ELEMENTI

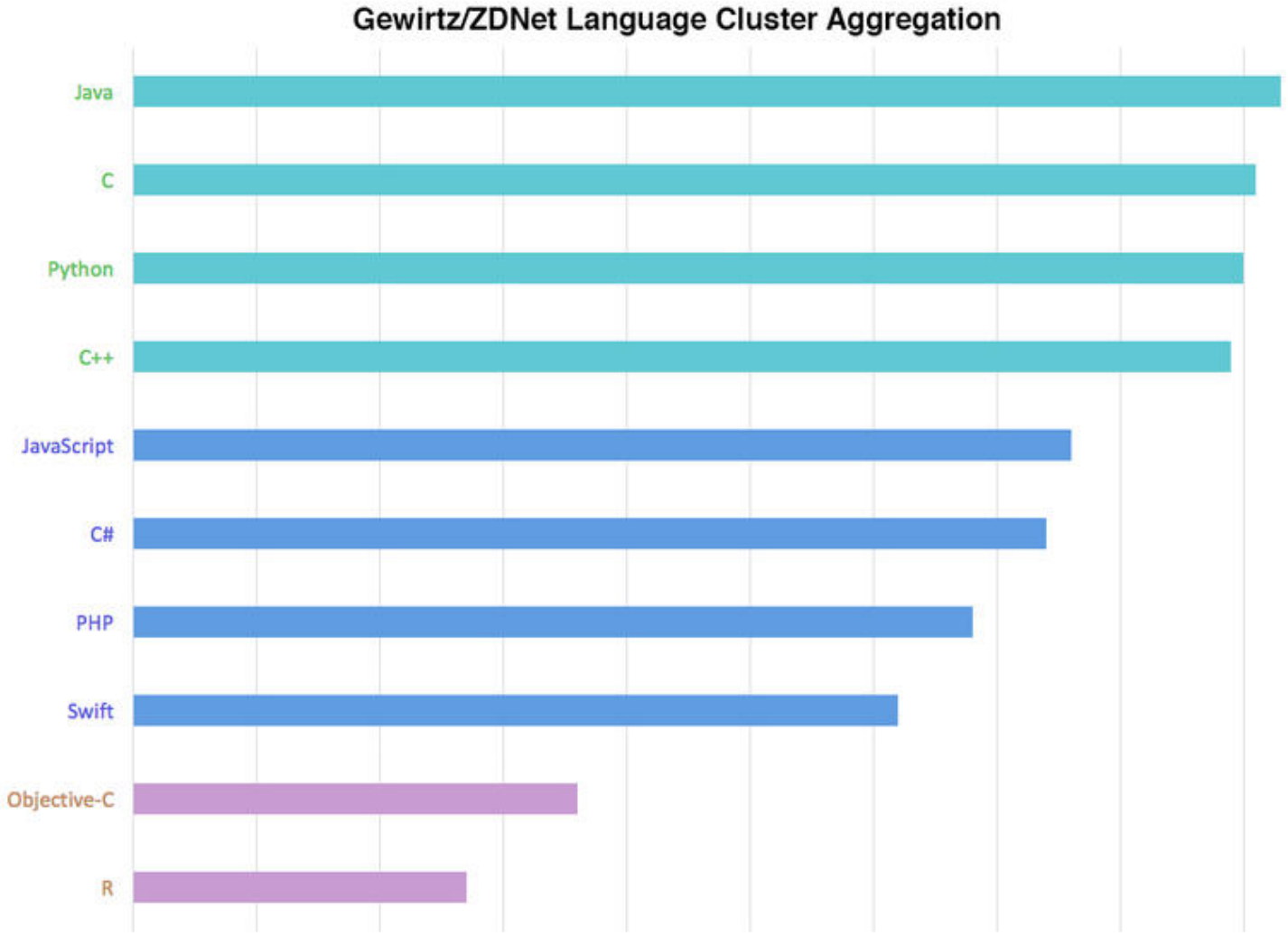
PYTHON JEZIKA

1. dan

Popularnost programskih jezika I

PYPL	Tiobe	CodingDojo	IEEE Jobs	IEEE Open	IEEE Trending
Java	Java	SQL	C	C++	C
Python	C	Java	Java	Python	C++
PHP	C++	JavaScript	Python	C	Python
C#	C#	C#	C++	Java	Java
JavaScript	Python	Python	JavaScript	Swift	Swift
C++	JavaScript	C++	C#	JavaScript	R
C	PHP	PHP	PHP	C#	JavaScript
Objective-C	Assembly	iOS	Ruby	Ruby	Ruby
R	VB.NET	Ruby/Rails	HTML	PHP	Go
Swift	Perl		Swift	Ruby	C#
Matlab	Delphi		Assembly	HTML	PHP
Ruby	Ruby		Ruby	Go	Scala
VBA	Swift		Scala	Scala	Arduino
Visual Basic	Objective-C		Shell	Objective-C	Assembly
Scala	Matlab		Perl	Shell	Shell
Perl	Groovy		SQL	Arduino	Objective-C
lua	Visual Basic		Objective-C	Assembly	HTML
Delphi	Ruby		Matlab	Matlab	Rust
Go	Go		Visual Basic	Lua	Haskell
Haskell	PL/SQL		Go	Perl	Visual Basic

Popularnost programskih jezika II

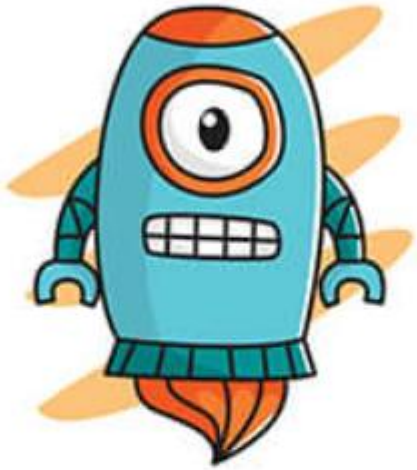


Šta je Python u odnosu na druge jezike?

C LANGUAGE

AGE: 44 YEARS
STRENGTH: USED FOR LOWER LEVEL SYSTEM, FAST RUN TIME PERFORMANCE
WEAKNESS: NOT USED FOR WEB DEVELOPMENT, SECURITY, NOT FOR DEVELOPING APPLICATIONS

PYTHON



AGE: 25 YEARS
STRENGTH: USAGE AMONG SECURITY, NETWORKING, AUTOMATION, LIBRARIES
WEAKNESS: POOR PACKAGING, SLOW

JAVA LANGUAGE

21 YEARS
TH: LOT OF EXTENSION/PLATFORMS, POPULAR JVM
SS: DIFFICULT TO COMPILE AND BUILD, EXPENSIVE THAN PHP

"You can't just copy-pase pseudocode into a program and expect it to work"

ProgrammerHumor.io

Osnove

- Nastao 1991 godine. Gvido van Rosum
- python.org
- Verzije
  - Python 2.7 - trenutno “end-of life" verzija
  - Python 3.10 – najčešće korištena verzija, 3.9+ se ne može koristiti na Win7 ili niže

Instalacija

- Windows - Downloads
- Linux - putem paket menadzera

Dva načina korisćenja

- REPL - read eval print loop

```
#! /usr/bin/env python
>>> print("doing REPL: \nHello World")
```

- Pokretanje skripte

```
#! /usr/bin/env python prvi.py
```

Leksičke konvencije:

Sintaksa i pravila pisanja Python programa

Elementi python jezika

Elementi python jezika

Struktura linije

- Svaki iskaz se završava u novoj liniji

```
a = b - a
b = b - a
a = b + a
```

- Dugi iskazi se mogu podijeliti u više redova:

```
a = math.sqrt(math.sin(b) + \
              math.cos(c) \
              )
```

- Osim kada se koriste (), {}, [] ...

```
a = {
              "a": 1,
              "b": 2,
              "c": 3
}
```

Elementi python jezika

Struktura linije

- Komentari pocinju sa #

```
# Ovo je komentar
```

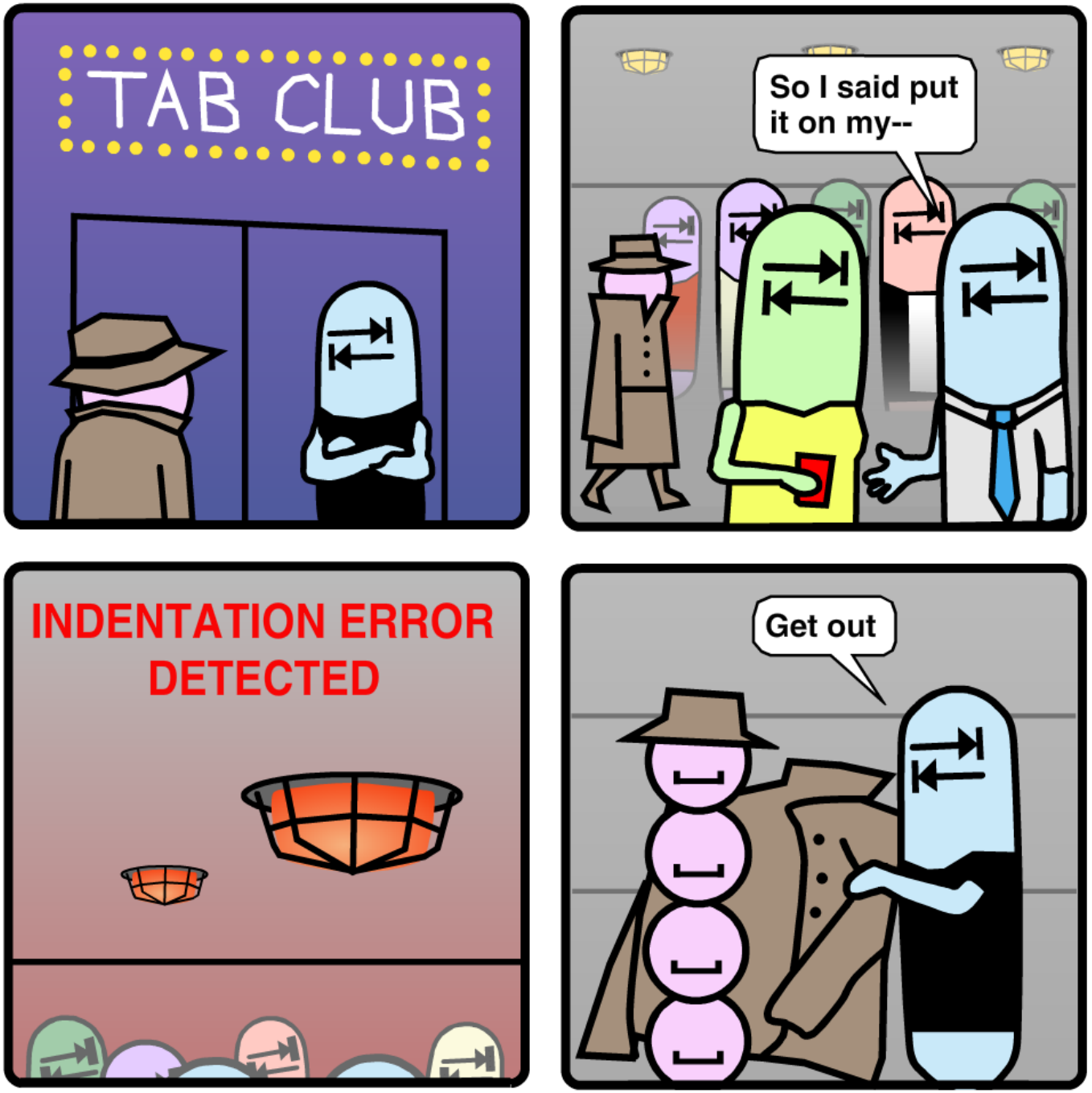
- Blokovi koda se odvajaju indentacijom

```
if a == 1:
        print "Ovo je uvučen kod"
```

- Indentacija je obavezna i mora biti konzistentna!

```
if a:
        iskaz1()
        iskaz2() # konzistentna indentacija
else:
        statement3()
    statement4() # nekonzistentna indentacija (greska)
```

- Preporucuje se 4 space-a, ne tab



Elementi python jezika

Identifikatori i rezervisane reči

- Imena promenljivih, funkcija, klasa, modula
- Mogu počinjati slovom ili donjom crtom
- Mogu sadržati brojeve
- Specijalni znaci kao *@,$, %* i rezervisane reci se ne mogu koristiti
- Rezervisane reci:

```
and as assert break class continue
def del elif else except exec finally
for from global if import in is lambda
nonlocal not or pass print raise return
try while with yield
```

- Identifikatori koji pocinju donjom crtom često imaju specijalno značenje. Više o tome kasnije...

Elementi python jezika

Literali

- Numerički literali
  - Boolean *(True, Flase)*
  - Integer *(1, 65536 ,0xbfffffc0)*
    - Proizvoljne dužine
  - Pokretni zarez *(1.54, 52. , .42, 1.2334e+02)*
  - Kompleksni brojevi *(4 + 5j)*

Elementi python jezika

Literali

- String literali
  - Sekvenca karaktera
  - Unutar jednostrukih ili dvostrukih navodnika
    - *"ovo je string" , 'i ovo je string'*
    - Proizvoljne dužine
  - Escape sekvence na koje smo navikli
    - *\\ \n \t \' \" \x41*
  - Stringovi su ASCII i predstavljeni kao niz bajtova
  - UTF-8 podrška

```
# -*- coding: UTF-8 -*-
print("тестирање ћирилице")
```

Elementi python jezika

Kolekcije

- Liste - *[1,2,3,4,5,6]*
- N-torke - *(1,4,2,7)*
- Riječnici - *{"a":1, "b":2, "c":3}*
- Skupovi – *{2, 3, 5}*

Elementi python jezika

Liste

- mogu da sadrže mješovite podatke
  a = ["asd",1, True]
- indeksiranje počinje od 0
  print(a[0])
- može se menjati *a[0] = 4*
- Za dodavanje *append* metoda

```
a.append(4)
```

Elementi python jezika

Torke

- efikasnija implementacija listi
  a = ("asd",1, True)
- Immutable - ne mogu se mijenjati
  a[0] = 4 Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  TypeError: 'tuple' object does not support item assignment
- *Ali ipak mogu:*
  a = a + (9,)

Elementi python jezika

Riječnici

- Key-value pair
- Vrijednosti se pristupa po ključu

```
import math
r = {"boja": "plava", "precnik": 3000}
print(r["boja"])
r["povrsina"] = r["precnik"] * 2 * math.pi
print(r)
```

Elementi python jezika

Operacije nad rečnicima

- Vrijednosti ključeva moraju biti immutable tipovi
- Indeksiranje

```
vrijednost = recnik[kljuc]
```

- dodjela vrijednosti

```
recnik[kljuc] = vrednost
```

- Brisanje iz rječnika del rjecnik[kljuc]

```
del recnik[kljuc]
```

Operacije nad rječnicima

- Testiranje pripadnosti

kljuc in rjecnik

```
rjecnik = {} # napravimo prazan rjecnik
rjecnik["naziv"] = "jabuka"
rjecnik["cijena"] = 4
print(rjecnik["naziv"])
print("cijena" in rjecnik)
del rjecnik["cijena"]
print("cijena" in rjecnik)
print(len(rjecnik))
```

Elementi python jezika

Skupovi

```
firme = {"Lacoste", "Ralph Lauren"}
firme.add("Cajevac")  # dodavanje firme
print(firme)
it_firme = ['Apple', ‘Google', 'Apple']
print(it_firme)
firme.update(it_firme)  # dodavanje liste
print(firme) # nema duplih
```

Elementi python jezika

Operacije nad skupovima

- Vrijednosti u skupu nemaju poredak
- Vrijednosti u skupu su jedinstvene
- Imaju posebne operacije
  - Unija |
  - Presjek &
  - Razlika -
  - Simetrična razlika ^

Operacije nad skupovima

- Primjer

```
s1 = {1,2,3,4,1,5,1,6}
print("s1", s1)
s2 = {4,5,6,7,8,9}
print("s2", s2)
unija = s1 | s2
print("unija", unija)
presjek = s1 & s2
print("presjek", presjek)
```

Elementi python jezika

Identitet i tip objekta

- Sve u programu je objekat.
- Objekat ima identitet, tip i vrijednost *a = 42*
- Python je strogo i dinamički tipiziran.
- Tip objekta predstavlja njegovu internu reprezentaciju.
- Objekat konkretnog tipa nazivamo instancom.
- Objekat može biti:
  - Mutable - ako vrijednost može da mu se izmjeni (primjer lista)
  - Immutable - ako ne može da se mijenja (primjer string)

Elementi python jezika

Identitet i tip objekta

- Ključne riječi:
  - *id()* - identitet objekta, memorijska lokacija
  - *is* - da li su dva objekta u stvari isti objekat?
  - *type()* - tip objekta
- Primjer poredjenja objekata:

```
if a is b:
   print("a i b imaju isti identitet")
if a == b:
   print("a i b imaju istu vrijednost")
if type(a) is type(b):
   print("a i b su istog tipa")
```

Reference i garbage collection

- Za svaki objekat se održava lista referenci
- Broj referenci se inkrementira dodjeljivanjem objekta novoj promjenljivoj ili dodavanjem u kolekciju.
- Smanjuje se kada promjenljiva izadje iz opsega ili joj se dodijeli drugi objekat.
- Memorija objekta biva oslobodjena (garbage collection) nakon sto broj referenci padne na nulu.

```
a = 37 # novi objekat
b = a # povecava broj referenci - a i b pokazuju na 37
c = []
c.append(b) # povecava broj referenci jer c[0] pokazuje na 37
b = 4 # smanjuje broj referenci na vrijednost 37, jer b sada pokazuje na 4
```

Reference i kopije

- Pri dodjeli vrijednosti, prave se nove reference.

```
a = b
```

- Za immutable objekte pravi se kopija.

```
s = "qwerty"
s1 = s
s = "abcde"
print(s1)
print(s)
```

Reference i kopije

- Za mutable objekte reference su ravnopravne.

```
a = [1,2,3,4,5,6]
b = a
b[4] = 1000
print(a)
print(b)
```

- Kopija mutable objekta (shallow copy – postoji i deep copy):

```
a = [1,2,3,4,5,6]
b = list(a)
b[4] = 1000
print(a)
print(b)
```

Elementi python jezika

First class objekti

- Svi objekti u Pythonu su "gradjani prve klase"
- To znači da su svi objekti koji imaju identifikator jednakog statusa
- Jasnije na primjeru:

```
import math
l = [1,2,3,math, "qwerty", math.sqrt]
kvadratni_korijen = l[5]
print(kvadratni_korijen(9))
```

Zadaci 1.

- Napraviti program koja za unesenu listu kreira novu listu bez duplih članova
- Napraviti program koji od unesenog skupa pravi dva skupa, jedan koji sadrzi samo brojeve I drugi koji sadrži sve druge tipove podataka
- Napraviti program koji za uneseni riječnik, pravi novi rječnik, čije su vrijednosti ključevi, a ključevi vrijednosti
- Napraviti program koji uneseni riječnik dijeli na skup ključeva i na listu vrijednosti

Operatori i izrazi:

Operacije nad brojevima, sekvencama, stringovima

Elementi python jezika

Operacije nad brojevima

- Standardni operatori se ponašaju očekivano:
  - +, -, /, *, %
- "Specijalni operatori" u Pythonu
  - //, **
- Bitwise operatori
  - <<, >>, &, |, ^, ~
- Obratiti pažnju da su integeri u pythonu "beskonačni"
- Ako je potrebno rukovanje "native" vrijednostima, koristiti *struct*
- Ugradjene funkcije: a*bs(), divmod(), pow(), round()*
- Uobičajene operacije poredjenja
  - <, <=, > , >= , == , *!=*

Elementi python jezika

Operacije nad sekvencama

- Sekvence su stringovi, liste i torke
- Dostupne su sljedeće operacije:
  - *s1* + *s2* - konkatenacija
  - *s* * *n* - ponavlja s n puta
  - *v1,v2,v3,v4* = *s1* - raspakivanje promenljivih
  - *s[i]* - indeksiranje
  - *s[i:j]* - isjecanje
  - *s[i:j:k]* - isjecanje sa korakom k
  - *x in s* - da li je x u s
  - *len(s)* - broj clanova s
  - *all(), any(), sum(),min(),max()*

Ponavljanje i kopije

- operator ponavljanja liste pravi shallow kopije
- Primer:

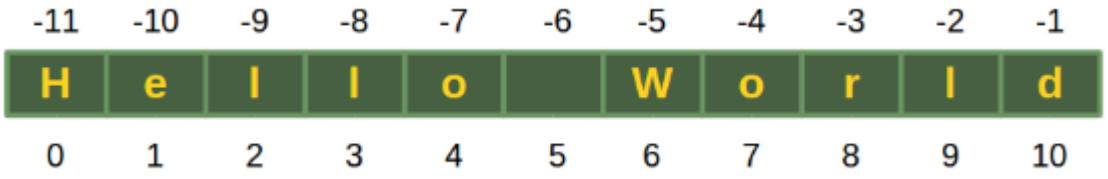
```
a = [1,2,3] # a je lista brojeva
b = [a] # b je lista koja sadrzi referencu na a
c = b * 4 # c sadrzi 4 reference na a
print(c)
a[1] = 10000
print(c)
```

Elementi python jezika

Isjecanja

- Parametri su opcioni
- Isjecanje je cirkularno
- Trikovi kod isecanja

```
s[:n] #od nultog do n-tog clana
s[n:] #od n-tog do kraja
s[-1:] #od posljednjeg do kraja
s[::-1] # invertovanje sekvence
s[::2] # svaki drugi clan sekvence pocev od prvog
```



Elementi python jezika

Konverzije

Funkcija	***Značenje***
int(x,(baza))	String x u integer iz baze
float(x)	String x u float
complex(r,i)	Kompleksni broj od r i I
str(x)	Bilo koji tip x u string
chr(i)	Integer u karakter (do 255)
ord(c)	Karakter u integer
hex(i)	Integer u hex string
bin(i)	Integer u bin string
oct(i)	Integer u oct string

Elementi python jezika

Logički izrazi

True	False
True	False
Bilo koji broj različit od 0	0
Kolekcija koja nije prazna	Kolekcija koja je prazna
	None

Operator	Značenje	Opis
x or y	Logičko ILI	Ako je x false, vraća y, u suprotnom x
x and y	Logičko I	Ako je x false, vraća x, u suprotnom y
not x	Logička negacija	Ako je x false, vraća 1, u suprotnom 0

Zadaci 2

- Napraviti program koji od zadatog niza pronalazi elemente koji su brojevi ne veći od jednog bajta i zamjenjuje te brojeve sa njihovom ASCII vrijednosti, odnosno karakterom
- Napraviti program koji provjerava da li je string palindrom
- Napraviti program koji od stringa pravi niz brojeva gdje svaki broj odgovara ASCII vrijednosti datog karaktera
- Podijeli string na podstringove gdje je delimiter zadati znak

Python kurs

Razvojni alati

Sadržaj

- pip
- Virtualenv
- IPython
- IDEs: Eclipse + PyDev
- PyCharm
- Setuptools

Not sure if C has taught me to think like a programmer

Or Python is just really easy

pip - osnovne komande

- Pretraga paketa po nazivu:

```
pip search dio_imena
```

- Instalacija paketa:

```
pip install ime_paketa
```

- Prikaz instaliranih paketa:

```
pip list
pip freeze
```

- Upgrade paketa:

```
pip install --upgrade ime_paketa
```

- Deinstalacija paketa:

```
pip uninstall ime_paketa
```

Virtualenv

- Problem sa zavisnošću i kolizijom između verzija.
- virtualenv omogućava kreiranje izolovanih Python okruženja sa svojim skupom paketa.
- Kada se aktivira određeno okruženje sistemski paketi kao i paketi iz drugih okruženja se ne vide.
- Kreiranje novog okruženja na windowsu:

```
C:\Users\srdjan>mkdir VirtualEnvs

C:\Users\srdjan>cd VirtualEnvs

C:\Users\srdjan\VirtualEnvs>virtualenv RTRK
New python executable in RTRK\Scripts\python.exe
Installing setuptools, pip...done.

C:\Users\srdjan\VirtualEnvs>
```

- Aktivacija virtuelnog okruženja

```
C:\Users\srdjan\VirtualEnvs>RTRK\Scripts\activate.bat
(RTRK) C:\Users\srdjan\VirtualEnvs>
```

- Listanje paketa u okruženju:

```
C:\Users\srdjan\VirtualEnvs>RTRK\Scripts\activate.bat
(RTRK) C:\Users\srdjan\VirtualEnvs>pip list
pip (1.5.6)
setuptools (3.6)

(RTRK) C:\Users\srdjan\VirtualEnvs>
```

IPython

- Interaktivni *shell* sličan standardnom
- Read-Eval-Print-Loop
- Razvoj kroz eksperimentisanje
- Instaliranje:

```
pip install ipython
```

- IPython mogućnosti:
  - Dopuna sa TAB tasterom
  - Istraživanje objekata sa ?
  - Autoreload modula
  - *Magic* funkcije

- Primjer sesije

```
C:\Users\srdjan>ipython2
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit
(Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 2.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print "Hello World"
Hello World
In [2]:
```

- Dopuna koda:
  - Pritisak na taster TAB

```
In [4]: import os

In [5]: os.pa
os.pardir os.path os.pathconf
os.pathconf_names os.pathsep

In [5]: os.pa
```

- Informacije o objektima
  - Iza naziva reference staviti znak "?"

```
In [7]: map?
Type: builtin_function_or_method
String form: <built-in function map>
Namespace: Python builtin
Docstring:
map(function, sequence[, sequence, ...]) -> list

Return a list of the results of applying the function to the items of
the argument sequence(s). If more than one sequence is given, the
function is called with an argument list consisting of the corresponding
item of each sequence, substituting None for missing values when not all
sequences have the same length. If the function is None, return a list
of the items of the sequence (or a list of tuples if more than one
sequence).

In [8]:
```

- Proširene informacije o objektima
  - Iza naziva reference staviti znak "??"

```
In [2]: import os
In [3]: os.path.abspath??
Type: function
String form: <function abspath at 0x7f723641b848>
File: /usr/lib/python2.7/posixpath.py
Definition: os.path.abspath(path)
Source:
def abspath(path):
"""Return an absolute path."""
if not isabs(path):
if isinstance(path, _unicode):
cwd = os.getcwdu()
else:
cwd = os.getcwd()
path = join(cwd, path)
return normpath(path)
```

- Reload modula:
  - Problem kod izmene koda posle import-a.
  - Dva načina:
    - *1. reload* funkcija

```
reload(moj_modul)
```

    - *2. autoreload* ekstenzija

```
%load_ext autoreload
%autoreload 2
```

Eclipse + PyDev

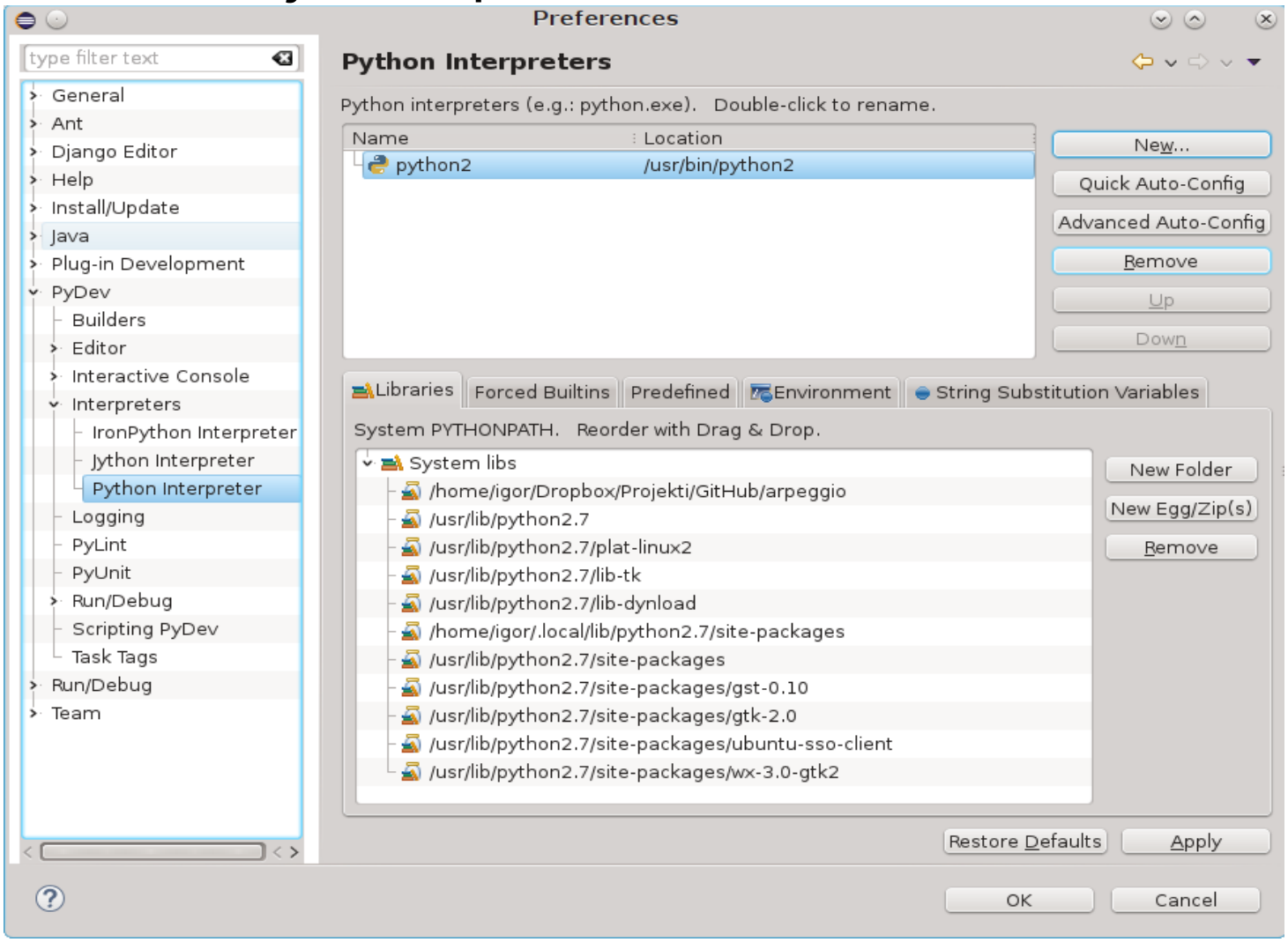
- Osnovne osobine
  - Slobodan softver otvorenog koda.
  - Dostupan kao skup plugin-a za Eclipse
  - Osnovne operacije: navigacija, strukturni prikaz, bojenje i dopuna koda...
  - Podrška za refaktorisanje.
  - Integrisani debager, interaktivna konzola, podrška za testiranje
  - Podrška za Jinja2 i Django template
  - Pisan u Javi, radi na svim vodećim OS

- Načini za instalaciju:
  - Eclipse distribucija sa već ugrađenim PyDev-om (npr. LiClipse)
  - Dropins zip arhiva
  - Update site: http://pydev.org/update

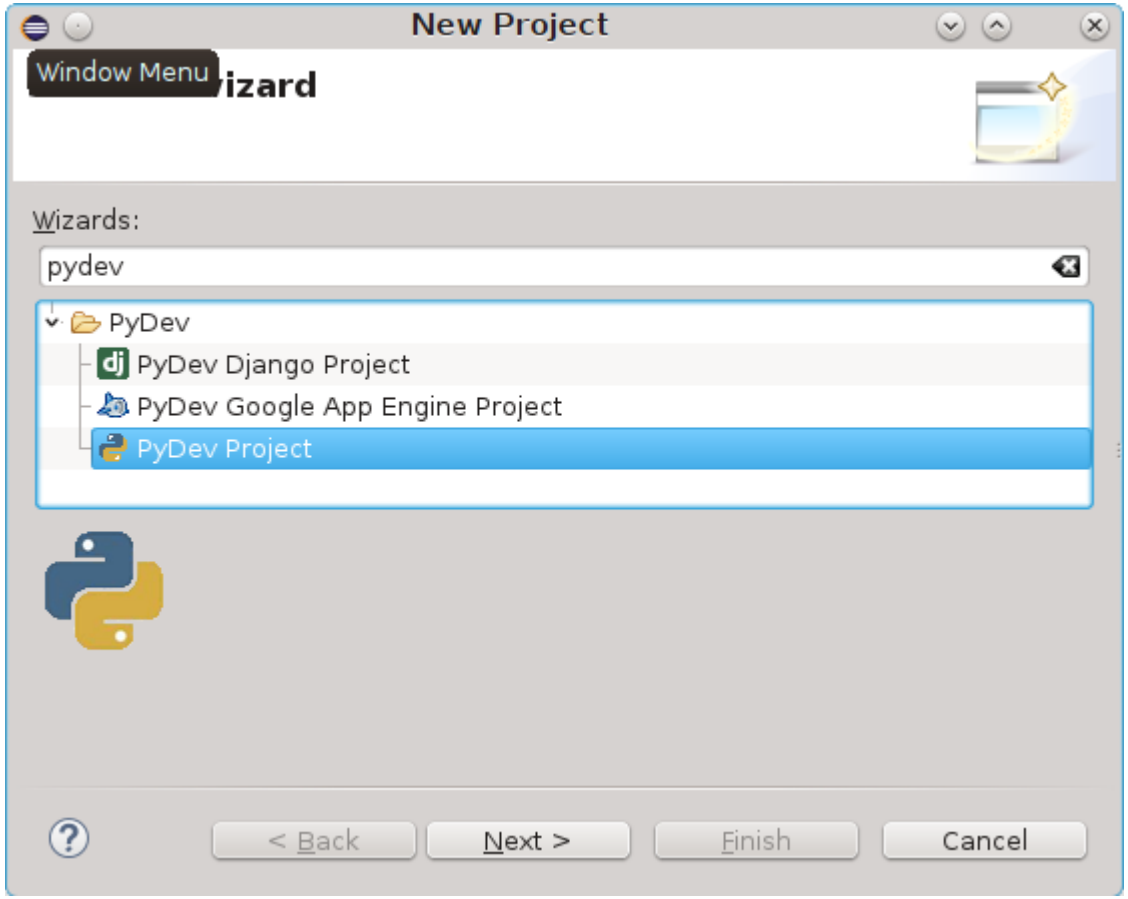
Konfiguracija

- Podešavanje interpretera:
  - Moguće je podesiti više interpretera (npr python 2 i 3).
  - Moguće je podesiti poseban interpreter za svaki projekat.
  - Obavlja se kroz standardni dijalog za konfigurisanje (*Window* > *Preferences*).
  - Potrebno je konfigurisati Python interpreter u sekciji *PyDev* > *Interpreter Python*
  - U većini slučajeva dovoljno je izabrati akciju *Auto Config*. Ukoliko Eclipse nije u stanju sam da pronađe Python interpreter to se može ručno definisati opcijom *New...*

- Podešavanje interpretera

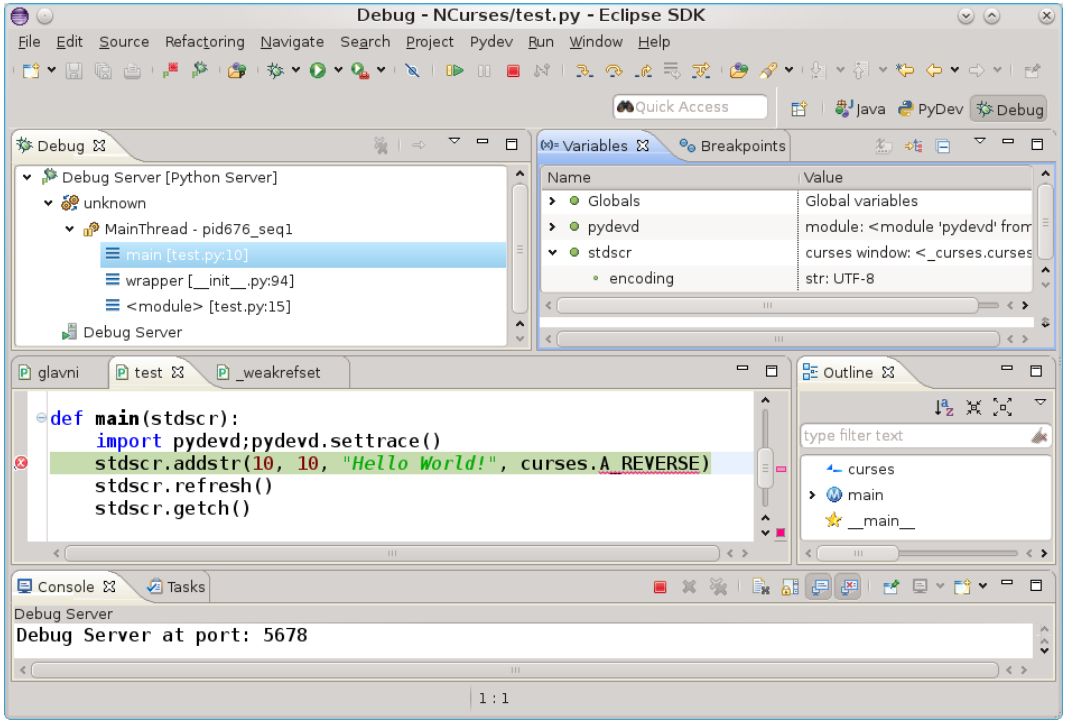


Kreiranje novog projekta



Debagovanje

- Debagovanje sa print i Python kodom
- Integrisani debager
  - Postavljanje prekidnih tačaka
  - Pokretanje debagera
  - Koračno izvršavanje i analiza varijabli



PyCharm

- https://www.jetbrains.com/pycharm/

Pakovanje i distribucija aplikacija u python-u

- Distutils
- Setuptools
- PyPi

Distutils

- Standardna biblioteka za upravljanje paketima.
- Dolazi uz instalaciju Pythona.
- Uglavnom se ne koristi, nego se koristi setuptools

Setuptools

- Naprednija verzija biblioteke za upravljanje paketima.
- Dobrim dijelom kompatiblina sa Distutils
- Na windows-u instalira se sa pip-om preko skripte get-pip.py

```
C:\>python get-pip.py
Downloading/unpacking pip
Downloading/unpacking setuptools
Installing collected packages: pip, setuptools
Successfully installed pip setuptools
Cleaning up...
```

- **Napomena:** Podesiti PATH da uključi Python Scripts folder.

setup.py fajl

- Metapodaci python paketa + informacije za *build*.
- Primer:

```
#!/usr/bin/env python

from setuptools import setup
#from distutils.core import setup

setup(  name='ImePaketa',
        py_modules = ['ime_modula']
version='1.0',
description='Opis paketa',
author='Ime i prezime autora',
author_email='mailautora@negde.com',
url='http://ulrprojekta.com/',
packages=['prvipaket', 'drugipaket',
                  'drugipaket.podpaket'], # ako je kod razvrstan
po folderima (paketima)
)
```

Instalacija iz setup.py

- Instalacija iz izvornog koda sa setup.py fajlom se obavlja komandom:

```
C:\Pygments-1.6> python setup.py install
copying pygments\styles\native.py -> build\lib\pygments\styles
...
Processing Pygments-1.6-py2.7.egg
creating c:\python27\lib\site-packages\Pygments-1.6-py2.7.egg
Extracting Pygments-1.6-py2.7.egg to c:\python27\lib\site-
packages
Adding Pygments 1.6 to easy-install.pth file
Installing pygmentize-script.py script to c:\python27\Scripts
Installing pygmentize.exe script to c:\python27\Scripts

Installed c:\python27\lib\site-packages\pygments-1.6-py2.7.egg
Processing dependencies for Pygments==1.6
Finished processing dependencies for Pygments==1.6

C:\Pygments-1.6>
```

Kreiranje installer-a iz setup.py

- Kreiranje binarnog installera za windows se obavlja sledećom komandom:

```
python setup.py py2exe
```

- Primer setup.py fajla:

```
from setuptools import setup
import py2exe
import os

setup(console=['file_name.py']
     ,data_files = [("template", templates)]
     ,options={
               "py2exe":{
                         "packages": ["jinja2"]
               }
     )
```

Instalacija za razvoj

- Ukoliko kôd koji želimo da instaliramo još uvijek razvijamo a želimo da izbjegnemo ponovnu instalaciju posle svake izmjene potrebno je da instaliramo paket na sljedeći način

```
python setup.py develop
```

- Za deinstalaciju razvojnog paketa koristi se:

```
python setup.py develop --uninstall
```

Python package index – PyPi

- PyPI (**Py**thon **P**ackage **I**ndex) predstavlja repozitorijum python paketa.
- Dostupan je na adresi https://pypi.python.org/
- Paketi se mogu pretraživati i prezimati putem web interfejsa ali i putem specijalizovanih alata.

Zadaci 3

- Na instaliranom PyCharm, kreirati novi projekat, koji se zove Dan01 i svu dosadašnju zadaću prebaciti u dati projekat

ELEMENTI

PYTHON JEZIKA

Funkcije

Funkcije

- Enkapsuliraju jedan odredjeni zadatak
- U Pythonu se definišu ključnom reči *def*
- Tijelo funkcije su izrazi koji se izvršavaju sekvencijalno

```
def xor(s1, s2):
    result = ""
    for b1,b2 in zip(s1,s2):
        result += chr(ord(b1) ^ ord(b2))
    return result
```

Parametri funkcije

- Neograničen broj parametara
- Parametri mogu imati podrazumijevane vrijednosti

```
def funkcija(a, b=0):
        return a+b
funkcija(4)
```

- Nakon prvog opcionog, svi ostali isto moraju biti opcioni
- Funkcije mogu imati promjenljivi broj parametara

```
def printf(fmt,*args):
        print(fmt%args)
printf("Moje ime je %s i imam %d godina","Pero",20)
```

- *args* predstavlja n-torku kojoj se može pristupati na uobičajen način, postoji i kwargs, koji predstavlja rječnik imenovanih argumenata. Pogledati primjer u **funkcije.py**
- *args* i *kwargs* mogu ići zajedno ali prvo *args*! Jer kwargs je generalizacija opcionih parametara (b=0 iz prvog primjera)

Prosljeđivanje parametara i povratne vrijednosti

- Mješavina "pass by value" i "pass by reference"
  - Ukoliko je proslijeđeni parametar immutable, može se smatrati da je "pass by value"
  - Ukoliko je mutable tip, ako mu se promijeni vrednost u funkciji, promjena je vidljiva i van funkcije
  - Preporuka je pisati "side-effect free" funkcije
    - Ulazne liste moraju proći kroz funkciju kao „read only“

```
def doubler(values):
    for i, old_value in enumerate(values):
        values[i] = old_value * 2
    return values
```

```
def doubler(values):
    new_values = []
    for value in values:
        new_values.append(value * 2)
    return new_values
```

Prosljeđivanje parametara i povratne vrijednosti

- Ključna reč *return* za povratne vrijednosti
  - *None* je podrazumjevana povratna vrijednost (kada nema ključne riječi *return)*
  - Više od jednog rezultata se može vratiti pomoću torki

```
def krug(poluprecnik):
        povrsina = 3.14 * (poluprecnik**2)
        obim = 2 * poluprecnik * 3.14
        return (povrsina,obim)
p,o = krug(4) # raspakivanje
print("Površina je %f a obim %f"%(p,o))
```

Lokalne i globalne promenljive

- Svaki poziv funkcije pravi novi lokalni namespace
- Interpreter promjenljivu po imenu traži prvo u lokalnom namespace-u pa zatim u globalnom
- Primjer:

```
a = 4
def test():
        a = 10 # osim toga sto nema smisla, takodje i zbunjuje
test()
print(a)
```

- Promjenljiva se eksplicitno proglasi globalnom pomoću *global*.

```
a = 4
def test():
        global a
        a = 10
test()
print(a)
```

Funkcije kao objekti

- Funkcije su "first class" objekti u Pythonu
- Mogu biti proslijeđene kao parametri ili vraćene kao rezultat
- Funkcije mogu biti definisane unutar drugih funkcija

```
def makeInc(x):
        def inc(y):
                return x+y
        return inc
inc10 = makeInc(10)
print( inc10(1) )
```

- Ovako definisane funkcije se nazivaju *closure* i obično se koriste za:
  - umjesto hardkodiranih konstanti
  - umjesto globalnih promjenljivih

Dekoratori

- Slično *closure*-u. Isto se definiše, sa razlikom da se dekoratoru prosljeđuje funkcija, koja se onda poziva u drugoj funkciji koja se pravi u dekoratoru
- Funkcija koja "obuhvata" drugu funkciju
- Funkcije koje se obuhvataju, počinju sa @

```
def something(f):
        def debug(*args,**kwargs):
                print("Pozivam funkciju %s"%f.__name__)
                return f(*args,**kwargs)
        return debug

@something
def test():
        print("test funkcija", 4)

@something
def druga_funkcija():
        print("druga funkcija", 10)
```

Dekoratori

- Korisni su kod operacija koje želimo u svakoj funkciji debug ispis
- logovanje u fajl
- ...
- Može biti više dekoratora i mogu imati parametre

Generatori i *yield*

- Generatori emituju sekvencu vrijednosti za iteracije.
- Rezultat se naznačuje pomoći *yield, j*asnije na primjeru:

```
def brojac(n):
        while n > 0:
                print("Stigao sam do %d"%n)
                yield n
                n -= 1
        return
```

- Poziva se *next()* metoda generatora.
  - Python 2 - generator.next(),
  - Python 3 – next(generator)
- Izvršenje se prekida nakon yeild i nastavlja sljedećim pozivom *next()*
- Obično se *next()* ne poziva eksplicitno, već u okviru *for, sum* i sličnim operacijama.

Korutine i *yield*

- Za razliku od generatora, korutine primaju vrijednosti.
- *yield* predstavlja vrijednost proslijeđenu korutini.
- Jasnije na primjeru:

```
def korutina():
        print("Cekam na podatak")
        try:
                while True:
                        n = (yield)
                        print("Primljeno %r"%n)
        except GeneratorExit:
                print("Kraj korutine")
```

- Neophodno je prvo pozvati *next()* kako bi se došlo do *yield*.
- Zatim sa *send()* poslati odgovarajuću vrijednost.

Elementi python jezika

Čemu sve to?

- Na prvi pogled sve sto mogu dekoratori, generatori i korutine može da se postigne i bez njih.
- ALI, pravilnim korišćenjem se dobija čistiji i efikasniji kod.
- Primjer - pipeline za obradu podataka:
  - Iz jednog skupa filtriraj podatke u podskup.
  - Obradi podatke i napravi novi podskup.
  - Još jedna obrada i još jedan podskup.
  - Prikaži rezultate.
- Implementacijom pomocu generatora nema privremenih listi/rječnika/promjenljivih.
- Efikasniji kod sa manje zauzeća memorije.
- Zgodno za potencijalnu paralelizaciju i distribuiranje.

Elementi python jezika

Čemu sve to?

- Pogledati primjer 12_pajplajn.py
- I od samih generatora se moze napraviti pajplajn, u ovom slučaju 4 generatora:
  1. Nalazi sve fajlove po odgovarajućem šablonu u odgovarajućem folderu - generator, generiše fajl po fajl
  2. Otvara ih jedan po jedan. Ima tu i malo više od filterisanja, jer za različite tipove imaju različiti otvarači. Naveden je lošprimjer, ali je tu da bi objasnio mogućnosti.
  3. Čita linije iz otvorenih fajlova – linija po linija
  4. Filtrira linije
- Promijeniti i pokrenuti

Sekvence i funkcije

- Česta je potreba da primenimo funkciju nad svim članovima liste
- Poseban operator nazvan *list comprehension*

```
brojevi = range(5)
kvadrati = [n * n for n in brojevi]
```

- Moguće je dodati i uslove:

```
brojevi = range(5)
parni_kvadrati = [n * n for n in brojevi if n%2 == 0 ]
```

Sekvence i funkcije

- Ili više sekvenci:

```
i = [1,2,3,4,5]
a = ['a','b','c','d','e']
z = [(x,y) for x in a for y in i if y > 2]
```

- Raspetljano, prethodni primjer u stvari izgleda ovako:

```
i = [1,2,3,4,5]
a = ['a','b','c','d','e']
z = []
for x in a:
        for y in i:
                if y > 2:
                        z.append((y,x))
```

Elementi python jezika

- Ni rjecnici nisu imuni na komprehenziju, napravimo novi rjecnik koji ima samo ime i platu:

```
sampleDict = {
  "name": "Kelly",
  "age":25,
  "salary": 8000,
  "city": "New york" }

keys = ["name", "salary"]

newDict = {k: sampleDict[k] for k in keys}
print(newDict)
```

Zadaci 4

- Prepraviti sve zadatke do sada urađene tako da su funkcije
- Napraviti funkciju koja dijeli dva ulazna parametra i vraća količnik, a zatim napraviti dekorator koji će provjeriti da li može da se dijeli: da li su oba podatka brojevi i da li je djeljenik različit od nule
- Napraviti *closure* funkciju koja za različit ulazni argument kreira funkciju za izračunavanja površina, kruga, kvadrata i pravougaonika.
- Napraviti generator koji generiše sve proste brojeve između dva zadata broja.
- Napraviti korutinu koja za dati generator pronalazi najbliži broj djeljiv sa 7

Zadaci 5

- Po uzoru na pajplajn primjer, napraviti pajplajn koji lista sve fajlove(jedan pajp), filteruje samo txt i py fajlove (drugi pajp), pronalazi (u Pajtonu liniju u kojoj piše print, u txt liniju koja nije prazna) treći pajp, i štampa je ali velikim slovima (kod pajtona štampa samo ono unutar navodnika printa)
- Koristeći generator prostih brojeva napraviti listu prostih brojeva čiji zbir cifri je djeljiv sa 7 - pomoću komprehenzije
- Pomoću komprehenzije i riječnika sa nazivom(ključ) i cijenom (vrijednost) proizvoda napraviti rječnik u kojem su cijene uvećane za PDV (17%)

ELEMENTI

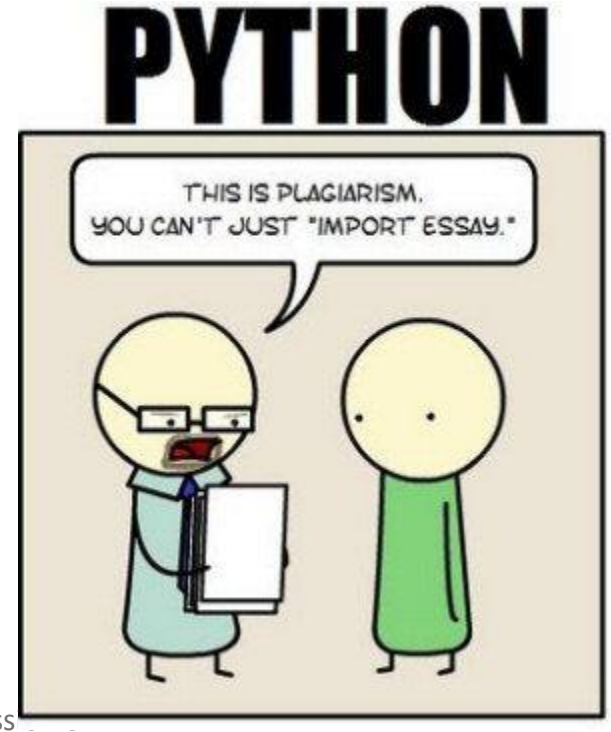
PYTHON JEZIKA

Moduli i Paketi

Moduli

- Veliki dijelovi koda, biblioteke, su organizovani u module.
- Standardna Python instalacija dolazi sa velikim brojem modula.
- Bilo koji Python izvorni kod može da se koristi kao modul.
- Za uključivanje modula u izvorni kod koristi se

```
import ime_modula
```



Moduli

- U modulu mogu biti definisane klase, funkcije i globalne promjenljive.
- Primjer:

```
#modul.py
def spam(eggs):
        print(eggs)
```

```
#program.py
import modul

#poziv funkcije iz modula
modul.spam("test")
```

Moduli i *import*

- Pri importovanju modula, modul u stvari biva izvršen.
- Ako se u modulu osim definicija klasa, promjenljivih i funkcija nalaze izvršni izrazi, biće izvršeni pri importovanju.
- Primjer:

```
class Spam(object):
        def __init__(self,eggs):
                self.eggs = eggs
print("Ovo ce biti izvrseno pri importovanju")
```

Moduli i import

- Ime importovanog modula može biti promijenjeno:

```
import spam as eggs
a = eggs.Spam("test")
```

- Importovanje ne mora biti na početku fajla, a može biti i uslovno:

```
if format == "xml":
    import xmlreader as reader
elif format == "csv":
    import csvreader as reader
    data = reader.read_data(filename)
```

Selektivno importovanje

- Kada ne želimo čitav modul.
- *from modul import definicija <as ime>*
- Izbjegavamo novi namespace.

```
from math import sqrt
sqrt(45) # importujemo samo jednu funkciju
from os import *
system("dir") # importovali smo sve, ali ih ne moramo pozivati sa os.
from socket import socket as sock
s = sock() # importovali jednu stvar I jos je preimenovali
```

Izvršavanje glavnog programa

- Kao što je već viđeno, program pokrećemo tako što proslijedimo ime fajla interpreteru.

```
python program.py
```

- Svaki modul implicitno definiše promjenljivu *__name__*.
- *__name__* sadrži samo ime modula.
- Pri pokretanju programa, početni "modul" ima ime *__main__*
- Na osnovu ovoga razlikujemo da li je modul importovan ili pokrenut kao glavni program.
- Ekvivalent *main()* funkcije u C-u:

```
def test():
        print("Ovo je test!“)
if __name__ == '__main__':
        test()
```

Putanje i nalaženje modula

- Pri importovanju se vrši pretraga za traženim modulom.
- Lista direktorijuma u kojima se vrši pretraga se nalazi u sys.path.
- Prvi direktorijum koji se pretražuje je trenutni, CWD.
- Kako je *sys.path* obična lista, može se dodati putanja za pretragu.
- Postoji podrška za arhive.

```
import sys
print(sys.path)  # stampa environment varijablu PATH
sys.path.append("neka_putanja")
sys.path.append("zipovani_moduli.zip")
sys.path.append("zipovani_moduli.zip/lib/")
```

Paketi

- Nekad je zgodno module organizovati u pakete, čime formiramo biblioteke.
- Tematski slični moduli se grupišu u pakete čime se smanjuje problem s namespace-ovima.
- Paket je u Pythonu predstavljen direktorijumom.
- Svaki paket sadrži *__init__.py* fajl, module i/ili podpakete.
- *__init__.py* može da sadrži kod za inicijalizaciju paketa, a može i da bude prazan.
- Biva pokrenut pri prvom importovanju modula iz paketa.

```
import Paket.PodPaket.modul1
k = Paket.PodPaket.modul1.Klasa()
import Paket.PodPaket.modul2 as modul2
k = modul2.Klasa()
```

Zadaci 6

- Napraviti pakete za sve dosadašnje zadatke (zadaci1, zadaci2, ... zadaci5) nema zadaci6! Prebaciti odgovarajuću zadaću u odgovarajuće pakete
- Napraviti u projektu poseban fajl main.py u rootu od projekta, u kojem ce se pokrenuti svi drugi zadaci, koji su pokretani u pojedinačnim fajlovima. Importi i sve to da radi.

ELEMENTI

PYTHON JEZIKA

IO
Argumenti, okruženje, fajlovi

Elementi python jezika

Komandna linija

- Uobičajeno je da programi pri pokretanju primaju argumente sa kompandne linije.
- Python interpreter argumente komandne linije smesta u *sys.argv* listu.
- Argumentima pristupamo kao i bilo kojoj drugoj listi:

```
import sys
if len(sys.argv) != 2:
        print("Usage: %s <n>"%sys.argv[0])
else:
        print("Prevaram broj %s u heksadecimalni: %s"%(sys.argv[1], hex(int(sys.argv[1]))))
```

Elementi python jezika

Komandna linija – pomocni alati - parseri

- Dva su najpoznatija argparse i optparse
- Optparse je bolji
- Pogledati primjere **15*.py**

Elementi python jezika

Promjenljive okruženja

- Često je neophodno pročitati promjenljive okruženja kao što su PATH, USER ili neke specifičnije. Iako smo vidjeli da i preko *sys* možemo doći do PATH,
- U modulu *os* rečnik *environ* sadrži promjenljive okruženja.

```
import os
print(os.environ["PATH"])
print(os.environ["USER"])
```

- Pošto je u pitanju riječnik, nove promjenljive dodajemo lako:

```
import os
os.environ["NOVA_PROMJENLJIVA"] = "NOVA_PROM_OKRUZENJA"
print(os.environ["NOVA_PROMJENLJIVA"])
```

Elementi python jezika

Rukovanje fajlovima

- Fajlovima se rukuje ugradjenom funkcijom *open(filename,options)*
- U opcijama specificiramo koji način pristupa želimo:

Oznaka	Značenje
r	read
w	write
a	append
b	binary file - modifikator
+	update
-	modifikator
U	newline modifikator

Pregled metoda fajl objekta

Metoda	*Značenje*
file.read([n])	Čitanje n bajtova ili ceo fajl.
file.readline([n])	Čita jednu liniju iz tekst fajla ili n bajtova linije
file.readlines()	Čita sve linije iz fajla u listu linija
file.write(s)	Upisivanje sekvence u fajl
f.writelines(l)	Upisivanje sekvence linija u
fajl f.close()	Zatvaranje fajla
f.tell()	Trenutna vrednost offseta unutar fajla
f.seek(offset)	Pomeranje na ofset fajla
f.flush()	Pražnjenje bafera
f.truncate(n)	Skraćivanje fajla na max n bajtova
f.fileno()	Broj fajl deskriptora
f.next()	Čita sledeću liniju, za iteracije

Kontekst menadžer za otvaranje fajlova

```
with open("fajl.txt") as f:
    data = f.read()
```

Elementi Python jezika

- Izuzeci – exception
- Podizanje izuzetka:
  - raise Exception()
- Reagovanje na izuzetak:

```
try:
    kontaktiraj_neki_sistem()
except:
    print("neuspjesno kontaktiranje sistema")
else:
    print("uspio sam kontaktirati sistem, idem dalje")
finally:
    print('Ako ima sta da se pocisti, ovdje cu biti u svakom slucaju')
```

Kontekst Menadžer

- Rješava nam try-catch-finally muke
- Ukoliko se trebamo nakačiti na neku bazu podataka, pa sav taj boiler-plate kod možemo sakriti u kontekst menadžer
- Otvaranje fajlova, smo već vidjeli
- Postoje 4 koncepta:
  - Otvori/zatvori – fajlovi i baze
  - Pokreni/zaustavi – pokreni tajmer i zaustavi
  - Zaključaj/otključaj – muteksi, semafori...
  - Promijeni/resetuj – stanje ili osobinu, ukoliko imamo

Kontekst Menadžer

```
class KontekstMenadzer:
    def __init__(self, bilo_koji_skup_parametara):
        pass # konstruktor nije obavezan

    def __enter__(self):
        # kreiraj objekat nad kojim ce biti vrsena manipulacija
        return objekat # na primjer to je taj objekat

    def __exit__(self, exc_type, exc_value, traceback):
        if objekat_uredan:
            pocisti_za_objektom()
        if exc_type:
            pocisti_krs_i_lom(exc_value, traceback)
        return True # ili False ako necemo ove greske da precutimo

with KontekstMenadzer(parametri) as onaj_objekat:
    onaj_objekat.na_poso()
    ...
```

```
instanca = KontekstMenadzer()
ctx = instanca.__enter__()
try:
    # radi nesto sa kontekstom
finally:
    # pocisti
    instance.__exit__()
```

Zadaci 7

- Napraviti program koji kreira fajl pod imenom <username>.txt sa sadržajem <password>. Ukoliko fajl postoji, provjeri se da li je sadržaj isti kao i <password> ukoliko nije, mijenja se, ukoliko jeste – ništa. Podatke <username> i <password> procitati iz *environment* varijabli.
- Zadatak 2.1 prepraviti da baca Exception ukoliko broj nije iz opsega printabilnih karaktera. Prepraviti i poziv (sada) funkcije.
- Prepraviti sve funkcije tako što se provjeri da li je tip podataka onaj koji je tražen, ukoliko nije, podići AttributeError. Prepraviti sve pozive da uzimaju ovo u obzir.

Zadaci 8

- Napraviti kontekst menadzer koji mjeri vrijeme provedeno unutar konteksta.

```
from time import perf_counter
```

Contact us

RT-RK Institute for Computer Based Systems
Narodnog fronta 23a
21000 Novi Sad
Serbia

www.rt-rk.com
info@rt-rk.com

RT-RK
COMPUTER BASED SYSTEMS