Python

U pythonu nema ; na kraju naredbe kao u perl-u i c-u

Komentari => sve iza znaka #

Komentari od više redova => """komentari od više redova"""

Print "Hello" -> na ekran se ispisuje Hello isto kao i print u perl-u

Python radi kao kalkulator tj. Bilo koji aritmetički izraz izračunava odmah i ispisuje rezultat

Hexa brojevi počinju sa 0x na početku a oktalni sa 0

Ukoliko se želi u ispis staviti neki posebni znak stavlja se \prije njega

exit() -> prijevremeni prekid programa

Program mora imati zaglavlje #!usr/bin/env python

Inicijaliziranje varijabli

x = broj -> inicijalizacija kao u perlu samo bez znakova @, %, &, $ ispred varijable, polja, hasha

x = 2

x*2 -> na ekran se ispisuje 4

Davanje imena varijablama iz vrijednosti drugih varijabla

```
var = "ovo"

vars()[var] = 123

print ovo
```

Dobivanje unosa od usera => input(...)

input("x: ") -> ako upišemo 10 ispiše se x: 10

Dobivanje unosa od usera => raw_input(...)

```
ime = raw_input("Kako se zoveš?" )

print "Bok, " + ime + "!" -> ispiše Bok, Darko! ukoliko smo unijeli Darko
```

Moduli

Moduli služe za uporabu funkcija koje se ne nalaze unutra standardnog modula -> import os

Import math -> uključuje modul u kojem se nalaze matematičke funkcije, funkcije uključenog modula se pozivaju tako da se stavi prvo ime modula pa . pa ime funkcije npr. math.sqrt(), no ako to isto uključimo ovako: from math import sqrt onda možemo direkto koristiti npr. sqrt()

import cmath -> sve funckije modula math samo što ove služe za kompleksne brojeve

Funkcije

** -> **potencija** 2**3 znači 2^3

Opis funkcija

abs(number)	Returns the absolute value of a number
cmath.sqrt(number)	Returns the square root; works with negative numbers
float(object)	Converts a string or number to a floating-point number
help()	Offers interactive help
input(prompt)	Gets input from the user
int(object)	Converts a string or number to an integer
long(object)	Converts a string or number to a long integer
math.ceil(number)	Returns the ceiling of a number as a float
math.floor(number)	Returns the floor of a number as a float
math.sqrt(number)	Returns the square root; doesn't work with negative numbers
pow(x, y[, z])	Returns x to the power of y (modulo z)
raw_input(prompt)	Gets input from the user, as a string
repr(object)	Returns a string representation of a value
round(number[, ndigits])	Rounds a number to a given precision
str(object)	Converts a value to a string

Backquotes => ``

služe za pokretanje varijabli npr.

```
temp = 42

print "The temperature is: " + `temp`        ->        ispiše se The temperature is: 42
```

Dugački stringovi => """string""" ili '''string'''

ako imamo string kroz više redova na početak i kraj se stavi ''' ili """

```
print ''' ovdje je početak

tu se nastavlja

Bok

ide sve do tuda'''
```

primjer iznad samo što ne stavljamo ''' ili """ nego na kraj svakog reda \

```
print " ovdje je početak \

tu se nastavlja \

Bok \

ide sve do tuda"
```

Sirovi stringovi => print r'izraz' ili \znak koji treba uključiti

path = 'C:\nowhere'

print path

c:

owhere

kako bi se izbjegla ta situacija moramo staviti to sa \ tj.

path = 'c:\\nowhere' -> ovako smo uključili jedan \

ili kod printa staviti print r'izraz'

Liste i polja

imena = ['Darko', 'Marko', ...] -> lista sa imenima

lista = [] -> prazna lista

ime = 'Darko'

ime[0] -> dobijemo D

Liste i stingovi

s = "string"

t = list(s)

t = ['s', 't', 'r', 'i', 'n', 'g']

Funkcije => len(lista), min(lista), max(lista), list(string), del imeliste[element]

len -> vraća duljinu liste

min -> vraća najmanji element

max -> vraća najveći element

list -> pretvara string u listu

del -> briše listu ili element liste

stringovi se mogu komadati kao da su njihovi dijelovi članovi polja

[-1] -> dobiva se posljednji član polja

komadanje polja => [od:do]

služi za dobivanje nekog dijela polja

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
numbers[3:6]

[4, 5, 6]

komadanje polja => [od:do:za koliki korak]

sve isto kao i gore samo ide se od do ne za 1 nego za korak određen u trećem dijelu

Kvantifikatori => + *

+ služi za dodavanje stringova/polja

[1, 2, 3] + [4, 5, 6] -> [1, 2, 3, 4, 5, 6]

'Bok ' + 'Bok' -> 'Bok Bok'

*** služi za ponavljanje**

[42] * 3 -> [42, 42, 42]

'Darko' * 2 -> DarkoDarko

Metode za liste

se koriste u obliku objekt.metoda(argumenti)

append -> dodaje element na kraj

lista = [1, 2]

lista.append(3)

lista = [1, 2, 3]

count -> broji koliko se puta pojavio zadani element

lista = ['to', 'be', 'or', 'not', 'to', 'be']

lista.count('to')

2

extend -> proširuje listu tako da doda elemente na kraj

a = [1, 2, 3]
b = [4, 5, 6]
a.extend(b) ili a + b
a
[1, 2, 3, 4, 5, 6]

index -> vraća index zadanog elementa

knights = ['We', 'are', 'the', 'knights', 'who', 'say', 'ni']
knights.index('who')
4

insert -> na određeni index stavlja element insert(index, element)

```
numbers = [1, 2, 3, 5, 6, 7]
numbers.insert(0, 'four')
numbers
[ 'four', 1, 2, 3, 5, 6, 7]
```

pop -> miče indeksirani element iz liste pop(index), pop() -> miče zadnji element

```
x = [1, 2, 3]
x.pop()
3
x
[1, 2]
x.pop(0)
1
x
[2]
```

remove -> koristi se kako bi izbacili prvo pojavljivanje elementa iz liste

```
x = ['to', 'be', 'or', 'not', 'to', 'be']
x.remove('be')
x
['to', 'or', 'not', 'to', 'be']
```

reverse -> metoda koja okrene cijelu listu

```
x = [1, 2, 3]
x.reverse()
x
[3, 2, 1]
```

sort -> sortira listu uzlazno

```
x = [4, 6, 2, 1, 7, 9]
x.sort()
x
[1, 2, 4, 6, 7, 9]
```

sorted -> vraća sortiranu kopiju liste

```
x = [4, 6, 2, 1, 7, 9]
y = sorted(x)
x
[4, 6, 2, 1, 7, 9]
y
[1, 2, 4, 6, 7, 9]
```

cmp -> sortira na način na koji joj se odredi ako x<y negativan broj vraća, x>y pozitivan broj vraća, x=y nula

```
cmp(42, 32)
1
```

```
cmp(99, 100)
-1
cmp(10, 10)
0
numbers = [5, 2, 9, 7]
numbers.sort(cmp)
numbers
[2, 5, 7, 9]

x = ['aardvark', 'abalone', 'acme', 'add', 'aerate']
x.sort(key=len)
x
['add', 'acme', 'aerate', 'abalone', 'aardvark']

x.sort(reverse=True)
x
[9, 7, 6, 4, 2, 1]
```

Formatiranje string-a

Operator specifičnog formata => % na lijevo stavimo varijablu a na desno vrijednost koja će nadopuniti

```
format = "Hello, %s. %s enough for ya?"
values = ('world', 'Hot')
print format % values
Hello, world. Hot enough for ya?

format = "Pi with three decimals: %.3f"
from math import pi
print format % pi
Pi with three decimals: 3.142

x = 4

y = 2

print '(%f, %f)' % (x, y)
```

Metode za stringove

find -> vraća poziciju zadanog elementa unutar stringa na krajnje lijevoj strani, ako nema vraća -1

```
'With a moo-moo here, and a moo-moo there'.find('moo')
7
title = "Monty Python's Flying Circus"
title.find('Monty')
0
```

join -> inverzna funkcija od split, spaja elemente neke sekvence

```
seq = ['1', '2', '3', '4', '5']
>>> sep.join(seq) # Joining a list of strings
'1+2+3+4+5'
```

lower -> vraća string tako da su sva slova mala

'Trondheim Hammer Dance'.lower()
'trondheim hammer dance'

upper -> isko ko lower samo sve veliko

title -> vraća string u obliku naslova

"that's all folks".title()
"That'S All, Folks"

replace -> zamijenimo svako pojavljivanje nekog elementa stringa drugim elementom

'ovo je stringovo'.replace('ovo', 'To')

'To je stringTo'

split -> funkcija koja razdvaja string po određenom graničniku

'1+2+3+4+5+6'.split('+')

['1', '2', '3', '4', '5', '6']

strip -> vraća string u kojemu su praznine s lijeve i desne strane maknute

' koje testiranje '.strip()

'koje testiranje'

```
names = ['gumby', 'smith', 'jones']
name = 'gumby '
if name in names: print 'Found it!'
```

Dictionaries/Hash

kreira se:

```
phonebook {}      -> prazan hash
phonebook = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}
```

dict -> služi za kreiranje hash-a

```
items = [('ime', 'darko'), ('godina', 22)]

d = dict(items)

d

{'age': 42, 'name': 'Gumby'}

x = {}
x[42] = 'Foobar'
x
{42: 'Foobar'}
```

Metode hash-a

len(d) -> vraća broj elemenata (parova) unutar d hash-a

d[k] -> vraća vrijednost od ključa k

del d[k] -> briše element sa k ključem

clear -> briše sve elemente hash-a

```
d = {}
d['name'] = 'Gumby'
d['age'] = 42
d
{'age': 42, 'name': 'Gumby'}

returned_value = d.clear()
d
{}
print returned_value
None
```

copy -> kopira stari hash u novi

```
x = {'username': 'admin', 'machines': ['foo', 'bar', 'baz']}
y = x.copy()
```

fromkeys -> kreira novi hash sa stari ključevima i nepoznatim vrijednostima

```
{}.fromkeys(['name', 'age'])
{'age': None, 'name': None}
```

has_key -> provjerava da li traženi hash ima ključ u sebi, ako ne vraća False inače True

```
d = {}
d.has_key('name')
False
d['name'] = 'Eric'
d.has_key('name')
True
```

items -> vraća listu vrijednosti hash-a u obliku (ključ, vrijednost)

```
d = {'title': 'Python Web Site', 'url': 'http://www.python.org', 'spam': 0}
d.items()
[('url', 'http://www.python.org'), ('spam', 0), ('title', 'Python Web Site')]
```

keys -> vraća listu ključeva zadanog hash-a

pop -> vraća vrijednost zadanog ključa te tada briše taj par ključ-vrijednost iz hash-a

```
>>> d = {'x': 1, 'y': 2}
>>> d.pop('x')
1
```

```
>>> d
{'y': 2}
```

popitem -> iz hash-a izbacuje zadani par ključ-vrijednost

```
d
{'url': 'http://www.python.org', 'spam': 0, 'title': 'Python Web Site'}
d.popitem()
('url', 'http://www.python.org')
d
{'spam': 0, 'title': 'Python Web Site'}
```

update -> nadograđuje hash-a s parovima iz drugog hash-a

```
d = {
'title': 'Python Web Site',
'url': 'http://www.python.org',
'changed': 'Mar 14 22:09:15 MET 2008'
}
x = {'title': 'Python Language Website'}
d.update(x)
d
{'url': 'http://www.python.org', 'changed':
'Mar 14 22:09:15 MET 2008', 'title': 'Python Language Website'}
```

values -> vraća listu vrijednosti hash-a

import naredba

1. import ime_modula

2. from ime_modula import ime_funkcije

3. from ime_modula import ime_f, ime_f2, ime_f3

4. from ime_modula import *

5. import ime_modula as alternativno_ime_modula

import => time

time.sleep(broj sekundi)

Blokovi u programu

```
ovo_je_linija

ovo_je_druga_lnija:

        ovo_je_unutrašnjost_bloka

        isto_sam_unutra

izašao_sam_van
```

Uvjeti

x == y	x equals y.
x < y	x is less than y.
x > y	x is greater than y.
x >= y	x is greater than or equal to y.
x <= y	x is less than or equal to y.
x != y	x is not equal to y.
x is y	x and y are the same object.
x is not y	x and y are different objects.
x in y	x is a member of the container (e.g., sequence) y.
x not in y	x is not a member of the container (e.g., sequence) y.

Petlje

If

```
name = raw_input('What is your name? ')
if name.endswith('Gumby'):
print 'Hello, Mr. Gumby'
```

If-else

```
name = raw_input('What is your name? ')
if name.endswith('Gumby'):
print 'Hello, Mr. Gumby'
else:
print 'Hello, stranger'
```

if-elif-else

```
num = input('Enter a number: ')
if num > 0:
print 'The number is positive'
elif num < 0:
print 'The number is negative'
else:
print 'The number is zero'
```

Operator pripadanja => in

```
dozvola = 'rw'

w in dozvola -> true

x in dozvola -> false

users = ['mlh', 'foo', 'bar']
raw_input('Enter your user name: ') in users
Enter your user name: mlh
True

database = [
['albert', '1234'],
```

```
['dilbert', '4242'],
['smith', '7524'],
['jones', '9843']
]
username = raw_input('User name: ')
pin = raw_input('PIN code: ')
if [username, pin] in database: print 'Access granted'
```

While

```
x = 1
while x <= 100:
print x
x += 1
```

For

```
words = ['this', 'is', 'an', 'ex', 'parrot']
for word in words:
print word

numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for number in numbers:
print number
```

range funkcija -> vraća nam broj elemenata ili listu elemenata

xrange ([*start*], *stop*[, *step*]) -> vrlo slična range samo što ne vraća listu nego objekt

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

for number in range(1,101):
print number

d = {'x': 1, 'y': 2, 'z': 3}
for key in d:
print key, 'corresponds to', d[key]

for key, value in d.items():
print key, 'corresponds to', value

for i in range(len(names)):
print names[i], 'is', ages[i], 'years old'

for index, string in enumerate(strings):
if 'xxx' in string:
strings[index] = '[censored]'
```

break -> izlaz iz petlje

```
from math import sqrt
```

```
for n in range(99, 0, -1):
root = sqrt(n)
if root == int(root):
print n
break
```

continue -> prekid trenunte iteracije i skok na iduću

```
for x in seq:
if condition1: continue
if condition2: continue
if condition3: continue
do_something()
do_something_else()
do_another_thing()
etc()
```

exec -> služi za pokretanje naredbi

eval -> evaluira izraz te vraća krajnju vrijednost

Korisničke Funkcije

def -> koristi se za definiranje funkcije

```
def hello(name):
return 'Hello, ' + name + '!'

print hello('world')
Hello, world!
print hello('Gumby')
Hello, Gumby!
```

dokumentiranje funkcije =>__doc__

```
def square(x):
'Calculates the square of the number x.'
return x*x
The docstring may be accessed like this:
square.__doc__
'Calculates the square of the number x.'
```

prijenos parametra

```
def change(n):
n[0] = 'Mr. Gumby'
names = ['Mrs. Entity', 'Mrs. Thing']
change(names)
names
['Mr. Gumby', 'Mrs. Thing']

def print_params(*params): -> uzimamo sve parametre i printamo
print params
```

Pokretanje nesigurnog dijela kod => try/except

```
try:
        x = input('Enter the first number: ')
        y = input('Enter the second number: ')
        print x/y
except ZeroDivisionError:
        print "The second number can't be zero!"

except TypeError:
        print "That wasn't a number, was it?"

try:
        x = input('Enter the first number: ')
        y = input('Enter the second number: ')
        print x/y
except (ZeroDivisionError, TypeError), e:
        print e

while True:
try:
        x = input('Enter the first number: ')
        y = input('Enter the second number: ')
        value = x/y
        print 'x/y is', value
except:
        print 'Invalid input. Please try again.'
else:
        break

try:
        1/0
except NameError:
        print "Unknown variable"
else:
        print "That went well!"
finally:
        print "Cleaning up."
```

Datoteke

otvaranje datoteke => open

open(name[, mode[, buffering]]) -> f = open(r'C:\text\somefile.txt')

modovi za otvaranje

r read

w write

a append

b binary

\+ read/write

Čitanje, pisanje, zatvaranje datoteke

```
f = open('file.txt', 'w')

f.write('Pisem u datoteku')

f.read()

f.close() -> zatvaranje
```

OOP

```
class <name>(superclass,...):                    # Assign to name
data = value                                     # Shared class data
def method(self,...):                            # Methods
self.member = value                              # Per-instance data

class Super:
        def method(self):
        print 'in Super.method'                  # Default behavior
        def delegate(self):
        self.action( )                           # Expected to be defined

class Inheritor(Super):                          # Inherit method verbatim
        pass

class Replacer(Super):                           # Replace method completely
        def method(self):
                print 'in Replacer.method'

class Extender(Super):                           # Extend method behavior
        def method(self):
                print 'starting Extender.method'
                Super.method(self)
                print 'ending Extender.method'

class Provider(Super):                           # Fill in a required method
        def action(self):
                print 'in Provider.action'
class Person:
        def setName(self, name):
        self.name = name

        def getName(self):
        return self.name

        def greet(self):
        print "Hello, world! I'm %s." % self.name

foo = Person()
bar = Person()
foo.setName('Luke Skywalker')
bar.setName('Anakin Skywalker')
foo.greet()
Hello, world! I'm Luke Skywalker.
bar.greet()
Hello, world! I'm Anakin Skywalker.
```

Ako želimo da se dio klase ne vidi stavimo __inaccessible / __secretive kod imena funkcije ili __accessible i sada se ta funkcija može koristiti samo unutar te klase

```
class Secretive:
def __inaccessible(self):
print "Bet you can't see me..."
def accessible(self):
print "The secret message is:"
self.__inaccessible()
```

issubclass -> želimo li saznati da li je neka klasa podklasa ili glavna

isssubclass(prva, druga)

isinstance -> da li je neki atribut instanca te klase

isinstance(atribut, klasa)

__class__ -> atribut, želimo li saznati u kojoj se klasi nalazi neki atribut

s.__class__

pass -> želimo li da neka klasa spoji više drugih

```
class kalkulator:
        def izracun(self, vrijednost):
                self.value = eval(vrijednost)
class printaj:
        def govor(self):
                print 'Bok moja vrijednost je:' + self.value
class obje(kalkulator, printaj):
        pass
```

hasattr -> želimo li saznati da li uopće postoji neka metoda

hasattr(metoda, argumenti)

__init__ -> konstruktor, posebna metoda klase koju ne treba zvati nego nakon pokretanja klase ona izvršava sve naredbe navedene u svom bloku

-> odmah nakon poziva klase poziva se init kako bi se napravilo ono šta je pod tom funkcijom

```
class klasa:
def __init__(self):
                print "Usao sam"
```

var = klasa()

'Usao sam'

__str__ -> definira kako će se objekt ponašati ako ga tretiramo kao string

def __str__(self):

__len__(self) -> vraća koliko je elemenata u kolekciji

__getitem__(self, key): -> vraća vrijednost koja odgovara ključu

__setitem__(self, key, value): -> sprema value s obzirom na key kako bi se kasnije mogla uzeti sa __getitem__

__delitem__(self, key): -> briše element asociran sa key

raise -> ukoliko želimo ispisati grešku možemo pomoću raise

raise greška

property -> vraća vrijednosti jedne ili više fumkcija unutar klase

size = property(getsize, setsize)

__getattribute__(self, name): -> Automatically called when the attribute name is accessed
__getattr__(self, name): -> Automatically called when the attribute name is accessed and the object has no such attribute.
__setattr__(self, name, value): -> Automatically called when an attempt is made to bind the attribute name to value.
__delattr__(self, name): -> Automatically called when an attempt is made to delete the attribute name

```
class Rectangle:
        def __init__(self):
                self.width = 0
                self.height = 0
        def __setattr__(self, name, value):
                if name == 'size':
                self.width, self.height = value
                else:
                self.__dict__[name] = value
        def __getattr__(self, name):
                if name == 'size':
                return self.width, self.height
                else:
                raise AttributeError
```

__iter__ -> vraća iterator, koji poziva objekt sa metodom zvanom next

class Fibs:

```
        def __init__(self):
                self.a = 0
                self.b = 1
        def next(self):
                self.a, self.b = self.b, self.a+self.b
                return self.a
        def __iter__(self):
                return self
```

```
it = Fibs()

it.next()
```

_ _init_ _	Constructor Object creation: X= Class( )
_ _del_ _	Destructor Object reclamation
_ _add_ _	Operator+ XY, X+= Y +
_ _or_ _	Operator | (bitwise OR) X | Y, X|= Y
_ _repr_ _,_ _str_ _	Printing, conversions print X, repr(X), str(X)
_ _call_ _	Function calls X ( )
_ _getattr_ _	Qualification X.undefined
_ _setattr_ _	Attribute assignment X.any = value
_ _getitem_ _	Indexing X[key], for loops and other iterations if no _ _iter_ _
_ _setitem_ _	Index assignment X[key] = value
_ _len_ _	Length len(X), truth tests
_ _cmp_ _	Comparison X == Y, X < Y
_ _lt_ _	Specific comparison X < Y (or else _ _cmp_ _)
_ _eq_ _	Specific comparison X== Y (or else _ _cmp_ _)
_ _radd_ _	Right-side operator + Noninstance + X
_ _iadd_ _	In-place (augmented) addition X+= Y (or else _ _add_ _)
_ _iter_ _	Iteration contexts for loops, in tests, list comprehensions, map, others

objektima unutar klase se pristupa sa -> .

```
class klasa():

        a = 2

b = klasa()

b.a -> 2
```