NB 03 - Python Fundamentals (Functions)

August 16, 2021

1 Functions without arguments

[10]:
```
def wish_birthday():
    print("Happy Birthday!!!")
```

[11]:
```
wish_birthday()
```

```
Happy Birthday!!!
```

2 Functions with arguments

[12]:
```
def add_vals(a,b):
    print(a+b)
```

[13]:
```
add_vals(20,10)
```

```
30
```

[14]:
```
add_vals(-10,40)
```

```
30
```

3 Default arguments

[15]:
```
def add_vals(a,b=10):
    print(a+b)
```

[16]:
```
add_vals(5)
```

```
15
```

[17]:
```
add_vals(20,10)
```

```
30
```

[18]:
```
def add_vals(a=10,b):
    print(a+b)
```

```
#This returns an error. Non default argument can not follow a default argument
```

```
      File "<ipython-input-18-a7e9e54bd437>", line 1
    def add_vals(a=10,b):
                ^
SyntaxError: non-default argument follows default argument
```

4 Return statement

[19]:
```
def add_vals(a,b):
    print(a+b)
```

[20]:
```
2+add_vals(4,5) #Print a value. But an error is given
```

```
9
```

```
   ␣
 ↪---------------------------------------------------------------------------

    TypeError                                 Traceback (most recent call␣
 ↪last)

    <ipython-input-20-7ecc473fe256> in <module>
----> 1 2+add_vals(4,5) #Print a value. But an error is given


    TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

[21]:
```
def add_val2(a,b):
    return a+b
```

[22]:
```
3+add_val2(10,20)
```

[22]: 33

[23]:
```
add_val2(add_val2(3,5),6)
```

[23]: 14

[24]:
```
def absulute_val(a):
    if a>=0:
        return a
    else:
        return -a
```

[25]:
```
absulute_val(10)
```

[25]:
```
10
```

[26]:
```
absulute_val(-20)
```

[26]:
```
20
```

[27]:
```
def add_prod_vals(a,b,c):
    return add_val2(a,b)*c
```

[28]:
```
add_prod_vals(2,3,4)
```

[28]:
```
20
```

5 Scope of the variables

[29]:
```
def ret_val():
    x=20
    return x

x=30

print(x)
print(ret_val())
print(x)
```

```
30
20
30
```

[30]:
```
def ret_val():
    global x
    x=20
    return x

x=30

print(x)
print(ret_val())
print(x)
```

```
30
20
20
```

6 map function

[31]:
```
L=[23,33,21,23,34]

def fun(x):
    return 5*x
```

[32]:
```
L2=map(fun,L)
```

[33]:
```
list(L2)
```

[33]:
```
[115, 165, 105, 115, 170]
```

[34]:
```
K=[[34,33,21,23,45],[56,67,32],[67,56,42]]

def fun2(L):
    return L[0]

P=map(fun2,K)
print(list(P))
```

```
[34, 56, 67]
```

7 lambda keyword

[35]:
```
def fun1(x):
    return 2*x

fun2=lambda x : 2*x
```

[36]:
```
fun1(5)
```

[36]:
```
10
```

[37]:
```
fun2(5)
```

[37]:
```
10
```

[38]:
```
fun3=lambda x,y : x+y
```

[39]:
```
fun3(10,20)
```

[39]:
```
30
```

8 Recursion

[40]:
```
def factorial(x):
    if x==0 or x==1:
        return 1
    else:
        return x*factorial(x-1)
```

[41]:
```
factorial(5)
```

[41]:
```
120
```

[42]:
```
def add_rec(x):
    if x>=0:
        if x==1:
            return 1
        else:
            return x+add_rec(x-1)
    else:
        print("Try positive value")
```

[43]:
```
add_rec(3)
```

[43]:
```
6
```

9 Non keyword arguments

[44]:
```
def fun(*args):
    for i in args:
        print(i, end=" ")
```

[45]:
```
fun(12,22,34,45)
```

```
12 22 34 45
```

10 Keyword arguments

[46]:
```
def fun(**kwargs):
    return kwargs
```

[47]:
```
fun(param1=34,param2=67)
```

[47]:
```
{'param1': 34, 'param2': 67}
```