljivu ***my_list*** kvadratima ovih brojeva, najpre bez, pa sa korišćenjem razumevanja listi, kao i korišćenjem ***map*** funkcije.

```
nums = [1,2,3,4,5,6,7,8,9,10]

# zadatak:
# Popuniti my_list sa n^2, za svako n u nums

# bez razumevanja listi
my_list = []
for num in nums:
    my_list.append(num*num)
print(my_list)

# sa razumevanjem listi
my_list = [n*n for n in nums]
print(my_list)

# sa map+lambda funkcijom
my_list = map(lambda n: n*n, nums)
print(list(my_list))
```

Izlaz:

```
>>>[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>>[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>>[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Napomena:

Funkcija map() pravi iterator koji računa funkciju koristeći argumente svih ite4rabilnih promenljivih. Zaustavlja se kada se iskoristi najkraći iterator.

```
numbers1 = [1, 2, 3]
numbers2 = [4, 5, 6]

result = map(lambda x, y: x + y, numbers1, numbers2)
print(list(result))
```

Izlaz:

```
>>>[5, 7, 9]
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMERI ZA RAZUMEVANJE LISTI SA DODATNIM USLOVIMA

Nekada je moguće koristiti filter funkciju umesto for funkcije pri sintaksi razumevanja listi.

Primer #3

Lista ***nums*** sadrži sve prirodne brojeve od 1 do 10. Popuniti promenljivu ***my_list*** samo parnim brojevima iz prve liste, najpre bez, pa sa korišćenjem razumevanja listi, kao i korišćenjem filter funkcije.

```
nums = [1,2,3,4,5,6,7,8,9,10]

# zadatak:
# Popuniti my_list sa n, za svako n u nums koje je parno

# bez razumevanja listi
my_list = []
for num in nums:
    if num % 2 == 0:
        my_list.append(num)
print(my_list)

# sa razumevanjem listi
my_list = [n for n in nums if n % 2 == 0]
print(my_list)

# sa filter+lambda funkcijom
my_list = filter(lambda n: n % 2 == 0, nums)
print(list(my_list))
```

Izlaz:

```
>>>[2, 4, 6, 8, 10]
>>>[2, 4, 6, 8, 10]
>>>[2, 4, 6, 8, 10]
```

Napomena:

Funkcija filter() vraća iterabilnu promenljivu sa onim elementima koji ispunjavaju uslov.

Return an iterator yielding those items of iterable for which function(item) is true. If function is None, return the items that are true.

```
# function that filters vowels
def fun(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable in letters):
        return True
    else:
        return False

# sequence
sequence = ['g', 'e', 'e', 'j', 'k', 's', 'p', 'r']

# using filter function
filtered = filter(fun, sequence)

print('The filtered letters are:')
for s in filtered:
    print(s)
```

Izlaz:

```
>>>e
>>>e
```

PRIMERI ZA RAZUMEVANJE LISTI SA UGNJEŽDENIM PETLJAMA

I ugnježdene petlje se mogu napisati u jednoj liniji koda korišćenjem sintakse razumevanja listi.

Primer #4

Vratiti par (slovo, broj) za svako slovo u "abcd" i za svaki broj u "1234", bez, i sa razumevanjem listi.

```
# zadatak:
# Popuniti my_list sa parovima (letter, num) za slova "abcd" i brojeve "1234"

# bez razumevanja listi
my_list = []
for letter in 'abcd':
    for num in range(4):
        my_list.append((letter,num))
print(my_list)

# sa razumevanjem listi
my_list = [(letter, num) for letter in 'abcd' for num in range(4)]
print(my_list)
```

Izlaz:

```
>>>[('a', 0), ('a', 1), ('a', 2), ('a', 3), ('b', 0), ('b', 1), ('b', 2), ('b', 3),
('c', 0), ('c', 1), ('c', 2), ('c', 3), ('d', 0), ('d', 1), ('d', 2), ('d', 3)]
>>>[('a', 0), ('a', 1), ('a', 2), ('a', 3), ('b', 0), ('b', 1), ('b', 2), ('b', 3),
('c', 0), ('c', 1), ('c', 2), ('c', 3), ('d', 0), ('d', 1), ('d', 2), ('d', 3)]
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

RAZUMEVANJE IMENIKA

Razumevanje imenika prati sličnu sintaksu kao kod razumevanja lista

Primer 5:

Date su dve liste:

```
names = ['Bruce', 'Clark', 'Peter', 'Logan', 'Wade']
heroes = ['Batman', 'Superman', 'Spiderman', 'Wolverine', 'Deadpool']
```

Napraviti imenik ***my_dict*** sa parovima **ključ: vrednost** = ***name***: ***hero*** iz datih listi, najpre bez, a posle sa razumevanjem liste (razumevanjem imenika).

```
# Razumevanje listi - imenici

names = ['Bruce', 'Clark', 'Peter', 'Logan', 'Wade']
heroes = ['Batman', 'Superman', 'Spiderman', 'Wolverine', 'Deadpool']

# Bez razumevanja imenika
my_dict = {}
for name, hero in zip(names, heroes):
    my_dict[name] = hero
```

```
print(my_dict)

# Sa razumevanjem imenika
my_dict = {name: hero for name, hero in zip(names, heroes)}
print(my_dict)
```

Izlaz:

```
>>>{'Bruce': 'Batman', 'Clark': 'Superman', 'Peter': 'Spiderman', 'Logan':
'Wolverine', 'Wade': 'Deadpool'}
>>>{'Bruce': 'Batman', 'Clark': 'Superman', 'Peter': 'Spiderman', 'Logan':
'Wolverine', 'Wade': 'Deadpool'}
```

Napomena:

Funkcija zip() vraća tuple-ove elemenata pojedinačnih iterabilnih promenljivih koje se ubace kao lista parametra. Funkcija staje kod najkraće iterabilne promenljive

```
# primer za zip:
list(zip('abcdefg', range(3), range(4)))

# zip objekat se mora pretvoriti u listu da bi se stampali tuple-ovi
>>>[('a', 0, 0), ('b', 1, 1), ('c', 2, 2)]
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 2

Dobre prakse Python programiranja

UVOD U DOBRU PRAKSU PROGRAMIRANJA

Uvek treba imati na umu da postoji dobra praksa programiranja koja može učiniti kod pisan na Python programskom jeziku boljim i optimalnijim.

Jedna od najvećih prednosti Python programskog jezika jeste jednostavna sintaksa i čitljivost koda.

Međutim, treba imati na umu da postoji dobra praksa programiranja (en. good coding practices) koja može učiniti kod pisan na Python programskom jeziku boljim i optimalnijim.

U nastavku jesu neke od teme dobrih praksi pri radu sa Python jezikom:

- Ternarna uslovna izjava (en. ternary conditionals)
- Dodavanje donjih crta (en. underscore placeholders)
- Kontekstni menadžeri (en. context managers)
- Funkcija ***enumerate()***
- Funkcija ***zip()***
- Funkcije ***help()*** i ***dir()***

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

TERNARNI USLOV

Kod ternarnog uslova (ternarnog operatora) moguće je izvršiti ispitivanje uslova u jednom redu.

Kod jednostavnih if/else naredbi, moguće je istu naredbu napisati kroz ternarni uslov, koji je jednostavnije sintakse.

Primer:

Napisati jednostavni program za proveru uslova ***condition***. Ukoliko je ***condition=True***, vrednost promenljive ***x*** postaje 1, ili postaje 0 ukoliko nije.

Uraditi program i kao *ternarni uslov*.

```
# ternarni uslov
condition = True

# prvi nacin
if condition:
    x = 1
else:
    x = 0

print(x)

# drugi nacin
x = 1 if condition else 0

print(x)
```

Primećuje se da je, slično kao kod razumevanje listi, ternarni uslov bliži govornom jeziku.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

DODAVANJE DONJIH CRTA

Python može ispisivati brojeve sa zapetom za razdvajanje hiljada.

Pri radu sa veoma velikim ili veoma malim brojevima, standardni izlaz ne grupiše izlaz po hiljadama.

Primer:

Broj milion (1000000) se može lakše prepoznati ukoliko se napiše (1.000.000), tj. ako se hiljade razdvajaju tačkom.

Napomena:

U srpskom jeziku, razlomljeni brojevi se razdvajaju decimalnom zapetom, dok u engleskom se razdvajaju decimalnom tačkom.

Izlaz u Python-u (ukoliko se ne koristi prikaz brojeva po naučnoj notaciji: 10^6) prikazivaće brojeve bez grupisanja:

```
>>> 1000000
```

Međutim, u Python jeziku moguće je pri pisanju vrednosti dodati donju crtu (en. underscore) koja neće menjati vrednost promenljive, ali je lakše programerima pisati velike ili male brojeve.

Primer: Napisati 10,000,000,000 korišćenjem donjih crta.

```
x = 10_000_000_000
print(x)
>>> 10000000000
```

Python takođe može ispisivati brojeve sa zapetom za razdvajanje hiljada ukoliko se štampa kao *f-string* sa dodatnim parametrom ***{:,}***

```
# razdvajanje hiljada

x = 10_000_000_000
y = 100_000_000_000
z = x + y
print(f'{z:,}')

# izlaz
>>> 110,000,000,000
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

KONTEKSTNI MENADŽERI

Upotrebom kontekstnih menadžera lakše se upravljaju resursi koji bi se posle korišćenja morali zatvoriti.

Pri radu sa kontrolom resursa, kao što su datoteke ili niti (en. thread), treba voditi računa o resursima da se resursi zatore nakon korišćenja.

Ukoliko se koriste kontekstni menadžeri, onda se unutar bloka kod kontekstnog menadžera izvršavaju operacije nad resursima, a čim se izađe iz tog bloka koda, resurs se zatvara.

Primer:

Otvoriti datoteku "test_file.txt" najpre bez, a nakon toga sa kontekstnim menadžerom. U ova slučaja pročitati liniju teksta i štampati

```
# bez kontekstnog menadzera

f = open('test_file.txt', 'r')
x = f.readline()
f.close()
print(x)

# sa kontekstnim menadzerom

with open('test_file.txt', 'r') as f:
    y = f.readline()

print(y)
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

FUNKCIJA ENUMERATE

Upotrebom funkcije enumerate moguće je dobiti i indeks i promenljivu unutar petlje.

Za razliku od programskih jezika koji su bazirani na C familiji jezika, za prolazak kroz for petlju nije neophodan brojač kao eksplicitna promenljiva, koji ujedno sliži i kao indeks elemenata unutar iterabilne promenljive.

Korišćenjem funkcije ***enumerate()*** može se odmah dobiti indeks neke iterabilne promenljive prilikom prolaska kroz petlju.

Primer:

Napisati program koji će prolaziti kroz petlju za sve elemente liste ***predmeti***. Unutar petlje ispisivati indeks i vrednost elementa.

Najpre uraditi zadatak bez ***enumerate*** funkcije, a potom sa ***enumerate*** funkcijom.

```
predmeti = ['CS324', 'CS220', 'CS225', 'SE325', 'IT331']

# bez enumarate
index = 0
for item in predmeti:
    print(f'{index} {item}')
    index += 1

# sa enumerate

for index, item in enumerate(predmeti):
    print(f'{index} {item}')
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZIP FUNKCIJA

Zip objekat nije moguće direktno odštampati, već je potrebno pretvoriti u listu.

Kada se potrebno da grupišemo elemente više listi, tako da se *n*-ti element jedne liste mapira sa *n*-tim elementom druge liste, može se koristiti funkcija ***zip().***

Funkcija zip napraviće zip objekat sa *tuple*-ovima koji sadrže elemente listi koje su parametri funkcije. Mapiranje se vrši dok se ne istroše elementi najkraće liste.

Zip objekat nije moguće direktno odštampati, već je potrebno pretvoriti u listu.

Primer:

Napraviti dve liste sa istim brojem elemenata. Štampati pojedinačne elemente kroz jednu *for petlju* korišćenjem ***zip*** funkcije nad listama.

```
# zip funkcija

predmeti = ['CS324', 'CS220', 'CS225', 'SE325', 'IT331']
ime_predmeta = ['Skripting jezici', 'Arhitektura racunara', 'Operativni sistemi',
'Upravljanje projektima razvoja softvera', 'Racunarke mreze i komunikacije']

# bez enumarate
for predmet, ime in zip(predmeti, ime_predmeta):
    print(f'{predmet} - {ime}')

print(zip(predmeti, ime_predmeta))
print(list(zip(predmeti, ime_predmeta)))
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

FUNKCIJE HELP() I DIR()

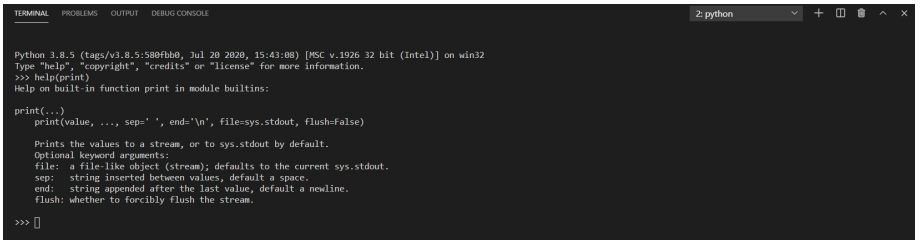
Funkcija help() omogućavaju programeru da nauče kako se radi sa nekim objektom, dok funkcija dir() ispisuje koje su funkcije dostupne u okviru modula ili paketa.

Funkcija help()

U Python konzoli moguće je potražiti pomoć oko specifičnog objekta.

Potrebno je u Python konzoli (ne u standardnom terminalu) ukucati:

```
>>> help(object)
```



Slika 2.1 Funkcija help(). [Izvor: Autor]

Kao izlaz, vratiće se objašnjenje objekta (ukoliko postoji)

Funkcija dir()

U Python konzoli moguće je naći koje su sve funkcije dostupne u okviru nekog paketa ili modula.

Potrenbno je najpre učitati (uvesti) željeni paket i/ili modul, i onda pozvati funkciju dir():

```
>>> import module
>>> dir(module)
```

Slika 2.2 Funkcije dir(). [Izvor: Autor]

Izlaz ove funkcije ispisaće kao listu sve funkcije koje je moguće pozvati kada se modul ili paket učita.

Poglavlje 3

Česti propusti pri programiranju u Python jeziku

UVOD U ČESTE PROPUSTE

Čest greške pri programiranju u Python jeziku jesu posledica prelaska sa drugog programskog jezika u Python, ili prelazak sa Python 2.x u Python 3.x.

Pri prelasku sa drugog programskog jezika na Python, ili pri prelasku sa Python 2.x na Python 3.x jezik, treba obratiti pažnju na greške koje se često dešavaju, a možda nisu odmah uočljive.

U nastavu su dati neki od čestih propusta pri programiranju u Python 3.x programskom jeziku:

- Konflikti u imenovanju
- Promenljivi podrazumevani argumenti
- Uvoz kompletnog modula ili paketa u lokalni imenski prostor

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

KONFLIKTI U IMENOVANJU DATOTEKA

Nekad python interpreter javlja grešku ukoliko je ime datoteke isto kao i ime modula standardne biblioteke.

Python standardna biblioteka sadrži veliki broj modula koji je moguće uvesti u radnu datoteku.

Jedan od čestih problema koji se javlja prilikom rada sa dodatnim modulima jeste čuvanje same datoteke kao ime modula koji se uvozi.

Primer:

U radnu datoteku uvesti modul math, i funkcije sin() i radians(). Napisati program koji računa sinus od datog ugla (unetog u stepenima), i sačuvati datoteku kao math.py. Diskutovati rezultate.

Napomena:

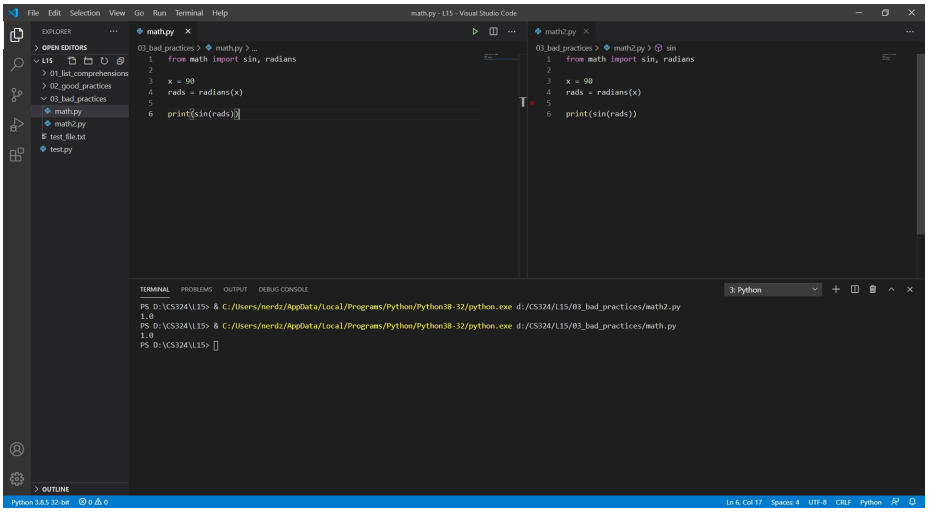
Python neće uvek javiti grešku o konfliktnim imenima. Zavisiće i od IDE-a i od verzije interpretera.

```
# datoteka se cuva kao math.py

from math import sin, radians

x = 90
rads = radians(x)

print(sin(rads))
```



Slika 3.1 Konflikti u imenovanju datoteka. [Izvor: Autor]

PROMENLJIVI PODRAZUMEVANI ARGUMENTI

Podrazumevani argumenti funkcije se izvršavaju samo jednom, i to u trenutku kreacije same funkcije.

Treba voditi računa o funkcijama koje imaju prazne iterabilne objekte kao podrazumevani argument. Ovakvi objekti jesu promenljivi ili mutabilni (en. mutable)

Primer:

Napisati funkciju ***stampaj_predmet(predmet, spisak=[])*** koja dodaje ***predmet*** na ***spisak*** i štampa ***spisak***.

Napraviti listu sa nekoliko predmeta i pozvati funkciju sa novim predmetom.

Zatim, pozvati funkciju više puta bez drugog argumenta. Diskutovati rezultate.

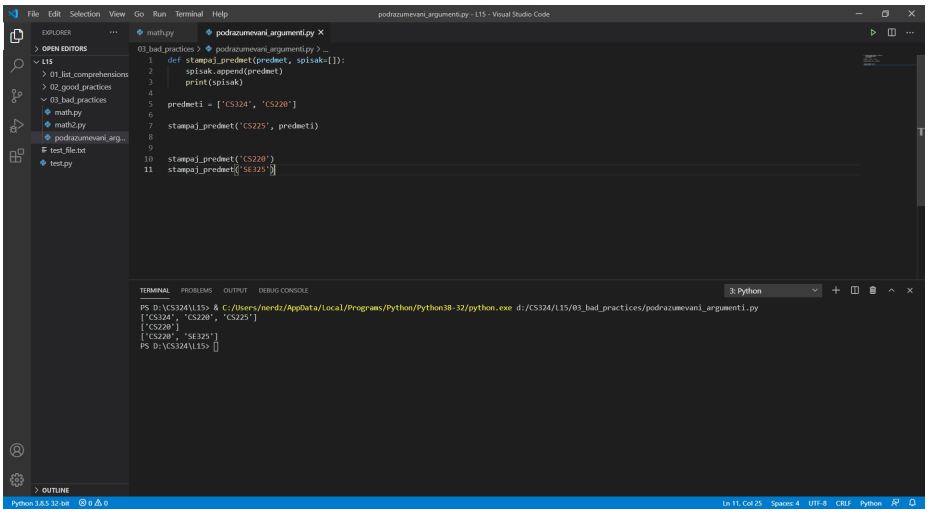
```
def stampaj_predmet(predmet, spisak=[]):
    spisak.append(predmet)
    print(spisak)
```

```
predmeti = ['CS324', 'CS220']

stampaj_predmet('CS225', predmeti)


stampaj_predmet('CS220')
stampaj_predmet('SE325')
```

U Python programskom jeziku, podrazumevani argumenti (argumenti funkcije koji imaju podrazumevanu vrednost) se izvršavaju **jednom,** u trenutku kreacije same funkcije.



Slika 3.2 Mutabilni podrazumevani argumenti. [Izvor: Autor]

UVOZ CELOG MODULA U LOKALNI IMENSKI PROSTOR

Uvoz celog modula ili paketa u lokalni imenski prostor treba koristiti samo kada je neophodno.

Prilikom uvoza paketa i modula u radnu datoteku, treba biti obazriv ukoliko se ceo paket ili modul uveze u imenski prostor.

Komanda za uvoz celog paketa ili modula je:

```
from module import *
```

Ukoliko se uvozi veći broj paketa ili modula, nije uvek jednostavno naći koja funkcija ili klasa je iz kog paketa ili modula.

Takođe, pojedini moduli imaju iste funkcije, i može se desiti greška da se želi koristiti funkcija iz jednog paketa, a zapravo se koristi funkcija drugog.

Primer:

Uvesti sve iz paket numpy, pa onda iz modula math.

Napraviti numpy niz ***a***, sa elementima koji su od 1 do 10 sa korakom 1. Napraviti novi niz ***b***, čiji su elementi ***sqrt()*** od niza ***a***. Štampati niz ***b***. Diskutovati rezultate.

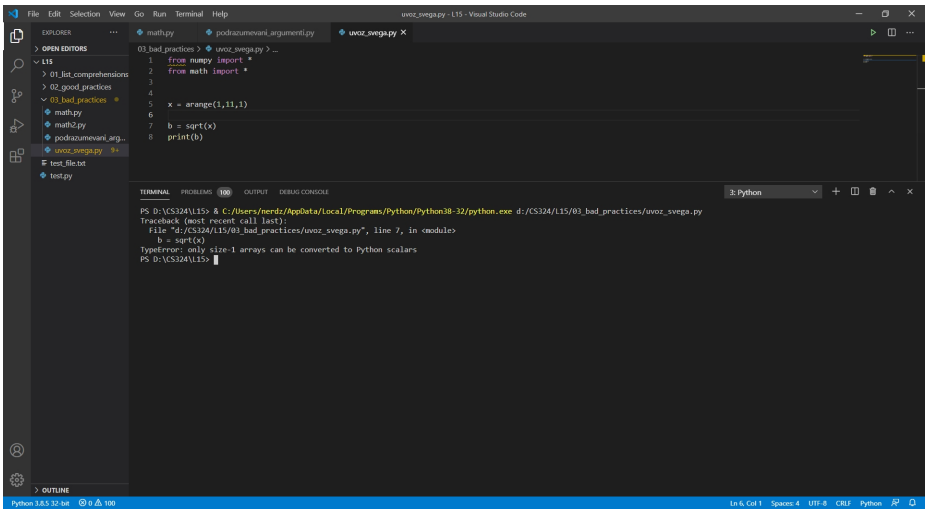
Napomena:

Neki IDE javiće da se ne koriste sve funckije koje su uvezene iz datog modula ili paketa.

```
from numpy import *
from math import *


x = arange(1,11,1)

b = sqrt(x)
print(b)
```



Slika 3.3 Greška pri uvozu svih funkcija u imenski prostor. [Izvor: Autor]

Poglavlje 4

Ostali popularni Python paketi

MODULARNOST = POPULARNOST

Osobina modularnosti ono što čini Python jezik toliko popularnim.

Slika 4.1 Modularnost Python jezika. [Izvor: python.org]

Python poseduje pregršt modula i paketa za različitu primenu. Upravo je osobina modularnosti ono što čini Python jezik toliko popularnim.

U nastavku biće reči o popularnim paketima koji nisu obrađivani u predmetu.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PAKETI ZA MAŠINSKO UČENJE

Najpopularniji paketi za mašinsko učenje i neuronske mreže jesu tensorflow, keras i pytouch

Slika 4.2 TensorFlow logo. [Izvor: https://www.tensorflow.org]

TensorFlow

TensorFlow je biblioteka koju je razvio Google Brain Team, i deo je svake aplikacije mašinskog učenja koju je Google razvio.

TensorFlow radi na principu velikog broja tenzorskih operacija, prvenstveno zbog neuronskih mreža koje se mogu lako predstaviti kao niz tenzorskih grafova.
https://www.tensorflow.org

Slika 4.3 Keras. [Izvor: https://keras.io]

Keras

Keras pruža mehanizam za predstavljanje modela neuronskih mreža, kao i alate za predstavljanje modela, procesiranja skupova podataka, vizualizaciju i prikaz podataka.

Keras se koristi u tandemu sa TensorFlow paketom, koji je zadužen za logiku mašinskog učenja.

https://keras.io

Slika 4.4 PyTorch. [Izvor: https://pytorch.org]

PyTorch

PyTorch prestavlja najveću biblioteku za mašinsko učenje za Python koje dozvoljava korišćenje grafičkog procesora (GPU) za ubrazano dobijanje rezultata. PyTorch podržava kreaciju dinamičkih grafova i automatko računanje gradijenata u algoritmima mašinskog učenja.

https://pytorch.org

PAKET ZA VEŠTAČKU INTELIGENCIJU

Popularni paketi za veštačku inteligenciju i duboko učenje jesu pybrain, caffe2 i hebel.

Slika 4.5 PyBrain. [Izvor: http://pybrain.org]

PyBrain

PyBrain sadrži algoritme za neuronske mreže koje se lako mogu koristiti i istrenirati bez velikog predznanja o načinu funkcionisanja neuronskih mreža. Cilj PyBrain paketa jesta napraviti jednostavne ali moćne algoritme za mešinsko učenje sa već generisanim okruženjem za testiranje i poređenje.

http://pybrain.org

Slika 4.6 Caffe2. [Izvor: https://caffe2.ai]

Caffe2

Caffe2 predstavlja radni okvir za duboko učenje koji je skalabilan i modularan. Pruža jednostavan način za eksperimentisanje sa algoritmima dubokog učenja i poseduje API-je za C++ i Python jezike.
https://caffe2.ai

Slika 4.7 Hebel. [Izvor: https://pypi.org/project/Hebel/]

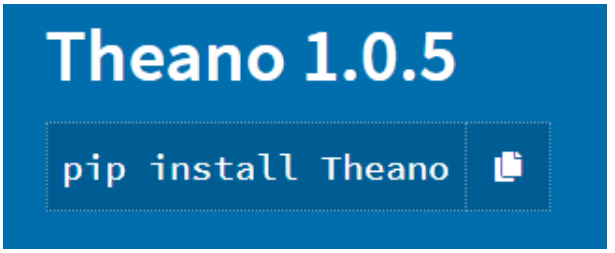
Hebel

Hebel predstavlja Pyzhon biblioteku za duboko učenje koje koristi GPU ubrzanje sa CUDA platformom i pyCUDA paketom. Hebel implementira neuroneke mreže za algoritme klasifikacije i regresije na jedan ili na više zadataka.

https://pypi.org/project/Hebel/

MATEMATIČKI PAKETI

Najpopularniji matematički paketi koji nisu obrađivani jesu statsmodels i seaborn, kao i theano.



Slika 4.8 Theano.[Izvor: https://pypi.org/project/Theano/]

Theano

Theano predstavlja biblioteku koja omogućava efikasno definisanje, optimizaciju i evaluaciju matematičkih izraza koji sadrže *n*-dimenzionalne nizove.

Theano prepoznaje nestabilne izraze i izvršava ih stabilnim algoritmima, što drugi paketi, kao što je NumPy, ne nude.

https://pypi.org/project/Theano/

Slika 4.9 Statsmodels. [Izvor: https://www.statsmodels.org]

Statsmodels

Statsmodels predstavlja modul koji pruža klase i funkcije za estimaciju različitih statističkih modela, kao i za statističke testove.

https://www.statsmodels.org

Slika 4.10 Seaborn. [Izvor: https://seaborn.pydata.org]

Seaborn

Seaborn predstavlja paket za vizuelizaciju statističkih veličina, koji pruža više mogućnosti od paketa od kojeg je izveden - matplotlib. Seaborn je blisko integrisan sa Pandas paketom i pandas DataFrame strukturama.

https://seaborn.pydata.org

WEB PAKETI

Najpopularniji web radni okviri jesu django, tornado i pyramid.

Slika 4.11 Django. [Izvor: https://www.djangoproject.com]

Django

Django predstavlja najpopularniji radni okvir za razvog veb aplikacija u Python jeziku. Django poseduje sve potrebne elemente za razvoj kompletne veb aplikacije, uključujući autentifikaciju, rute, šablone, migraciju baza kao i mapiranje objekata.
https://www.djangoproject.com

Slika 4.12 Tornado. [Izvor: https://www.tornadoweb.org/en/latest/]

Tornado

Tornado predstavlja radni okvir za razvoj veb aplikacija koji je asinhronog tipa, i predstavlja mikroservis koji omogućava veliki broj istovremenih konekcija. Tornado se zasniva na mogućnosti obrade velikog saobraćaja.

https://www.tornadoweb.org/en/latest/

Slika 4.13 [Pyramid. Izvor:https://trypyramid.com]

Pyramid

Pyramid predstavlja drugi po popularnosti radni okvir za razvoj veb aplikacija. Cilj pyramid-a jeste da se učini što više, sa što manje kompleksnosti. Pyramid zahteva Python 3 i ne može raditi na ranijim verzijama.

https://trypyramid.com

GUI PAKETI

Među popularnijim GUI paketima jesu tkinter i pyqt.

Slika 4.14 Tkinter. [Izvor: https://docs.python.org/3/library/tkinter.html]

Tkinter

Tkinter predstavlja podrazumevani radni okvir za razvoj aplikacija sa GUI-jem. TKinter je popularan zvog svoje jednostavnosti i korisničkog interfejsa.

https://docs.python.org/3/library/tkinter.html

Slika 4.15 PyQT. [Izvor: https://riverbankcomputing.com/software/pyqt/intro]

PyQT

PyQT predstavlja QT biblioteku za Python, i služi za razvoj QT aplikacija. PyQT je podržan na različite platforme (Windows, Unix/Linux, MacOS)

https://riverbankcomputing.com/software/pyqt/intro

Poglavlje 5

Pregled predmeta

OBRAĐENE TEME

Teme koje su obrađene na ovom predmetu predstavljaju dovoljnu osnovu za svakog ko želi da nastavi da se bavi programiranjem u Python jeziku,

Tokom semestra na predmetu CS324 - Skripting jezici, obrađene su sledeće teme:

- Osnove Python 3.x programskog jezika
- Sintaksa Python programskog jezika
  - Promenljive
  - Funkcije
  - Standardni ulaz/izlaz
  - Uslovna grananja
  - Petlje
- Proceduralno programiranje u Python jeziku
  - Korisničke funkcije
  - Strukture podataka
  - Standardni ulaz-izlaz (konzola)
- Objektno-orjentisano programiranje u Python jeziku
  - Klase i objekti
  - Atributi i metode
  - Nasleđivanje u Python jeziku
- Rad sa datotekama
- Rad sa bazama podataka
- Obrada izuzetaka
- Debug-ovanje u različitim IDE-ima

- Moduli, paketi i biblioteke u Python jeziku
- Standardna biblioteka
- Ugrađeni paketi u Python jeziku
- Paket NumPy za numeričko programiranje
- Paket pandas za rad sa tabelama
- Paket matplotlib za prikaz podataka
- Paket Pillow za rad sa slikama
- Paket SciPy i SciPy ekosistem
- Python i Data Science
- Python i mašinsko učenje - paket scikit-learn
- Python i razvoj veb aplikacija - paket Flask

Rečeno da je Python jezik koji se može primeniti na veliki broj oblasti upravo zbog svoje modularnosti.

Mnogi od popularnih paketa (prvenstveno za razvoj veb aplikacija) za Python nisu obrađeni na ovom predmetu, jer je nekada obrada jednog paketa toliko obimna da zahteva sopstveni predmet.

Ipak, teme koje su obrađene na ovom predmetu predstavljaju dovoljnu osnovu za svakog ko želi da nastavi da se bavi programiranjem u Python jeziku, i pruža dovoljno znanja za samostalno usavršavanje.

UNAPREĐENJE PREDMETA

Popularnost Python programskog jezika čine upravo paketi koji su savremeni i prave razvoj tehnologija.

Popularnost Python programskog jezika čine upravo paketi koji su savremeni i prave razvoj tehnologija.

Da bi se predmet CS324 - Skripting jezici poboljšao, potrebno je popuniti Google formu.

Iako je potrebna prijava sa Metropolitan nalogom, forma je anonimna.
https://forms.gle/8F7yikzJm8g8XSs69

Forma sadrži pitanja o tome koje su teme u okviru predmeta bile više i manje zanimljive, šta je nedostajalo u predmetu, kao i na to na šta treba više, a na šta manje obratiti pažnju.

Poglavlje 6

Pokazne vežba #15

OPIS TEORETSKOG DELA ISPITA

Na teoretskom delu ispita mogu se javiti više tipova pitanja.

- Esejska (duža) pitanja:

Ovakva pitanja zahtevaju od studenata da daju esejski odgovor od jednog do dva pasusa. Ukoliko je potrebno nacrtati i sliku ili dijagram uz odgovor, obično će se naglasiti.

- Esejska (kraća) pitanja:

Ovakva pitanja najčešće zahtevaju od studenata da nabroje sve ili nekoliko stavki vezane za neku temu. Ukoliko je potrebno nacrtati i sliku ili dijagram uz odgovor, obično će se naglasiti.

- Analizirati kod:

Kod ovakvih pitanja od studenata se očekuje da bez računara uspešno analizira kratak kod napisan u Python jeziku i da odgovori na prateća pitanja. Ponekad je potrebno i ručno napisati deo koda ili pseudokoda.

- Pronaći grešku u kodu:

Kod ovakvih pitanja od studenata se očekuje da bez računara pronađe grešku u delu koda koji je zadat, da objasni zbog čega će taj deo koda javiti grešku. Ponekad je potrebno i ručno napisati ispravan kod.

- Pitanja na zaokruživanje:

Kod ovih pitanja student treba zaokružiti tačan odgovor od više ponuđenih odgovora. Zaokruživanje pogrešnog odgovora ne donosi negativne poene.

- Matematičko pitanje ili zadatak iz oblasti rađene u nekoj od lekcija:

Ukoliko je neka od lekcija imala matematički deo (eksplicitna matematička notacija, izvođenje izraza i sl.), od studenta se očekuje da bez korišćenja literature uradi pitanje ili zadatak.

OPIS PRAKTIČNOG DELA ISPITA

Praktični deo ispita sastoji ili iz većeg broja kraćih, ili manjeg broja dužih zadataka.

Kod praktičnog dela ispita, student radi zadatke u izabranom Python okruženju instaliranom na računaru. Svi zadaci mogu se odraditi u bilo kom razvojnom okruženju.

Studenti mogu koristiti ugrađene opise funkcija u samom IDE-u. Ukoliko se koristi poseban eksterni modul (NumPy, matplotlib, pandas), onda student dobija i spisak najčešće korišćenih komandi iz tog paketa.

Prilikom izrade zadataka, kod treba imati komentare koji prate tok zadatka.

Za pojedine zadatke koje traže daju izlaz, moguće je dati kao primer ulazne podatke, kao i odgovarajuće izlazne podatke.

Datoteke izvornog koda uz odgovarajuće screenshot-ove, treba smestiti u jedan direktorijum, zipovatii poslati predmetnom asistentu i profesoru.

Ukoliko postoji rad sa datotekama treba i sve datoteke poslati zajedno sa izvornim kodom i screenshot-ovima.

- Kratak zadatak koji ispituje jednu funkcionalnost

Kod ovakvih zadataka od studenata se očekuje da napišu program koji će izvršavati datu (najčešće jednu) funkcionalnost.

- Duži zadatak koji ispituje više funkcionalnosti

Kod ovakvih zadataka od studenata se očekuje da napišu program ili programe koji će izvršavati jednu ili više funkcionalnosti, najčešće kao celina.

Praktični deo ispita sastoji ili iz većeg broja kraćih, ili manjeg broja dužih zadataka.

Poglavlje 7

Individualne vežbe

INDIVIDUALNE VEŽBE #15

Individualne vežbe #15 traju 2 časa (90 minuta)

Individualne vežbe #15 predviđene su za usmenu odbranu projekata, kao i za diskusiju o prethodnim pokaznim i individualnim vežbama.

Poglavlje 8

Domaći zadatak

DOMAĆI ZADATAK #15

Domaći zadatak #15 okvirno se radi 2h

Domaći zadatak #15 daje se u dogovoru sa predmetnim nastavnikom i/ili asistentom.

Predaja domaćeg zadatka:

Tradicionalni studenti:

Domaći zadatak treba dostaviti najkasnije 7 dana nakon predavanja, za 100% poena. Nakon toga poeni se umanjuju za 50%.

Internet studenti:

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganje ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.
Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

Poglavlje 9

Zaključak

ZAKLJUČAK

Zaključak lekcije #15

Rezime:

U 15. lekciji najpre obrađene su napredne teme razumevanje listi i imenika. Korišćenjem ovakve sintakse različite for petlje se mogu napisati uglavnom u jednom redu.

Zatim, obrađene su dobre prakse programiranja u Python jeziku, tzv. tips & tricks, koji mogu olakšati svakodnevni rad pri programiranju u Python jeziku.

Nakon dobrih praksi, obrađeno je nekoliko čestih propusta koji programeri u Python jeziku prave.

Konačno, navedeni su popularni paketi u Python jeziku koji nisu detaljno obrađivani u predmetu.

Na kraju je dat pregled predmeta, kao i forma za studente u cilju poboljšanja predmeta.

Literatura:

- David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, Learning Python, 5th Edition, O’Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
- Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.