



CS324 - SKRIPTING JEZICI

Python biblioteke i moduli

Lekcija 07

PRIRUČNIK ZA STUDENTE

CS324 - SKRIPTING JEZICI

Lekcija 07

PYTHON BIBLIOTEKE I MODULI

- ✓ Python biblioteke i moduli
- ✓ Poglavlje 1: Uvod u biblioteke, pakete i module
- ✓ Poglavlje 2: Menadžer Python paketa - PIP
- ✓ Poglavlje 3: Paketi math i datetime
- ✓ Poglavlje 4: Regularni izrazi u Python-u – Paket re
- ✓ Poglavlje 5: Python i JSON datoteke – Paket json
- ✓ Poglavlje 6: Python i relacione baze podataka - Paket sqlite
- ✓ Poglavlje 7: Pokazna vežba #7
- ✓ Poglavlje 8: Invividualna vežba #7
- ✓ Poglavlje 9: Domaći zadatak #7
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

❖ Uvod

UVOD

Python poseduje pregršt besplatnih biblioteka i modula za specifične svrhe koje se lako mogu instalirati i koristiti.

U sedmoj lekciji biće reči o modulima i bibliotekama koje Python koristi.

Ono što je specifično za jedan scripting jezik kao što je Python, jeste da poseduje pregršt besplatnih biblioteka i modula za specifične svrhe. Upravo je raznovrsnost biblioteka i modula učinilo Python toliko popularnim za opštu primenu. Pre svega, potrebno je znati razliku između *paketa* i *modula*. Bilo koja python datoteka predstavlja jedan modul, i ime tog modula je isto kao i ime datoteke (bez .py ekstenzije). Paket predstavlja skup odnosno kolekciju modula. Paket kao takav je zapravo direktorijum Python modula koji takođe sadrži i dodatnu datoteku za inicijalizaciju, da bi se paket razlikovao od običnog direktorijuma koji sadrži Python datoteke.

Moduli i paketi se mogu jednostavno dodati koristeći python instalater paketa, ili skraćeno PIP. Korišćenjem PIP modula vrlo lako se mogu ubaciti dodatni paketi, i to direktno iz komandne linije Pythona. Kada je potrebno da se neki modul ili paket da uveze u trenutnu datoteku, koristi se komanda import, posle koje se navede ime paketa. Paket se može i skraćeno nazvati u trenutnoj datoteci, i kao takav koristiti u nastavku. U nastavku lekcije biće obrađeni paketi, odnosno moduli za matematiku i datum i vreme, za rad sa JASON datotekama, kao i za povezivanje sa relacionim bazama podataka.

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

▼ Poglavlje 1

Uvod u biblioteke, pakete i module

PYTHON PROGRAMSKO JEZIK I OSOBINA MODULARNOSTI

Python poseduje mogućnost da se radi sa konkretnim funkcijama, metodama, pa i celim radnim okvirima prilikom pisanja programa za specifičnu primenu.

Prilikom pisanja programa u bilo kom programskom jeziku (pa i u Python programskom jeziku), veoma je neefikasno uvek pisati sve funkcionalnosti programa u jednoj izvornoj datoteci.

Podelom funkcionalnosti programa na više datoteka uvodi se strukture u pisanju i organizaciju programa. Na ovaj način jednostavnije i efikasnije je pisati glavni program, dok se pojedine funkcionalnosti mogu pozvati iz eksternih programa.

Može se reći da je srž Python jezika upravo osobina modularnosti, tj. mogućnost da se radi sa konkretnim funkcijama, metodama, pa i celim radnim okvirima prilikom pisanja programa za specifičnu primenu.

Ovu osobinu Python je nasledio od MODULA-3 programskog jezika, koji se upravo i bazira na modularnosti.

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

MODUL U PYTHON JEZIKU

U Python programskom jeziku, svaka Python datoteka sa ekstenzijom .py predstavlja jedan modul.

Modul (en. **module**) predstavlja deo softvera koji poseduje specifičnu funkcionalnost.

U Python programskom jeziku, svaka Python datoteka (datoteka sa ekstenzijom .py) predstavlja jedan modul. Ime modula biće isto kao i ime datoteka.

Modul, kao i svaka druga Python datoteka, može imati promenljive, funkcije, klase sa atributima i metodama koje su već definisane i implementirane.

Uvoz modula

Modul se može uvesti korišćenjem ključne reči import, nakon čega se navodi ime datoteke koje se uvozi. Postoje različiti načini uvoza modula

Modul se ceo može uvesti, i to bez preimenovanja. Tada je, prilikom poziva dela koda iz tog modula, u glavnom programu navedemo i ime modula, pa deo koda koji uvozimo. Ovaj pristup može biti složen ukoliko je ime modula dugačko ili ako se ne nalazi u istom direktorijumu. Zbog toga se može preimenovati modul zbog lakoće pisanja.

Modul ili deo modula se može uvesti u imenski prostor glavnog programa, i onda nije neophodno navesti ime modula prilikom svakog poziva dela koda.

Primer: Uvoz modula (15 minuta)

Napisati program koji će u sebi sadržati dve funkcije:

- d6roller() - vraća nasumičnu celobrojnu vrednost od 1 do 6
- d20roller() - vraća nasumičnu celobrojnu vrednost od 1 do 20

Takođe, u ovom programu je definisana i klasa dice_roller(broj_strana) sa sledećim metodama:

konstruktor sa parametrom o broju strana

- roll() - vraća nasumičnu celobrojnu vrednost od 1 do unetog broja strana
- roll_many(broj_bacanja) - vraća zbir višestrukog bacanja
- roll_adv() - vraća veći rezultat od dva bacanja
- roll_dis() - vraća manji rezultat od dva bacanja

U odvojenoj datoteci uvesti:

- ceo modul bez preimenovanja
- ceo modul sa imenovanjem
- samo klasu dice_roller (u postojeći imenski prostor)
- ceo modul u postojeći imenski prostor

UVOZ MODULA - PRIMER

Primer za uvoz modula u zasebnu datoteku

```
# datoteka za import modula
import random

# funkcija za simulaciju bacanje kockica
def d6roller():
    return random.randint(1,6)

# funkcija za simulaciju bacanje 20-strane kockice
def d20roller():
    return random.randint(1,20)

# klasa za bacanja razlicitih kockica
```

```
class dice_roller():

    def __init__(self, number_of_sides):
        self.number_of_sides = number_of_sides

    def roll(self):
        return random.randint(1, self.number_of_sides)

    def roll_many(self, number_of_rolls):
        self.result = 0
        self.number_of_rolls = number_of_rolls
        for _ in range(self.number_of_rolls):
            self.result += random.randint(1, self.number_of_sides)
        return self.result

    def roll_adv(self):
        return max(random.randint(1, self.number_of_sides), random.randint(1,
self.number_of_sides))

    def roll_dis(self):
        return min(random.randint(1, self.number_of_sides), random.randint(1,
self.number_of_sides))
```

```
# uvoz bez preimenovanja
import cs324_L07_01_import_modula
print(cs324_L07_01_import_modula.d6roller())

# uvoz sa preimenovanjem
import cs324_L07_01_import_modula as dices
print(dices.d20roller())

# uvoz konkretnog dela koda u trenutni imenski prostor
from cs324_L07_01_import_modula import dice_roller
x = dice_roller(20)
print(x.roll())

# uvoz svega u trenutni imenski prostor
from cs324_L07_01_import_modula import *
print(d20roller())
```

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

PAKET U PYTHON JEZIKU

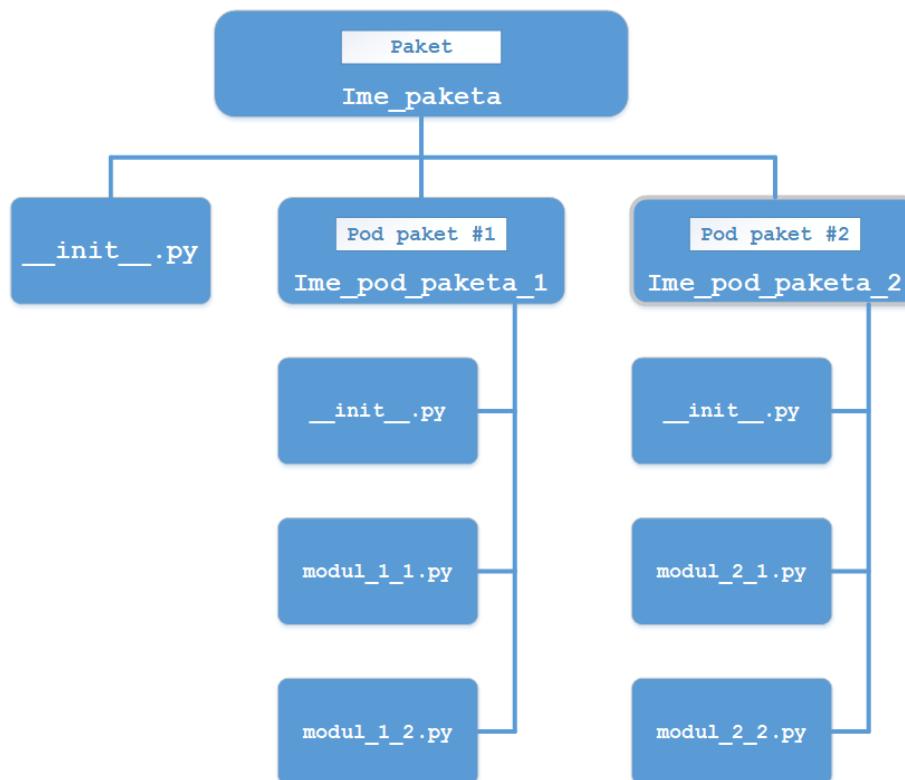
Paket u Python jeziku predstavlja kolekciju modula unutar direktorijuma, a direktorijum mora sadržati konstruktor paketa.

Paket (en. **package**) u Python jeziku predstavlja kolekciju modula unutar direktorijuma.

Treba istaći da bilo koji direktorijum sa Python modulima (direktorijum sa datotekama sa .py ekstenzijom) ne predstavlja modul sam po sebi, već mora ispuniti dodatan uslov da poseduje datoteku konstruktora modula, koja će biti jedna prazna `__init__.py` datoteka. Na taj način Python interpreter razumeće da je u pitanju paket, koji se može uvesti.

Kao i kod strukture direktorijuma, i paketi mogu sadržati pod-pakete (en. sub-packet), ali unutar svakog paketskog direktorijuma treba postojati paketni konstruktor.

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



Slika 1.1 Struktura Python paketa i modula. [Izvor: Autor]

UVOZ UGRAĐENOG PAKETA

Python interpreter uvek će prepoznati ugrađene pakete koji uvek može uvesti.

Python 3 jezik dolazi sa mnoštvo ugrađenih paketa za specifične svrhe.

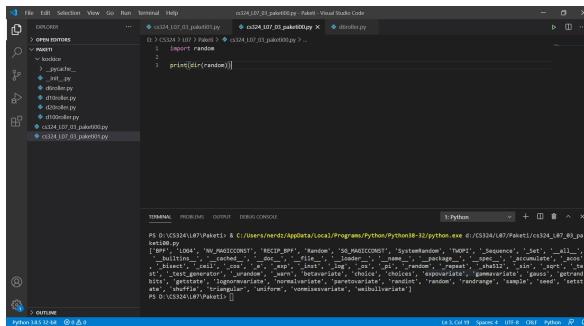
Kod ugrađenih paketa, iako se ne nalaze u direktorijumu u kome je glavni program, Python interpreter prepoznaće da se radi o ugrađenim paketima i moći će da ih uveze uvek.

Do sada su obrađeni primeri koji su uvozili paket `math` (za rad sa matematičkim funkcijama) i paket `random` (za rad sa nasumičnim brojevima).

Moguće je videti šta od funkcija i klasa sadrži korišćenjem funkcije `dir()`.

Primer: izlistati moguće funkcije paketa radnom. (2 minuta)

```
import random  
  
print(dir(random))
```



Slika 1.2 Lista mogućih funkcija paketa random. [Izvor: Autor]

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

UVOD KORISNIČKOG PAKETA

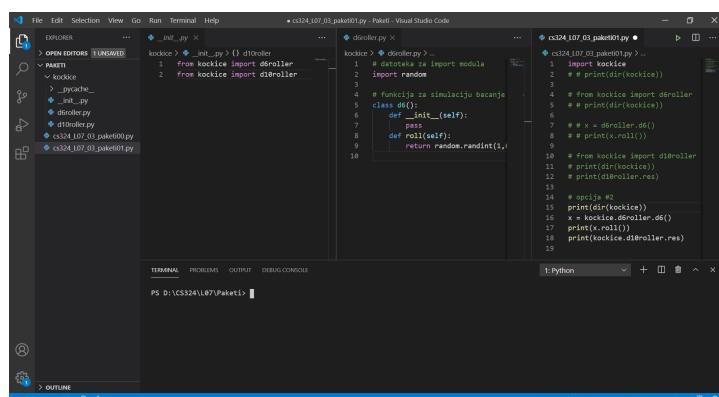
Kod uvoza korisničkih paketa potrebno je uvesti pojedinačne module, a ne samo ime paketa.

Korisnički paket se na vrlo sličan način može uvesti u trenutni radni direktorijum.

Postupak je sledeći:

- Napraviti prazni paketni konstruktor
 - U glavnom programu, korišćenjem naredbe `import`, uvesti korisnički paket po imenu
 - Nakon toga, uvesti pojedinačne module u imenski prostor korišćenjem naredbi `from [package] import [module]`

Alternativno, moguće je unutar paketnog konstruktora navesti pojedinačne naredbe uvoza da bi na taj način korisnički moduli uvek bili dostupni.



Slika 1.3 Uvoz korisničkih modula preko paketa. [Izvor: Autor]

Primer: (10 minuta)

Napraviti paket kockice koje će imati pojedinačne module za vraćanje nasumičnog broja za dve različite kockice. U glavnom programu uvesti paket i isprobati funkcionalnost.

```
# --- moduli paketa kockice (pojedinačne datoteke) ---
# -----
# modul #1 d6roller.py
import random

# funkcija za simulaciju bacanje kockica
class d6():
    def __init__(self):
        pass
    def roll(self):
        return random.randint(1,6)
# -----
# modul #2 d10roller.py
import random
res = random.randint(1,10)
# -----
# glavni program main.py
import kockice
print(dir(kockice))

from kockice import d6roller
x = d6roller.d6()
print(x.roll())

from kockice import d10roller
print(d10roller.res)

print(dir(kockice))
```

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

PYTHON BIBLIOTEKA

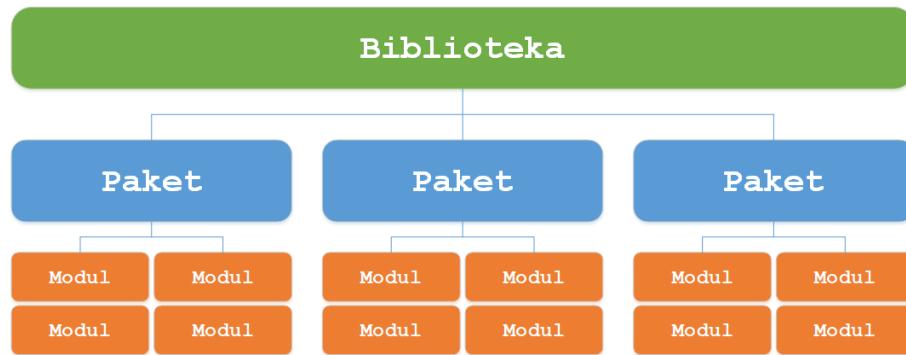
Skup paketa (sa modulima) i dodatnim podešavanjima predstavlja Python biblioteku. Biblioteka koja se koristi prilikom pokretanja Python-a jeste standardna biblioteka.

Kao što je paket predstavlja skup modula, tako i biblioteka (en. library) predstavlja skup paketa.

Moguće je napraviti sopstvenu biblioteku koja bi bila skup paketa i modula, ali i dodatnih podešavanja prilikom pokretanja IDE-a.

Zapravo, podešavao bi se virtuelni prostor (en. virtual environment).

Korisničke biblioteke su izvan skupa ovog predmeta.



Slika 1.4 Odnos modula, paketa i biblioteka u Python programskom jeziku. [Izvor: Autor]

Standardna Python biblioteka

Standardna Python biblioteka se sastoji iz više različitih komponenti. Sadrži najpre tipove podataka koji se mogu smatrati da su "srž" programskog jezika, npr. brojevi i liste.

Biblioteka takođe sadrži ugrađene funkcije i izuzetke - objekte koji se mogu koristiti u svom kodu pisanom na Python jeziku bez potrebe za naredbom import.

Najveći deo biblioteke sadrži skup modula. Neki moduli su napisani na C jeziku i ugrađeni su u sam Python interpreter, dok su neki moduli napisani na Python jeziku i uvoze se u izvornom obliku.

Pojedini moduli služe kao interfejs operativnom sistemu i hardveru, ili specifičnim aplikacionim domenima, kao što je www.

Pojedini moduli dostupni su u svakoj verziji Python instalacije i za svaki operativni sistem, dok su neki dostupni samo kroz određene opcije konfigurisanja. Više o standardnoj biblioteci na stranici:

<https://docs.python.org/3/library/>

✓ Poglavlje 2

Menadžer Python paketa - PIP

UVOD U PIP

PIP menadžer datoteka se poziva u komandnoj liniji i pruža detaljnu kontrolu nad instaliranim paketima u Python jeziku.

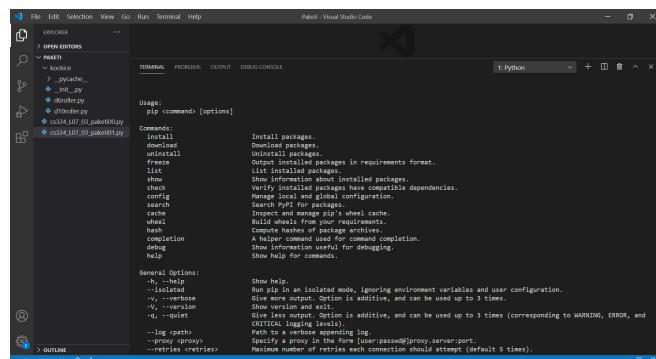
Do sada je bilo reči o ugrađenim i korisnički-definisanim paketima za Python.

Postavlja se pitanje: Kako ubaciti tuđe pakete i kako im pristupiti?

PIP menadžer paketa (en. **PIP - pip installs packets**) jeste softver koji dolazi uz instalaciju Python programskog jezika sa zadatkom da upravlja eksternim paketima.

Koristi se u komandnoj liniji i iako ima samo nekoliko osnovnih komandi, pruža detaljnu kontrolu nad instaliranim paketima. Prilikom instalacije Python jezika, instalira se i **pip**, i može se pristupiti komandom **pip**

```
C:\pip
```



Slika 2.1 Pokretanje pip menadžera u konzoli. [Izvor: Autor]

Osnovne komande

PIP nudi sledeće osnovne komande:

- **install** - instalira pakete
- **download** - preuzima pakete
- **uninstall** - izbacuje pakete
- **freeze** - pokazuje instalirane pakete u formatu potražnje
- **list** - pokazuje instalirane pakete
- **show** - pokazuje informaciju o instaliranom paketu
- **check** - proverava zavisnosti instaliranih paketa

- **config** - upravlja lokalnom i globalnom konfiguracijom
- **search** - pretraga paketa
- **hash** - računa heš vrednosti arhiva paketa
- **completion** - opcija za auto-complete
- **debug** - pokazuje informaciju za debagovanje
- **help** - pomoć

PRETRAGA I INSTALACIJA NOVOG PAKETA

Pretraga, pregled, instalacija novih paketa preko PIP-a je jednostavno i izvršava se svega u nekoliko komandi.

Pretraga paketa

Pretraga paketa vrši se pomoću komande `search`, nakon koje treba navesti ime paketa.

Ukoliko postoji paket sa tim nazivom, pip vraća naziv, verziju, i opis paketa.

```
pip search ime_paketa
```

Instalacija paketa

Instalacija paketa vrši se pomoću komande `install`, nakon koje treba navesti ime paketa.

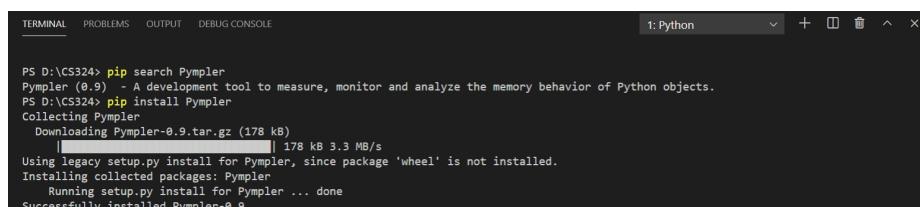
```
pip install ime_paketa
```

Pregled instaliranih paketa

Pregled paketa vrši se pomoću komande `list`, nakon koje pip vraća listu instaliranih paketa u obliku tabele, gde su u jednoj koloni paketi, a u drugoj koloni verzije.

```
pip list
```

Primer: potražiti i instalirati paket Pympler



```
PS D:\CS324> pip search Pympler
Pympler (0.9) - A development tool to measure, monitor and analyze the memory behavior of Python objects.
PS D:\CS324> pip install Pympler
Collecting Pympler
  Downloading Pympler-0.9.tar.gz (178 kB)
    [██████████] 178 kB 3.3 MB/s
Using legacy setup.py install for Pympler, since package 'wheel' is not installed.
Installing collected packages: Pympler
  Running setup.py install for Pympler ... done
Successfully installed Pympler-0.9
```

Slika 2.2 Pretraga i instalacija paketa Pympler. [Izvor: Autor]

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

AŽURIRANJE INSTALIRANIH PAKETA

Provera paketa koji nisu ažurirani vrši se pomoću komande `list`, sa dodanim argumentom `--outdated`.

Paketi se često ažuriraju, te je potrebno znati da li na računaru na kojem se piše program instalirana najnovija verzija paketa.

Provera paketa koji nisu ažurirani vrši se pomoću komande `list`, sa dodanim argumentom `--outdated`:

```
pip list --outdated
```

Sa dodatnim argumentom, `list` komanda vratiće tabelu instaliranih paketa u jednoj koloni, sa izdanjem novije verzije u drugoj koloni.

Package	Version	Latest	Type
colorama	0.4.3	0.4.4	wheel
isort	5.4.2	5.6.4	wheel
lazy-object-proxy	1.4.3	1.5.1	wheel
setuptools	47.1.0	50.3.2	wheel
toml	0.10.1	0.10.2	wheel

Slika 2.3 Lista paketa koji nisu ažurirani. [Izvor: Autor]

Ažuriranje paketa

Ažuriranje paketa moguće je takođe kroz komandu `install`, ali sa dodatnim argumentom `-U`.

```
pip install -U ime_paketa
```

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

PREGLED INSTALIRANIH PAKETA, IZVOZ U DATOTEKU I BRISANJE PAKETA

Dobra praksa jeste imati spisak potrebnih paketa u posebnoj datoteci, ili u komentaru samog programa.

Komanda `list` vraća sve instalirane pakete u obliku tabele, ali nekada nije poželjno na ovaj način predstaviti sve pakete.

Često se dešava da treba javiti koji su sve paketi potrebni da bi se program pokrenuo. Zbog toga je efikasnije koristiti komandu `freeze`, koja, kao i `list`, vraća spisak instaliranih paketa, ali u drugom obliku.

```
pip freeze
```

Dobra praksa jeste izvesti ovako formatiran spisak u posebnu datoteku, što je moguće dodavanjem komande:

```
pip list > izlazna_datoteka.ekstenzija
```

Na ovaj način, datoteka se može dodati u projektnu dokumentaciju, ili i dodati kao komentar u izvornom kodu koji koristi paket, kao komentar.

Brisanje paketa

[Brisanje paketa](#) vrši se pomoću komande `uninstall`, nakon koje treba navesti ime paketa.

```
pip uninstall ime_paketa
```

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

✓ Poglavlje 3

Paketi math i datetime

UVOD U MATH PAKET

Math je ugrađeni paket i nije neophodno instalirati naknadno.

Paket math je ugrađen paket, što znači da nije potrebno naknadno instalirati paket nakon instalacije Python programskog jezika.

Međutim, funkcije paketa math nisu direktno dostupne u imenskom prostoru, te je neophodno koristiti sledeću komandu:

```
# rad sa realnim brojevima
import math
```

Pitanje:

U zadatku koji računa rešenja kvadratne jednačine, zog čega je bilo neophodno koristiti cmath, a ne math?

Ukoliko se radi sa kompleksnim brojevima, potrebno je umesto *math* koristiti paket *cmath*.

```
# rad sa kompleksnim brojevima
import cmath
```

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

UVOD U DATETIME PAKET

Datetime paket predstavlja glavni paket za rad sa datumima i vremenskim zapisima.

Paket datetime jeste glavni paket za rad sa datumima i vremenom. Ovaj paket je veoma bitan prilikom razvoja aplikacija koje imaju u sebi neke vremenske zapise.

Ovaj paket je takođe ugrađen i ne treba se posebno instalirati, ali je potrebno ručno uvesti u imenski prostor.

```
import datetime
```

Rad sa datumima i vremenom je jako bitan u bilo kojoj **real-time** aplikaciji, ali i prilikom rada sa datotekama, kada je potrebno iz datoteke uneti string i parsirati kao datum, ili obrnuto, kada treba datum da se pretvori u string koji treba upisati u datoteku.

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

✓ 3.1 Rad sa math paketom

FUNKCIJE PAKETA MATH

Ugrađeni paket math sadrži matematičke funkcije koje su definisane C standardom

Ugrađeni paket **math** sadrži matematičke funkcije koje su definisane C standardom - funkcije koje su identične (ili jako slične) onima uz C familije jezika.

U nastavku je dato objašnjenje pojedinih funkcija paketa math.

Funkcija ***floor()*** vraća najблиži ceo broj **manji** od unetog broja.

```
# funkcija floor vraca najblizi manji ceo broj
x = 3.4566
print(math.floor(x))
-----
output:
>>> 3
```

Funkcija ***ceil()*** vraća najблиži ceo broj **veći** od unetog broja.

```
# funkcija ceil vraca najblizi veci ceo broj
x = 3.4566
print(math.ceil(x))
-----
output:
>>> 4
```

Funkcija ***sqrt()*** vraća kvadratni koren unetog broja.

```
# funkcija kvadratnog korena
print(math.sqrt(2))
-----
output:
>>> 1.4142135623730951
```

Funkcija ***isqrt()*** vraća kvadratni koren broja, i to najблиži ceo broj manji od rezultata.

```
# funkcija celobrojnog kvadratnog korena
print(math.isqrt(2))
-----
output:
>>> 1
```

Funkcija `pow()` vraća stepen prvog argumenta na stepen drugog argumenta.

```
# funkcija stepenovanja
print(math.pow(2,6))
-----
output:
>>> 64
```

Eksponencijalna funkcija `exp()` i logaritamska funkcija sa proizvoljnom bazom `log()`:

```
# eksponencijalna funkcija
print(math.exp(5))
# logaritamska funkcija
print(math.log(100,10))
```

Broj `pi` i broj `e` se mogu uvesti iz math paketa

```
from math import pi
from math import e
```

PRIMER ZA RAD SA PAKETOM MATH

Sledi primer za rad sa paketom math i trigonometrijskim funkcijama.

Primer (5 minuta):

Napraviti funkciju `trig('f', 'stepeni')`, koja vraća sin, cos, tan, i cot.

f može biti **s**, **c**, **t**, i **ct**, a za ostale unose vraća grešku.

Koristiti math paket. **Napomena:** math trigonometrijske funkcije za ulaz uzimaju vrednost u radijanima.

```
import math

def trig(f, degrees):
    if f == 's':
        return math.sin(math.radians(degrees))
    elif f == 'c':
        return math.cos(math.radians(degrees))
    elif f == 't':
        return math.tan(math.radians(degrees))
    elif f == 'ct':
        return 1/math.tan(math.radians(degrees))
```

```
else:
    print('Treba uneti s, c, t, ili ct')
```

The screenshot shows a Visual Studio Code interface. On the left, the file structure shows a folder 'Lekt' containing 'Lekt', 'Math', and 'trig.py'. The 'trig.py' file contains Python code for trigonometric functions. On the right, a 'Trigonometry Ratio Table' is displayed, showing values for sine, cosine, tangent, cotangent, secant, and cosecant at various angles from 0 to 360 degrees.

```
trig.py
import math

def trig(f, degrees):
    if f == 's':
        return math.sin(math.radians(degrees))
    elif f == 'c':
        return math.cos(math.radians(degrees))
    elif f == 't':
        return math.tan(math.radians(degrees))
    elif f == 'ct':
        print('Treba uneti s, c, t, ili ct')
    else:
        print('Niste uneli s, c, t, ili ct')

# print(trig('s', 90))
# print(trig('c', 90))
# print(trig('t', 90))
# print(trig('ct', 90))
```

Slika 3.1.1 Primer zadatka sa trigonometrijskim funkcijama. [Izvor: Autor]

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

3.2 Rad sa datetime paketom

FUNKCIJE PARSIRANJA DATUMA

Često je potrebno da se string parsira kao datum ili vreme, ili da se datum ili vreme parsira kao datetime objekat.

Pri radu sa datumima i vremenskim zapisima poželjno je koristiti paket datetime. Paket sadrži klase kreirane specifično za rad sa datumima, pravilnim predstavljanjem i konverzijom datuma i vremena, kao i uvid u vremenske zone.

Često je potrebno da se string parsira kao datum ili vreme, ili da se datum ili vreme parsira kao datetime objekat.

Funkcija parsiranja strftime()

Funkcija parsiranja *datetime.strptime(str, directives)* konvertuje string u kome je napisan datum u datetime objekat. Prvi argument jeste sam string, a drugi jeste string sa direktivama koje opisuju format datuma.

```
from datetime import datetime

datum1 = 'November 13, 2020'
datum1_dt_objekat = datetime.strptime(datum1, '%B %d, %Y')
print(datum1_dt_objekat)
```

Direktive formata datuma

Python datetime paket podržava veliki broj formata datuma. Ceo spisak direktiva je moguće naći na <http://strftime.org/>, dok su česti primeri dati u nastavku

Direktiva	Objašnjenje	Primer
<hr/>		
%a	Dan, kraci format.	Mon
%A	Dan, cela rec.	Monday
%w	Radni broj dana u nedelji (pon. je 0). 0	
%d	Dan u mesecu.	30
%b	Mesec, kraci format.	Sep
%B	Mesec, cela rec.	September
%y	Godina, poslednje dve cifre.	21
%Y	Godina, celo broj.	2021

FUNKCIJE PARSIRANJA VREMENA

Parsiranje vremena je, kao i za datume, moguće kroz funkciju datetime.strptime().

Parsiranje vremena

Funkcija `datetime.strptime()` može parsirati i vreme, a ne samo datume.

```
from datetime import datetime

datum1 = 'November 13, 2020'
datum1_dt_objekat = datetime.strptime(datum1, '%B %d, %Y')
print(datum1_dt_objekat)
print(type(datum1_dt_objekat))

vremel1 = '21:52:20'
vremel1_dt_objekat = datetime.strptime(vremel1, '%H:%M:%S')
print(vremel1_dt_objekat)
print(type(vremel1_dt_objekat))

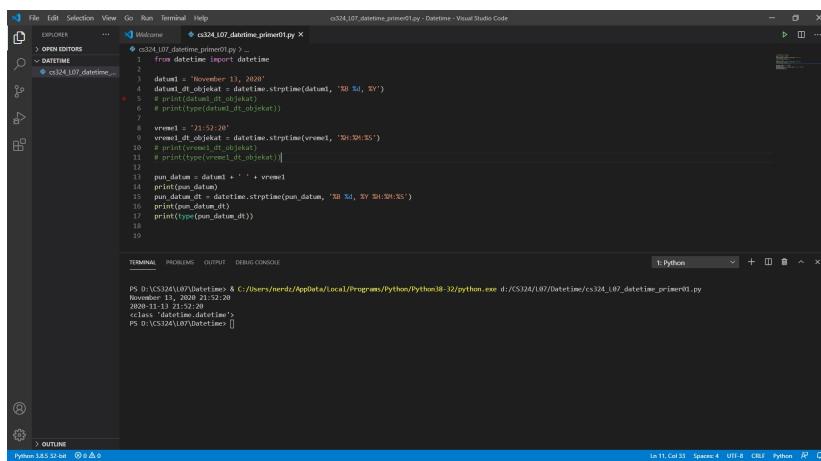
pun_datum = datum1 + ' ' + vremel1
print(pun_datum)
pun_datum_dt = datetime.strptime(pun_datum, '%B %d, %Y %H:%M:%S')
print(pun_datum_dt)
print(type(pun_datum_dt))
```

Direktive za parsiranje vremena

Kao i za datume, postoje direktive za vreme.

Direktiva	Objašnjenje	Primer
<hr/>		
%H	Sat (24h) sa vodecom nulom.	07
%-H	Sat (24h) bez vodeće nule.	7
%I	Sat (12h) sa vodecom nulom.	07
%-I	Sat (12h) bez vodeće nule.	7

%p	Pre podne ili posle podne (AM/PM).	AM
%M	Minuti sa vodecom nulom.	06
%-M	Minuti bez vodece nule.	6
%S	Sekunde sa vodecom nulom.	09
%-S	Sekunde bez vodece nule.	9
%f	Mikrosekunde	000000



Slika 3.2.1 Parsiranje stringova u datum i vreme. [Izvor: Autor]

PARSIRANJE IZ DATUMA U STRING

Funkcija `datetime.strptime()` može parsirati datum i vreme u string.

Parsiranje datuma u string

Funkcija `datetime.strptime()` može parsirati datum i vreme u string.

Na ovaj način moguće je štampati string regularno, i u format koji je definisan direktivama, ali od strane korisnika.

Preporuka:

Koristiti datetime formate unutar programa, a tek pri štampiparsovati u string.

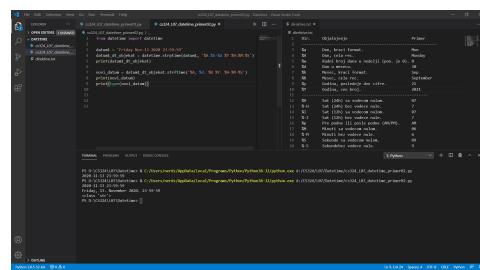
```

from datetime import datetime

datum1 = 'Friday Nov-13 2020 23:59:59'
datum1_dt_objekat = datetime.strptime(datum1, '%A %b-%d %Y %H:%M:%S')
print(datum1_dt_objekat)

novi_datum = datum1_dt_objekat.strftime('%A, %d. %B %Y. %H-%M-%S')
print(novi_datum)
print(type(novi_datum))

```



Slika 3.2.2 Parsiranje datum i vreme u string. [Izvor: Autor]

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

▼ Poglavlje 4

Regularni izrazi u Python-u – Paket re

UVOD U REGULARNE IZRAZE

Pri korišćenju regularnih izraza pretraga teksta ili provera korektnog unosa teksta predstavlja olakšicu pri programiranju.

Pri radu sa tekstualnim podacima često postoji potreba da se izvuku pojedini podaci iz samog teksta.

Češće, potrebno je izvući određen deo teksta koji zadovoljava određen uslov, a koji nije direktno pretraživan.

Umesto direktnog traženja dela teksta u dokumentu, poželjno je koristiti regularne izraze (en. **Regular expressions, Regex**).

Regularni izraz predstavlja niz karaktera koji definišu šablon pretraživanja.

Pri korišćenju regularnih izraza pretraga teksta ili provera korektnog unosa teksta predstavlja olakšicu pri programiranju.

Validacija unosa korektne email adrese ili korektne šifre koja sadrži minimalan broj karaktera ostvarivo je pomoću **regex** izraza.

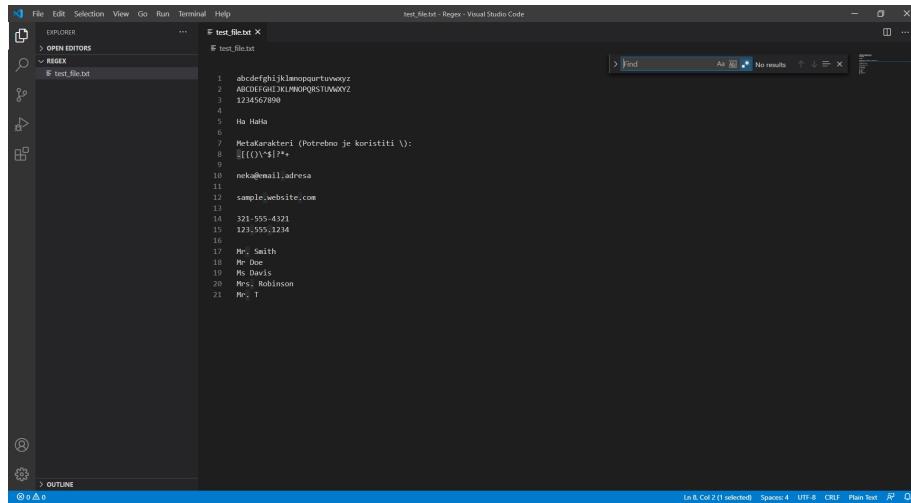
Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRETRAGA KORIŠĆENJEM REGULARNIH IZRAZA

Svaki bolji tekstualni editor ima mogućnost korišćenje regularnih izraza.

Pretraga pomoću regularnih izraza u VSCode

Prilikom pretrage (CTRL+F) potrebno je štiklirati **.*** da bi se pretraga vršila sa regularnim izrazima. Moguće je pritisnuti i ALT+R.



Slika 4.1.1 Pretraga pomoću regularnih izraza. [Izvor: Autor]

Unutar polja za pretragu moguće je uneti regularni izraz koji se traži, korišćenjem ključnih karaktera u nastavku slajda.

```

.      - Any Character Except New Line
\d     - Digit (0-9)
\D     - Not a Digit (0-9)
\w     - Word Character (a-z, A-Z, 0-9, _)
\W     - Not a Word Character
\s     - Whitespace (space, tab, newline)
\S     - Not Whitespace (space, tab, newline)

\b     - Word Boundary
\B     - Not a Word Boundary
^     - Beginning of a String
$     - End of a String

[]    - Matches Characters in brackets
[^ ]  - Matches Characters NOT in brackets
|     - Either Or
( )  - Group

Quantifiers:
*     - 0 or More
+     - 1 or More
?     - 0 or One
{3}   - Exact Number
{3,4} - Range of Numbers (Minimum, Maximum)

```

PRIMERI KORIŠĆENJA REGULARNIH IZRAZA

Dati su primeri korišćenja regularnih izraza za pretragu unutar IDE-a

Pretraga svih brojeva unutar datoteke.

```
\d
```

Pretraga svih karaktera koji nisu brojevi unutar datoteke.

```
\D
```

Pretraga brojeva sa tri cifre nakon koje sledi bilo koji karakter, pa još tri cifre, pa bilo koji karakter, pa proizvoljan broj cifara

```
\d{3}.\d{3}.\d+
```

Pretraga stringova koji počinju sa karakterima "Ha"

```
\bHa
```

Pretraga veb domena i pod-domena, i to web stranica koja se završavaju sa .com

```
[\w.]+\.com
```

```
abcdefghijklmnpqrstuvwxyz  
ABCDEFGHIJKLMNPQRSTUVWXYZ  
1234567890
```

```
Ha HaHa
```

MetaKarakteri (Potrebno je koristiti \):
. [{()^\$|?*+}

```
neka@email.adresa
```

```
sample.website.com
```

```
321-555-4321  
123.555.1234
```

```
Mr. Smith  
Mr Doe  
Ms Davis  
Mrs. Robinson  
Mr. T
```

DETALJNIJE OBJAŠNJENJE KORIŠĆENJA REGULARNIZH IZRAZA U VISUAL STUDIO CODE

Sledi autorski video o korišćenju regularnih izraza u Visual Studio Code razvojnom okruženju.

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

✓ 4.1 Rad sa re paketom

KLASE I FUNKCIJE PAKETA RE

Korišćenjem klase `re.compile` definiše se pretraga, a metodom `.finditer()` se vrši pretraga nad tekstom.

Sirovi stringovi

Nekada je potrebno štampanje stringova koji će uključiti i specijalne karaktere kao što su *tab*, *novi red*, i sl.

Funkcija za štampanje stringova koji pokazuju "nevidljive" karaktere razlikuje se samo u navođenju karaktera '**r**' ispred samog stringa.

Ovo su tzv.[sirovi stringovi](#) (en.[raw strings](#))

```
# Stampanje obicnog stringa
print('\t String koji ne pokazuje tab')

# Stampanje sirovog stringa
print(r'\t String koji pokazuje tab')
```

Pri radu sa regularnim izrazima upotreba sirovih stringova je česta, te je bitno razlikovati štampanje običnih i sirovih stringova.

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

Python koristi [paket re](#) za rad sa regularnim izrazima.

Definisanje šablonu

Pri radu sa regularnim izrazima, nakon uvoza paketa `re`, potrebno je napraviti objekat klase `re.compile()`, čiji je parametar (sirovi) string ili regularni izraz koji se traži.

Pretraga šablonu

Nakon toga, nad objektom se poziva metoda `finditer()`, čiji je parametar tekst koji se pretražuje. Ova metoda jeste iterabilni tip koji će sadržati sve "pogotke" unutar pretrage.

Štampanje šablonu

Nakon pronalaženja šablonu koji je definisan stringom ili regularnim izrazom, moguće je štampanje rezultata. Kako je rezultat prethodne metode iterabilni tip podataka, kroz petlju je moguće stampati sve rezultate.

PRIMER: IZVLAČENJE EMAIL ADRESA IZ DATOTEKE.

Regularni izrazi se mogu koristiti za pretragu dela teksta vrlo efikasno.

Primer: (15 minuta)

Tekst sa strane sačuvati u dokument `tekst_za_pretragu.txt`

Korišćenjem regularnih izraza (re paketa), napisati program koji će pronaći sve email adrese, štampati ih, i sačuvati u novu datoteku, `email.txt`

```
import re

with open('tekst_za_pretragu.txt', 'r') as f:
    text_uvoz = f.read()

sablon = re.compile(r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-\.]+')

pogodak = sablon.finditer(text_uvoz)

for i in pogodak:
    with open('email.txt', 'a') as fw:
        fw.write(i.group(0) + '\n')
```

Objašnjenje šablonu:

```
[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-\.]+
```

String će pretražiti sve alfa-numeričke karaktere, donju i srednju crtlu, (email korisničko ime) nakon kojeg sledi karakter `@`, pa ponovo alfa-numeričke karaktere (domen), pa tačku `.`, pa, pa ponovo alfa-numeričke karaktere (domen najvišeg nivoa) .

```
-----
Tekst za pretragu
-----
Dave Martin
615-555-7164
173 Main St., Springfield RI 55924
davemartin@bogusemail.com

Charles Harris
800-555-5669
969 High St., Atlantis VA 34075
charlesharris@bogusemail.com

Eric Williams
560-555-5153
806 1st St., Faketown AK 86847
laurawilliams@bogusemail.com

Corey Jefferson
900-555-9340
826 Elm St., Epicburg NE 10671
coreyjefferson@bogusemail.com

Jennifer Martin-White
714-555-7405
```

212 Cedar St., Sunnydale CT 74983
jenniferwhite@bogusemail.com

Erick Davis
800-555-6771
519 Washington St., Olympus TN 32425
tomdavis@bogusemail.com

Neil Patterson
783-555-4799
625 Oak St., Dawnstar IL 61914
neilpatterson@bogusemail.com

Laura Jefferson
516-555-4615
890 Main St., Pythonville LA 29947
laurajefferson@bogusemail.com

Maria Johnson
127-555-1867
884 High St., Braavos ME 43597
mariajohnson@bogusemail.com

Michael Arnold
608-555-4938
249 Elm St., Quahog OR 90938
michaelarnold@bogusemail.com

Michael Smith
568-555-6051
619 Park St., Winterfell VA 99000
michaelsmith@bogusemail.com

Erik Stuart
292-555-1875
220 Cedar St., Lakeview NY 87282
robertstuart@bogusemail.com

Laura Martin
900-555-3205
391 High St., Smalltown WY 28362
lauramartin@bogusemail.com

Barbara Martin
614-555-1166
121 Hill St., Braavos UT 92474
barbaramartin@bogusemail.com

Linda Jackson
530-555-2676
433 Elm St., Westworld TX 61967
lindajackson@bogusemail.com

Eric Miller
470-555-2750
838 Main St., Balmora MT 56526
stevemiller@bogusemail.com

Dave Arnold
800-555-6089
732 High St., Valyria KY 97152
davearnold@bogusemail.com

Jennifer Jacobs
880-555-8319
217 High St., Old-town IA 82767
jenniferjacobs@bogusemail.com

Neil Wilson
777-555-8378
191 Main St., Mordor IL 72160
neilwilson@bogusemail.com

Kurt Jackson
998-555-7385
607 Washington St., Blackwater NH 97183
kurtjackson@bogusemail.com

Mary Jacobs
800-555-7100
478 Oak St., Bedrock IA 58176
maryjacobs@bogusemail.com

Michael White
903-555-8277
906 Elm St., Mordor TX 89212
michaelwhite@bogusemail.com

Jennifer Jenkins
196-555-5674
949 Main St., Smalltown SC 96962
jenniferjenkins@bogusemail.com

Sam Wright
900-555-5118
835 Pearl St., Smalltown ND 77737
samwright@bogusemail.com

Objašnjenje koda:

Najpre se čita datoteka **tekst_za_pretragu.txt** korišćenjem kontekstnog menadžera, i smešta u novu promenljivu.

Pravi sa objekat šablon koji za parametar ima sirovi string koji pretražuje email adresu.

Nakon toga, otvoriće novu datoteku **email.txt**, za pisanje (dodavanja sadržaja) i upisivati samo pogotke.

PRETRAGA EMAIL ADRESA KORIŠĆENJEM REGULARNIH IZRAZA U PYTHONU

Sledi video za primer pretrage email adresa iz jedne datoteke putem regularnih izraza, i smeštanje u novu datoteku.

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

✓ Poglavlje 5

Python i JSON datoteke – Paket json

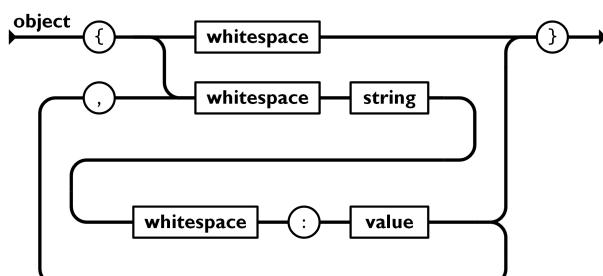
STRUKTURA JSON DATOTEKE

JSON datoteka je otvorena struktura podataka koje koriste razni programski jezici za razmenu podataka između aplikacija.

JSON datoteka (en. **JavaScript Object Notation**) predstavlja tekstualnu datoteku koja služi za razmenu podataka. Laka je za čitanje i pisanje i od strane ljudi i od strane aplikacija, i može se lako generisati i parsirati.

JSON Objekat

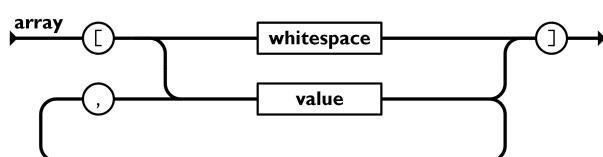
Objekat predstavlja neuređen skup parova ime-vrednost. Objekat počinje sa otvorenom velikom zagradom i završava se sa zatvorenom velikom zagradom. Posle svakog imena sledi dvotačka, a parovi ime-vrednost se razdvajaju zapetom.



Slika 5.1 JSON objekat. [Izvor: json.org]

JSON niz

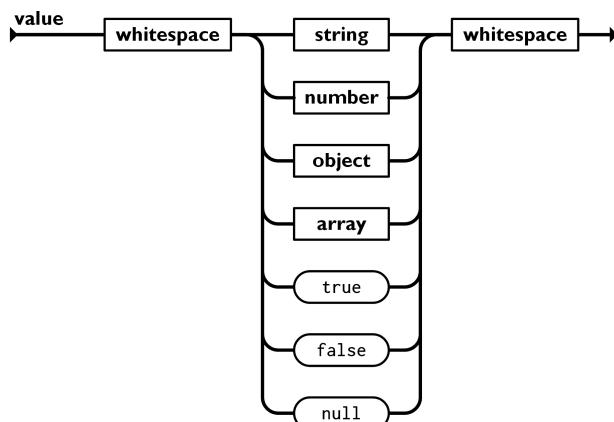
Niz se sastoji od skupa vrednosti. Niz počinje sa otvorenom srednjom zagradom, završava se sa zatvorenom srednjom zagradom, a vrednosti su razdvojene zapetom.



Slika 5.2 JSON niz. [Izvor: json.org]

JSON vrednost

Vrednost može biti string pod navodnicima, broj, logička vrednost, prazna vrednost, objekat ili niz. Moguće je ugnezdati ovakve strukture.



Slika 5.3 JSON vrednost. [Izvor: json.org]

PYTHON I PAKET JSON

Struktura JSON datoteka vrlo je slična Python imenicima, i moguće je konvertovati različite tipove JSON podataka u ugrađene tipove u Python jeziku.

U Python programskom jeziku, rad sa JSON datotekama zahteva uvoz paketa json.

Učitavanje JSON datoteka

JSON datoteka se može uvesti u Python na sledeći način:

```
import json

# JSON datoteka se nalazi u promenljivoj json_primer

podaci = json.loads(json_primer)
```

Struktura JSON datoteka vrlo je slična Python imenicima, i moguće je konvertovati različite tipove JSON podataka u ugrađene tipove u Python jeziku.

Štampanje JSON datoteka

Štampanje promenljive iz Python jezika u JSON datoteku može se postići na sledeći način:

```
import json

novi_podaci = json.dumps(ime_promenljive)
```

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

Slika 5.4 JSON/Python konverzija. [Izvor: Autor.]

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

PREUZIMANJE JSON DATOTEKA ZA API

Čest je slučaj da se JSON datoteke preuzimaju sa stranih veb lokacija da bi se dobilo na funkcionalnosti programa.

Čest je slučaj da se JSON datoteke preuzimaju sa stranih veb lokacija da bi se dobilo na funkcionalnosti programa.

Nekada se JSON datoteke mogu preuzeti besplatno, a nekada je potrebno prijaviti se na servis ili platiti da bi se dobio API ključ (en. API key)

Primer:

Napisati program koji računa konverziju valuta. Trenutni kurs preuzeti sa <https://exchangeratesapi.io>

Za učitavanje stranice koristiti

```
from urllib.request import urlopen
```

Unutar glavnog programa napisati funkciju za računanje EUR u USD. Štampati rezultat.

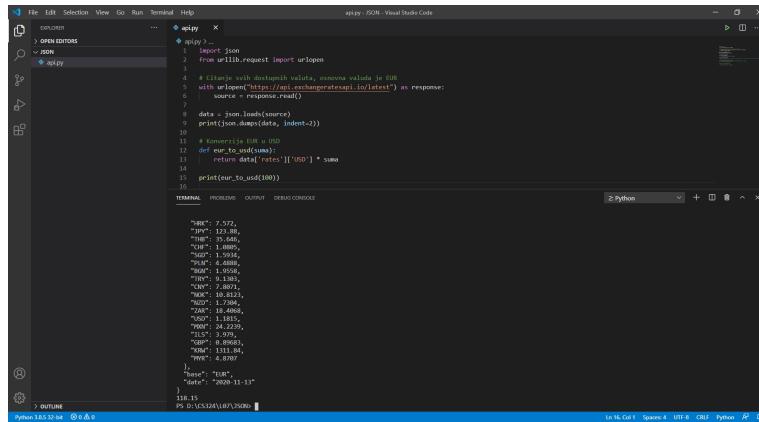
```
import json
from urllib.request import urlopen

# Citanje svih dostupnih valuta, osnovna valuta je EUR
with urlopen("https://api.exchangeratesapi.io/latest") as response:
    source = response.read()

data = json.loads(source)
print(json.dumps(data, indent=2))
```

```
# Konverzija EUR u USD
def eur_to_usd(suma):
    return data['rates']['USD'] * suma

print(eur_to_usd(100))
```



Slika 5.5 Preuzimanje JSON datoteke sa Interneta i korišćenje u programu. [Izvor: Autor]

PRIMER KONVERZIJE VALUTA

Sledi video sa konverzijom valuta, učitavanjem JSON datoteke sa Interneta.

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

▼ Poglavlje 6

Python i relacione baze podataka - Paket sqlite

KRATAK PREGLED SQLITE BAZA PODATAKA

SQLite je jedna on najčešćih implementacija baza podataka na svetu.

SQLite predstavlja biblioteku koja implementira funkcionalnost SQL baza podataka bez potrebe za serverom. SQLite je **open-access** i može se koristiti za privatne i komercijalne svrhe.

SQLite je jedna on najčešćih implementacija baza podataka na svetu.

Za razliku od ostalih SQL baza podataka, SQLite ne poseduje serverski proces. SQLite čita i upisuje direktno na datoteke na disku, unutar jedne datoteke.

SQLite se ne mora instalirati pre korišćenja. Ne postoji procedura podešavanja i instalacije, jer ne postoji potreba za konfiguracionim datotekama. Zbog toga ne postoji **troubleshooting**.

Glavne osobine SQLite baza podataka jesu sledeće:

- **Serverless** arhitektura (ne postoji potreba za serverskim procesom)
- Baza podataka se nalazi u jednoj datoteci
- Datoteka baze podataka može se koristiti na različitim platformama
- Kompaktna veličina baze podataka
- SQL izjave se kompajliraju u kod virtuelne mašine
- SQLite je otvorena za korišćenje i u privatne i u komercijalne svrhe

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.

RAD SA PAKETOM SQLITE3 ZA RAD SA JEDNOSTAVNIM RELACIONIM BAZAMA PODATAKA

Nije neophodno postavljati lokalne servere ili povezivati se na baze podataka na Internetu pri radu sa SQLite bazama i sqlite3 paketom.

Paket sqlite3 je deo standardne biblioteke, I nije je potrebno naknadno instalirati. Paket je veoma lak za korišćenje iz razloga što baza podataka može biti i datoteka na disku, ili može postojati samo u RAM memoriji tokom testiranja funkcionalnosti programa. Drugim rečima, nije neophodno postavljati lokalne servere ili povezivati se na baze podataka na Internetu pri radu sa SQLite bazama i SQLite paketom.

Povezivanje sa bazom podataka

Najpre je potrebno izvršiti povezivanje sa bazom podataka tako što se pravi objekat za povezivanje, kome se prosleđuje ime datoteke baze podataka

```
import sqlite3
conn = sqlite3.connect("ime_baze_podataka.db")
```

Pravljenje kursora za izvršenje SQL naredbi

Nakon povezivanja sa bazom podataka, potrebno je napraviti kurzor (en. cursor) koji će moći da izvršava naredbe ka povezanoj bazi podataka.

Kada se napravi cursor, moguće je izvršiti SQL naredbe metodom [.execute\(\)](#).

```
c = conn.cursor()
```

Unos podataka u bazu

Nakon izvršenja SQL naredbi, pravi "unos" podataka u bazu vrši se metodom [.commit\(\)](#), koja je metoda objekta povezivanja.

Zatvaranje veze sa bazom podataka

Dobra praksa jeste zatvaranje veze sa bazom podataka nakon završenog rada tako što se pozove metoda [.close\(\)](#) nad objekat povezivanja.

```
import sqlite3
conn = sqlite3.connect("ime_baze_podataka.db")
c = conn.cursor()

c.execute("... SQL naredbe ... ")

conn.commit()

conn.close()
```

PRAVLJENJE BAZE PODATAKA U RAM MEMORIJI RAČUNARA

Pri testiranju Python programa koji rade sa SQL bazama podataka, moguće je koristiti RAM memoriju računara za smeštanje baze podataka.

Pravljenje baze podataka u RAM memoriji računara

Kada se jednom napravi baza podataka kao i tabele, nije moguće ponovo izvršiti te komande, jer se javlja greška

```
sqlite3.OperationalError: table ime_baze_podataka already exists
```

Pri testiranju Python programa koji rade sa SQL bazama podataka, moguće je koristiti RAM memoriju računara za smeštanje baze podataka, i na taj način prilikom svakog izvršenja programa, praviće se nova baza podataka.

```
conn = sqlite3.connect(':memory:')
```

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

UBACIVANJE I AŽURIRANJE VREDNOSTI BAZE PODATAKA U PYTHON-U

Nakon povezivanja sa bazom podataka, kroz .execute() metodu moguće je izvršiti bilo koju SQL naredbu.

Primer: Baza Podataka student (20 minuta)

Koristiti sqlite3 paket i napraviti bazu podataka student u RAM memoriji računara.

Napraviti tabelu *student* sa kolonama **ime (tekst)**, **prezime (tekst)**, **brojIndeksa (broj)**, **email (text)** i **godinaStudiranja (broj)**.

Napraviti klasu *student(ime,prezime,broj_indeksa)*, sa metodama *dodati_email()* koja dodeljuje studentu email adresu kao *ime.prezime.broj_indeksa@metropolitan.ac.rs*, *stampati_sve()* koja stampa ime, prezime, broj indeksa i email adresu, kao i metode *dodati_u_bazu()*, koja dodaje studenta sa svim atributima u bazu podataka. Prilikom dodavanja svakom studentu je godina studiranja 1.

Napraviti funkciju za ažuriranje godine studiranja u bazu podataka *azurirati_godinu(broj_indeksa, azurirana_godina)*.

Isprobati sve funkcionalnosti.

```
import sqlite3
conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute("""CREATE TABLE studenti(
                ime text,
                prezime text,
                brojIndeksa integer,
                email text,
```

```
        godinaStudiranja integer
        )"""
class student:

    def __init__(self, ime, prezime, broj_indeksa):
        self.ime = ime
        self.prezime = prezime
        self.broj_indeksa = broj_indeksa

    def dodati_email(self):
        self.email = '{}.{}.{}@metropolitan.ac.rs'.format(self.ime.lower(),
        self.prezime.lower(), self.broj_indeksa)

    def stampati_sve(self):
        return "Student('{}', '{}', {}, {})".format(self.ime, self.prezime,
        self.broj_indeksa, self.email)

    def dodati_u_bazu(self):
        with conn:
            c.execute("INSERT INTO studenti VALUES (:ime, :prezime, :brojIndeksa,
        :email, :godinaStudiranja",
            {'ime': self.ime, 'prezime': self.prezime, 'brojIndeksa':
        self.broj_indeksa, 'email': self.email, 'godinaStudiranja': 1})

    def azurirati_godinu(broj, azurirana_godina):
        with conn:
            c.execute("UPDATE studenti SET godinaStudiranja = :godinaStudiranja WHERE
        brojIndeksa = :brojIndeksa", {'godinaStudiranja': azurirana_godina, 'brojIndeksa':
        broj})

stud1 = student('Petar', 'Petrovic', 1813)
stud1.dodati_email()
stud1.dodati_u_bazu()
conn.commit()

print('-----')
c.execute("""SELECT * FROM studenti WHERE prezime='Petrovic'""")
print('Prvobitna vrednost :')
print(c.fetchone())

print('Nova vrednost:')
azurirati_godinu(1813,4)
c.execute("""SELECT * FROM studenti WHERE prezime='Petrovic'""")
print(c.fetchone())

conn.close()
```

PRIMER KREIRANJA BAZE, UBACIVANJA I AŽURIRANJA VREDNOSTI KROZ PYTHON SQLITE3 PAKET

*Sledi video sa primerom kreiranja i ažuriranja vrednosti baze
poadataka "studenti"*

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

✓ Poglavlje 7

Pokazna vežba #7

ZADATAK #1

Prvi zadatak odnosi se na regulane izraze

Zadatak #1 (10 minuta)

Napisati program koji će putem regularnih izraza proveriti da li je uneta email adresa validna ili ne.

Validna adresa može sadržati veći broj alfa-numeričkih karaktera, srednju i donju crtu za korisničko ime i domen, dok može sadržati samo alfa-numeričke karaktere za top-level domen.

Program stalno postavlja upit, dok se ne unese validna email adresa.

```
import re

pattern = r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-\.]+'

while True:
    email = input('Uneti email: ')
    if re.search(pattern,email):
        print('Validna email adresa!')
        break
    else:
        print('Nije validna email adresa!')
```

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

ZADATAK #2

Prvi zadatak odnosi se rad sa SQLite bazama podataka

Zadatak #2 (30 minuta)

Pomoću Python sqlite3 paketa napraviti bazu podataka *predmeti* koja sadrži kolone **sifraPredmeta (tekst)**, **punolmePredmeta (tekst)**, **godinaStudiranja (ceo broj)** i **FITsmer (text)**.

Napraviti klasu `fit_predmet` sa podrazumevanim konstruktorom. Napraviti metodu za dodavanja atributa **sifra_predmeta**, **puno_ime_predmeta**, **godina_studiranja** i **fit_smer**.

Zatim, napraviti metodu koja vraća sve atribute klase u listu.

Izvršiti naredbe za povezivanje kreiranje baze podataka u RAM memoriju računara. Instancirati nekoliko objekata klase `fit_predmet`. Ubaciti vrednosti atributa instanci klase u bazu podataka. Izbrisati jedan red iz baze podataka.

Posle svake izvršene naredbe štampati sadržaj tabele `predmeti`.

```
import sqlite3
conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute("""CREATE TABLE predmeti(
                sifraPredmeta text,
                punoImePredmeta text,
                godinaStudiranja integer,
                smer text
            )""")

class fit_predmet:
    def __init__(self):
        pass

    def popuniti_predmet(self, sifra_predmeta, puno_ime_predmeta,
godina_studiranja, fit_smer):
        self.sifra_predmeta = sifra_predmeta
        self.puno_ime_predmeta = puno_ime_predmeta
        self.godina_studiranja = godina_studiranja
        self.fit_smer = fit_smer

    def stampati_predmet(self):
        return [self.sifra_predmeta, self.puno_ime_predmeta,
self.godina_studiranja, self.fit_smer]

predmet01 = fit_predmet()
predmet01.popuniti_predmet('CS324', 'Skripting Jezici', 3, 'IT')
print(predmet01.stampati_predmet())
predmet02 = fit_predmet()
predmet02.popuniti_predmet('CS225', 'Operativni sistemi', 4, 'SI')
print(predmet02.stampati_predmet())

def ubaci_u_bazu(predmet):
    with conn:
        c.execute("INSERT INTO predmeti VALUES (?, ?, ?, ?)",
(predmet.sifra_predmeta, predmet.puno_ime_predmeta,
predmet.godina_studiranja, predmet.fit_smer))

ubaci_u_bazu(predmet01)
ubaci_u_bazu(predmet02)
```

```
conn.commit()
# provera funkcionalnosti
print('-----')
c.execute("""SELECT * FROM predmeti""")
print(c.fetchall())

c.execute("DELETE FROM predmeti WHERE smer = 'IT'")
print('-----')
c.execute("""SELECT * FROM predmeti""")
print(c.fetchall())

c.close()
```

▼ Poglavlje 8

Individualna vežba #7

ZADACI ZA INDIVIDUALNU VEŽBU #7

Zadaci se rade ukupno 80 minuta

Individualna vežba #7 odnosi se na rad sa različitim paketima unutar Python standardne biblioteke. Procenjeno trajanje individualnih vežbi iznosi 80 minuta.

Zadatak #1 (35 minuta)

Napisati program koji ispisuje meni picerije u datoteku.

Napraviti klasu *pizza* sa konstruktorom (*ime, prečnik_u_cm, cena*). Instancirati više objekata sa različitim vrednostima.

Napraviti funkciju koja će štampati sve vrednosti u datoteku *picerija.txt*, kao i trenutni datum (koristiti paket *datetime* za dobijanje datuma)

Zatim, unutar klase napraviti novu metodu *cena_po_cm2()*, koja će računati cenu po kvadratnom centimetru za konkretnu instancu.

Napraviti funkciju koja će učitati sve vrednosti iz datoteke i izračunati koja pica ima najbolji odnos cene po površini. Funkcija treba da štampa:

```
'Najbolji odnos cena po povrsini ima pizza {ime} od {prečnik} cm, i cene od {cena} dinara.'
```

Zadatak #2 (45 minuta)

Napisati koji proverava ispravnost unete email adrese i šifre na sledeći način:

Regularnim izrazom proveriti da li je uneta regularna email adresa validna. Validna adresa može sadržati veći broj alfa-numeričkih karaktera, srednju i donju crtu za korisničko ime i domen, dok može sadržati samo alfa-numeričke karaktere za top-level domen.

Program javlja kada se unese validna ili nevalidna adresa. Ako se unese nevalidna adresa, program se završava.

Kada se unese validna adresa, tražiti da se unese šifra korišćenjem:

```
from getpass import getpass
sifra = getpass()
```

Šifra treba biti minimalno dužine 8 karaktera, treba sadržati mala i velika slova, cifre, i specijalne karaktere - _ # @ .

Program javlja ako šifra nije validna, ali nastavlja sa radom dok se ne unese validna šifra.

Email, kao i sve unete šifre (validne i nevalidne), sačuvati u datoteku **email_sifre.txt**

✓ Poglavlje 9

Domaći zadatak #7

DOMAĆI ZADATAK

Domaći zadatak #7 se okvirno radi 2.5h

Zadatak #1

Dati su podaci u tekstualnoj datoteci na lokaciji:

https://github.com/CoreyMSchafer/code_snippets/blob/master/Regular-Expressions/data.txt

Koristeći regularne izraze i paket `re`, izvući sve adrese koje kreću cifrom koja je jednaka Vašem broju indeksa, i snimiti u tekstualnu datoteku **adrese.txt**

Ukoliko je broj indeksa **1234**, treba izvući adrese:

- **433** Elm St., Westworld TX 61967
- **478** Oak St., Bedrock IA 58176
- ...

Zadatak #2

Koristeći paket sqlite3, kreirati bazu podataka **predmeti** u memoriji računara, sa kolonama **sifra**, **punolme**, **profesor**, **godinaStudiranja**.

Popuniti tabelu svim predmetima koje ste slušali do sada i napravite funkciju koja pretražuje bazu po predmetnom profesoru i vraća rezultate.

Tradicionalni studenti:

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

Internet studenti:

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja 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 10

Zaključak

ZAKLJUČAK

Zaključak lekcije #7

Rezime:

U ovoj lekciji bilo je reči o paketima za Python 3 programski jezik. Pojedini paketi jesu deo standardne biblioteke, dok se drugi moraju instalirati pomoću PIP paketa.

Bilo je reči o ugrađenim paketima math i datetime, re, json, i sqlite3.

Osim toga, kroz paket PIP moguće je upravljati instaliranim paketima, ažurirati pakete za koje postoji nova verzija, kao i instalirati nove pakete.

Primeri u okviru lekcije kao i zadaci za individualni rad treba da osposobe studente za rad u Python 3.x jeziku pri radu sa različitima paketima.

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.