Tutoriat 5

Fișiere

Deschiderea unui fișier

Se asociază o variabila unui fișier.

Se folosești funcția open.

Apel: <variabila_fișier>=open("<cale>","<mod>").

Funcția open întoarce un obiect asociat fișierului(dacă fișierul se află în directorul curent, calea fișierului este chiar nume sau).

Exista 3 moduri :

- "r" => read(citire), genereaza eroare dacă fișierul nu exista.
- "w" => write(scrie), suprascrie conținutul fișierului (fișierul este creat automat dacă acesta nu exista).
- "a" => append, scrie la finalul fișierului (fișierul este creat automat dacă acesta nu exista).
- "x" => create, creează fișierul, genereaza eroare dacă fișierul exista deja.

Prelucrarea datelor din fișiere

Citire:

Metoda	Valoare returnată
<variabila_fișier>.readline()	Returnează linia curentă din fișier sub forma de string. (returnează stringul vid dacă s-a ajuns la final de fișier)
<variabila_fișier>.readlines()	Returnează liniile din fișier sub forma lista de string. Include \n la finalul fiecărui string,excepție ultima linie din fișier.
<variabila_fișier>.read()	Returnează conținutul fișierului sub forma de string.

Scriere:

Metoda	Rezultat
<variabila_fișier>.write(<string>)	Scrie în fișier <string> Nu pune automat \n la final.
<variabila_fișier>.write(<colecție_string>)	Scrie în fișier <colecție_string> Nu delimitează valorile din colecție.

Inchiderea fișierelor.

Se folosește funcția close().

Apel: <variabila_fișier>.close().

Exemple:

cod	date.in	output
#1 f=open("date.in",'r') s=f.read() print(s)	Test citire din fisier	Test citire din fisier #Obs : Se observă că citește # și \n
#2 f=open("date.in",'r') s=f.readline() print(s,end="") print("test")	Test citire din fisier	Test citire test #Obs : Se observă că citește # și \n

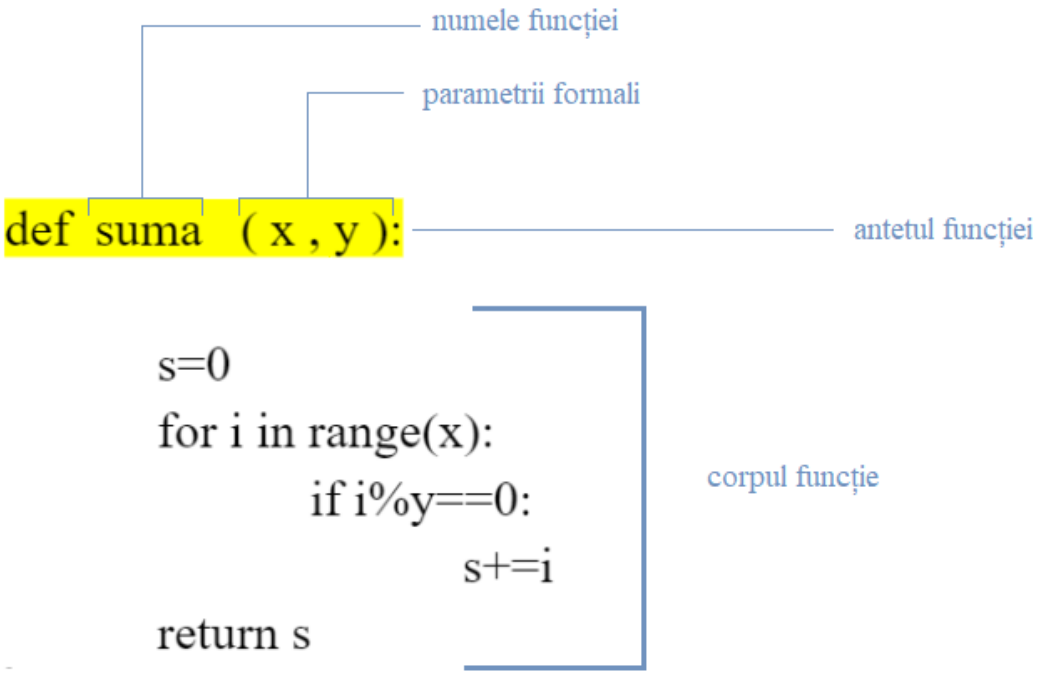
cod	date.out
f=open("date.out",'w') f.write(1)	Obs : genereaza eroare deoarece write poate scrie în fișier doar obiecte de tipul string

Functii/Subprograme

Definire:

def <nume_functie> (<parametrul_1>,<parametrul_2>,..,<parametrul_n>):
    <instructiuni>

Exemplu:



Apel:

suma(10,2) — 10 ,2 parametrii actuali.

Parametrii:

-funcțiile pot avea număr variabil de parametrii,

-sunt prefixate de operatorul *.

! În python nu putem avea 2 funcții cu același nume și cu număr diferit de parametrii (va fi luată în considerare doar ultima funcție declarată)

-pot avea valori default.

Exemplu :

```
def  suma   ( x , y=2 ):
                s=0
                for i in range(x):
                        if i%y==0:
                s+=i
                 return s


print( suma( 10 , 2 ) )   =>20
print ( suma( 10 ) )      =>20
```

Metode de specificare a parametrilor:

```
def f( x, y , z , t ):
        return x+y+z+t;
```

Metoda	Explicație	Exemplu
prin poziție	se respecta numărul și ordinea	f(1, 2, 3, 4)
prin nume	se respecta numărul de parametrii, dar nu și ordinea (foarte des utilizat la apelul funcțiilor predefinite)	f(y=3,z=1,x=2,t=4)
poziție și nume	inati cei prin poziție apoi cei prin nume.	f(1,2,t=5,z=3)

Returnarea valorilor:

-default o funcție returnează None,

- o funcție poate returna mai multe valori (nu neapărat de același tip) împachetate într-un tuple

-rezultatul returnat poate fi despachetat în mai multe variabile.

Transmiterea parametrilor:

Parametrii se transmit prin referința la obiect, dacă referinta parametrului formal se modifica valoarea parametrului actual nu se modifică, dacă se modifica valoarea parametrului formal se modifica valoare și parametrului actual.

Ex:

```
def f(n,lista):
        n+=1;
        lista.append(10);

li=[]
n=0
f(n,li)
print(n,li) => n=0  li=[10]
```

Variabile globale:

Sunt definite folosind sintagma **global <nume_variabila>**.

Pentru a fi folosite în funcții, variabila trebuie declarata folosind sintagma **global <nume_variabila>**.

Ex:

```
global x
x=10
y=10
def f():
    global x
    x=20
    y=20
f()

print(x,y) => x=20, y=10
```