

FUNCTION

Pratik Joshi



Function in Python

Function is a block of codes which perform a specific task.

Syntax:

```
def function_name(arg):  
    Statement1 #Function Body  
    return val
```

- Function name is uniquely identified throughout the python program
- Arguments and return statement are option

Example:

```
def test():  
    print("Hi from function")
```

```
test() #Calling the method
```



Arguments and Return

```
def sum(a,b):  
    sum=a+b  
    return sum    #returning result  
op=sum(5,9) #Calling function with argument
```

Default Argument:

```
def fun(name,country='India')  
    print(name,'is from',country)
```

```
fun('Pratik') #op: Pratik is from India
```

```
fun('Trump','USA') #Op: Trump is from USA
```

```
fun(name='Modi') #Op: Modi is from India
```

```
fun(country='Russia',name='Putin') #Op: Putin is from Russia
```

Arbitrary Arguments

If the number of arguments are unknown, a * can be added before the parameter name in function definition.

```
def fun(*names):  
    print("The first PM is",names[0])  
def('Modi','Trump','Putin') #OP: The first PM is Modi
```

Arbitrary Keyword Arguments:

If the number of keyword arguments is unknown, ** can be added before function parameter.

```
def fun(**kwargs):  
    print("Favourite color is,kwargs['color']")  
fun(color='Blue',food='soup',sport='Badminton') #Op: Favourite color is Blue
```

Function Recursion

When a defined function calls itself, it is called function recursion.

#Factorial of a number using Function Recursion

```
def fact_rec(num):  
    fact=1  
    if num>1:  
        fact=num*fact_rec(num-1)  
    return fact  
print(fact_rec(5)) #120
```

#Sum of first 10 number using Recursion

```
def sum_rec(num):  
    sum=0  
    if num>0:  
        sum=num+sum_rec(num-1)  
    return sum  
print(sum_rec(10)) #55
```

Pratik Joshi



Lambda Function

Lambda is an anonymous function which can be defined without name. It can take any number of arguments, but can only have one expression.

Syntax:

```
lambda arguments : expression
```

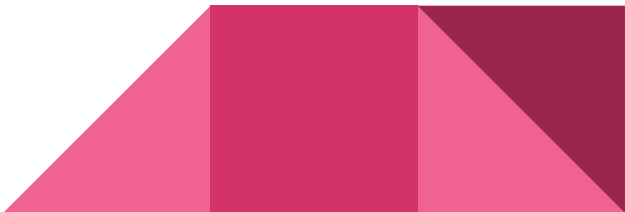
Example:

```
x=lambda a:a*2  
print(x(5)) #op:10
```

```
y=lambda a,b:a*b  
print(y(5,8)) #op:40
```

```
z=lambda a,b,c:a*b*c  
print(z(10,6,2)) #op:120
```

Pratik Joshi



Lambda Inside Another Function

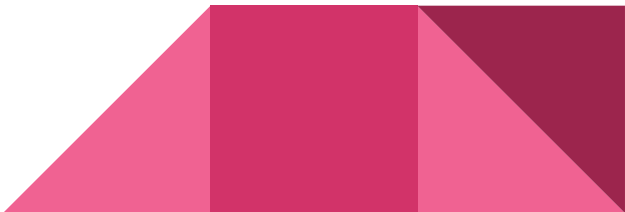
The Use of lambda is better shown when it is used as an anonymous function inside another function.

```
def pow(n):  
    return lambda a:a**n
```

```
sqr=pow(2)  
cube=pow(3)  
quad=pow(4)
```

```
print(sqr(5)) #op:25  
print(cube(5)) #op:125  
print(quad(5)) #op:625
```

Pratik Joshi



MODULES

Pratik Joshi



Custom Module

It is a file containing python codes and functions, same as a code library.

- To create a module write python code a file and save as .py extension.
- To use that module : import modulename
- To create alias of module: import modulename as mn, here mn is alias of that module

test_module.py

=====

```
x=[5,75,80]
```

```
def test():
```

```
    print("Hi from module")
```

Use of Module

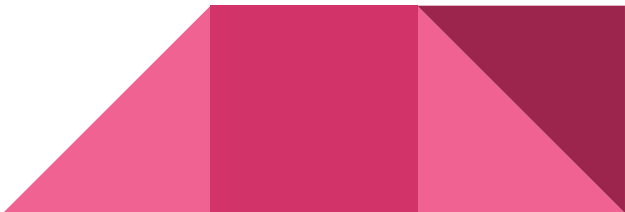
=====

```
import test_module as tm
```

```
tm.test()          #Op: Hi from module
```

```
print(tm.x[0])     #Op: 5
```

Pratik Joshi



Built-in Modules

Python has several built-in modules, which can be imported as per requirement.

Math Module

```
import math
```

```
print(math.sqrt(625))  
print(math.ceil(2.4))  
print(math.floor(2.4))  
print(math.pi)  
print(math.e)  
print(math.inf)
```

```
#Op:25 (Square root of a number)
```

```
#Op:3 (Round the value upwards to an integer)
```

```
#Op:2 (Round the value downward to an integer)
```

```
#Return the value of mathematical constant pi i.e 3.14159...
```

```
#Return the value of mathematical constant e i.e 2.7182818...
```

```
#Return infinity inf
```



Random Module

```
import random
print(random.random())           #Return random floating number between 0 and 0.0
print(random.random())
print(random.randint(1,10))      #Return random integer between 1 to 10
print(random.randrange(1,20,2))  #Return random integer between 1 to 20 with step 2
```

Time Module

```
import time
print(time.time())               #Return Epoch time: 1621249083.0091047
print(time.ctime())              #Return Time Mon May 17 16:28:03 2021
epoch = 1621249051.8006983
print(time.ctime(epoch))         #Returns Human readable time
print("First line")
time.sleep(2)                    #Delays for 2 seconds
print("Second line")
```

Sys Module

```
import sys
```

```
print(sys.version)      #Return version of python interpreter
print(sys.path)         #Return system python path
print(sys.modules)     #Return name of existing python modules
print(sys.builtin_module_names) #Return name of built-in modules
print(sys.copyright)    #Return copyright informations
print(sys.exit)        #Exit the current python interpreter
```

Command Line Arguments:

```
print(len(sys.argv))    #Return total numbers of command line arguments
print(sys.argv[1])     #Return 1st command line argument
print(sys.argv[2])     #Return 2nd command line argument
```


```
python test.py Hello 5
```

Here, Total Argument: 3, 1st Argument: Hello, 2nd Argument: 5

PIP

- PIP is package installer for python.
- Package is container of all the files needed for a module.
- To check pip version: `pip --version`
- Install pip from <https://pypi.org/project/pip/>
- From python version 3.4 and later pip included by default.
- Install a package using pip: `pip install pandas`
- Uninstall a package: `pip uninstall pandas`
- To check list of installed package: `pip list`

Hands-On 4

1. Develop a script for a guess-lottery game.
 2. Develop a menu driven program to calculate area of a circle, volume of cylinder, volume of cone by taking required input from user. (Use Function)
 3. Take state name as input from command line argument and show details of that state (CM Name, Capital, Population) using a dictionary.
 4. Compare time of execution between factorial of 899 with and without function recursion.
 5. Develop a quiz game like KBC which will ask user at least 10 questions with 4 options and show the score after closure of game. The game should stop if user gives any wrong answer.
 6. Enable Login, Signup and Logout feature using Function.
- 

EXCEPTION HANDLING

Pratik Joshi



Error and Exception

- Error caused by not following proper syntax of any language. This is also called syntax error.
- A python program terminates as soon as it encounters an unhandled error.
- Errors that occur at runtime even if syntax is correct are called Exception or logical error.
- Whenever these types of runtime errors occur, Python creates an exception object. If not handled properly, it prints the error and some details about it.

```
>>> 1/0
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ZeroDivisionError: division by zero
```

- Built-in Exception in python : `print(dir(locals()['__builtins__']))`

try...except Block

A try...except block is used to handle exception in python.

Syntax:

```
try:  
    statement1  
    statement2  
except:  
    handle the exception
```

Example:

```
try:  
    1/0  
except:  
    print("Can't divide by zero")
```

Pratik Joshi



Execution of try..except

- First try clause is executed i.e. the code between try and except clause.
- If there is no exception, then only try clause will run, except clause is finished.
- If any exception occurred, try clause will be skipped and except clause will run.
- If any exception occurs, but the except clause within the code doesn't handle it, it is passed on to the outer try statements. If the exception left unhandled, then the execution stops.
- A try statement can have more than one except clause.

Catching Specific Exception:

```
try:
    print(x)
except NameError:
    print("x is not defined")      #If NameError
except:
    print("Something else went wrong")  #If Some other error
```

Else and Finally Block

- Else block will be executed if no exception occurs.
- Finally block will be executed regardless of error and will be executed ultimately.

```
try:
    1/0
except:
    print("Error occurred")
else:
    print("No error")
finally:
    print("try...except finishes")
```

Printing the Exception:

```
try:
    1/0
except Exception as e:
    print("Error: ",e)    #division by zero
```

Pratik Joshi



Raise An Exception

- Python enable developer to create their own exception by using **raise** keyword.
- This can also be called user defined Exception.

Example:

```
try:
    num=input("Enter a positive integer: ")
    if int(num)<0:
        raise ValueError
except ValueError :
    print("Sorry! Number is negative or Zero")

num=input("Enter a Number: ")
if int(num)!=9:
    raise Exception("Number is not my Luck Number")
```