In [2]:

```
# 1)Write a function called factorial that takes a positive integer as input and returns
def factorial():
    n=int(input("Enter the Number :"))
    fact=1
    if n<0:
        print("the factorial does not have negative number")
    else:
        for i in range(1,n+1):
            fact=fact*i
        print(f"The factorial of {fact}")
factorial()
```

```
Enter the Number :3
The factorial of 6
```

In [3]:

```
# 2)Write a function called is_palindrome that takes a string as input and returns True i
def is_palindrome():
    n="RACECAR"
    reverse=n[::-1]
    if n==reverse:
        print("Yes")
    else:
        print("No")
is_palindrome()
```

```
Yes
```

In [4]:

```
# 3)Write a function called remove_duplicates that takes a list as input and returns a ne
def remove_duplicates():
    n=[1,2,3,4,5,5,6,6,7,7,8,7,8,9,9,1,11]
    new_list=[]
    for i in n:
        if i not in new_list:
            new_list.append(i)
    print(new_list)
remove_duplicates()
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 11]
```

In [5]:

```
# 4)Write a function called common_elements that takes two lists as input and returns a n
def common_elements():
    n=[1,2,3,4,5]
    n1=[5,6,7,8,9]
    l=set(n)
    l1=set(n1)
    new_list=list(l & l1)
    print(new_list)
common_elements()
```

```
[5]
```

In [38]:

```
# 5)Write a function called binary_search that takes a sorted list and a target value as
# and returns the index of the target value in the list, or -1 if the value is not found.
pos=-1
def binary_search(list,n):
    global pos
    l=0
    u=len(list)-1
    while l<=u:
        mid=(l+u)//2
        if list[mid]==n:
            pos=mid
            return True
        else:
            if list[mid]<n:
                l=mid+1
            else:
                u=mid-1
    return False
list=[4,5,6,7,8,45,99]
n=4
if binary_search(list,n):
    print("found at ",pos)
else:
    print("Not Found")
```

```
found at  0
```

In [44]:

```
# 6)Write a function called matrix_transpose that takes a 2D matrix (a list of lists) as
# and returns the transpose of the matrix.
def Transpose():
    matrix=[[1,2,3],
            [4,5,6]]
    result=[[0,0],
           [0,0],
           [0,0]]
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            result[j][i]=matrix[i][j]
    return result

Transpose()
```

Out[44]:

```
[[1, 4], [2, 5], [3, 6]]
```

In [47]:

```
#  7)Write a function called find_missing_number that takes a list of integers from 1 to
# (inclusive) with one number missing, and returns the missing number.
def find_missing_number():
    n=[1,2,4,5,6,7,8]
    l=len(n)
    m=l+1
    total=m*(m+1)//2
    return total-sum(n)
find_missing_number()
```

Out[47]:

```
3
```

In [49]:

```
# 8)Write a function called flatten_list that takes a nested list as input and returns a
def flatten_list():
    n=[[1],[1,2,3],[4,5,6]]
    new_list=[]
    for sub_list in n:
        for num in sub_list:
            new_list.append(num)
    print(new_list)
flatten_list()
```

```
[1, 1, 2, 3, 4, 5, 6]
```

In [61]:

```
# 9)Write a function called capitalize_words that takes a sentence as input and returns a
def capitalze_word():
    word="python is a high-level, general-purpose programming language"
    for words in word.split():
        n=words.capitalize()
        print(n,end="\n")
capitalze_word()
```

```
Python
Is
A
High-level,
General-purpose
Programming
Language
```

In [64]:

```
# 10)Write a function called is_anagram that takes two strings as input and returns True
# two strings are anagrams (contain the same characters in any order), and False
# otherwise.
def is_anagram(st1,str2):
    if (sorted(str1)==sorted(str2)):
        return True
    else:
        return False
str1="king"
str2="care"
is_anagram(str1,str2)
```

Out[64]:

```
False
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: