SoC-Day8

July 25, 2018

1 SoC Day 8: Python, Python, Python

why Python? interpreter, compiler, cross platform script language
Raspberry Pi - a perfect pythonic platform example

1.1 1. Sintaksa

1.1.1 komentari:

```
In [ ]: #this is a comment
        #comment could be written like this

        # but it's really easy to read with leading space
```

1.1.2 dodjela vrijednosti, definicija varijable:

```
In [1]: a = 5
        a = 5 + 7
        a = 9 + 5
```

1.1.3 grananje, uvlaenje:

```
In [2]: if a == 5:
            print("if you don't need any braces")
            print("why do I keep bracing print statement?")

        # when does if work?
        # what operators do work?

        # indentation matters:

        if a == 5: print("don't do this, my friend. Oneliners suck. They are here only to impress newbies")

        print("C anyone?"); print("Another ugly oneliner")

        if a == 5:
            print("just keep everything tidy")
```

```
# LISTEN TO PEP8, BOYS
# https://www.python.org/dev/peps/pep-0008/?
```

C anyone?
Another ugly oneliner

1.2 2. Semantika

Dinamiki tipizirani jezik podrazumjeva tip podataka varijable prema njezinim vrijednostima

1.2.1 Varijable - ugraeni tipovi:

In [3]:
```
int a = 5 # nope
```

```
  File "<ipython-input-3-ccc7acec22f7>", line 1
    int a = 5 # nope
        ^
SyntaxError: invalid syntax
```

In [4]:
```
a = int() # ok. But why?
```

In [5]:
```
a = 5 # this is a way to go
```

In [7]:
```
s = str("bye bye")
```

In [8]:
```
s = "even better"
```

In [9]:
```
b = 3.1415926 # works out of the box.
```

In [12]:
```
type(a)
```

Out[12]: int

In [13]:
```
class(b) # RRRRR, nope...
```

```
  File "<ipython-input-13-e9c810050fb1>", line 1
    class(b) # RRRRR, nope...
          ^
SyntaxError: invalid syntax
```

In [14]:
```
# int, bool, float, None
varC = None
```

1.2.2 integer, float

In [21]:
```
# chr, +, -, =, %, //, str,
a = 48
chr(a)

a = a + 5
a += 5
a = a - 5

print(a)
```

```
53
```

1.3 Kolekcije

In [ ]:
```
# string - format, +, [], len, find, index, isalnum,
# islower, isnumeric, isupper, upper, lower, ord
```

In [ ]:
```
# list - len, [1], [1::], [1:], append, pop, extend, +
```

In [ ]:
```
# tuple
```

In [ ]:
```
# dict
```

In [ ]:
```
# range
```

In [ ]:
```
# set - add one, update iterable, discard one, remove iter
# pop rand, | un, & inters, - diff, union -> in an new set
```

In [ ]:
```
# unpack -- a, b = tuple
```

1.4 Ugraene funkcije:

In [ ]:
```
# print

# apply

# len

# type

# dict

# dir

# zip

# enumerate
```

```
# sum

# help

# chr

# ord
```

1.5 Petlje

In [38]:
```
# for - string, range, -=, enumerate, zip, reverse itd

 # for i lokalan namespace

for i in range(100):
    i = i - 1
print("gotov")
```

```
gotov
```

In [ ]:
```
# while: bool, int, None, condition
uvijet = True

while uvijet:
    print("blabla")
```

1.6 Korisnike funkcije

In [114]:
```
# def, dir, void, return, yield, argumenti, named argumenti,
 # unpack, __next__,
 # nested
 # return ,

 # procedura
 def foo():
     print("bar")

 # bar() = foo()

 def foo2(a=0, b=0, c=0):
     print(a, b, c)

 def foo3(a, b):
     a += (2, 3)
     b.append(7)
```

```
a = (1, 2, 3)
b = [1, 2, 3]
# print(foo3(a, b))
# print(a, b)

# generatori
spremnik = 0

def neradeca_funkcija(n):
    for i in range(n):
        yield i

a = neradeca_funkcija(10)
```

In [119]:
```
next(a)
```

Out[119]: 4

In [136]:
```
for i in range(10):
    i += 1

z = 50

def jos_jedna():
    global z
    z += 1
    print(z)


    return z

def jos_jos_jedna(n: int=10):

    def brojac():
        return n
        #return list(range(n))

    return brojac

funkcija = jos_jos_jedna()
funkcija()
```

Out[136]: 10

1.7 Garbage Collection, Performance, Alternatives, Open Source

Is Python GILtty of something? Cython, Jyton... Anaconda, miniconda WinPython

1.8 Globalne, lokalne varijable

In [ ]:
```
# naredba global
# for, if, def
```

1.9 Moduli

In [143]:
```
# import, *, some, dir, as --> mtk

# math

import math
import math as mh
from math import pi
# from math import * PEP8 - ne raditi!

mh.sin(90)
print(pi)
```

```
3.141592653589793
```

1.10 Objekt, konstruktor, metoda

In [151]:
```
# wooof

class Dog:

    def __init__(self, name="Generic dog"):
        self.name = name

    def woof(self):
        print("Dog named \"{}\" says \"woof woof\"".format(self.name))

    def walk(self):
        print("this dog is walking!")

pas = Dog("Minnie")
pas.woof()

type(pas)
```

```
Dog named "Minnie" says "woof woof"
```

Out[151]:
```
__main__.Dog
```

In [171]:
```
class Pravokutnik(object):
    def __init__(self, side_a=1, side_b=1):
```

```
            self.a = side_a
            self.b = side_b

        def povrsina(self):
            return self.a * self.b

        def opseg(self):
            return 2 * self.a + 2 * self.b

    class Kvadrat(Pravokutnik):
        def __init__(self, single_side=1):
            super().__init__(single_side, single_side)

            # ili

            # Pravokutnik.__init__(self, single_side, single_side)

            # ili

            # self.side_a = single_side
            # delf.side_b = single_side


    if __name__ == "__main__":
        mp = Kvadrat(1)
        print(mp.opseg())
        dir(mp)
```

```
4
```

In [173]:
```
import mojModul
import mojModul as ml

moj_kvadrat = mojModul.Kvadrat(5)
moj_kvadrat.opseg()
```

Out[173]: 20

1.11 Datoteke, Streamovi i With statement

In [174]:
```
# open
# with
# close
# read

out = open("datotekaZaPisanje.txt", "w")

out.write("neki tekst")
```

```
out.close()
# wb, rb, r, a, w
# write
# close
```

In [175]:
```
# out.write("hi")

with open("novaDatoteka.txt", "w") as file_stream:
    file_stream.write("zdravo")
```

In [ ]:
```
# out.close()
```

In [ ]:
```
with open("blabla", "w") as datoteka:
    datoteka.write("jeah")
```

1.12 Tips & Tricks

In [183]:
```
# palindrom, generator, [], {}, lambda,
# kopiranje funkcija, kopiranje objekata

a = lambda x: x**2

a(2)

moja_lista = [i**2 for i in range(10)]
moja_lista

moja_lista2 = list(moja_lista)

import copy

dir(copy)
```

Out[183]:
```
['Error',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_copy_dispatch',
 '_copy_immutable',
 '_deepcopy_atomic',
 '_deepcopy_dict',
 '_deepcopy_dispatch',
```

```
 '_deepcopy_list',
 '_deepcopy_method',
 '_deepcopy_tuple',
 '_keep_alive',
 '_reconstruct',
 'copy',
 'deepcopy',
 'dispatch_table',
 'error']
```

1.13 Pickle

In [179]:
```
import pickle

d = dict()
d["blabla"] = 16

with open("moj_rijecnik.p", "wb") as stream:
    pickle.dump(d, stream)

with open("moj_rijecnik.p", "rb") as stream:
    d_ucitani = pickle.load(stream)

print(d_ucitani)
```

```
{'blabla': 16}
```

1.14 Copy vs. DeepCopy

In [195]:
```
a = [1, 2, 3, 4, 5, 6]
b = [6, 5, 4, 3, 2, 1]

c = [1, 2, [1, 2, 3, a, [b]]]
d = c
e = copy.copy(c)
f = copy.deepcopy(c)

c[0] = "HA"
c[1] = "HAHA"
c[2][3][3] = "738942659234862398465239846523984652398465239846"


print(c) # original
print(d) # 'copied' with "=" statement
print(e) # copied with copy.copy statement
print(f) # copied with copy.deepcopy (aka. True Copy) statement
```

['HA', 'HAHA', [1, 2, 3, [1, 2, 3, '738942659234862398465239846', 5, 6], [[6, 5, 4, 3, 2, 1]]]]
['HA', 'HAHA', [1, 2, 3, [1, 2, 3, '738942659234862398465239846', 5, 6], [[6, 5, 4, 3, 2, 1]]]]
[1, 2, [1, 2, 3, [1, 2, 3, '738942659234862398465239846', 5, 6], [[6, 5, 4, 3, 2, 1]]]]
[1, 2, [1, 2, 3, [1, 2, 3, 4, 5, 6], [[6, 5, 4, 3, 2, 1]]]]