Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

FUNCTIONS-2

Agenda:

1. Types of Parameters

3.TYPES OF PARAMETERS

Parameters or Arguments:

- Parameters are inputs to the function.
- If a function contains parameters, then at the time of calling, compulsory we should provide values, otherwise, we will get error.

Types of Parameters in Python:

1. Positional Parameters:

- In the case of positional arguments, number of arguments must be same.
- In the case of positional arguments, order of the arguments is important.

2. Keyword (i.e., Parameter name) Parameters:

- In the case of keyword arguments, order of the arguments is not important.
- In the case of keyword arguments, number of arguments must be same.

3. Default Parameters:

- You can define default value for the arguments.
- If you are not passing any argument, then default values by default will be considered.
- After default arguments you should not take normal arguments. (i.e., Default arguments you need to take at last)

4. Variable length Parameters:

- Sometimes we can pass variable number of arguments to our function, such type of arguments are called variable length arguments.
- We can declare a variable length argument with * symbol as follows:

def f1(*n):

- We can call this function by passing any number of arguments including zero number. Internally all these values represented in the form of tuple.

Eg: Write a function to take name of the student as input and print wish message by name.

Program:

```
def wish(name):
    print("Hello",name," Good Morning")
wish("Karthi")
wish("Sahasra")
```

Output:

```
Hello Karthi Good Morning
Hello Sahasra Good Morning
```

Eg: Write a function to take number as input and print its square value.

Program:

```
def squareIt(number):
    print("The Square of",number,"is", number*number)
squareIt(4)
squareIt(5)
squareIt(7)
```

Output:

```
The Square of 4 is 16
The Square of 5 is 25
The Square of 7 is 49
```

return statement

- Function can take input values as parameters and executes business logic, and returns output to the caller with return statement.
- **Python function can return any number of values at a time by using a return statement.**
- Default return value of any python function is **None**.

Eg:

```
def wish():
    print('hello')
#print(wish())
wish()
```

Output: hello

```
def wish():
    print('hello')
print(wish())
#wish()
```

Output:

hello

None

Simple Example Programs

Q 1. Write a function to accept 2 numbers as input and return sum.

```
def add(x,y):
    return x+y
result=add(10,20)
print("The sum is",result)
print("The sum is",add(100,200))
```

Output:

The sum is 30

The sum is 300

Note: If we are not writing return statement then default return value is **None.**

Eg:

```
def f1():
    print("Hello")
f1()
print(f1())
```

Output:

```
Hello
Hello
None
```

Q 2. Write a function to check whether the given number is even or odd?

```
def even_odd(num):
    if num%2==0:
        print(num,"is Even Number")
    else:
        print(num,"is Odd Number")
even_odd(10)
even_odd(15)
```

Output:

10 is Even Number

15 is Odd Number

Q 3. Write a function to find factorial of given number.

```
def fact(num):
    result=1
    while num>=1:
        result=result*num
        num=num-1
    return result
for i in range(1,5):
    print("The Factorial of",i,"is :",fact(i))
```

```
The Factorial of 1 is : 1
The Factorial of 2 is : 2
The Factorial of 3 is : 6
The Factorial of 4 is : 24
```

Returning multiple values from a function

- In other languages like C,C++ and Java, function can return atmost one value. But in Python, a function can return any number of values.

```
def calc(a,b):          # Here, 'a' & 'b' are called positional arguments
    sum = a + b
    sub = a - b
    mul = a * b
    div = a / b
    return sum,sub,mul,div

a,b,c,d = calc(100,50)          # Positional arguments
print(a,b,c,d)
```

```
150 50 5000 2.0
```

Alternate Way:

```
def calc(a,b):                          # Positional Arguments
    sum = a + b
    sub = a - b
    mul = a * b
    div = a / b
    return sum,sub,mul,div

t = calc(100,50)
for x in t:
    print(x)
```

```
150
50
5000
2.0
```

```
def calc(a,b):                          # keyword arguments Arguments
    sum = a + b
    sub = a - b
    mul = a * b
    div = a / b
    return sum,sub,mul,div

t = calc(a = 100, b = 50)               # keyword arguments Arguments
for x in t:
    print(x)
```

```
150
50
5000
2.0
```

```
def calc(a,b):                          # keyword arguments Arguments
    sum = a + b
    sub = a - b
    mul = a * b
    div = a / b
    return sum,sub,mul,div

t = calc(b = 50, a = 100)               # keyword arguments Arguments
for x in t:
    print(x)
```

```
150
50
5000
2.0
```

Some more examples on keyword arguments:

I.

```
def calc(a,b):                  # keyword arguments
    sum = a + b
    sub = a - b
    mul = a * b
    div = a / b
    return sum,sub,mul,div
t = calc(100, b = 50)           # It is perfectly valid
for x in t:
    print(x)
```

```
150
50
5000
2.0
```

II.

```
def calc(a,b):                    # keyword arguments
    sum = a + b
    sub = a - b
    mul = a * b
    div = a / b
    return sum,sub,mul,div
t = calc(b = 50, 100) # It is invalid, because positional argument should follow keyword argument.
for x in t:                       # first keyword argument then positional argument is not allowed.
    print(x)
```

SyntaxError: positional argument follows keyword argument.

III.

```
def calc(a,b):                  # keyword arguments
    sum = a + b
    sub = a - b
    mul = a * b
    div = a / b
    return sum,sub,mul,div
t = calc(50, a = 50)            # It is also invalid
for x in t:
    print(x)
```

TypeError: calc() got multiple values for argument 'a'

Another Example:

```
def wish(name,msg):
    print('Hello',name,msg)
wish(name = 'Karthi',msg = 'Good Morning')
#order is not important, but no.of arguments is important
wish(msg = 'Good Morning',name = 'Karthi')
```

Output:

```
Hello Karthi Good Morning
Hello Karthi Good Morning
```

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