Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

FUNCTIONS-3

Agenda:

1. Types of Parameters (contd..)

3.TYPES OF PARAMETERS

Example Programs on default parameters.

I.

```
def wish(name ='Guest',msg):
# After default argument, we should not take non-default argument
    print('Hello',name,msg)
```

```
  File "<ipython-input-15-1e1d7a3c73c0>", line 1
    def wish(name ='Guest',msg):        # After default argument, we should
 not take non-default argument
             ^
SyntaxError: non-default argument follows default argument
```

II.

```
def wish(msg,name ='Guest'):
    print(msg,name)
wish('Hello','Karthi')
```

Output: Hello Karthi

III.

```
def wish(msg,name ='Guest'):
    print(msg,name)
wish('Hello')
```

Output: Hello Guest

IV.

```
def wish(msg,name ='Guest'):
    print(msg,name)
wish()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-993af2d34958> in <module>
      2     print(msg,name)
      3
----> 4 wish()

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

Note:

You can give any number of default arguments.

Eg.:

```
def wish(name ='Guest',msg='Good Morning'):
    print('Hello',name,msg)
wish()
```

Output: Hello Guest Good Morning

Eg.

```
def wish(marks,age,name = 'Guest', msg = 'Good Morning'):
    print('Student Name:',name)
    print('Student Age:',age)
    print('Student Marks:',marks)
    print('Message:',msg)
wish(99,48,'Karthi')                    # Valid
```

```
Student Name: Karthi
Student Age: 48
Student Marks: 99
Message: Good Morning
```

Eg.

```
def wish(marks,age,name = 'Guest', msg = 'Good Morning'):
    print('Student Name:',name)
    print('Student Age:',age)
    print('Student Marks:',marks)
    print('Message:',msg)
wish(age=48,marks = 100)                # valid
```

```
Student Name: Guest
Student Age: 48
Student Marks: 100
Message: Good Morning
```

Eg.

```
def wish(marks,age,name = 'Guest', msg = 'Good Morning'):
    print('Student Name:',name)
    print('Student Age:',age)
    print('Student Marks:',marks)
    print('Message:',msg)
wish(100,age=46,msg='Bad Morning',name='Karthi')          # valid
```

```
Student Name: Karthi
Student Age: 46
Student Marks: 100
Message: Bad Morning
```

Eg.

```
def wish(marks,age,name = 'Guest', msg = 'Good Morning'):
    print('Student Name:',name)
    print('Student Age:',age)
    print('Student Marks:',marks)
    print('Message:',msg)
wish(marks=100,46,msg='Bad Morning',name = 'Karthi')
# invalid, first keyword argument then positional argument is not allowed.
```

SyntaxError: positional argument follows keyword argument

Eg.

```
def wish(marks,age,name = 'Guest', msg = 'Good Morning'):
    print('Student Name:',name)
    print('Student Age:',age)
    print('Student Marks:',marks)
    print('Message:',msg)
wish(46,marks=100,msg='Bad Morning',name = 'Karthi')     # invalid
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-e767702586d4> in <module>
      5     print('Message:',msg)
      6
----> 7 wish(46,marks=100,msg='Bad Morning',name = 'Karthi')  # invalid

TypeError: wish() got multiple values for argument 'marks'
```

Programs on variable length parameters:

```
def sum(a,b):
    print(a+b)
sum(10,20)        ➔ 30
```

Now it is working correctly. After some time my requirement is as follows:

sum(10,20,30)

```
def sum(a,b):
    print(a+b)          # This sum() function we can't use for the new requirement.
sum(10,20,30)
```

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

```
def sum(a,b,c):
    print(a+b+c)            # we have to go for another sum() function
sum(10,20,30)      ➔ 60
```

Now it is working correctly. After some time my requirement is as follows:

sum(10,20,30,40)

```
def sum(a,b,c):
    print(a+b+c)
sum(10,20,30,40)
```

TypeError: sum() takes 3 positional arguments but 4 were given.

Once again the same problem. we should go for another sum() function.

```
def sum(a,b,c,d):
    print(a+b+c+d)          # we have to go for another sum() function
sum(10,20,30,40)            ➔100
```

- If you change the number of arguments, then automatically for every change, compulsorily we need to go for new function unnecessarily. Because of this length of the code is going to increase. To overcome this problem we should go for **variable length arguments.**

Eg.

```
def sum(*n):    # Here, 'n' is a variable length argument.
    result =0
    for x in n:
        result = result + x
    print(result)
sum(10,20,30,40)        ➔100
```

Eg.

```
def sum(*n):        # Here, 'n' is a variable length argument.
    result =0
    for x in n:
        result = result + x
    print(result)
sum(10,20,30)              ➔60
```

```
def sum(*n): # Here, 'n' is a variable length argument.
    result =0
    for x in n:
        result = result + x
    print(result)
sum(10,20)                 ➔30
```

```
def sum(*n):                # Here, 'n' is a variable length argument.
    result =0
    for x in n:
        result = result + x
        print(result)
sum(10)                     ➔10

def sum(*n):                # Here, 'n' is a variable length argument.
    result =0
    for x in n:
        result = result + x
    print(result)
sum()                       ➔0
```

Eg.

```
def sum(*n):                    # Here, 'n' is a variable length argument.
    result =0
    for x in n:
        result = result + x
    print('The Sum is : ', result)
sum(10,20,30,40)
sum(10,20,30)
sum(10,20)
sum(10)
sum()
```

```
The Sum is :  100
The Sum is :  60
The Sum is :  30
The Sum is :  10
The Sum is :  0
```

Note : Same function is used for variable number of arguments.

Key Point 1:

- We can mix variable length arguments with positional arguments.
- You can take positional arguments and variable length arguments simultaneously.

Eg.

```
def sum(name,*n):
    result =0
    for x in n:
        result = result + x
    print("The Sum by", name, ": ", result)
sum('Robin',10,20,30,40)
sum('Rahul',10,20,30)
sum('Sachin',10,20)
sum('Sourav',10)
sum('Karthi')
```

```
The Sum by Robin :  100
The Sum by Rahul :  60
The Sum by Sachin :  30
The Sum by Sourav :  10
The Sum by Karthi :  0
```

Note:

- **Rule :** After variable length argumenst,if we are taking any other arguments then we should provide values as keyword arguments.

Eg.

```
def sum(*n,name):
    result =0
    for x in n:
        result = result + x
    print("The Sum by", name, ": ", result)
sum('Robin',10,20,30,40)
sum('Rahul',10,20,30)
sum('Sachin',10,20)
sum('Sourav',10)
sum('Karthi')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-b8733ebba999> in <module>
      4         result = result + x
      5     print("The Sum by", name, ": ", result)
----> 6 sum('Robin',10,20,30,40)
      7 sum('Rahul',10,20,30)
      8 sum('Sachin',10,20)

TypeError: sum() missing 1 required keyword-only argument: 'name'
```

Eg.

```
def sum(*n,name):
    result =0
    for x in n:
        result = result + x
    print("The Sum by", name, ": ", result)
sum(name = 'Robin',10,20,30,40)
sum(name = 'Rahul',10,20,30)
sum(name = 'Sachin',10,20)
sum(name = 'Sourav',10)
sum(name ='Karthi')
```

```
  File "<ipython-input-40-19134896e44a>", line 6
    sum(name = 'Robin',10,20,30,40)
                                  ^
SyntaxError: positional argument follows keyword argument
```

Eg.

```
def sum(*n,name):
    result =0
    for x in n:
        result = result + x
    print("The Sum by", name, ": ", result)
sum(10,20,30,40,name = 'Robin')
sum(10,20,30,name = 'Rahul')
sum(10,20,name = 'Sachin')
sum(10,name = 'Sourav')
sum(name ='Karthi')
```

```
The Sum by Robin :  100
The Sum by Rahul :  60
The Sum by Sachin :  30
The Sum by Sourav :  10
The Sum by Karthi :  0
```

Another Example:

```
def f1(n1,*s):
    print(n1)
    for s1 in s:
        print(s1)
f1(10)
f1(10,20,30,40)
f1(10,"A",30,"B")
```

```
10
10
20
30
40
10
A
30
B
```

Conclusion:

- After variable length arguments, if you are taking any other argument, then we have to provide values as key word arguments only.
- If you pass first normal argument and then variable arguments, then there is no rule to follow. It works correctly.

Key Point 2:

Keyword variable length arguments :

- Now, Suppose if we want to pass any number of keyword arguments to a function, compulsorily we have to identify the difference with the above case (i.e., Passing of any number of positional arguments).
- We can declare key word variable length arguments also. For this we have to use **.
- We can call this function by passing any number of keyword arguments. Internally these keyword arguments will be stored inside a dictionary.

Eg.

```
def display(**kwargs):
    for k,v in kwargs.items():
        print(k,"=",v)
display(n1=10,n2=20,n3=30)
display(rno=100,name="Karthi",marks=70,subject="Python")
```

```
n1 = 10
n2 = 20
n3 = 30
rno = 100
name = Karthi
marks = 70
subject = Python
```

Case Study:

```
def f(arg1,arg2,arg3=4,arg4=8):
    print(arg1,arg2,arg3,arg4)

f(3,2)       #3 2 4 8

f(10,20,30,40)   # 10,20,30,40

f(25,50,arg4=100)   # 25  50 4 100

f(arg4=2,arg1=3,arg2=4)   # 3 4 4 2

#f()      # TypeError: f() missing 2 required positional arguments: 'arg1' and 'arg2'

#f(arg3=10,arg4=20,30,40)    SyntaxError: positional argument follows keyword argument

#f(4,5,arg2=6)  #TypeError: f() got multiple values for argument 'arg2'

#f(4,5,arg3=5,arg5=6)  #TypeError: f() got an unexpected keyword argument 'arg5'
```

```
3 2 4 8
10 20 30 40
25 50 4 100
3 4 4 2
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You