PythonCourse_4_Functions

April 18, 2021

0.1 Functions

```
[4]: def fact(a):                        #Function Definiton
         a_fact = 1
         for i in range(1, a + 1):
             a_fact = a_fact * i
         return a_fact                  #Can return a value
```

```
[6]: fact(4)                             #Function Call
```

```
[6]: 24
```

```
[8]: ##Calculate nCr

     n = int(input())
     r = int(input())
     n_fact = fact(n)
     r_fact = fact(r)
     n_r_fact = fact(n-r)

     ans = n_fact //(r_fact * n_r_fact)
     print(ans)
```

```
4
2
6
```

```
[9]: ## Funtion to check if no is Prime
     def isPrime(n):
         for d in range(2, n):
             if (n % d == 0):
                 break
         else:
             return True # Executes only when the comple range of For is done
                         # Once Return statement done, statements below no longer␣
      ↪executed in the func
         return False # Can be kept here as the else has a return
```

```
[11]: isPrime(10)
```

[11]: False

[12]:
```
## Function to print prime nos from 2 to n
def printPrimetillN(n):
    for k in range (2, n + 1):
        is_k_prime = isPrime(k) #Function call inside Function
        if is_k_prime:
            print(k)
```

[13]:
```
printPrimetillN(20)
```

```
2
3
5
7
11
13
17
19
```

[17]:
```
def func(a):
    a = a + 10
    return a
a = 5
func(a)  # Have not assigned to any variable
print(a) # Therefore, 'a' still retains 5
print(func(a))
```

```
5
15
```

Scope of Variables

[19]:
```
a1 = 5 # Global variable
def f1():
    print('Inside Function')
    print(a1) # Global variable can be ACCESSED within the function

print('Outside Function')
print(a1) # Global variable can be ACCESSED outside the function
f1()
```

```
Outside Function
5
Inside Function
5
```

```
[22]: a2 = 5 # Global variable
def f2():
    print('Inside Function')
    b2 = 10 #Local variable
    print(b2) # Local variable can be ACCESSED within the function

print('Outside Function')
#print(b2) # Local variable CANNOT be ACCESSED outside the function␣
 ↪-->NameError: name 'b2' is not defined
f2()
```

```
Outside Function
Inside Function
10
```

```
[24]: def f3():
    print('Inside Function')
    print(a3)

print('Outside Function')
a3 = 5
f3()
#a3 = 5 # Global variable --> Has to be define before Function Call
print(a3)
```

```
Outside Function
Inside Function
5
5
```

```
[29]: a4 = 5
def f4():
    print('Inside Function')
    a4 = 6  # Global variable CANNOT be changed as such within the function
            # Python assumes that a new local variable is being created
    print(a4)
    print(id(a4)) # Address is different in this case

print('Outside Function')
print(a4)
print(id(a4))
f4()
print('After Function Execution')
print(a4)
print(id(a4))
```

```
Outside Function
5
8791248938912
```

```
Inside Function
6
8791248938944
After Function Execution
5
8791248938912
```

[30]:
```python
## To use and change the global variable inside the function
a5 = 5
def f5():
    global a5 # Specify global varaible type
    print('Inside Function')
    a5 = 6  # Global variable CAN now be changed
    print(a5)
    print(id(a5)) # Address gets changed since in python it is value based

print('Outside Function')
print(a5)
print(id(a5))
f5()
print('After Function Execution')
print(a5)
print(id(a5))
```

```
Outside Function
5
8791248938912
Inside Function
6
8791248938944
After Function Execution
6
8791248938944
```

Default Parameters

[31]:
```python
def sum1(a,b,c):
    return a + b + c

sum1(2,3) #Passing lesser Args--> Error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-5a1831598276> in <module>
      2     return a + b + c
      3
----> 4 sum1(2,3)
```

```
TypeError: sum1() missing 1 required positional argument: 'c'
```

[32]:
```
def sum2(a,b,c):
    return a + b + c

sum2(2, 3, 4, 5) #Passing More Agrs --> Error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-7422ef67f4af> in <module>
      2     return a + b + c
      3
----> 4 sum2(2, 3, 4, 5)

TypeError: sum2() takes 3 positional arguments but 4 were given
```

[34]:
```
def sum3(a,b,c = 0): # C takes default value of 0
    print('c',c)
    return a + b + c

print('sum',sum3(2, 3))
```

```
c 0
sum 5
```

[35]:
```
def sum4(a,b,c = 0): # C takes default value of 0
    print('c',c)   #value of C is passed from the func call
    return a + b + c

print('sum',sum4(2, 3, 4))
```

```
c 4
sum 9
```

[36]:
```
def sum5(a,b = 0,c):  # Default args only allowed at end
    return a + b + c

print('sum',sum5(2, 3, 4))
```

```
  File "<ipython-input-36-fec3797e1fd8>", line 1
    def sum5(a,b = 0,c):
           ^
SyntaxError: non-default argument follows default argument
```

```
[37]: def sum6(a,b,c = 0, d = 5):
          return a + b + c + d

      print('sum',sum6(2, 3, d= 0)) #The specified value can be changed
```

```
sum 5
```

```
[39]: def sum7(a,b,c = 0, d = 5):
          return a + b + c + d

      print('sum',sum6(d = 2,a = 3, c= 0, b = 1)) #When specifying the values, order␣
       ↪does not matter
```

```
sum 6
```

```
[2]: def printTable(start,end,step):
     #Implement Your Code Here
         for i in range(start, end + 1, step):
             c_val = int((5/9)*(i - 32))
             print(i, c_val)

     s = int(input())
     e = int(input())
     step = int(input())
     printTable(s,e,step)
```

```
23
45
5
23 -5
28 -2
33 0
38 3
43 6
```