PROGRAMIRANJE U INTEGRISANIM TEHNOLOGIJAMA

Oznaka predmeta: FPR

Predavanje broj: 2

Nastavna jedinica: PYTHON,

Nastavne teme:

Liste. Metode liste. N-torka. Funkcije za n-torke. Rečnici. Funkcije za rečnike. Metode rečnika. Funkcije. Argumenti funkcije (Required arguments, Keyword arguments, Default arguments, Variable-length arguments). Lambda funkcije. Opseg promenljivih Moduli. Paketi. Datum i vreme (struct_time, time, localtime, asctime, clock, ctime, gmtime, sleep, calendar, altzone, strftime, strptime, tzset). Atributi time-a. Funkcije modula calendar (firstdayweek, calendar, month, isleap, leapdays, monthrange, prcal, prmonth, setfirstweekday, timegm, weekday). Input, eval. Fajlovi (open, modovi, buffering, close, write, read, seek). Metode modula os (rename, remove, mkdir, rmdir, getcwd, chdir). Metode objekta file.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

Steven Lott: "Functional Python Programming", Packt Publishing, 2015.

Python: liste

- Osnovna struktura podataka u Python-u je sekvenca. Lista je tip sekvence.
- Lista se predstavlja članovima koji se navode kao CSV unutar srednjih zagrada.

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"];
```

- Pristupanje vrednostima u listi je kao što sledi:

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
```

```
list1[0]: physics
list2[1:5]: [2, 3, 4, 5]
```

- Ažuriranje liste:

```
list = ['physics', 'chemistry', 1997, 2000];
print ("Value available at index 2 : ", list[2]);
list[2] = 2001;
print ("New value available at index 2 : ", list[2]);
```

```
Value available at index 2 : 1997
New value available at index 2 : 2001
```

Python: liste

- Brisanje elemenata liste:

```
list1 = ['physics', 'chemistry', 1997, 2000];
print (list1); del list1[2];
print ("After deleting value at index 2 : ")
print (list1);
```

```
['physics', 'chemistry', 1997, 2000]
After deleting value at index 2 :
['physics', 'chemistry', 2000]
```

- Osnovne operacije sa listom:

Python izraz	Rezultat	Opis
`len([1, 2, 3])`	`3`	Broj elemenata liste
`[1, 2, 3] + [4, 5, 6]`	`[1, 2, 3, 4, 5, 6]`	Konkatenacija
`['Hi!'] * 4`	`['Hi!', 'Hi!', 'Hi!', 'Hi!']`	Ponavljanje
`3 in [1, 2, 3]`	`True`	Pripadnost
`for x in [1, 2, 3]:` `print (x, end=' ')`	`1 2 3`	Iteracija

Python: liste

- Indeksiranje i odsecanje liste:

```
L = ['spam', 'Spam', 'SPAM!'];
```

Python izraz	Rezultat	Opis
`L[2]`	`'SPAM!'`	Ofset ide od nule
`L[-2]`	`'Spam'`	Negativan indeks se broji od kraja liste
`L[1:]`	`['Spam', 'SPAM!']`	Odsečak od indeksa 1 do kraja liste

Ugrađene funkcije za liste:

- koristi se (list1 > list2) - (list1 < list2) umesto `cmp(list1, list2)`, (videti p02)
  - poredi elemente dve liste
  - ako su elementi istog tipa vrši se poređenje i vraća se rezultat
  - ako su elementi različitih tipova
    - proverava se da li su brojevi.
      - ako jesu, svode se na isti tip, porede i vraća se rezultat.
      - ako je jedan element broj drugi element je veći (brojevi su "najmanji" tip)
      - ako nisu, elementi se sortiraju alfabetski.
  - ako se dosegne kraj jedne liste, a druge ne, duža lista je veća
  - ako se iscrpe obe liste i imaju iste podatke vraća se 0.

Python: liste

```
list1, list2 = [123, 'xyz'], [456, 'abc']
print ((list1 > list2) - (list1 < list2));    # -1
print ((list2 > list1) - (list2 < list1));    # 1
list3 = list2 + [786];
print ((list2 > list3) - (list2 < list3))          # -1
```

- len(list)

```
list1, list2 = [123, 'xyz', 'zara'], [456, 'abc']
print ("First list length : ", len(list1));
print ("Second list length : ", len(list2));
```

```
First list length :  3
Second lsit length :  2
```

- max(list)

```
list1, list2 = [123, 'xyz', 'zara', 'abc'], [456, 700, 200]
print ("Max value element : ", max(list1));
print ("Max value element : ", max(list2));
```

```
Max value element : zara
Max value element : 700
```

- min(list), za prethodno date liste

```
print ("Min value element : ", min(list1)); # Min value element : 123
print ("Min value element : ", min(list2)); # Min value element : 200
```

Python: liste, metode liste

- list(sequence)
  - konvertuje sekvencu u listu

```
aTuple = (123, 'xyz', 'zara', 'abc');
aList = list(aTuple)
print ("List elements : ", aList)
```

```
List elements : [123, 'xyz', 'zara', 'abc']
```

Metode liste:

- list.append(obj)

```
aList = [123, 'xyz', 'zara', 'abc'];
aList.append( 2009 );
print ("Updated List : ", aList);
```

```
Updated List : [123, 'xyz', 'zara', 'abc', 2009]
```

- list.count(obj)

```
aList = [123, 'xyz', 'zara', 'abc', 123];
print ("Count for 123 : ", aList.count(123));
print ("Count for zara : ", aList.count('zara'));
```

```
Count for 123 : 2
Count for zara : 1
```

- list.extend(sequence)

```
aList = [123, 'abc', 123];        bList = [2009, 'manni'];
aList.extend(bList);     print ("Extended List : ", aList) ;
```

```
Extended List : [123, 'abc', 123, 2009, 'manni']
```

Python: metode liste

- list.index(obj)

```
aList = [123, 'xyz', 'zara', 'abc'];
print ("Index for xyz : ", aList.index( 'xyz' )) ;
print ("Index for zara : ", aList.index( 'zara' )) ;
```

```
Index for xyz : 1
Index for zara : 2
```

- list.insert(index, obj)

```
aList = [123, 'xyz', 'zara', 'abc']
aList.insert( 3, 2009)
print ("Final List : ", aList)
```

```
Final List : [123, 'xyz', 'zara', 2009, 'abc']
```

- list.pop(obj=list[-1])

```
aList = [123, 'xyz', 'zara', 'abc'];
print ("A List : ", aList.pop());
print ("B List : ", aList.pop(2));
```

```
A List : abc
B List : zara
```

- list.remove(obj)

```
aList = [123, 'xyz', 'zara', 'abc', 'xyz'];
aList.remove('xyz'); print ("List : ", aList);
aList.remove('abc'); print ("List : ", aList);
```

```
List : [123, 'zara', 'abc', 'xyz']
List : [123, 'zara', 'xyz']
```

Python: metode liste, tuple (n-torka)

- list.reverse()

```
aList = [123, 'xyz', 'zara', 'abc', 'xyz'];
aList.reverse();
print ("List : ", aList);
                List : ['xyz', 'abc', 'zara', 'xyz', 123]
```

- list.sort([func])

```
aList = [123, 'xyz', 'zara', 'abc', 'xyz'];
aList.sort();
print ("List : ", aList);
                List : [123, 'abc', 'xyz', 'xyz', 'zara']
```

Tuple (n-torka):

- N-torka predstavlja sekvencu neizmenljivih objekata.
- Elementi n-torke su CSV vrednosti navedene u malim zagradama.

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
tup4 = ();          # prazna n-torka
tup5 = (123,);      # n-torka sa jednim elementom
```

- Elementi n-torke su indeksirani od 0.

Python: tuple (n-torka)

- Pristupanje vrednostima n-torke.
- Ne mogu se menjati vrednosti članova n-torke.

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# ne moze tup1[0] = 100;
tup3 = tup1 + tup2; print (tup3); # konkatenacija MOZE
                                  (12, 34.56, 'abc', 'xyz')
```

- Nije moguće uklanjati pojedine elemente n-torke:

```
tup = ('physics', 'chemistry', 1997, 2000);
print (tup);
del tup;
print ("After deleting tup : ")
print (tup);
        ('physics', 'chemistry', 1997, 2000)
        After deleting tup :
        Traceback (most recent call last):
                File "test.py", line 9, in <module> print tup;
        NameError: name 'tup' is not defined
```

- Osnovne operacije sa n-torkama obuhvataju: konkatenaciju, vraćanje dužine n-torke, ponavljanje n-torke, provera pripadnosti elementa, iteracija po elementima n-torke.

Python: tuple (n-torka)

Python izraz	Rezultat	Opis
`len((1, 2, 3))`	`3`	Broj elemenata n-torke
`(1, 2, 3) + (4, 5, 6)`	`(1, 2, 3, 4, 5, 6)`	Konkatenacija
`('Hi!') * 4`	`('Hi!', 'Hi!', 'Hi!', 'Hi!')`	Ponavljanje
`3 in (1, 2, 3)`	`True`	Pripadnost
`for x in (1, 2, 3):` `print (x,end=' ')`	`1 2 3`	Iteracija

- Indeksiranje i odsecanje n-torke:

```
L = ('spam', 'Spam', 'SPAM!');
```

Python izraz	Rezultat	Opis
`L[2]`	`'SPAM!'`	Ofset ide od nule
`L[-2]`	`'Spam'`	Negativan indeks se broji od kraja n-torke
`L[1:]`	`('Spam', 'SPAM!')`	Odsečak od indeksa 1 do kraja n-torke

Python: funkcije za n-torke

Funkcije za n-torke su kao što sledi:

- koristi se (tuple1>tuple2)-(tuple1<tuple2) umesto cmp(tuple1, tuple2)
  - poredi elemente dve n-torke
  - ako su elementi istog tipa vrši se poređenje i vraća se rezultat
  - ako su elementi različitih tipova
    - proverava se da li su brojevi.
      - ako jesu, svode se na isti tip, porede i vraća se rezultat.
      - ako je jedan element broj drugi element je veći (brojevi su "najmanji" tip)
      - ako nisu, elementi se sortiraju alfabetski.
  - ako se dosegne kraj jedne n-torke, a druge ne, duža n-torka je veća
  - ako se iscrpe obe n-torke i imaju iste podatke vraća se 0.

```
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')
print ((tuple1>tuple2)-(tuple1<tuple2) ); # -1
print ((tuple2>tuple1)-(tuple2<tuple1) ); # 1
tuple3 = tuple2 + (786,);
print ((tuple2>tuple3)-(tuple2<tuple3) )  # -1
```

- len(tuple)

```
tuple1, tuple2 = (123, 'xyz', 'zara'), (456, 'abc')
print (len(tuple1)); # 3
print (len(tuple2)); # 2
```

Python: : funkcije za n-torke

- max(tuple)

```
tuple1, tuple2 = (123, 'xyz', 'zara', 'abc'), (456, 700, 200)
print ("Max value element : ", max(tuple1));
print ("Max value element : ", max(tuple2));
```

```
Max value element : zara
Max value element : 700
```

- min(tuple)

```
tuple1, tuple2 = (123, 'xyz', 'zara', 'abc'), (456, 700, 200)
print ("min value element : ", min(tuple1));
print ("min value element : ", min(tuple2));
```

```
min value element : 123
min value element : 200
```

- tuple(sequence)
  - pretvara sekvencu u n-torku

```
aList = [123, 'xyz', 'zara', 'abc'];
aTuple = tuple(aList)
print ("Tuple elements : ", aTuple)
```

```
Tuple elements : (123, 'xyz', 'zara', 'abc')
```

Python: dictionary (rečnik)

- Rečnik je sekvenca koja se zapisuje u velike zagrade u kojima se navode parovi ključ-vrednost.
- Par ključ vrednost je razdvojen sa : dok su parovi odvojeni CSV formatom.

```
dict = {'Name': 'Jane', 'Age': 30, 'Class': 'First'};
print ("dict['Name']: ", dict['Name']);
print ("dict['Age']: ", dict['Age']);
print ("dict['Alice']: ", dict['Alice']);
```

```
dict['Name']: Jane
dict['Age']: 30
dict['Alice']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
        print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

- Ažuriranje elemenata rečnika:

```
dict = {'Na': 'Joe', 'Age': 37};
del dict['Na']; # uklanja ulaz 'Na'
dict.clear();   # uklanja sve ulaze
del dict ;      # brise recnik
print ("dict['Age']: ", dict['Age']);
print ("dict['School']: ", dict['School']);
```

```
dict['Age']:
Traceback (most recent call last):
  File "main.py", line 9, in
    print "dict['Age']: ",
    dict['Age'];
TypeError: 'type' object has no
attribute '__getitem__'
```

Python: dictionary (rečnik), funkcije

- Vrednosti u rečniku nemaju ograničenja, ali ključevi imaju:
  - ključ mora biti jedinstven inače se računa samo poslednji

```
dict = {'Name': 'Jane', 'Age': 30, 'Name': 'Manni'};
print ("dict['Name']: ", dict['Name']);
```

```
dict['Name']: Manni
```

  - ključ mora biti neizmenljiv

```
dict = {['Name']: 'Jane', 'Age': 30};
print ("dict['Name']: ", dict['Name']);
```

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: list objects are unhashable
```

Ugrađene funkcije koje barataju rečnikom:

- poređenje koje je zadržano je dict==dict2

```
dict1 = {'Name': 'Zara', 'Age': 7};
dict2 = {'Name': 'Mahnaz', 'Age': 27};
dict3 = {'Name': 'Zara', 'Age': 7};
print (dict1==dict2)     # False
print (dict1==dict3))    # True
```

Python: dictionary (rečnik), funkcije i metode

- len(dict)

```
dict = {'Name': 'Zara', 'Age': 7};
print ("Length : %d" % len (dict))       # Length : 2 para n-v
```

- str(dict)

```
dict = {'Name': 'Zara', 'Age': 7};
print ("Equivalent String : %s" % str (dict))
                Equivalent String : {'Age': 7, 'Name': 'Zara'}
```

- type(variable)

```
dict = {'Name': 'Zara', 'Age': 7};
print ("Variable Type : %s" % type (dict))
                Variable Type : <class 'dict'>
```

Metode rečnika:

- dict.clear()

```
dict = {'Name': 'Zara', 'Age': 7};
print ("Start Len : %d" % len(dict)) # Start Len : 2
dict.clear()
print ("End Len : %d" % len(dict))   # End Len : 0
```

- dict.copy()

```
dict1 = {'Name': 'Zara', 'Age': 7}; dict2 = dict1.copy()
print (dict2);  # {'Age': 7, 'Name': 'Zara'}
```

Python: dictionary (rečnik), metode

- dict.fromkeys()

```
seq = ('name', 'age', 'sex')
dict = dict.fromkeys(seq)
print ("New Dictionary : %s" % str(dict))
dict = dict.fromkeys(seq, 10)
print ("New Dictionary : %s" % str(dict))
```

```
New Dictionary : {'age': None, 'name': None, 'sex': None}
New Dictionary : {'age': 10, 'name': 10, 'sex': 10}
```

- dict.get(key, default=None)

```
dict = {'Name': 'Zara', 'Age': 27}
print ("Value : %s" % dict.get('Age'))            # Value : 27
print ("Value : %s" % dict.get('Sex', "Never"))   # Value : Never
```

- dict.has_key(key)

```
dict = {'Name': 'Zara', 'Age': 27}
print ("Value : %s" % dict.has_key('Age'))        # Value : True
print ("Value : %s" % dict.has_key('Sex'))        # Value : False
```

- dict.items()

```
dict = {'Name': 'Zara', 'Age': 27}
print ("Value : %s" % dict.items())
```

```
Value : [('Age', 7), ('Name', 'Zara')]
```

Python: dictionary (rečnik), metode

- dict.keys()

```
dict = {'Name': 'Zara', 'Age': 27}
print ("Value : %s" % dict.keys())
```

```
Value : ['Name', 'Age']
```

- dict.setdefault(key, default=None) #postavi ako nema

```
dict = {'Name': 'Zara', 'Age': 27}
print ("Value : %s" % dict.setdefault('Age', None))
print ("Value : %s" % dict.setdefault('Sex', 'f'))
```

```
Value : 27
Value : 'f'
```

- dict.update(dict2)

```
dict = {'Name': 'Zara', 'Age': 27}
dict2 = {'Sex': 'female' }
dict.update(dict2)
print ("Value : %s" % dict)
```

```
Value : {'Name': 'Zara', 'Age': 27, 'Sex': 'female'}
```

- dict.values()

```
dict = {'Name': 'Zara', 'Age': 27}
print ("Value : %s" % dict.values())
```

```
Value : ['Zara', 27]
```

Python: funkcije

- Slede pravila za definisanje funkcije u Python-u:
  - Blok funkcije počinje ključnom rečju def iza koje sledi ime funkcije i male zagrade (( )).
  - Svi ulazni parametri ili argumenti trebalo bi da budu navedeni unutar pomenutih zagrada. Moguće je i definisati parametre unutar ovih zagrada.
  - Prva naredba funkcije može biti opcionalna naredba koja predstavlja dokumentacioni string funkcije (docstring).
  - Blok koda unutar svake funkcije počinje iza dvotačke (:). Ovaj blok je uvučen.
  - Iz funcije se izlazi naredbom return [expression] gde se opciono vraća izraz iz funkcije onome koji ju je pozvao.

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

primer:

```
def printme( str ):
   "This prints a passed string into this function"
   print (str)
   return
```

Python: funkcije

- Pozivanje funkcije je ili iz Python prompta ili iz drugih funkcija.

```
# definicija funkcije
def printme( str ):
   "This prints a passed string into this function"
   print (str);
   return;
# pozivi funkcije
printme("I'm first call to user defined function!");
printme("Again second call to the same function");
```

```
I'm first call to user defined function!
Again second call to the same function
```

- U Python-u se svi argumenti prenose po "referenci" (paziti šta znači dodela):

```
# definicija funkcije
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist.append([1,2,3,4]);
   print ("Values inside the function: ", mylist)
   return
# poziv funkcije
mylist = [10,20,30];
changeme( mylist );
print ("Values outside the function: ", mylist)
```

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

Python: funkcije

- U sledećem primeru argument je prosleđen preko reference a onda je referenca prepisana unutar pozvane funkcije:

```
# definicija funkcije
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # dodela za novu referencu
   print ("Values inside the function: ", mylist)
   return
# poziv funkcije
mylist = [10,20,30];
changeme( mylist );
print ("Values outside the function: ", mylist)
```

```
Values inside the function: [1, 2, 3, 4]
Values outside the function: [10, 20, 30]
```

- Argumenti funkcije:
  Funkcija se može pozvati sa sledećim tipovima formalnih argumenata:
  - Required arguments
  - Keyword arguments
  - Default arguments
  - Variable-length arguments

Python: funkcije, *required arguments*

- Zahtevani argumenti (*Required arguments*) su argumenti prosleđeni funkciji u pravilnom redosledu pri čemu broj argumenata koji se prosleđuju funkciji odgovara definiciji funkcije:

```
# definicija funkcije
def printme( str ):
   "This prints a passed string into this function"
   print (str);
   return;
# poziv funkcije
printme("OK");
printme();
```

```
OK
Traceback (most recent call last):
  File "main.py", line 11, in
    printme();
TypeError: printme() takes exactly 1 argument (0 given)
```

- Ključne reči - argumenti, odnose se na identifikatore formalnih argumenta u funkciji i korišćenja tih identifikatora za kopiranje stvarnih argumenta.
  - Ovo je omogućeno u Python-u tako da se može promeniti redosled kojim su navedeni identifikatori formalnih argumenta sa dodelama stvarnih argumenata.

Python: funkcije, *keyword arguments*

```
# Primer 1
# definicija funkcije
def printme( str ):
   "This prints a passed string into this function"
   print (str);
   return;
# poziv funkcije
printme( str = "My string");
```

```
My string
```

```
# Primer 2
# definicija funkcije
def printinfo( name, age ):
   "This prints a passed info into this function"
   print ("Name: ", name);
   print ("Age ", age);
   return;
# poziv funkcije
printinfo( age=50, name="miki" );
printinfo( age=100, name="" );
```

```
Name: miki
Age 50
Name:
Age 100
```

Python: funkcije, *Default arguments, Variable-length arguments*

- Podrazumevani argumenti (*Default arguments*) se navode u definiciji funkcije tako da je moguće da se ne navedu odgovarajući stvarni argumenti u pozivu funkcije.

```
def printinfo( name, age = 35 ):
   "This prints a passed info into this function"
   print ("Name: ", name);
   print ("Age ", age);
   return;
printinfo( age=50, name="miki" );
printinfo( name="miki" );
```

```
Name: miki
Age 50
Name: miki
Age 35
```

- Varijabilni broj argumenata (*Variable-length arguments*) omogućuje da funkciji prosledimo proizvoljan broj argumenta:

```
def functionname([formal_args,] *var_args_tuple ):
   "function_docstring"
   function_suite
   return [expression]
```

Zvezdica (*) ispred identifikatora varijable omogućuje prihvatanje svih vrednosti dodatnih argumenta (ova n-torka ostaje prazna ako nema dodatnih argumenata).

Python: funkcije, *Variable-length arguments, lambda*

```
def printinfo( arg1, *vartuple ):
   "This prints a variable passed arguments"
   print ("Output is: ")
   print (arg1)
   for v in vartuple:
      print (v,end=' ')
   return;
printinfo( 10 );
printinfo( 70, 60, 50 );
```

```
Output is:
10
Output is:
70
60 50
```

Anonimne (lambda) funkcije

- Naziv su dobile jer se ne deklarišu standardno sa def, već se koristi ključna reč lambda.
- Lambda prima proizvoljno argumenata a vraća jedan izraz. Ne sadrži komande ili višestruke izraze.
- Lambda funkcije koriste sopstveni prostor imena.

Python: lambda

- Sintaksa lambda funkcije:

```
lambda [arg1 [,arg2,.....argn]]:expression
```

```
# primer
sum = lambda arg1, arg2: arg1 + arg2;
print ("Value of total : ", sum( 10, 20 ))
print ("Value of total : ", sum( 20, 20 ))
```

```
Value of total : 30
Value of total : 40
```

- Naredba return:

```
def sum( arg1, arg2 ):
   total = arg1 + arg2
   print ("Inside the function : ", total)
   return total;
total = sum( 10, 20 );
print ("Outside the function : ", total)
```

```
Inside the function : 30
Outside the function : 30
```

- Opseg važenja promenljivih:
  - globalne promenljive (definisane su van funkcije)
  - lokalne promenljive (definisane su unutar funkcije)

Python, opseg promenljivih, moduli

- Primer globalne i lokalne varijable:

```
total = 0;                          # globalna.
def sum( arg1, arg2 ):
   total = arg1 + arg2;             # lokalna.
   print ("Inside the function local total : ", total)
   return total;
sum( 10, 20 );
print ("Outside the function global total : ", total)
```

```
Inside the function local total : 30
Outside the function global total : 0
```

Moduli u Python-u:

- Koriste se za logičko organizovanje koda. Modul predstavlja fajl sa Python kodom.
- Modul može definisati: funkcije, klase i promenljive.
- Neka je funkcija `print_func` smeštena u fajl `support.py` i neka je sadržaj kao što sledi:

```
def print_func( par ):
   print ("Hello : ", par)
   return
```

- Bilo koji Python-ov fajl može biti tretiran kao modul korišćenjem naredbe import:

```
import module1[, module2[,... moduleN]
```

Python, opseg promenljivih, moduli

- Primer importa modula i korišćenja:

```
import support
support.print_func("Zara")
```

```
Hello : Zara
```

- Modul se učitava samo jednom.
- Naredba `from` omogućuje uvoz specifičnih atributa iz modula u tekući prostor imena (namespace).

```
from modulname import name1[, name2[, ... nameN]]
```

primer: uvoz funkcije fibonnaci iz modula fib

```
from fib import fibonacci
```

- Naredba `from` ne uključuje modul fib u tekući prostor imena već samo član fibonnaci iz modula fib u tabelu globalnih simbola.
- Moguće je importovati sva imena iz modula u tekući prostor imena korišćenjem naredbe `from` kao što sledi:

```
from modname import *
```

Python, opseg promenljivih, moduli

- Prilikom importa modula, Python interpreter traži dati modul kao što sledi:
  - prvo se proverava da li se modul nalazi u tekućem folderu
  - ako se modul ne pronađe, Python onda traži modul u svakom folderu navedenom u sistemskoj promenljivoj PYTHONPATH.
  - ako i dalje nije našao modul onda Python proverava da li je modul u podrazumevanom folderu (npr. na linux-u /usr/local/lib/python/).
- Staza za traženje modula je navedena u sistemskom modulu sys kao sys.path promenljiva.
  - Promenljiva sys.path sadrži: tekući folder, PYTHONPATH i instalaciono zavisne staze.
- Promenljiva PYTHONPATH (varijable okruženja):
  - sastoji se od liste foldera.
  - sintaksa PYTHONPATH-a je ista kao i promenljive PATH.
    - primer tipične PYTHONPATH varijable u Windows-u:

```
set PYTHONPATH=c:\python30\lib;
```

    - primer tipične PYTHONPATH u linux-u:

```
set PYTHONPATH=/usr/local/lib/python
```

Python, opseg promenljivih

- Naredba Python-a može pristupiti promenljivoj u lokalnom prostoru imena i u globalnom prostoru imena.
- Ako lokalna varijabla ima isto ime kao globalna varijabla onda lokalna varijabla skriva globalnu varijablu.
- Svaka funkcija ima svoj lokalni prostor imena.
- Metode klase takođe imaju pravila vezana za opseg kao i obične funkcije.
- Kod Python-a bilo koja varijabla koja je dodeljena u okviru funkcije predstavlja lokalnu varijablu.
- Međutim, da bi se dodelila vrednost globalnoj varijabli unutar funkcije mora se upotrebiti naredba global.
  - npr. global VarName označava da je VarName globalna varijabla u Python-u.

```
Money = 2000
def AddMoney():
   # global Money #ako se ova linija ukljuci sve ce raditi
   Money = Money + 1
print (Money)
AddMoney()
print (Money)
```

ako bi ostao komentar # global Money, Python bi dojavio grešku da lokalna promenljiva nije definisana.

Python, opseg promenljivih, moduli

- Funkcija locals() pozvana unutar funkcije vratiće sva imena kojima se pristupa lokalno iz te funkcije.
- Funkcija globals() pozvana unutar funkcije vratiće sva imena kojima se može pristupiti globalno iz te funkcije.
- Povratni tip obe navedene funkcije je rečnik (imena mogu biti ekstrahovana korišćenjem funkcije keys()).
- Ugrađena funkcija dir(modul) vraća sortiranu listu stringova koji predstavljaju imena definisana u modulu.
- Lista sadrži imena svih modula, varijabli i funkcija koji su definisani u datom modulu.

```
import math
content = dir(math)
print (content);
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf',
'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi',
'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

Navedene su i specijalne string varijable __name__ (odnosi se na naziv modula)...

Python, moduli, paketi

- Funkcija modula importlib.reload(ime_modula) ponovo importuje dati modul.
- Paketi u Python-u predstavljaju hijerarhisku strukturu fajlova u folderu koji definišu okruženje Python aplikacije koje sadrži module i potpakete.
- Neka je fajl `Pots.py` dostupan u `Phone` folderu sa sadržajem kao što sledi:

```
def Pots():
    print ("I'm Pots Phone")
```

i neka postoje dva fajla koji kao i prethodni imaju i svoje istoimene funkcije:

`Phone/Isdn.py`	`sadrži funkciju Isdn()`
`Phone/G3.py`	`sadrži funkciju G3()`

Sada je potrebno kreirati još jedan fajl u `Phone` direktorijumu:

`Phone/__init__.py`

- Da bi se omogućilo da sve navedene funkcije budu dostupne pri importu paketa `Phone`, potrebno je eksplicitno staviti `from-import` naredbe u fajl `__init__.py` kao što sledi:

```
from Pots import Pots
from Isdn import Isdn
from G3 import G3
```

Python, opseg promenljivih, moduli

- Nakon dodavanja prethodnih linija u fajl `__init__.py`, dobiće se dostupnost svih navedenih funkcija pri importu paketa Phone.

```
import Phone
Phone.Pots()
Phone.Isdn()
Phone.G3()
```

```
I'm Pots Phone
I'm 3G Phone
I'm ISDN Phone
```

- U prethodnom primeru dato je uključenje po jedne funkcije iz svakog fajla, a na isti način se može uključiti više funkcija.
- Mogu se definisati i različite Python klase u navedenim fajlovima a onda kreirati posebni paketi koji ih uključuju.

Python: datum i vreme

- U Python-u funkcija *time.time()* vraća tekuće sistemsko vreme koje se računa od 1.1.1970. godine.

```
import time; # potrebno je ukljuciti ovaj modul
ticks = time.time()
print ("Number of ticks since 12:00am, January 1, 1970:", ticks)
```

```
Number of ticks since 12:00am, January 1, 1970: 1445942131.585914
```

- Vremenska n-torka

Indeks	Polje	Vrednost
0	4-digit year	Npr. 2008
1	Month	1 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 59
6	Day of Week	0 to 6 (0 is Monday)
7	Day of year	1 to 366 (Julian day)
8	Daylight savings	-1, 0, 1, npr. DST = -1

Python: datum i vreme, struct_time

- Prethodna 9-torka je ekvivalentna strukturi **struct_time:**

Indeks	Atributi	Vrednosti
0	tm_year	2008
1	tm_mon	1 to 12
2	tm_mday	1 to 31
3	tm_hour	0 to 23
4	tm_min	0 to 59
5	tm_sec	0 to 61 (60 or 61 are leap-seconds)
6	tm_wday	0 to 6 (0 is Monday)
7	tm_yday	1 to 366 (Julian day)
8	tm_isdst	-1, 0, 1, -1 means library determines DST

- Translacija iz vrednosti datih u sekudama proteklim od 1.1.1970. godine u vremensku 9-torku je kao što sledi:

```
import time; localtime = time.localtime(time.time())
print ("Local current time :", localtime)
```

```
Local current time : time.struct_time(tm_year=2015, tm_mon=10, tm_mday=27,
tm_hour=11, tm_min=41, tm_sec=21, tm_wday=1, tm_yday=300, tm_isdst=0)
```

Python: datum i vreme, localtime, calendar, altzone

- Dohvatanje formatiranog vremena:

```
import time
asctime = time.asctime( time.localtime(time.time()) )
print ("Local current time :", asctime )
```

```
Local current time : Tue Jan 13 10:17:09 2009
```

- Prikaz kalendara za dati mesec

```
import calendar
calm = calendar.month(2015, 7)
print ("Here is the calendar:"); print (calm);
```

```
Here is the calendar:
     July 2015
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

Funkcije i varijable modula time:

- time.altzone - ofset lokalne DST vremenske zone u sekundama zapadno od UTC

```
import time
print ("time.altzone %d " % time.altzone)
```

```
time.altzone -7200
```

Python: datum i vreme,

- time.asctime([tupletime])

```
import time
t = time.localtime()
print ("time.asctime(t): %s " % time.asctime(t))
```

```
time.asctime(t): Thu Jul 16 14:47:44 2015
```

- time.clock( )

```
import time
def procedure():
    time.sleep(2.5)
# measure process time
t0 = time.clock()
procedure()
print (time.clock() - t0, "seconds process time")
# measure wall time
t0 = time.time()
procedure()
print (time.time() - t0, "seconds wall time")
```

```
2.5001568558090175 seconds process time
2.500142812728882 seconds wall time
```

- time.ctime([secs])

```
import time
print ("time.ctime() : %s" % time.ctime())
```

```
time.ctime() : Tue Feb 17 10:00:18 2009
```

Python, datum i vreme

- time.gmtime([secs])

```
import time
print ("time.gmtime() : %s" % time.gmtime())
```

```
time.gmtime() : time.struct_time(tm_year=2015, tm_mon=10, tm_mday=27,
tm_hour=11, tm_min=9, tm_sec=42, tm_wday=1, tm_yday=300, tm_isdst=0)
```

- time.localtime([secs])

```
import time
print ("time.localtime() : %s" % time.localtime())
```

```
time.localtime() : time.struct_time(tm_year=2015, tm_mon=10, tm_mday=27,
tm_hour=12, tm_min=12, tm_sec=29, tm_wday=1, tm_yday=300, tm_isdst=0)
```

- time.mktime(tupletime)

```
import time
t = (2009, 2, 17, 17, 3, 38, 1, 48, 0)
secs = time.mktime( t )
print ("time.mktime(t) : %f" % secs)
print ("asctime(localtime(secs)): %s" % time.asctime(time.localtime(secs)))
```

```
time.mktime(t) : 1234915418.000000
asctime(localtime(secs)): Tue Feb 17 17:03:38 2009
```

- time.sleep(secs)

```
import time
print ("S: %s" % time.ctime()); time.sleep(5)
print ("E: %s" % time.ctime())
```

```
S: Thu Jul 16 15:31:10 2015
E: Thu Jul 16 15:31:15 2015
```

Python, datum i vreme

- time.strftime(fmt[,tupletime])
  - formatira ispis datuma i vremena

```
import time
t = (2009, 2, 17, 17, 3, 38, 1, 48, 0)
t = time.mktime(t)
print (time.strftime("%b %d %Y %H:%M:%S", time.gmtime(t)))
                              Feb 18 2009 00:03:38
```

- time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')

```
import time
struct_time = time.strptime("30 Nov 00", "%d %b %y")
print ("returned tuple: %s " % struct_time)
        returned tuple: (2000, 11, 30, 0, 0, 0, 3, 335, -1)
```

- time.time( )

```
import time
print ("time.time(): %f " % time.time())
print (time.localtime( time.time() ))
print (time.asctime( time.localtime(time.time()) ) )
```

```
time.time(): 1445946474.565745
time.struct_time(tm_year=2015, tm_mon=10, tm_mday=27, tm_hour=12, tm_min=47,
tm_sec=54, tm_wday=1, tm_yday=300, tm_isdst=0)
Tue Oct 27 12:47:54 2015
```

Python: popis formata za datum i vreme

- %a - abbreviated weekday name
- %A - full weekday name
- %b - abbreviated month name
- %B - full month name
- %c - preferred date and time representation
- %C - century number (the year divided by 100, range 00 to 99)
- %d - day of the month (01 to 31)
- %D - same as %m/%d/%y
- %e - day of the month (1 to 31)
- %g - like %G, but without the century
- %G - 4-digit year corresponding to the ISO week number (see %V).
- %h - same as %b
- %H - hour, using a 24-hour clock (00 to 23)
- %I - hour, using a 12-hour clock (01 to 12)
- %j - day of the year (001 to 366)
- %m - month (01 to 12)
- %M - minute
- %n - newline character
- %p - either am or pm according to the given time value
- %r - time in a.m. and p.m. notation
- %R - time in 24 hour notation

Python: popis formata za datum i vreme

- %S - second
- %t - tab character
- %T - current time, equal to %H:%M:%S
- %u - weekday as a number (1 to 7), Monday=1.
  Warning: In Sun Solaris Sunday=1
- %U - week number of the current year, starting with the first Sunday as the first day of the first week
- %V - The ISO 8601 week number of the current year (01 to 53), where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week
- %W - week number of the current year, starting with the first Monday as the first day of the first week
- %w - day of the week as a decimal, Sunday=0
- %x - preferred date representation without the time
- %X - preferred time representation without the date
- %y - year without a century (range 00 to 99)
- %Y - year including the century
- %Z or %z - time zone or name or abbreviation
- %% - a literal % character

Python, datum i vreme

Atributi time-a:

- `time.timezone` — atribut predstavlja ofset u sekundama lokalne vremenske zone (bez DST) u odnosu na UTC (>0 u Americi; <=0 u većini evropskih država, u Aziji i Africi).
- `time.tzname` — atribut predstavlja deo uređenog para koga čine lokalana vremenska zona i bez ili sa DST, respektivno.

Funkcije modula calendar:

- `calendar.calendar(year,w=2,l=1,c=6)`

  Vraća mulitlinijski string koji predstavlja formatirani kalendar navedene godine (year). Meseci su raspoređeni u 3 kolone odvojene sa c razmaka. Parametar w određuje širinu datuma u karakterima. Svaka linija je duga 21*w+18+2*c. Parametar l predstavlja broj linija za sedmicu.
- `calendar.firstweekday()`

  Vraća tekuću postavku startnog dana u sedmici. Podrazumevano je 0 što znači (Ponedeljak).
- `calendar.isleap(year)`

  Vraća odgovor da li je godina (year) prestupna (True – ako jeste, inače False).

Python, datum i vreme

- `calendar.leapdays(y1,y2)`

  Vraća ukupan broj prestupnih dana u opsegu dve navedene godine (y1, y2).

- `calendar.month(year,month,w=2,l=1)`

  Vraća višelinijski string koji predstavlja kalendar navedenog meseca u navedenoj godini. Sedmica je u jednoj liniji ispisa. Zaglavlje ima dve linije (naziv ima dve linije (mesec, dani u sedmici). Parametar w se odnosi na širinu svakog datuma u karakterima. Svaka linija ima dužinu 7*w+6. Parametar l se donosi na broj linija ispisa za svaku sedmicu.

- `calendar.monthcalendar(year,month)`

  Vraća listu lista celih brojeva. Podlista predstavlja sedmicu. Dani van meseca imaju vrednost 0 a pripadani dani imaju dodeljen redni broj tog dana u mesecu.

```
calendar.monthcalendar(2016,10)
[[0, 0, 0, 0, 0, 1, 2], [3, 4, 5, 6, 7, 8, 9], [10, 11, 12, 13, 14, 15, 16],[17,
18, 19, 20, 21, 22, 23], [24, 25, 26, 27, 28, 29, 30], [31, 0, 0, 0, 0, 0, 0]]
```

- `calendar.monthrange(year,month)`

  Vraća uređen par celih brojeva. Prvi broj pokazuje indeks prvog dana u mesecu u podlisti sedmice. Drugi broj pokazuje koliko navedeni mesec ima dana.

```
calendar.monthrange() → (5,31) indeks prvog dana u sedmici, broj dana u mesecu
```

- `calendar.prcal(year,w=2,l=1,c=6)`

  Kao print(calendar.calendar(year,w,l,c))

Python, datum i vreme

- `calendar.prmonth(year,month,w=2,l=1)`

  Kao print(calendar.month(year,month,w,l))

- `calendar.setfirstweekday(weekday)`

  Postavljanje koda prvog dana u sedmici. Kodovi su 0 (Monday) do 6 (Sunday).

- `calendar.timegm(tupletime)`

  Suprotno od time.gmtime. Prihvata vremensku 9-torku, a vraća broj sekundi koji je protekao od 1.1.1970. godine.

- `calendar.weekday(year,month,day)`

  Vraća kod dana u sedmici za dati datum. Kodovi su 0 (Monday) do 6 (Sunday). Meseci idu od 1 do 12.

Python, I/O

- Osnovna naredba za ispis na ekran je print:

```
print ("Hello, Python!");
print ("Python is really a great language,", "isn't it?");
```

```
Hello, Python!
Python is really a great language, isn't it?
```

- Python-ove ugrađene naredbe za prihvatanje sa standardnog ulaza:
  - input([prompt])
    - funkcija čita liniju sa standardnog ulaza i vraća je kao string bez oznake za novi red na kraju.

```
str = input("Enter your input: ");
print ("Received input is : ", str)
```

```
Enter your input: Hello Python
Received input is : Hello Python
```

    - za evaluaciju ulaza koristiti funkciju eval:

```
str = input("Unesi izraz: ");
lista = eval(str);
print (lista)
```

```
Unesi izraz: [x*5 for x in range(2,10,2)]
[10, 20, 30, 40]
```

Python, I/O

- Python obezbeđuje osnovne funkcije za manipulaciju fajlovima. Koristi se fajl objekat.
- Ugrađena funkcija open služi za otvaranje fajla pre njegove manipulacije.
  - Ova funkcija kreira fajl objekat a onda se dalje manipulacije fajlom sprovode pozivom odgovarajućuh metoda ovog objekta.

```
fo = open(file_name [, access_mode][, buffering])
```

  - file_name: predstavlja string koji sadrži ime fajla kome se želi pristupiti.
  - access_mode: određuje mod u kom će fajl biti otvoren (read, write, append, ...). Ovo je opcionalni parametar a njegova podrazumevana vrednost je read (r).
  - buffering:
    - ako je vrednost za baferovanje postavljena na 0 nema baferovanja fajla.
    - ako je vrednost baferovanja postavljena na 1, izvršava se line buffering pri pristupanju fajlu.
    - Ako je vrednost >1 onda ona znači veličinu bafera. Ako je vrednost negativna onda je veličina bafera na podrazumevanoj sistemskoj vrednosti.

Python, I/O, modovi otvaranja fajla

Mod	Opsi
r	Čitanje. File pointer gleda na početak fajla. Podrazumevani mod.
rb	Čitanje u binarnom formatu. File pointer gleda na početak fajla.
r+	Čitanje i pisanje. File pointer gleda na početak fajla.
rb+	Čitanje i pisanje u binarnom formatu. File pointer gleda na početak fajla.
w	Pisanje. Obrisaće postojeći fajl, odnosno, kreiraće novi fajl za pisanje ako takav fajl ne postoji.
wb	Pisanje u binarnom formatu. Obrisaće postojeći fajl, odnosno, kreiraće novi fajl za pisanje u binarnom formatu ako takav fajl ne postoji.
w+	Čitanje i pisanje. Obrisaće postojeći fajl, odnosno, kreiraće novi fajl za čitanje i pisanje ako takav fajl ne postoji.
wb+	Čitanje i pisanje u binarnom formatu. Obrisaće postojeći fajl, odnosno, kreiraće novi fajl za čitanje i pisanje u binarnom format ako takav fajl ne postoji.
a	Dodavanje. File pointer gleda na kraj postojećeg fajla, odnosno, kreira se novi fajl za pisanje ako takav ne postoji.

Python, I/O, modovi otvaranja fajla, atributi file object-a

Mod	Opsi
ab	Dodavanje u binarnom formatu. File pointer gleda na kraj postojećeg fajla, odnosno, kreira se novi fajl za pisanje u binarnom formatu ako takav ne postoji.
a+	Dodavanje i čitanje. File pointer gleda na kraj postojećeg fajla, odnosno, kreira se novi fajl za čitanje i pisanje ako takav ne postoji.
ab+	Dodavanje i čitanje u binarnom formatu. File pointer gleda na kraj postojećeg fajla, odnosno, kreira se novi fajl za čitanje i pisanje ako takav ne postoji.

- Atributi *file* object-a

Atribute	Opis
file.closed	Vraća true ako je fajl zatvoren, inače vraća false.
file.mode	Vraća mod pristupa u kom je fajl otvoren.
file.name	Vraća ime fajla.

Python, I/O

- Primer:

```
fo = open("foo.txt", "wb")
print ("Name of the file: ", fo.name)
print ("Closed : ", fo.closed)
print ("Opening mode : ", fo.mode)
```

```
Name of the file: foo.txt
Closed : False
Opening mode : wb
```

- Metod `close()` file object-a zatvara fajl.
  - Python automatski zatvara fajl ako se referencirani objekt dodeli drugom fajlu.
  - Bolje je koristiti eksplicitno metod `close()`.

```
fileObject.close();
```

Primer:

```
fo = open("foo.txt", "wb")
print ("Name of the file: ", fo.name)
fo.close()
print(fo.closed)
```

```
Name of the file: foo.txt
True
```

Python, I/O

- Za čitanje i pisanje fajlova koriste se metode:
  - write()
    - write metod upisuje string u otvoreni fajl.
    - write metod ne dodaje znak za novi red ('\n') na kraj stringa

```
fileObject.write(string);
```

primer:

```
fo = open("foo.txt", "wb")
fo.write(b"Python is a great language.\nYeah its great!!\n")
fo.close()
```

```
Python is a great language. Yeah its great!!
```

  - read()
    - metod read čita string iz otvorenog fajla.

```
fileObject.read([count]);
```

prosleđeni parametar je broj bajtova koji će biti pročitani iz otvorenog fajla počevši od početka fajla a ako parametar count nije naveden pokušava se čitanje što više podatka (možda do kraja fajla).

```
fo = open("foo.txt", "r+")
str = fo.read(10);
print ("Read String is : ", str)
fo.close()
```

```
Read String is : Python is
```

Python, I/O

- Pozicioniranje unutar fajla:
  - *tell()* metod vraća tekuću poziciju u fajlu.
  - *seek(offset[, from])* metod menja tekuću fajl poziciju.
    - *offset* argument označava pomeraj pozicije u fajlu.
    - *from* argument označava odakle se računa pomeraj.

      0 pomeraj se računa od početka fajla
      1 pomeraj je od tekuće pozicije u fajlu
      2 pomeraj se računa od kraja fajla

      (f.seek(-3, 2)).

```
fo = open("foo.txt", "r+")
str = fo.read(10);
print ("Read String is : ", str)
position = fo.tell();
print ("Current file position : ", position)
position = fo.seek(0, 0);
str = fo.read(10);
print ("Again read String is : ", str)
fo.close()
```

```
Read String is : Python is
Current file position : 10
Again read String is : Python is
```

Python, I/O

- Python-ov modul **os** obezbeđuje metode za procesiranje operacija nad fajlovima.
  - Za korišćenje modula dovoljno je uraditi import ovog modula.
- Preimenovanje fajlova obavlja metoda rename().
- Metoda *rename()* ima 2 argumenta, tekući naziv fajla i novi naziv fajla.
  (os.rename(current_file_name, new_file_name)).

```
import os
# preimenovanje test1.txt u test2.txt
os.rename( "test1.txt", "test2.txt" )
```

- Brisanje fajlova obavlja metoda remove() kojoj se prosleđuje ime fajla koji se briše (`os.remove(file_name)`).

```
import os
os.remove("text2.txt")
```

- Modul **os** ima nekoliko metoda za kreiranje, uklanjanje i menjanje direktorijuma.
- Metod *`mkdir()`* kreira u tekućem direktorijumu direktorijum čiji je naziv prosleđen kao argument.

```
os.mkdir("newdir")
```

Primer:

```
import os
os.mkdir("test")
```

Python, I/O

- Metod *chdir()* menja tekući direktorijum, jedini argument je naziv novog tekućeg direktorijuma.

```
os.chdir("newdir")
```

Primer:

```
import os
os.chdir("/home/newdir")
```

- Metod *getcwd()* ispisuje tekući radni direktorijum.

```
os.getcwd()
```

Primer:

```
import os
print(os.getcwd())
```

- Metod *rmdir()* uklanja direktorijum čiji je naziv prosleđen ako parametar.
  - Pre uklanjanja direktorijuma, ceo sadržaj unutar direktorijuma mora biti uklonjen.

```
os.rmdir('dirname') #paziti na os.system("rm -rf dirname")
```

Primer:

```
import os
os.rmdir( "/tmp/test" )  # ako je /tmp/test prazan
```

Python: I/O, neke metode file object-a

- `file.close()`, zatvara fajl.
- `file.flush()`, prazni interni bafer (kao fflush kod stdio)
- `file.fileno()`, vraća celobrojnu vrednost file descriptor-a.
- `file.read([size])`, čita najviše *size* bajtova fajla (manje ako ranije dosegne EOF).
- `file.readline([size])`, čita unutar linije fajla (uzeće i završni '\n').
- `file.readlines([sizehint])`, vraća listu linija fajla. Opcionalni parametar sizehint označava koliko će se ukupno bajtova pročitati računajući sve linije.
- `file.seek(offset[, whence])`, postavljanje tekuće pozicije u fajlu.
- `file.tell()`, vraća tekuću poziciju u fajlu.
- `file.truncate([size])`, odseca fajl do na maksimalno datu dužinu.
- `file.write(str)`, upisuje string u fajl (vraća dužinu upisa).
- `file.writelines(sequence)`, upisuje sekvencu stringova u fajl.

Python, metode file object-a

```
fo = open("foo.txt", "wb")
print ("Name of the file: ", fo.name)
fo.close()
```

```
Name of the file: foo.txt
```

```
fo = open("foo.txt", "wb")
print ("Name of the file: ", fo.name)
fo.flush()
fo.close()
```

```
Name of the file: foo.txt
```

```
fo = open("foo.txt", "wb")
print ("Name of the file: ", fo.name)
fid = fo.fileno()
print ("File Descriptor: ", fid)
fo.close()
```

```
Name of the file: foo.txt
File Descriptor: 3
```

Python, metode file object-a

```
fo = open("foo.txt", "r")
for index in range(5):
   line = fo.readline()
   print ("Line No %d - %s" % (index, line))
fo.close()
```

```
Line No 0 - This is 1st line

Line No 1 - This is 2nd line

...sve do 5-te linije sa uračunatim \n
```

```
# sadrzaj foo.txt fajla
za 4 primera koji slede:
This is 1st line
This is 2nd line
This is 3rd line
This is 4th line
This is 5th line
```

```
fo = open("foo.txt", "r")
line = fo.read(10)
print ("Read Line: %s" % (line))
fo.close()
```

```
Read Line: This is 1s
```

```
fo = open("foo.txt", "r")
line = fo.readline()
print ("Read Line: %s" % (line))
line = fo.readline(5)
print ("Read Line: %s" % (line))
fo.close()
```

```
Read Line: This is 1st line
Read Line: This
```

Python, metode file object-a

```
fo = open("foo.txt", "r")
line = fo.readlines()
print ("Read Line: %s" % (line))
line = fo.readlines(2)
print ("Read Line: %s" % (line))
fo.close()

Read Line: ['This is 1st line\n', 'This is 2nd line\n', 'This is 3rd
line\n', 'This is 4th line\n', 'This is 5th line\n']
Read Line: []
```

•Metod seek() postavlja tekuću poziciju u fajlu u odnosu na početak, tekuću vrednost ili kraj fajla (0, 1 i 2 respektivno).

- Nema povratne vrednosti.
- Ako je fajl otvoren za dodavanje korišćenjem 'a' or 'a+', bilo koja seek() operacija biće poništena pri sledećem upisu.
  - Ako je fajl otvoren za pisanje u modu dodavanja 'a' seek() nema uticaja osim ako je u pitanju operacija čitanja (mode 'a+').

Python, I/O

```
fo = open("foo.txt", "r")
line = fo.readline()
print ("Read Line: %s" % (line))
# Again set the pointer to the beginning
fo.seek(0, 0)
line = fo.readline()
print ("Read Line: %s" % (line))
# Close opend file
fo.close()
```

```
Read Line: This is 1st line
Read Line: This is 1st line
```

```
fo = open("foo.txt", "r")
line = fo.readline()
print ("Read Line: %s" % (line))
# Get the current position of the file.
pos = fo.tell()
print ("Current Position: %d" % (pos))
fo.close()
```

```
Read Line: This is 1st line
Current Position: 18
```

Python, I/O

```
fo = open("foo.txt", "r+")
line = fo.readline()
print ("Read Line: %s" % (line))
# Now truncate remaining file.
fo.truncate()
# Try to read file now
line = fo.readline()
print ("Read Line: %s" % (line))
# Close opend file
fo.close()
```

```
Read Line: This is 1st line
Read Line:
```

Python, I/O

```
fo = open("foo.txt", "r+")
str = "This is 6th line"
# Write a line at the end of the file.
fo.seek(0, 2)
line = fo.write( str )

# Now read complete file from beginning.
fo.seek(0,0)
for index in range(6): # napisite resenje koje nije hard-kodovano
   line = fo.readline()
   print ("Line No %d - %s" % (index, line))
# Close opend file
fo.close()
```

```
Name of the file: foo.txt
Line No 0 - This is 1st line
Line No 1 - This is 2nd line
Line No 2 - This is 3rd line
Line No 3 - This is 4th line
Line No 4 - This is 5th line
Line No 5 - This is 6th line
```

Python, I/O

```
fo = open("foo.txt", "r+")
seq = ["This is 6th line\n", "This is 7th line"]
# Write sequence of lines at the end of the file.
fo.seek(0, 2)
line = fo.writelines( seq )
# Now read complete file from beginning.
fo.seek(0,0)
for index in range(7): # napisite resenje koje nije hard-kodovano
   line = fo.readline()
   print ("Line No %d - %s" % (index, line))
# Close opend file
fo.close()
```

```
Line No 0 - This is 1st line
Line No 1 - This is 2nd line
Line No 2 - This is 3rd line
Line No 3 - This is 4th line
Line No 4 - This is 5th line
Line No 5 - This is 6th line
Line No 6 - This is 7th line
```

- Automatsko postavljanje rada sa fajlom (metode __enter__ i __exit__ radimo kasnije)

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