Merni sistemi u računarstvu, http://automatika.etf.rs/sr/13e053msr

Python programiranje

Vanredni profesor dr Nadica Miljković, kabinet 68, nadica.miljkovic@etf.rs

Korišćen je Programski jezik Python: skripta za studente telekomunikacija, prof. M. Bjelice, 2016, http://www.etf.bg.ac.rs/etf_files/udzbenici/python.pdf za pripremu ove prezentacije.

Slike za naslovni slajd: http://2.bp.blogspot.com/-Y7f2e4YgAWM/UTwEDap796I/AAAAAAAAAhw/DGbcYwqDmkk/s1600/pyserial-linux.png, https://cdn.instructables.com/FA1/KNIO/HAWSI7Z8/FA1KNIOHAWSI7Z8.MEDIUM.gif i https://media.wired.com/photos/592650d8cfe0d93c4742fbf9/master/pass/windowsxp.jpg.

ŠTA JE PYTHON?

Python

- Python je programski jezik (https://en.wikipedia.org/wiki/Python_(programming_language)). Slobodan softver. Modularan. Praktičan. Ovde: za merenja sa serijskog porta, vizuelizaciju podataka (npr. Lisažuova figura, histogrami) i dr.
- Na MSR predmetu su instalirane jedna po jedna biblioteka. Sada, pomoću pip-a možete instalirati koje god želite i/ili koje su Vam potrebne.
- Međutim, postoje i tzv. *Custom Distributions* instalacije koje dolaze sa skupom paketa, npr.:
  - Python(x,y), http://www.pythonxy.com/, dec. 2017.
  - Sage, http://www.sagemath.org/, dec. 2017.
- Više programskih paradigmi je "pomešano" u ovom programskom jeziku, a najvažnije su objektno-orijentisano i funkcionalno programiranje.
- **VAŽNO: Ova prezentacija nema zadatak da uči studente programiranju, već da istakne neke elemente programskog jezika Python s obzirom da je to sve popularniji programski jezik u merenjima koja su zasnovana na primeni računara.**

Python – deo istorije

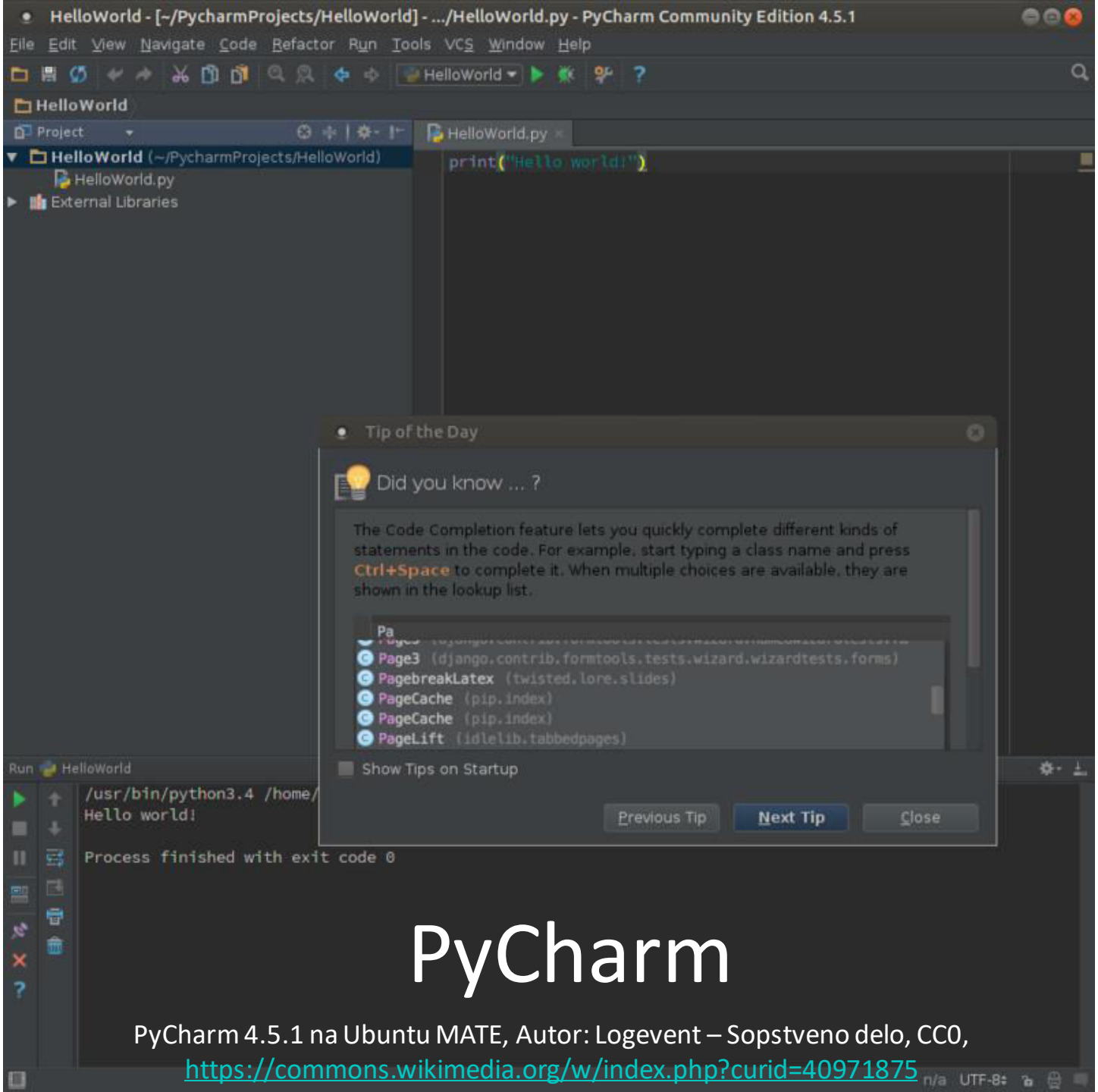
- Kreirano ga je Guido van Rossum i prva verzija je objavljena 1991. godine.
- Guido je rekao o kreiranju Python-a (https://en.wikipedia.org/wiki/History_of_Python):

> ...In December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of *Monty Python's Flying Circus*).

- Realno? Svima želim *Merry Christmas*!

Python Zen filozofija

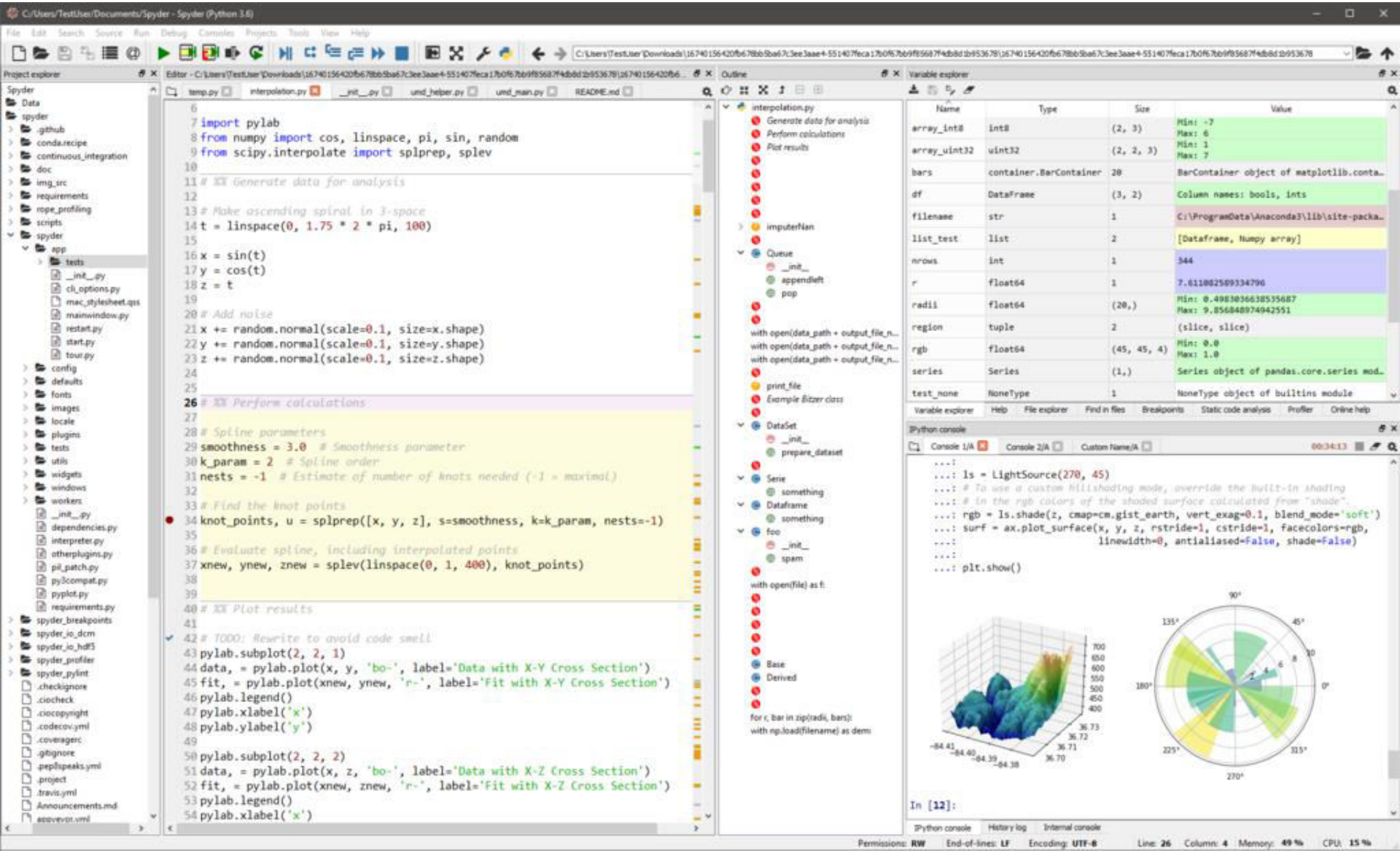
- Principi ove filozofije su izlistani na linku https://en.wikipedia.org/wiki/Zen_of_Python, a ovde samo neki od njih:
  - *Beautiful is better than ugly.*
  - *Simple is better than complex.*
  - *Complex is better than complicated.*
  - *Readability counts.*
  - *Errors should never pass silently.*
  - *Unless explicitly silenced.*
  - *If the implementation is hard to explain, it's a bad idea.*
  - *If the implementation is easy to explain, it may be a good idea.*



PyCharm

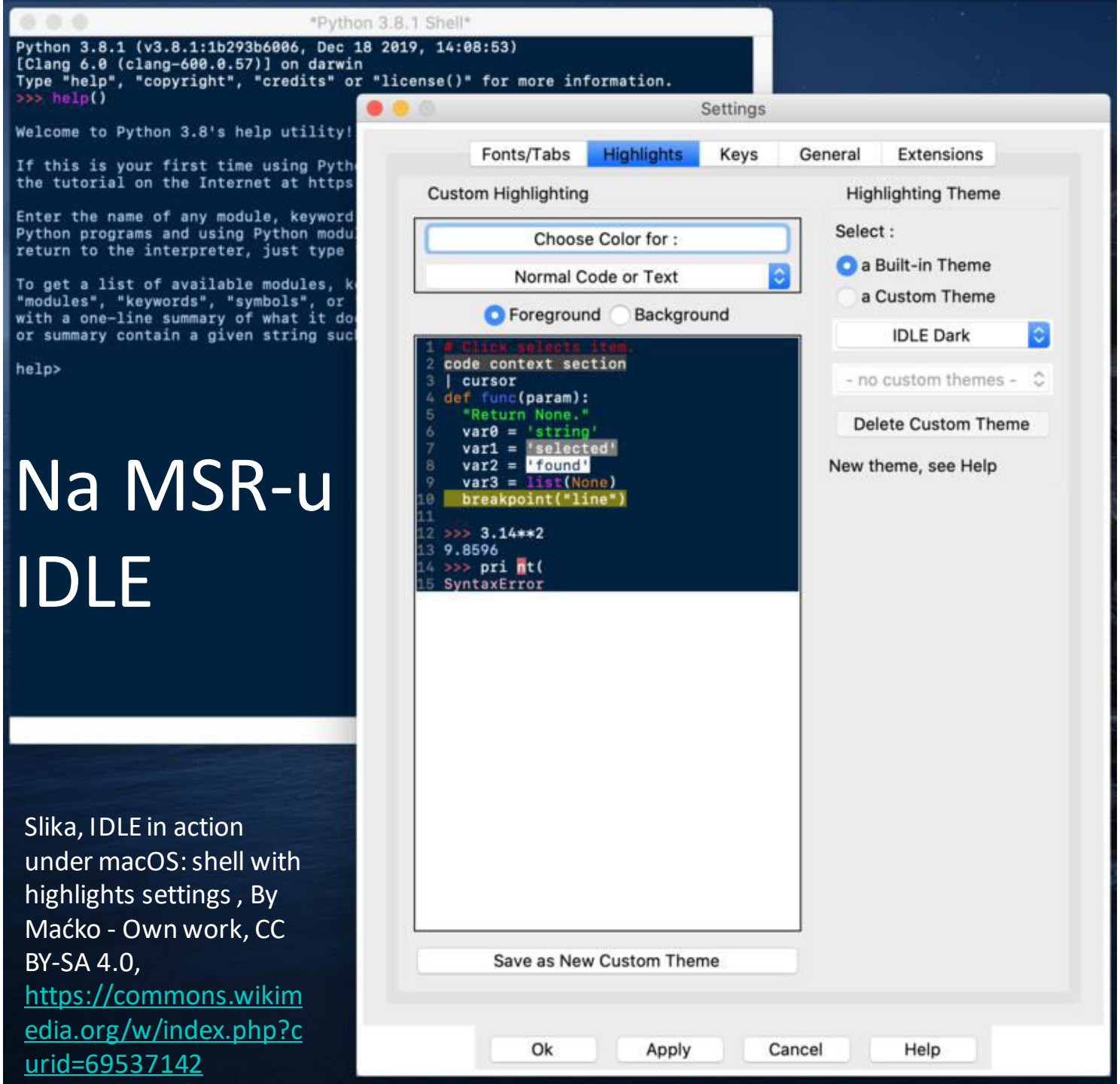
PyCharm 4.5.1 na Ubuntu MATE, Autor: Logevent – Sopstveno delo, CC0, https://commons.wikimedia.org/w/index.php?curid=40971875

Spyder



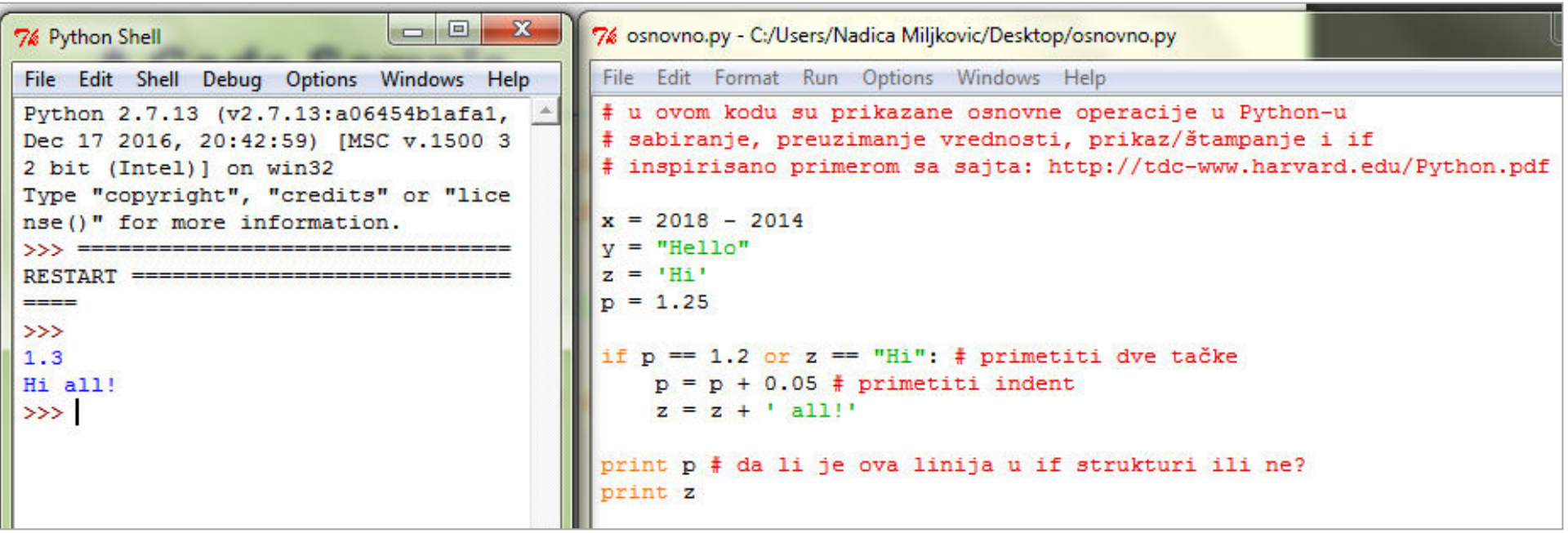
By C.A.M. Gerlach - Spyder IDE Main Website, MIT, https://commons.wikimedia.org/w/index.php?curid=18469698

Na MSR-u IDLE



Slika, IDLE in action under macOS: shell with highlights settings , By Maćko - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=69537142

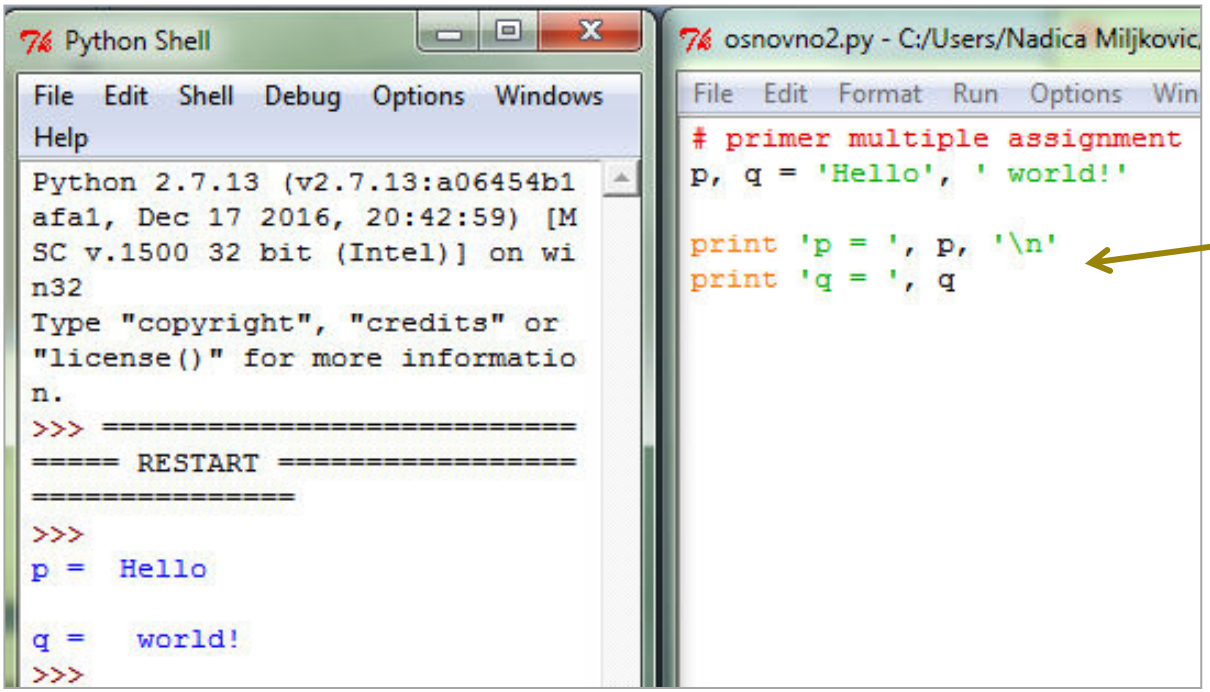
Python osnove



- Primetiti da: na kraju izraza nema “;”, da je string moguće definisati na dva načina, kako se izgleda *if* struktura, da ne postoje posebne komande za početak i kraj, kako izgledaju logički operatori?
- Dodatno, primetiti da se za spajanje stringova (eng. *concatenation*) koristi operator “+”. Negaciji odgovara operator *not*.
- VAŽNO: kod prve dodele vrednosti nekoj promenljivoj dolazi do njene inicijalizacije.
- Na slici je prikazan VIDLE for VPython koji je i preporučen studentima na 13E053MSR predmetu. Takođe, ovde je dat primer u 2.7.

Python, osnovni tipovi podataka

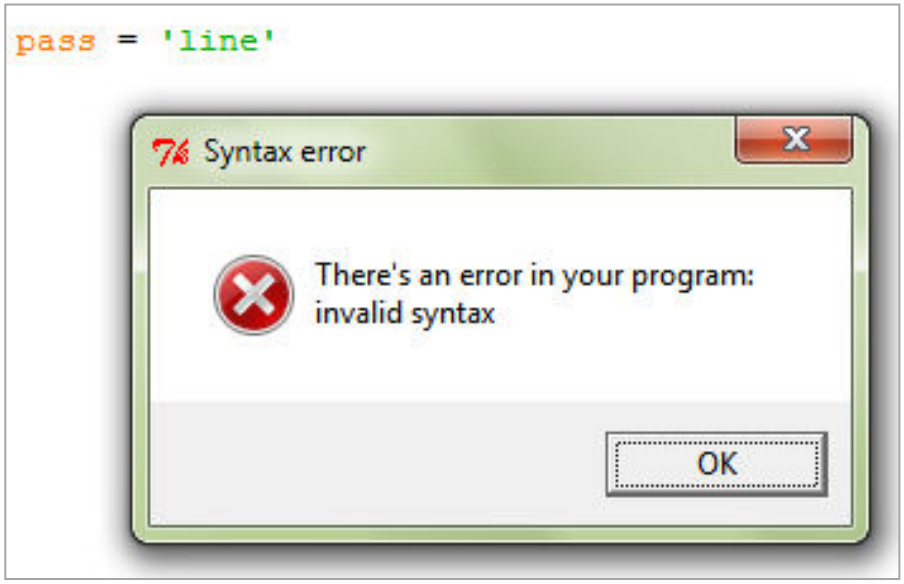
```
""" Ovo je primer višelinijskoj komentara
koji se može koristiti po želji"""
print y
# jednolinijski komentar
```



prelazak u novu liniju i više parametara za prikaz ...
U novom Python-u treba dodati zagrade!

- Celobrojni tip podataka (eng. *integer*) je podrazumevani tip za sve brojeve.
- Na prethodnom slajdu promenljiva *p* je tipa *float*. Zašto?
- Za *string*-ove se mogu koristiti “”, ali i ‘’.
- Jednolinijski komentari počinju sa tarabom “#”. Međutim, višelinijski komentari počinju sa “”” (pogledati sliku gore).
- *Multiple assignment* je jedna od osobina koda u Python-u (pogledati sliku dole).

Imena promenljivih



- Imena su *case sensitive* i ne mogu početi sa brojem, ali mogu sadržati slova, brojeve i donje crte.
- Rezervisane reči se ne mogu koristiti za imena promenljivih.
- Primeri rezervisanih reči su (a možda Vam i budu potrebne):
  - *and*, *assert*, *break*, *class*, *continue*, *def*, *del*, *elif*, *else*, *except*, *exec*, *finally*, *for*, *from*, *global*, *if*, *import*, *in*, *is*, *lambda*, *not*, *or*, *pass*, *print*, *raise*, *return*, *try*, *while* ...
  - Ne morate da ih pamtite sve, Python će promeniti boju rezervisane reči, a i javiće Vam poruku kao na slici.

Reference semantics i pokazivači

```
# reference semantics
x = [10, 9, 8, 7, 6]
y = x
print 'x = ', x
print 'y = ', y

x.append(5)
print 'x = ', x
print 'y = ', y
```

```
x =  [10, 9, 8, 7, 6]
y =  [10, 9, 8, 7, 6]
x =  [10, 9, 8, 7, 6, 5]
y =  [10, 9, 8, 7, 6, 5]
>>>
```

- Šta se desilo sa *x* i *y* na slici?
- Iako je korisno, budite oprezni!
- Kog su tipa *x* i *y* sa slike? Lista, naravno. Može se proveriti komandom *type*.
  - *x* je zapravo pokazivač na listu brojeva od 6 do 10, a *y* je pokazivač na *x*.
  - Da li je moguće dodati svakom elementu liste *x* broj 1 ili neki drugi i kako? Da li u tom slučaju *x* menja vrednost ili pokazuje na neke druge brojeve?

Rešenje

```
# reference semantics
x = [10, 9, 8, 7, 6]
y = x
print 'x = ', x
print 'y = ', y

x.append(5)
print 'x = ', x
print 'y = ', y

print type(x)

# dodavanje vrednosti 1 svakom elementu liste
x = [vr + 1 for vr in x]
print 'x = ', x
```

```
x =  [10, 9, 8, 7, 6]
y =  [10, 9, 8, 7, 6]
x =  [10, 9, 8, 7, 6, 5]
y =  [10, 9, 8, 7, 6, 5]
<type 'list'>
x =  [11, 10, 9, 8, 7, 6]
>>>
```

Liste i indeksiranje

```
x =  [11, 10, 9, 8, 7, 6]
x[0] =  11
x[-2] =  7
x[:2] =  [11, 10]
x[2:] =  [9, 8, 7, 6]
x[-1:2] =  []
Da bi bile napravljene dve kopije liste
y = x[:]
A, da bi obe promenljive pokazivale na istu listu
z = x
>>>
```

- U Python-u postoje *immutable* i *mutable* tipovi podataka tj. nepromenljivi i promenljivi... Postoji i tip podataka *tuple* koji je nepromenljiv. Operacije koje se mogu primeniti na listu, mogu se primeniti i na *tuple* (lista se definiše uglastim zagradama, a *tuple* običnim).
- Više o *tuple* na sajtu: https://www.tutorialspoint.com/python/python_tuples.htm
- Pojedinačnim elementima liste, *tuple*-a i stringova se pristupa uglastim zagradama []. Indeksi počinju od 0. Negativni indeksi označavaju da se broji s desna (pogledati sliku).
- Još neke opcije indeksiranja su prikazane na slici.

in operator

```
# in operator
mojString = 'ananas'
print 'ana' in mojString
print 'x = ', x
print (7 in x)
print (77 in x)
```

```
True
x =  [11, 10, 9, 8, 7, 6]
True
False
>>>
```

- Zgodno. Primeri korišćenja *in* operatora su prikazani na slici.
- Koristi se da se proveri da li se neka vrednost pojavljuje u listi, ali i da se proveri da li neki podstring pripada drugom stringu.
- Ovaj operator se koristi i unutar *for* petlje.
- Ranije je pokazana upotreba operatora + za spajanje tzv. konkatenaciju stringova. Analogno, koristi se i za konkatenaciju listi u *tuple*-a.

“*” operator

```
x =  [11, 10, 9, 8, 7, 6]
x * 3 =  [11, 10, 9, 8, 7, 6, 11, 10, 9, 8, 7, 6, 11, 10, 9, 8, 7, 6]
>>>
```

- Operator “*” kreira novu listu, *tuple* ili string, tako što vrši ponavljanje određen broj puta.
- Na primer, na slici je prikazano kako je moguće kreirati listu ponavljanjem 3 puta svih elemenata u listi *x*.

Liste: *append* i *insert*

```
# append i insert funkcije
print 'x = ', x

x.append(56)
print 'x.append(56) = ', x

x.insert(1, 55)
print 'x.insert(1, 55) = ', x
```

```
x =  [10, 9, 8, 7, 6]
x.append(56) =  [10, 9, 8, 7, 6, 56]
x.insert(1, 55) =  [10, 55, 9, 8, 7, 6, 56]
>>>
```

- Primeri primene funkcija *append* i *insert* nad listama su prikazani na slikama.
- Primetiti da se pozivaju sa tačkom nakon promenljive i da se nova vrednost dodeljuje listi za koju se poziva.
- Može se, pored ove dve funkcije, koristiti i *extend* funkcija? Koja je njena uloga?
- Dodatno postoje i: *index* (na izlazu daje indeks na kome se prvi put pojavljuje element koji je prosleđen kao argument), *count* (broj pojavljivanja arguementa), *remove* (briše se prvo pojavljivanje argumenta), *reverse* (menja se redosled elemenata), *sort*, *clear*, *copy*, ...
- Za konverziju npr. iz liste u *tuple* i obrnuto koriste se istoimene funkcije: *list()* i *tuple()*.

Rečnici, korisničke funkcije, ...

```
def <name>(arg1, arg2, ..., argN):
  <statements>
  return <value>

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

- Postoje i rečnici (eng. *dictionaries*) i o njima ovde neće biti puno reči. Više o strukturama podataka u Python-u generalno na: https://docs.python.org/3/tutorial/datastructures.html.
- Za definisanje funkcija koristi se ključna reč *def*, kao na slici.
- Slika je preuzeta sa: http://tdc-www.harvard.edu/Python.pdf.
- Definisanje podrazumevanih vrednosti se vrši definisanjem poziva funkcije:
  - def times(x = 1, y = 1) ...
- Ako korisnik ne definiše sam koja će biti izlazna vrednost iz funkcije, onda će Python svakako vratiti *None*.

Kontrola toka i strukture u Python-u

```
sensor = 1;
if sensor > 2:
    print 'Upalite svetlo!'
elif sensor < 0.5:
    print 'Ugasite svetlo!'
else:
    print 'Neka ostane kako je bilo'
print 'Ovo je izvan if strukture i štampa se sigurno'

vr = 1
while vr < 5:
    if vr >= 1:
        vr += 0.5
        continue
    vr = vr + 1
    if vr > 4.5:
        break

for x in range(15):
    if x < 3:
        x += 1
        continue
    print(x)
```

```
Neka ostane kako je bilo
Ovo je izvan if strukture i štampa se sigurno
3
4
5
6
7
8
9
10
11
12
13
14
>>>
```

- Na slici su prikazane osnovne strukture u Python-u.
- Obratiti pažnju na ":" na kraju *if*, *while* i *for* strukture!
- Postoji i *assert*. To pogledajte sami.
- Šta je rezultat koda sa slike? Koliko je *vr* na kraju *while* petlje? Za domaći?

Moduli

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

Now enter the Python interpreter and import this module with the following command:

```
>>> import fibo
```

- Moduli omogućavaju da se efikasnije organizuje kod u Python-u ("širina" i "visina" koda?). Najčešće se promenljive i funkcije definišu u modulima.
- Na slici je dat primer definisanja i "uvoženja" modula sa sajta: https://docs.python.org/3/tutorial/modules.html.

Klase i objekti

```
class Oscilloscope(object):
    #
    def __init__(self, address = '/dev/ttyS0'):
        try:
            self.port = serial.Serial(address)
            self.port.timeout = 5
            scopeid = self.ask('*idn?')
            if scopeid == '':
                print 'Cannot find oscilloscope at', address
            self.write('header 0')
            self.write('data:encdg ascii')
        except:
            print
            print 'Cannot open port', address
            print
    #
    def __del__(self):
        self.port.close()
    #
    def write(self, string):
        self.port.write(string + '\n')
    #
    def read(self):
        return self.port.readline()
    #
    def readchar(self):
        return self.port.read()
    #
    def ask(self, question):
        self.write(question)
        return self.read()
```

- Enkapsulacija, polimorfizam, nasleđivanje ... poznato?
- Na slici je dat primer definisanja klase osciloskop u kodu za predmet Električna merenja na Katedri za elektroniku: http://tnt.etf.rs/~oe2em/.

Dodatno

```
>>> try:
...     1 / 0
... except:
...     print('That was silly!')
... finally:
...     print('This gets executed no matter what')
...
That was silly!
This gets executed no matter what
```

```
fileptr = open('filename')
somestring = fileptr.read()
for line in fileptr:
   print line
fileptr.close()
```

```
>>> a = 1
>>> b = 2.4
>>> c = 'Tom'
>>> '%s has %d coins worth a total of $%.02f' % (c, a, b)
'Tom has 1 coins worth a total of $2.40'
```

Primer je sa: http://tdc-www.harvard.edu/Python.pdf.

Neke korisne funkcije

- *len()* – vraća dužinu niza
- repeat()
- *ceil()*
- *radians()*
- *sin()*
- *exp()*
- Konstante: *pi* i *e*
- *capitalize()*
- *islower()*
- *ilsupper()*
- *split()*
- *read()*
- *write()*
- *close()*
- Preporučene liste funkcija:
  - http://www.astro.up.pt/~sousasag/Python_For_Astronomers/Python_qr.pdf
  - https://perso.limsi.fr/pointal/_media/python:cours:abregepython-english.pdf

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>> 17 // 3  # floor division
5
```

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

```
# Python 3: Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!

# Input, assignment
>>> name = input('What is your name?\n')
>>> print('Hi, %s.' % name)
What is your name?
Python
Hi, Python.
```

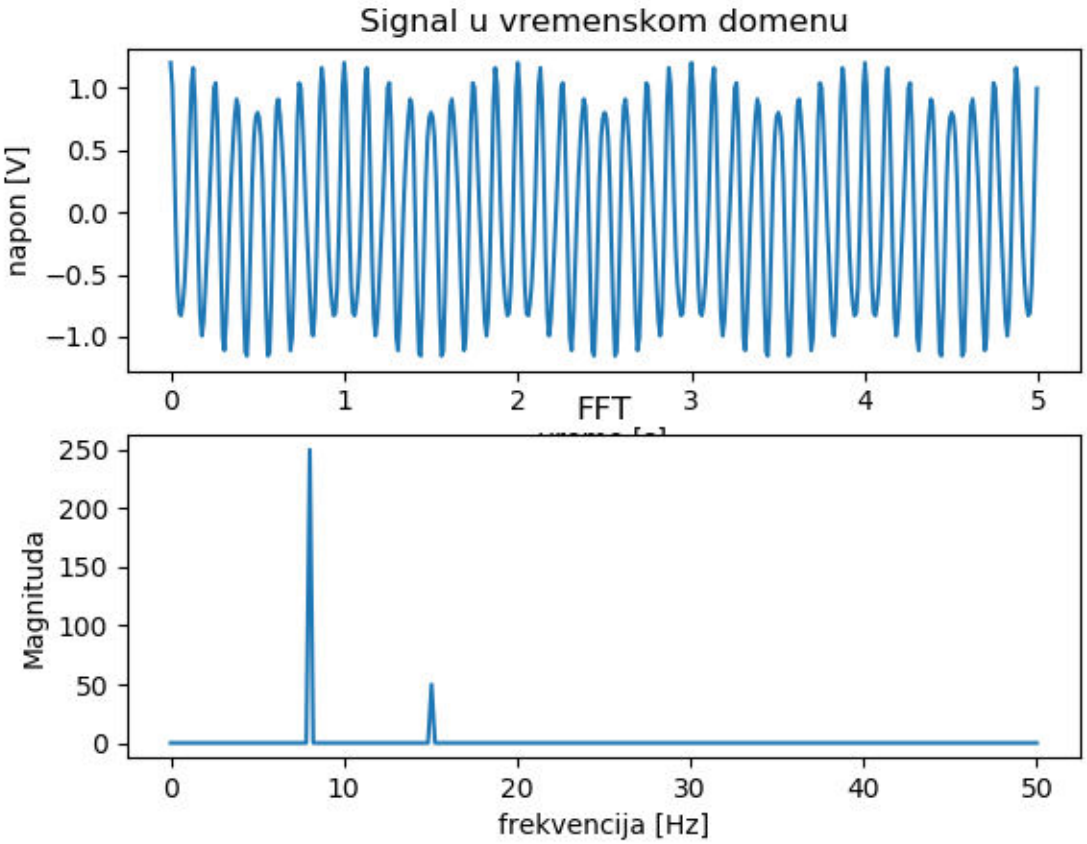
```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

Online console from PythonAnywhere

https://www.python.org/

PYTHON BIBLIOTEKE NA MSR

numpy

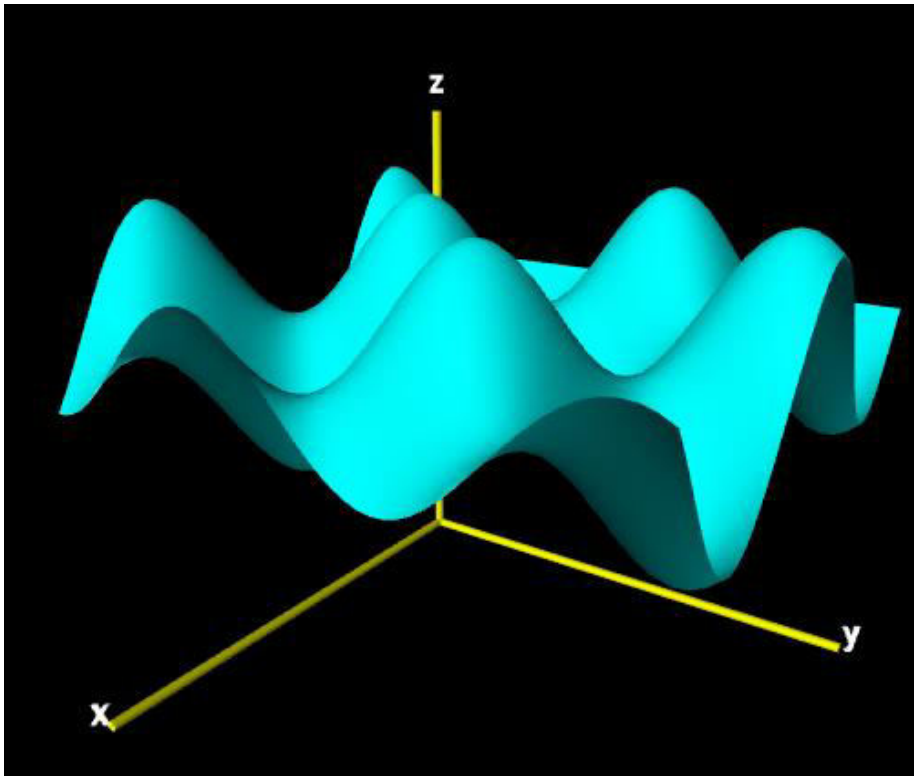


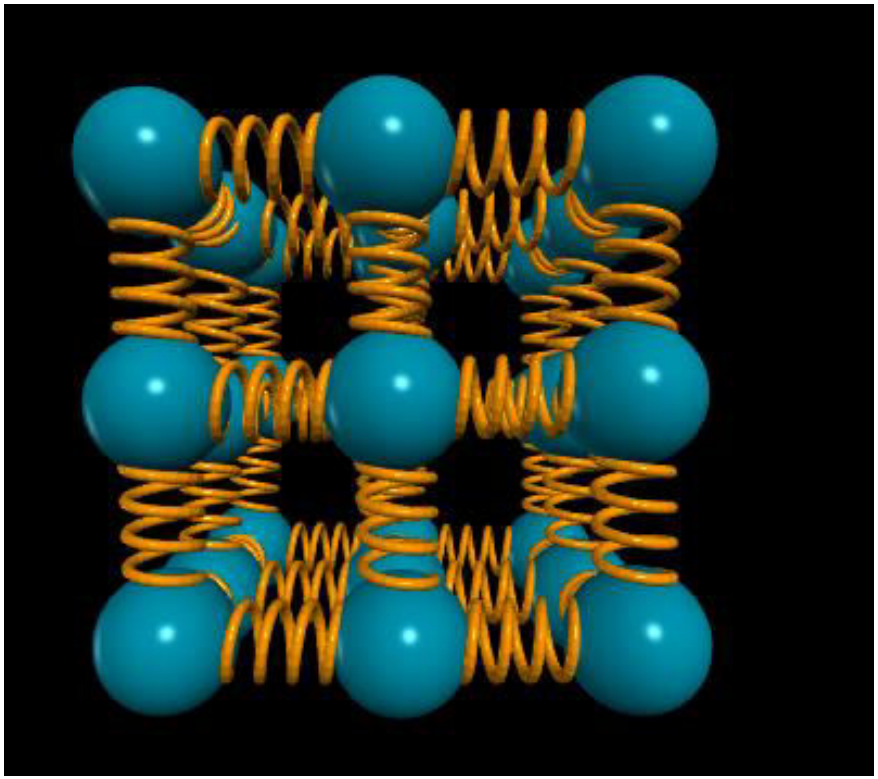
- Kaže se da ova biblioteka sadrži funkcije koje omogućavaju funkcionalnost sličnu Matlab-u (relativno brze operacije sa nizovima i matricama, linearna algebra)
- Više informacija na: http://www.numpy.org/
- Primer računanja FFT (eng. *Fast Fourier Transform*) za jedan sintetički signal je prikazan na slici.
- Koju informaciju o signalu pruža FFT?

```
# ovaj kod je inspirisan online primerom sa sajta:
# https://stackoverflow.com/questions/15382076/plotting-power-spectrum-in-python

import numpy as np
import matplotlib.pyplot as plt

fs = 100.0 # definisati frekvenciju odabiranja
t = np.arange(0, 5, 1/fs) # kreirati vreme u trajanju od 5 sekundi
# zbor kosinusa sa osnovnim frekvencijama od 8 i 15 Hz
x = np.cos(2 * np.pi * 8 * t) + 0.2 * np.cos(2 * np.pi * 15 * t)

# Furijeova transformacija
xF = np.fft.fft(x)
N = len(xF)
xF = xF[0:N/2] # posmatra se do Nikvista (fs/2)
fr = np.linspace(0, fs/2, N/2) # definiše se frekvencijska karakteristika

# crtanje grafika
fig = plt.figure()

ax = fig.add_subplot(211)
ax.plot(t, x)
plt.title('Signal u vremenskom domenu')
plt.xlabel('vreme [s]')
plt.ylabel('napon [V]')

ax = fig.add_subplot(212)
ax.plot(fr, abs(xF))
plt.title('FFT')
plt.ylabel('Magnituda')
plt.xlabel('frekvencija [Hz]')

plt.show()
```

A kod?

vpython





- VPython je biblioteka koja se koristi za programiranje relativno jednostavnih 3D animacija i displeja (eng. *3D Programming for Ordinary Mortals*)
- Više na: http://vpython.org/
- Neki od primer uključuju fizičke eksperimente i biološke strukture.
- Primeri sa http://www.glowscript.org/#/user/GlowScriptDemos/folder/Examples/ su prikazani na slikama (http://www.glowscript.org/#/user/GlowScriptDemos/folder/Examples/program/Plot3D,).

pyserial

- Više na: https://pythonhosted.org/pyserial/.
- Ovo je biblioteka koja omogućava pristup serijskom portu. Na MSR predmetu, pristup podacima koje je Arduino kod postavio na serijski port.
- Moguće je čitati podatke (eng. *read*) sa serijskog porta, ali ih je moguće i "slati" tj. upisivati na serijski port (eng. *write*).
- Za one koje zanima da pristupaju virtuelnim serijskim portovima, mogu pogledati: http://com0com.sourceforge.net/.
- Način na koji mi pristupamo podacima iz Python-a preko serijskog porta se naziva snifovanje (eng. *sniffing*).

matplotlib

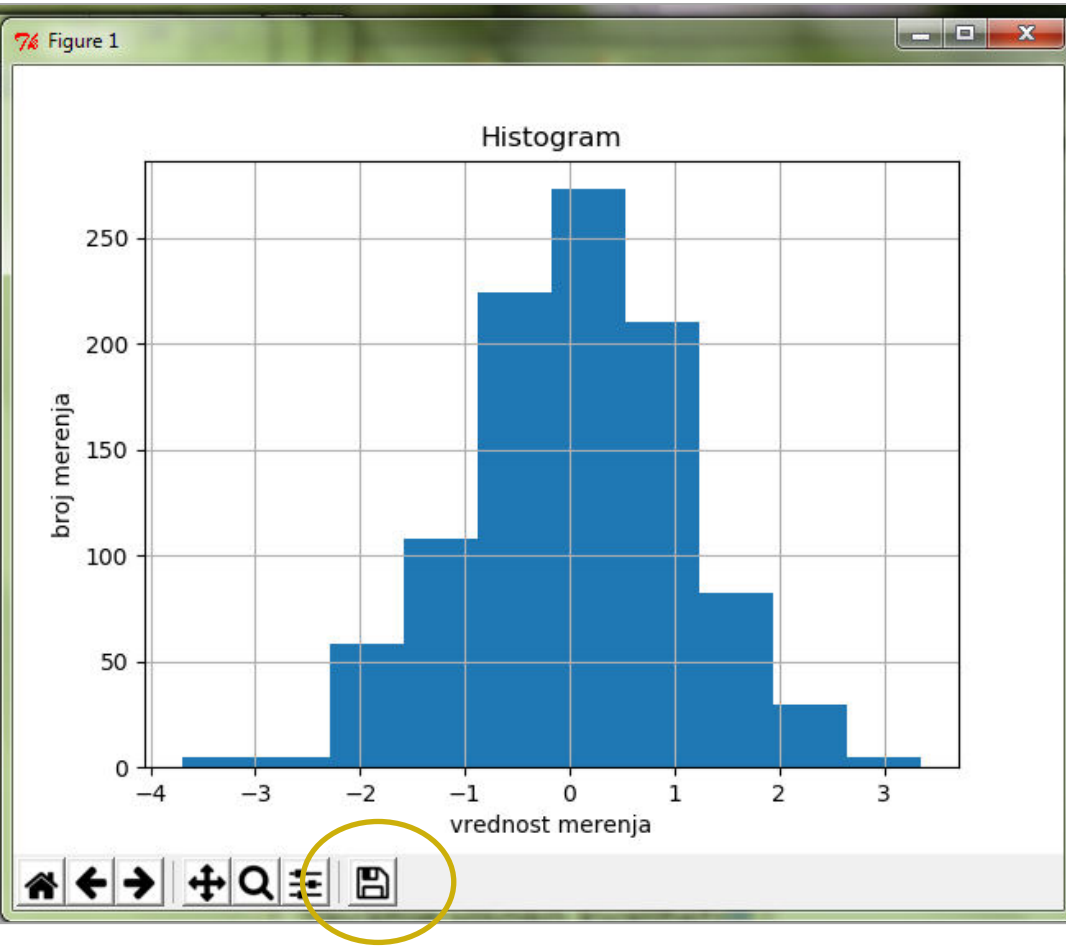
- Ova biblioteka sadrži niz funkcija koje omogućavaju iscrtavanje visoko kvalitetnih grafika.
- U ranijim primerima je pokazano kako je moguće koristiti ovu biblioteku.
- Osim prikaza 2D grafika, koristi se i za prikaz 3D grafika, ali i za prikaz slika.
- Horizontalni barovi se prikazuju primenom *barh()* funkcije.
- Više na: http://matplotlib.org/.

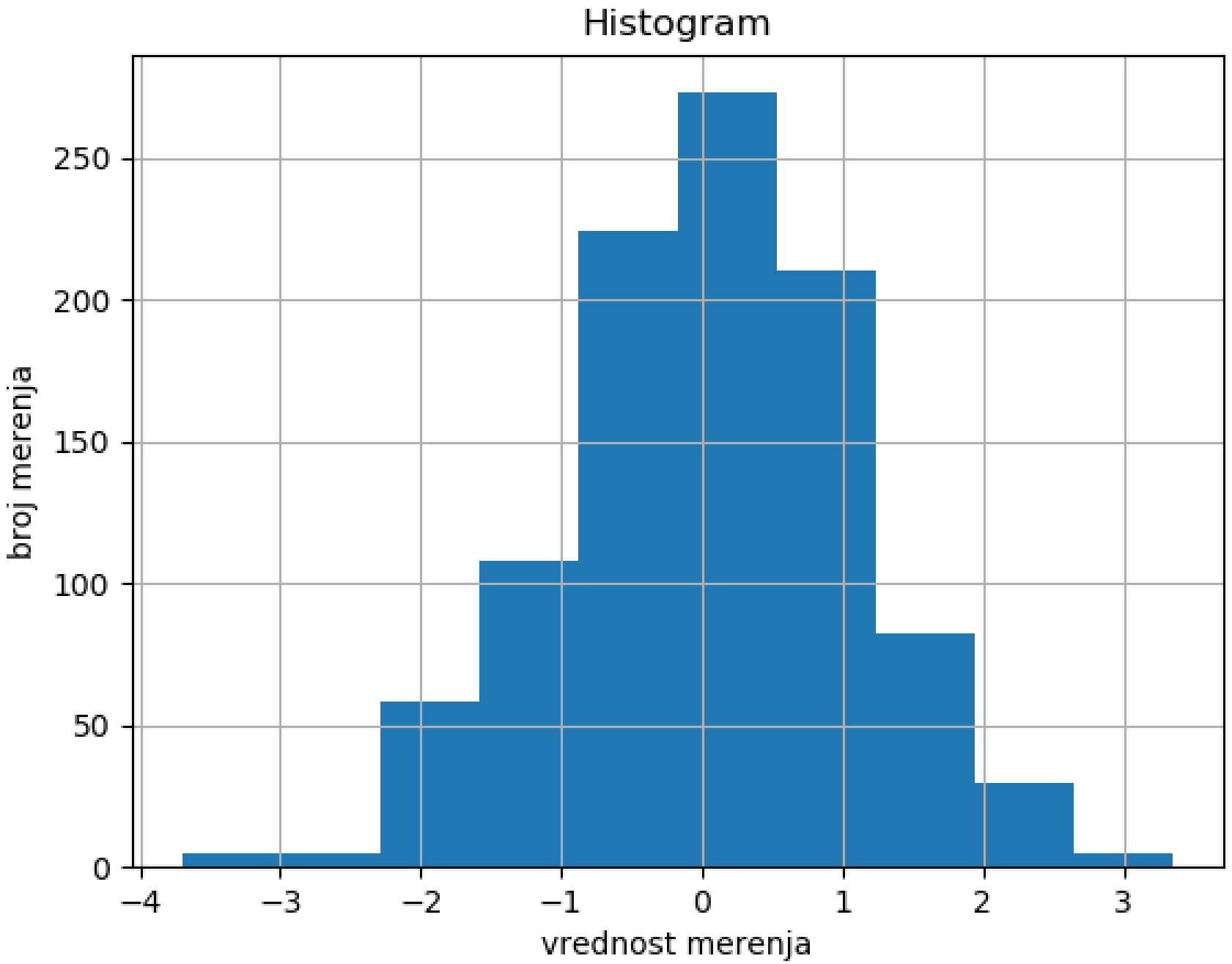
Primer, *matplotlib*

racun2.py - C:\Users\Nadica Miljkovic\Desktop\MSR2017\skripta\racun2.py

File Edit Format Run Options Windows Help

```
import numpy as np
import matplotlib.pyplot as plt

# promenljiva u koju se smeštaju učitane vrednosti
mer = []

# čitanje podataka iz fajla
f = open('C:/Users/Nadica Miljkovic/Desktop/merenje2.txt', 'r')
for line in f:
    mer.append(float(line.strip()))

# korišćenje numpy biblioteke za osnovne parametre
print np.mean(mer)
print np.std(mer)

# prikaz histograma
plt.hist(mer)
plt.xlabel('vrednost merenja')
plt.ylabel('broj merenja')
plt.grid(True)
plt.title('Histogram')
plt.show()
```



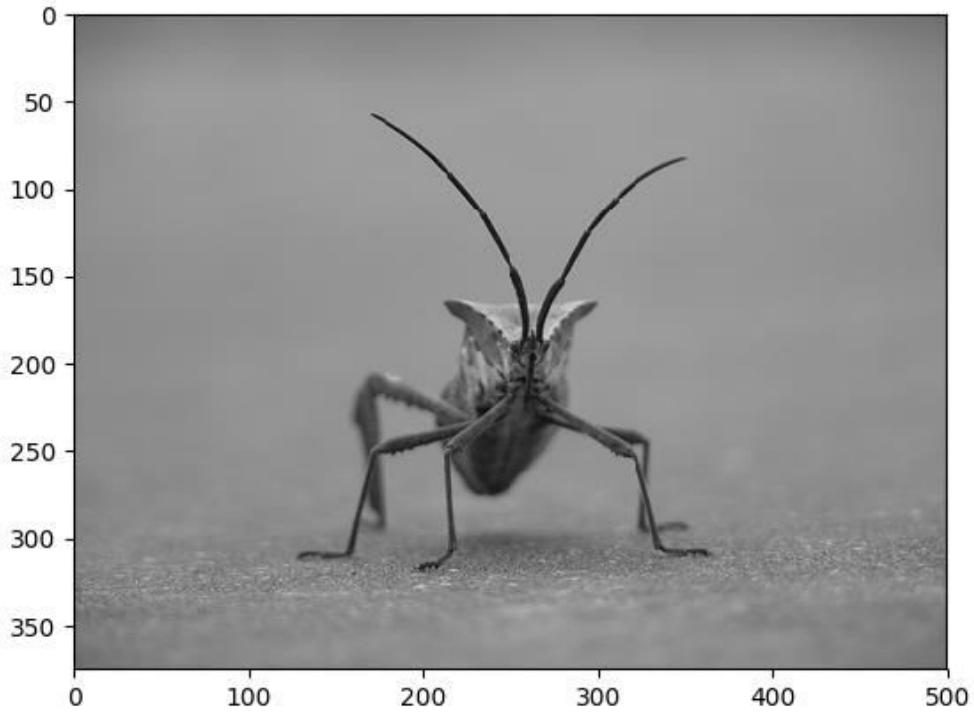

Histogram
broj merenja
vrednost merenja
0
50
100
150
200
250
−4
−3
−2
−1
0
1
2
3

numpy i *matplotlib* za prikaz/analizu slika

Plotting numpy arrays as images

So, you have your data in a numpy array (either by importing it, or by generating it). Let's render it. In Matplotlib, this is performed using the `imshow()` function. Here we'll grab the plot object. This object gives you an easy way to manipulate the plot from the prompt.

```
imgplot = plt.imshow(img)
```



Prikazana je slika sa sajta: http://matplotlib.org/tutorials/introductory/images.html#sphx-glr-tutorials-introductory-images-py.

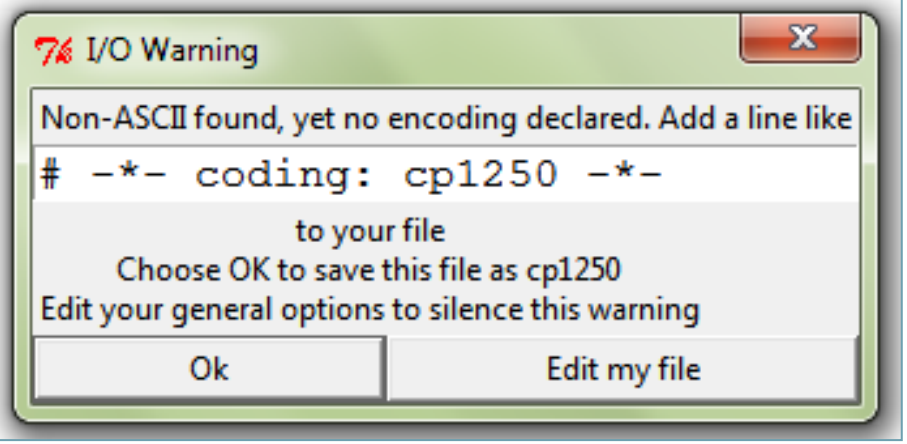
ENCODING I PYTHON

Python

- Stringovi su najčešće korišćeni tipovi podataka u Python-u.
- ASCII skup karaktera je najčešće dovoljan za pisanje osnovnih programa. Međutim, ponekad je potrebno koristiti i druge karaktere.
- U verzijama Python-a koje su >2 postoji Unicode string za rad sa Unicode podacima (više na: https://en.wikipedia.org/wiki/Unicode) koji koriste enkodiranje i dekodiranje.
- Kada unutar Python koda postoji rad sa Unicode stringovima u zaglavlju mora da stoji:
- # -*- coding: utf-8 -*-
- Detalje pogledati na: https://www.pythoncentral.io/python-unicode-encode-decode-strings-python-2x/.

Gde može da nastane problem?

```
kT = [0.1, 0.2, 0.5, 1, 1.5, 2.5, 5]
M = 30 # V
uB = [vr*M/(100 * (3**0.5)) for vr in kT]
uB = [round(vr, 2) for vr in uB]

print 'Za ispitivane klase tačnosti od: ', kT, ','
print 'merne nesigurnosti su: ', uB, '.'
```



- Problem ili izazov?
- Zašto se javlja obaveštenje sa slike?
- Kakav problem može da nastupi kao rezultat operacija enkodiranja i dekodiranja?

cp1250?

Python comes with a number of codecs built-in, either implemented as C functions or with dictionaries as mapping tables. The following table lists the codecs by name, together with a few common aliases, and the languages for which the encoding is likely used. Neither the list of aliases nor the list of languages is meant to be exhaustive. Notice that spelling alternatives that only differ in case or use a hyphen instead of an underscore are also valid aliases.

Many of the character sets support the same languages. They vary in individual characters (e.g. whether the EURO SIGN is supported or not), and in the assignment of characters to code positions. For the European languages in particular, the following variants typically exist:

- an ISO 8859 codeset
- a Microsoft Windows code page, which is typically derived from a 8859 codeset, but replaces control characters with additional graphic characters
- an IBM EBCDIC code page
- an IBM PC code page, which is ASCII compatible

Codec	Aliases	Languages
ascii	646, us-ascii	English
big5	big5-tw, csbig5	Traditional Chinese
big5hkscs	big5-hkscs, hkscs	Traditional Chinese
cp037	IBM037, IBM039	English
cp424	EBCDIC-CP-HE, IBM424	Hebrew
cp437	437, IBM437	English
cp500	EBCDIC-CP-BE, EBCDIC-CP-CH, IBM500	Western Europe
cp737		Greek
cp775	IBM775	Baltic languages
cp850	850, IBM850	Western Europe
cp852	852, IBM852	Central and Eastern Europe
cp855	855, IBM855	Bulgarian, Byelorussian, Macedonian, Russian, Serbian
cp856		Hebrew
cp857	857, IBM857	Turkish

Slika je sa sajta: https://docs.python.org/2.4/lib/standard-encodings.html.

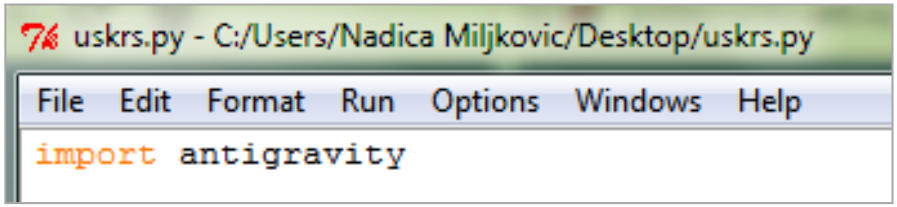
ZA KRAJ

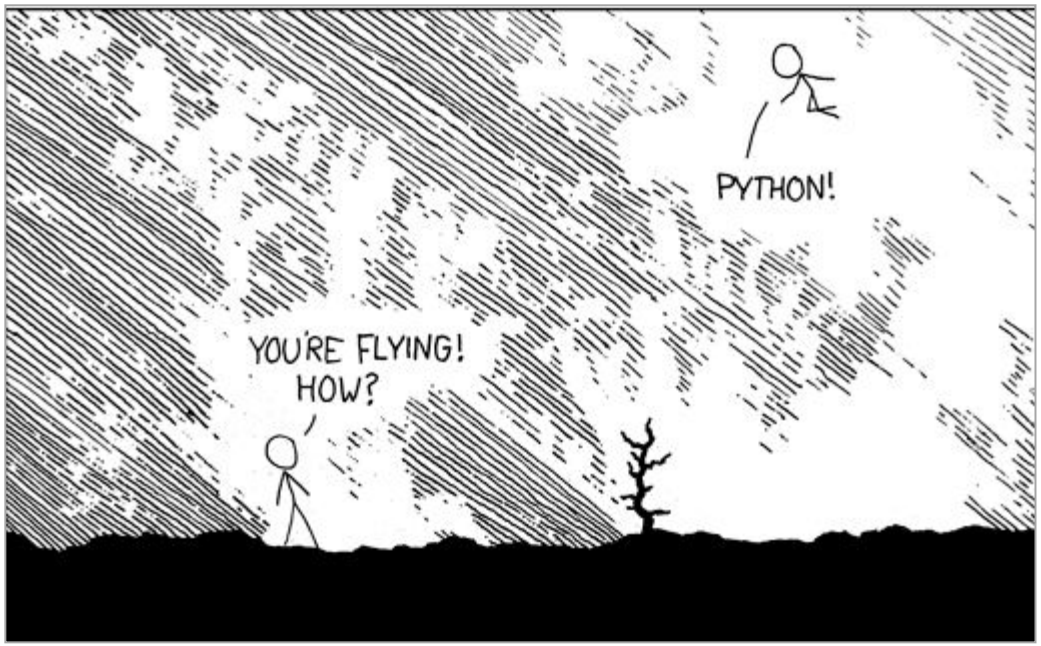
Slajdovi predavanja po pozivu

- Digitalni merni instrumenti i programabilna instrumentacija
- Merenje električnih veličina I deo
- Merenje električnih veličina II deo
- Senzori
- Merna nesigurnost tipa A
- Merna nesigurnost tipa A - drugi deo (pogledati i video Uncertainities, Lecture 1 of Classical Mechanics 8.01 by Lewin. Originally from MIT OpenCourseWare, CC BY-NC-SA 3.0 US)
- Merna nesigurnost tipa B (Uputstvo za pisanje izveštaja i primer laboratorijskog izveštaja, može se pogledati na: https://rodzah.wordpress.com/)
- "Programski jezik Python: primena u merenjima" predavanje prof. Predraga Pejovića: prezentacija i *handout*, DOI 10.5281/zenodo.1320794

- Od nekoga ko zna mnogo o Python-u ...
- Sa praktičnim primerima ...

Ne još ...





- Da li znate šta je *easter egg*?
- Ima ih i Python.
- Primer sa slike. Neke druge nađite sami, pa javite i meni ...

Za pripremu ove prezentacije

- korišćeni su materijali sa sledećih sajtova:
  - Introduction to Python, http://tdc-www.harvard.edu/Python.pdf, pristupljeno decembra 2017
  - http://matplotlib.org/tutorials/introductory/sample_plots.html#sphx-glr-tutorials-introductory-sample-plots-py, pristupljeno decembra 2017
  - i drugih koji su izlistani u prethodnim slajdovima.