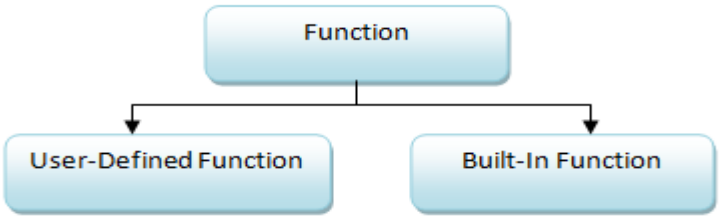
LECTURE THREE

Function

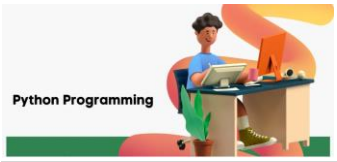
- function is a group of related statements that performs a specific task.
- Functions help break our program into smaller chunks. it avoids repetition and makes the code reusable.
- As our program grows larger and larger, functions make it more organized and manageable.



Python Built-in Functions

Python has several functions that are readily available for use. These functions are called built-in functions.

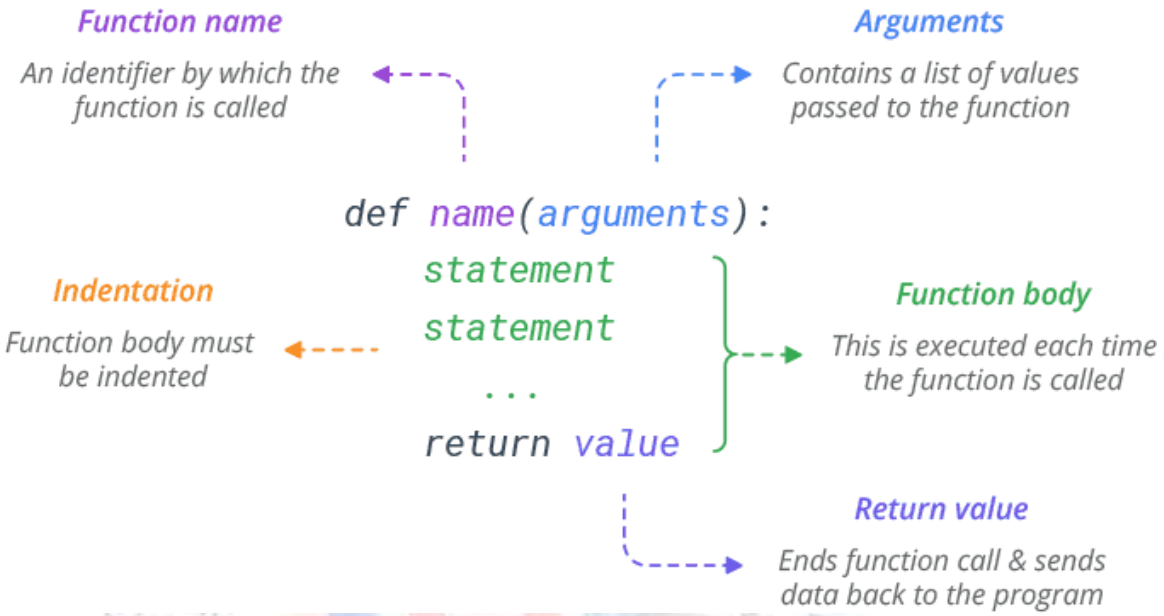
Built-in Functions in Python						
abs()	classmethod()	filter()	id()	max()	property()	str()
all()	compile()	float()	input()	memoryview()	range()	sum()
any()	complex()	format()	int()	min()	repr()	super()
ascii()	delattr()	frozenset()	isinstance()	next()	reversed()	tuple()
bin()	dict()	getattr()	issubclass()	object()	round()	type()
bool()	dir()	globals()	iter()	oct()	set()	vars()
bytearray()	divmod()	hasattr()	len()	open()	setattr()	zip()
bytes()	enumerate()	hash()	list()	ord()	slice()	__import__()
callable()	eval()	help()	locals()	pow()	sorted()	
chr()	exce()	hex()	map()	print()	staticmethod()	



User Defined Function

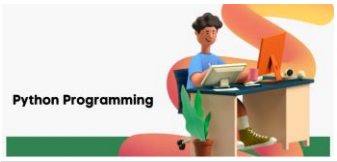
Functions that we define ourselves to do certain specific task are referred as user-defined functions.

- Syntax of Function



Advantages of user-defined functions

1. Abstraction: User-defined functions help to decompose a large program into small segments which makes program easy to understand, maintain and debug.
2. Reusability: If repeated code occurs in a program. Function can be used to include those codes and execute when needed by calling that function.
3. Organizing: Programmers working on large project can divide the workload by making different functions.



Types of user-defined Functions

a- Return function vs non return function: -

Non Return function	Return function

def fun():
  print('hello')
fun()
	def fun():
	  return 'hello'
	print(fun())

b- Arguments: -

Non Return function	Return function

def fun(x):
  print(x*2)
fun(3)
fun(2.4)
	def fun(x):
	  return x*2
	print(fun(3))
	print(fun(2.4))

c- Default Arguments: -

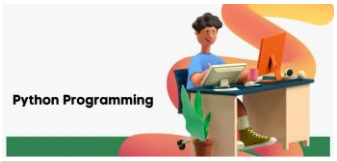
Non Return function	Return function

def fun(x=5):
  print(x*2)
fun(3)
fun(2.4)
fun()
	def fun(x=5):
	  return (x*2)
	print(fun(3))
	print(fun(2.4))
	print(fun())

d- Unknown number of Arguments: -

Non Return function	Return function

def fun(*x):
  print(sum(x))
fun(1,2,3)
fun()
	def fun(*x):
	  return sum(x)
	print(fun(1,2,3))
	print(fun())



Example(1): Write Python Program to check the number is positive or negative by using function.

Non Return function	Return function

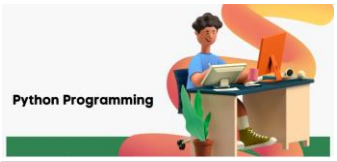
def check(x):
  if x<0:
  print('neg')
  else:
  print('pos')
n = int(input("number = "))
check(n)
	def check(x):
	  if x<0:
	  return ('neg')
	  else:
	  return('pos')
	n = int(input("number = "))
	print(check(n))

Example(2): Write Python Program to calculate the following equation by using function.

$$Y= x + x^3 + x^5 + \ldots\ldots x^n$$

Non Return function	Return function

def calculate(x, n):
  result = 0
  for i in range(1, n+1, 2):
  result += x**i
  print(result)
x = int(input(" x= "))
n = int(input("n= "))
calculate(x, n)
	def calculate(x, n):
	  result = 0
	  for i in range(1, n+1, 2):
	  result += x**i
	  return result
	x = int(input(" x= "))
	n = int(input("n= "))
	print(calculate(x, n))



Example(3): Write Python Program to calculate area and perimeter of the square using function.

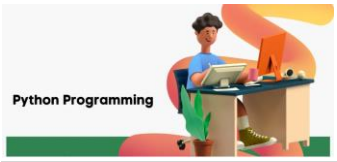
Non Return function	Return function

def area(a):
  print('area = ', a*a)
def perimeter(a):
  print('perimeter = ', a*4)
 	
x=int(input("Enter the Slide : "))
area(x)
perimeter(x)
	def area(a):
	  return(a*a)
	def perimeter(a):
	  return (a*4)
	x=int(input("Slide ="))
	print('area = ',area(x))
	print('perimeter =',perimeter(x))

Example(4): Write Python Program to check if the number is prime or not using Return function

```
def check(number):
    if number <= 1:
        return False
    elif number == 2:
        return True
    elif number % 2 == 0:
        return False
    else:
        for i in range(3, int(number**0.5) + 1, 2):
            if number % i == 0:
                return False
        return True

num = int(input("Enter a number "))
if check(num)==True:
    print(num, "is a prime number.")
else:
    print(num, "is not a prime number.")
```

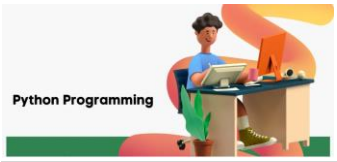


Recursive Function

- The process in which a function calls itself is called Recursive Function.
- Advantages of using Recursive Function:
  1. Complicated function can be split down into smaller sub-problems.
  2. Sequence creation is simpler through recursion than utilizing any nested iteration.
  3. function code looks simple and effective.
- Disadvantages of using Recursive Function:
  1. A lot of memory and time is taken through recursive calls.
  2. Recursive functions are challenging to debug.
-

Syntax:

```
def func(): <--
            |
            | (recursive call)
            |
    func() ----
```



Example(5): write python program to Print numbers from 1 to n without the help of loops(Recursive Function)

```
def printNos(n):
    if n > 0:
        printNos(n - 1)
        print(n, end=" ")

n = int(input(" n="))
printNos(n)
```

Example(6): write python program to find factorial of given number using Recursive Function

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n-1)

n = int(input(" number="))
print("Factorial of", n, "is",
factorial(n))
```