Functions

- is a set/block of code that can be used again and again whenever a function is called.

Two types of functions :

1. In-built / Predefined functions
2. User defined functions

In [1]:
```
def greet():       # defining a function
    print("Hello")
```

In [2]:
```
greet()
```

```
Hello
```

In [4]:
```
def greet(name):       # defining a function
    print("Hello",name)
```

In [6]:
```
greet("Rahul")
```

```
Hello Rahul
```

In [9]:
```
# write a function that will take one number from the user
# and print whether it is even or odd

def eveodd(num):
    if num%2==0:
        print("The number is even")
    else:
        print("The number is odd")

eveodd(int(input()))    # giving the user input when function is called
```

```
45
The number is odd
```

In [14]:
```
a = 100
def eveodd():
    print(a)
    x = int(input())
    if x%2==0:
        print("The number is even")
    else:
        print("The number is odd")

eveodd()
```

```
100
34
The number is even
```

In [15]:
```
x    # is a local variable so it cannot be accessed outside the function
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\BHUPEN~1\AppData\Local\Temp/ipykernel_17228/1561318907.py in <module>
----> 1 x    # is a local variable so it cannot be accessed outside the functio
n

NameError: name 'x' is not defined
```

Global Variable : Which are accessible everywhere
Local Variable : Whose scope is limited only to a particular function

In [51]:
```
# write a function to check prime & not prime
flag = 0
def prime(num):
    global flag
    for i in range(2,num): # 2,3,4
        if num%i==0:
            flag = 1
            break
        else:
            flag = 0

    if flag == 0:
        print("prime")
    else:
        print('Not prime')


```

In [52]:
```
prime(2)
```

```
prime
```

In [65]:
```
# Method: 2

def prime(num):
    for i in range(2,num):
        if num%i == 0:
            print('Not prime')
            break
    else:
        print(" prime")

```

In [66]:
```
prime(5)
```

```
 prime
```

In [ ]:
```
# Write a function to check wheather a string is palindrome or not ?
```

In [70]:
```
def palindrome(word):
    if word == word[::-1]:
        print("Palindrome")
    else:
        print("Not Palindrome")

palindrome('nitin')
```

```
Palindrome
```

In [71]:
```
palindrome('python')
```

```
Not Palindrome
```

Global & Local Variable

In [73]:
```
a = 10    # global variable
def sample():
    print(a)

sample()
```

```
10
```

In [75]:
```
a = 10    # global variable
def sample():
    a = 50
    a = a+1
    print(a)

sample()
```

```
51
```

In [76]:
```
print(a)
```

```
10
```

In [77]:
```
a = 10    # global variable
def sample():
    global a
    a = a+1
    print(a)

sample()
```

```
11
```

In [78]:
```
a
```

Out[78]:
```
11
```

Return Keyword

In [80]:
```
a = print(20)

print(a)
```

```
20
None
```

In [82]:
```
b = palindrome("Apple")

print(b)
```

```
Not Palindrome
None
```

In [83]:
```
def palin(x):
    if x == x[::-1]:
        return "palindrome"
    else:
        return "Not Palindrome"
```

In [87]:
```
a = palin('nitin')
print(a)
```

```
palindrome
```

In [91]:
```
def sample():
    return 1
    return 2
    return 3


sample()
```

Out[91]: 1

In [92]:
```
def sample():
    print( 1)
    print( 2)
    print( 3)


sample()
```

```
1
2
3
```

In [93]:
```
def sample():
    print("Hello")
    return
    print("Bye")


sample()
```

```
Hello
```

In [102]:
```
def sumofnums(numbers):
    temp = 0
    for i in numbers:
        temp+=i
    return temp

summ = sumofnums([2,3,4,4])
```

In [103]:
```
summ
```

Out[103]: 13

In [2]:
```
# write a funtion which accepts a string as a argument and displays total nu

def lower_count(word):
    count = 0
    for i in word:
        if i.islower():
            count+=1
    return count

lower_count('AbcDef')
```

Out[2]: 4

In [4]:
```
def up_low(word):
    countU = 0
    countL = 0
    for i in word:
        if i.islower():
            countL+=1
        else:
            countU+=1
    return countU,countL

up_low('AbcDef')
```

Out[4]: (2, 4)

In [6]:
```
'a'.isnumeric()
```

Out[6]: False

Patterns

In [13]:
```
# *
# **
# ***

for r in range(0,3):
    for c in range(0,r+1):
        print('*', end="")
    print()
```

```
*
**
***
```

In [14]:
```
def triangle(row):
    for r in range(0,row):
        for c in range(0,r+1):
            print('*', end="")
        print()
```

In [18]:
```
triangle(5)
```

```
*
**
***
****
*****
```

In [22]:
```
# 1
# 23
# 456

def num_triangle(row):
    n = 0
    for r in range(0,row):
        for c in range(0,r+1):
            print(n, end="")
            n+=1
        print()

num_triangle(5)
```

```
0
12
345
6789
1011121314
```

In [26]:
```
# 1
# 22
# 333
# 4444

def pattern(row):
    for r in range(0,row):
        for j in range(0,r+1):
            print(r+1,end = "")
        print()

pattern(5)
```

```
1
22
333
4444
55555
```

In [27]:
```
# trick

for i in range(1,5):
    print('*'*i)
```

```
*
**
***
****
```

In [ ]:
```

```