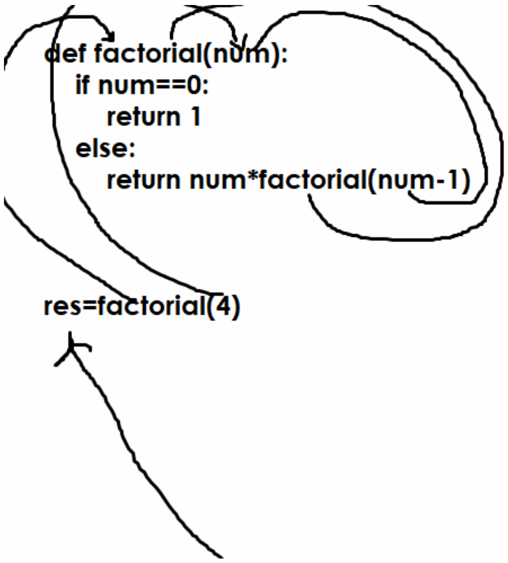
Example:

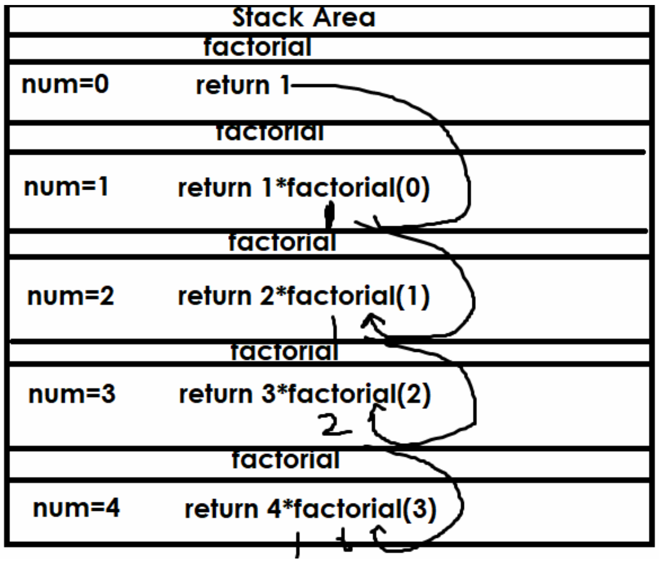
```
def factorial(num):
    if num==0:
        return 1
    else:
        return num*factorial(num-1)



number=int(input("Enter any number "))
result=factorial(number)
print(f'Factorial of {number} is {result}')
```

Output

```
Enter any number 4
Factorial of 4 is 24

Enter any number 3
Factorial of 3 is 6
```



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

res=factorial(4)
```



Example:

```
def print_num(num):
    if num>0:
        print_num(num-1)
    print(num)


print_num(3)
```

Output

```
0
1
2
3
```

Example:

```
def sum_num(num):
    if num==0:
        return 0
    else:
        return num+sum_num(num-1)



result=sum_num(5)
print(result)
```

Output

```
15
```

Example:

```
# sum of digits of input number using recursion

def sum_digits(num):
    if num==0:
        return 0
    else:
        return (num%10)+sum_digits(num//10)


number=int(input("Enter any number "))
res=sum_digits(number)
```

```
print(res)
```

Output
```
Enter any number 123
6
```

bytes and bytearray

bytes data type

Bytes objects are immutable sequences of single bytes. Since many major binary protocols are based on the ASCII text encoding, bytes objects offer several methods that are only valid when working with ASCII compatible data and are closely related to string objects

Bytes is an immutable sequence data type, after creating bytes object changes cannot done. Bytes object does not support mutable operations like,

1. append()
2. extend()
3. remove()
4. del
5. sort()
6. insert()
7. pop()

How to create bytes object?

Single quotes: `b'still allows embedded "double" quotes'`
Double quotes: `b"still allows embedded 'single' quotes"`
Triple quoted: `b'''3 single quotes'''`, `b"""3 double quotes"""`
Example:
```
str1="ABC"
print(str1,type(str1))
b1=b'ABC'
print(b1,type(b1))

for x in str1:
    print(x)
```

```
for y in b1:
    print(y)
```

Output

```
ABC <class 'str'>
b'ABC' <class 'bytes'>
A
B
C
65
66
67
```

In addition to the literal forms, bytes objects can be created in a number of other ways:

- A zero-filled bytes object of a specified length: `bytes(10)`
- From an iterable of integers: `bytes(range(20))`
- Copying existing binary data via the buffer protocol: `bytes(obj)`

Example:

```
>>> b1=bytes(10)
>>> print(b1)
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
>>> b2=b'\x41'
>>> print(b2,type(b2))
b'A' <class 'bytes'>
>>> b2[0]
65
>>> for x in b1:
    print(x)


0
0
0
0
0
0
0
```

```
0
0
0
```

Example

```
>>> b3=bytes([97,98,99,100,101])
>>> print(b3,type(b3))
b'abcde' <class 'bytes'>
>>> for x in b3:
    print(x,end=' ')


97 98 99 100 101
>>> b4=bytes(range(65,71))
>>> print(b4,type(b4))
b'ABCDEF' <class 'bytes'>
>>> for x in b4:
    print(x,end=' ')

...
65 66 67 68 69 70
>>> b5=bytes([1.5,2.5])
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    b5=bytes([1.5,2.5])
TypeError: 'float' object cannot be interpreted as an integer

>>> b6=bytes(range(10,21))
>>> print(b6)
b'\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14'
>>> b7=bytes(range(65,68))
>>> print(b7)
b'ABC'
>>> for value in b6:
...     print(value,end=' ')
...
...
10 11 12 13 14 15 16 17 18 19 20
```

Example:

```
>>> b1=bytes(range(65,70))
>>> b2=bytes(b1)
>>> print(b1,type(b1))
b'ABCDE' <class 'bytes'>
>>> print(b2,type(b2))
b'ABCDE' <class 'bytes'>
```

Example:

```
>>> b1=bytes("ABC","utf-8")
>>> print(b1)
b'ABC'
>>> for value in b1:
    print(value,end=' ')


65 66 67
```

Example:

```
>>> str1="PYTHON"
>>> b1=str1.encode()
>>> print(str1,type(str1))
PYTHON <class 'str'>
>>> print(b1,type(b1))
b'PYTHON' <class 'bytes'>
>>> for ch in str1:
    print(ch,end=' ')


P Y T H O N
>>> for x in b1:
...     print(x,end=' ')
...
...
80 89 84 72 79 78
>>> str2=b1.decode()
>>> print(str2,type(str2))
PYTHON <class 'str'>
```

Bytearray

Bytearray is a mutable sequence data type

Bytearray allow modifying content after creation using mutable operations.
Bytearray support the following mutable operations.

1. append()
2. extend()
3. remove()
4. insert()
5. pop()
6. clear()
7. sort()
8. del

bytearray object is created using bytearray type

Creating an empty instance: bytearray()
Creating a zero-filled instance with a given length: bytearray(10)
From an iterable of integers: bytearray(range(20))
Copying existing binary data via the buffer protocol: bytearray(b'Hi!')

Example:

```
>>> b1=bytearray()
>>> print(b1,type(b1))
bytearray(b'') <class 'bytearray'>
>>> b1.append(65)
>>> b1.append(66)
>>> b1.append(67)
>>> print(b1)
bytearray(b'ABC')
>>> b1.remove(65)
>>> print(b1)
bytearray(b'BC')
>>> b1[0]=65
>>> print(b1)
bytearray(b'AC')
>>> b2=bytearray(range(65,70))
>>> print(b2)
bytearray(b'ABCDE')
>>> b3=bytearray("ABCDE",'utf-8')
>>> print(b3,type(b3))
bytearray(b'ABCDE') <class 'bytearray'>
```

```
>>> b4=bytearray([65,66,67])
>>> print(b4,type(b4))
bytearray(b'ABC') <class 'bytearray'>
>>> b5=bytearray(b4)
>>> print(b5,type(b5))
bytearray(b'ABC') <class 'bytearray'>
```