Functions:

===========

Need of functions:

------------------

1> code reusablity

2>Length of code decreses means performance improved

Types of functions:

----------------------

two types of functions

1.Built in funtions /pre defined functions:

-------------------------------------------

print()

id()

len()

eval()

2.user defined functions/ customized functions:

-----------------------------------------------

Based on our requirements we can define a functions

syntax:

---------

def function\_name():

'''Doc string'''

body

return value

function\_name()

Function parameters:

---------------------

def f1(parameters (formal argumnets)):

-------------

-----------------

--------------

f1(parameters (actual arguments ))

eg:

def wish():

print("Hello")

wish()

wish()

wish()

wish()

output:

Hello

Hello

Hello

Hello

return statement:

-----------------

without return:

--------------

If not writing return statement default return value is None

eg:

def wish():

print("Hello")

x=wish()

print(x)

output:

Hello

None

with return:

------------

eg:

def wish():

print("Hello")

return "Good morning"

x=wish()

print(x)

output:

Hello

Good morning

eg:

def wish():

print("Hello")

return "Good morning"

print(wish())

output:

Hello

Good morning

eg:

def wish():

a=10

b=20

return a+b

print(wish())

output:

30

eg:

def wish():

a=10

b=20

return

print(wish())

output:

None

return multiple values:

-----------------------

eg:

def cal(a,b):

total=a+b

sub=a-b

mul=a\*b

div=a/b

fdiv=a//b

return total,sub,mul,div,fdiv

r=cal(10,20)

print(r)

print(type(r))

output:

(30, -10, 200, 0.5, 0)

<class 'tuple'>

Note: if function returns multiple values in the format of tuple

Types of arguments:

------------------

1.postional arguments:

----------------------

1>order of passing of arguments is important

2>Number of actual and formal arguments must be same

3>order of arguments is important other wise we get wrong result

eg:

def total(a,b):

return a+b, a-b

print(total(10,20))

print(total(20,10))

output:

(30, -10)

(30, 10)

eg:

def total(a,b):

return a+b, a-b

print(total(10,20))

print(total(20))

output:

(30, -10)

Traceback (most recent call last):

File "test.py", line 4, in <module>

print(total(20))

TypeError: total() missing 1 required positional argument: 'b'

2.key word arguments:

---------------------

1>order is not important

eg:

def total(a,b):

return a+b, a-b

x=total(a=10,b=20)

print(x)

x=total(b=20,a=10)

print(x)

output:

(30, -10)

(30, -10)

3.default arguments:

--------------------

eg:

def wish(name):

print("Hello:",name)

wish("Raj")

wish("Ram")

wish("Khan")

output:

Hello: Raj

Hello: Ram

Hello: Khan

eg:

def wish(name):

print("Hello:",name)

wish("Raj")

wish("Ram")

wish("Khan")

wish()

output:

Hello: Raj

Hello: Ram

Hello: Khan

Traceback (most recent call last):

File "test.py", line 7, in <module>

wish()

TypeError: wish() missing 1 required positional argument: '

eg:

def wish(name="Guest"):

print("Hello:",name)

wish("Raj")

wish("Ram")

wish("Khan")

wish()

output:

Hello: Raj

Hello: Ram

Hello: Khan

Hello: Guest

eg:

def wish(name="Guest"):

if name=="Jagan":

print("Hello",name,"How are you")

else:

print("Hello",name)

wish()

wish("Raj")

wish("Jagan")

output:

Hello Guest

Hello Raj

Hello Jagan How are you

4.variable length arguments:

----------------------------

eg:

def calc(a,b):

print(a+b)

calc(10,20)

output:

30

eg:

def calc(a,b):

print(a+b)

calc(10,20,30)

output:

Traceback (most recent call last):

File "test.py", line 3, in <module>

calc(10,20,30)

TypeError: calc() takes 2 positional arguments but 3 were given

eg:

def f1(\*n):

print(n)

print(type(n))

f1()

f1(1)

f1(1,2)

f1(1,2,3)

f1(1,2,3,4)

output:

()

<class 'tuple'>

(1,)

<class 'tuple'>

(1, 2)

<class 'tuple'>

(1, 2, 3)

<class 'tuple'>

(1, 2, 3, 4)

<class 'tuple'>

eg:

def total(\*n):

total=0

for x in n:

total=total+x

print("Total:",total)

total()

total(10)

total(10,20,30)

total(10,20,30,40,50,60,70,80,10)

output:

Total: 0

Total: 10

Total: 60

Total: 370

5.variable length key word arguments:

-------------------------------------

eg:

def f1(\*\*kwargs):

print(kwargs)

print(type(kwargs))

f1()

output:

{}

<class 'dict'>

eg:

def f1(\*\*kwargs):

print(kwargs)

f1(A=10,B=20)

f1(name="jagan",roll=222,addrs="ATP")

output:

{'A': 10, 'B': 20}

{'name': 'jagan', 'roll': 222, 'addrs': 'ATP'}

Types of variable or scope of variable:

-----------------------------------------

1.Global variable

2.Local variable

1.Global variable:

-------------------

->To define variable outside of function are known as global variables

->The global variables we can access any where inside our code

2.Local variable:

------------------

->To define variable inside of function are known as local variables

->The local variables available only inside of the function

eg:

x=200

y=400

def f1():

a=100

b=20

print(a)

print(b)

print(x)

print(y)

f1()

print(x)

print(y)

output:

100

20

200

400

200

400

eg:

x=200

y=400

def f1():

a=100

b=20

print(a)

print(b)

print(x)

print(y)

f1()

print(x)

print(y)

print(a)

print(b)

output:

100

20

200

400

200

400

Traceback (most recent call last):

File "test.py", line 14, in <module>

print(a)

NameError: name 'a' is not defined

eg:

x=200

y=400

def f1():

a=100

b=20

print(a)

print(b)

print(x)

print(y)

def f2():

c=1000

d=2000

print(c)

print(d)

print(x)

print(y)

f1()

f2()

print(x)

print(y)

output:

100

20

200

400

1000

2000

200

400

200

400

eg:

x=200

y=400

def f1():

a=100

b=20

print(a)

print(b)

print(x)

print(y)

def f2():

c=1000

d=2000

print(c)

print(d)

print(a)

print(x)

print(y)

f1()

f2()

output:

100

20

200

400

1000

2000

Traceback (most recent call last):

File "test.py", line 20, in <module>

f2()

File "test.py", line 16, in f2

print(a)

NameError: name 'a' is not defined

global key word:

---------------

We can make it local variables to global variables

eg:

x=200

y=400

def f1():

global a

a=100

b=20

print(a)

print(b)

print(x)

print(y)

def f2():

c=1000

d=2000

print(c)

print(d)

print(a)

print(x)

print(y)

f1()

f2()

print(a)

output:

100

20

200

400

1000

2000

100

200

400

100

eg:

x=200

y=400

def f1():

global a

a=100

b=20

print(a)

print(b)

print(x)

print(y)

def f2():

c=1000

d=2000

print(c)

print(d)

print(a)

print(x)

print(y)

f2()

f1()

output:

1000

2000

Traceback (most recent call last):

File "test.py", line 21, in <module>

f2()

File "test.py", line 17, in f2

print(a)

NameError: name 'a' is not defined

eg:

x=200

y=400

def f1():

global a

a=100

b=20

print(a)

print(b)

print(x)

print(y)

def f2():

c=1000

d=2000

print(c)

print(d)

print(a)

print(x)

print(y)

f1()

f2()

output:

100

20

200

400

1000

2000

100

200

400

Anonymous functions or lambda functions:

----------------------------------------

1>The function don't have name

2>Instant use (only one time use)

eg:

def squre(n):

return n\*n

print(squre(4))#squre(4)------>16 -------->16

print(squre(5))

x=squre(6)

print(x)

output:

16

25

36

lambda function syntax:

----------------------

d= lambda n:n\*n

eg:

d=lambda n:n\*n

print(d(4))

print(d(6))

output:

16

36

eg:

s= lambda a,b:a+b

print(s(2,4))

output:

6

eg:

s= lambda a,b: a if a>b else b

print(s(2,3))

print(s(50,20))

output:

3

50

The Function as argument to another functions

1>filter(function,sequence)

2>map(function,sequence)

3>reduce(function,sequence)

1>filter(function,sequence):

----------------------------

eg:

def Even(n):

if n%2==0:

return True

else:

return False

l=[1,2,3,4,5,6,7,8,9]

l1=[]

for n in l:

if Even(n)==True:

l1.append(n)

print(l1)

output:

[2, 4, 6, 8]

eg:

def Even(n):

if n%2==0:

return True

else:

return False

l=[1,2,3,4,5,6,7,8,9]

l1=filter(Even,l)

print(l1)

print(list(filter(Even,l)))

output:

<filter object at 0x000001244BFBEA58>

[2, 4, 6, 8]

eg:

def Even(n):

if n%2==0:

return True

else:

return False

l1=list(filter(Even,[1,2,3,4,5,6,7,8,9]))

print(l1)

output:

[2, 4, 6, 8]

eg:

l=lambda n:n%2==0

print(l(2))

print(l(3))

output:

True

False

eg:

l1=filter(lambda n:n%2==0,[1,2,3,4,5,6,7,8,9])

print(l1)

print(list(l1))

output:

<filter object at 0x00000217CB4DE780>

[2, 4, 6, 8]

eg:

l1=list(filter(lambda n:n%2==0,[1,2,3,4,5,6,7,8,9]))

print(l1)

output:

[2, 4, 6, 8]

eg:

l= list(filter(lambda name:name[0]=="A",["Ant","Apple","Bat","Cat","Axe"]))

print(l)

output:

['Ant', 'Apple', 'Axe']

2>map(function,sequence):

-------------------------

eg:

def sqre\_no(n):

return n\*n

l=[1,2,3,4,5,6]

l1=[]

for n in l:

l1.append(sqre\_no(n))

print(l)

print(l1)

output:

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

[1, 4, 9, 16, 25, 36]

eg:

def sqre\_no(n):

return n\*n

l1=map(sqre\_no,[1,2,3,4,5,6])

print(l1)

print(list(l1))

output:

<map object at 0x00000221AF7BEA58>

[1, 4, 9, 16, 25, 36]

eg:

l1=list(map(lambda n:n\*n ,[1,2,3,4,5,6]))

print(l1)

output:

[1, 4, 9, 16, 25, 36]

eg:

l1=[1,2,3,4,5]

l2=[10,20,30,40,50]

l3=list(map(lambda x,y:x\*y ,l1,l2))

print(l3)

output:

[10, 40, 90, 160, 250]

eg:

l3=list(map(lambda x,y:x\*y ,[1,2,3,4,5],[10,20,30,40,50]))

print(l3)

output:

[10, 40, 90, 160, 250]

3>reduce(function,sequence):

-----------------------------

eg:

r=reduce(lambda x,y:x+y ,[1,2,3,4,5])

print(r)

output:

Traceback (most recent call last):

File "test.py", line 2, in <module>

r=reduce(lambda x,y:x+y ,[1,2,3,4,5])

NameError: name 'reduce' is not defined

eg:

from functools import \*

r=reduce(lambda x,y:x+y ,[1,2,3,4,5])

print(r)

output:

15