In [1]:
```
a='sathi'
a
```

Out[1]:
```
'sathi'
```

In [5]:
```
a='ssaathiii'
str = "sample string"
def getMaxOccuringChar(str):
print ("Max occurring character is " + getMaxOccuringChar(str))
```

```
  File "<ipython-input-5-e27b4a74edb5>", line 4
    print ("Max occurring character is " + getMaxOccuringChar(str))
        ^
IndentationError: expected an indented block
```

In [7]:
```
#13
ASCII_SIZE = 256

def getMaxOccuringChar(str):

    count = [0] * ASCII_SIZE


    max = -1
    c = ''


    for i in str:
        count[ord(i)]+=1;

    for i in str:
        if max < count[ord(i)]:
            max = count[ord(i)]
            c = i

    return c


str = "sample string"
print ("Max occurring character is " + getMaxOccuringChar(str))
```

```
Max occurring character is s
```

In [10]:
```
#14
NO_OF_CHARS = 256


def toMutable(string):
    List = []
    for i in string:
        List.append(i)
    return List


def toString(List):
    return ''.join(List)


def removeDups(string):
    bin_hash = [0] * NO_OF_CHARS
    ip_ind = 0
    res_ind = 0
    temp = ''
    mutableString = toMutable(string)


    while ip_ind != len(mutableString):
        temp = mutableString[ip_ind]
        if bin_hash[ord(temp)] == 0:
            bin_hash[ord(temp)] = 1
            mutableString[res_ind] = mutableString[ip_ind]
            res_ind+=1
        ip_ind+=1


    return toString(mutableString[0:res_ind])


string = "geeksforgeeks"
print (removeDups(string))
```

```
geksfor
```

In [11]:
```
#15
my_tuple = (1, 2, 3,4)
print(my_tuple)
```

```
(1, 2, 3, 4)
```

In [12]:
```
list(my_tuple)
```

Out[12]:
```
[1, 2, 3, 4]
```

In [16]:
```
#16
listx = list(my_tuple)
#use different ways to remove an item of the list
listx.remove(2)
print(listx)
```

```
[1, 3, 4]
```

In [17]:
```
#17
tuple(listx)
```

Out[17]:
```
(1, 3, 4)
```

In [1]:
```
#24
num = 7

# uncomment to take input from the user
#num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
       factorial = factorial*i
   print("The factorial of",num,"is",factorial)
```

```
The factorial of 7 is 5040
```

In [2]:
```
#25
# Program to display the Fibonacci sequence up to n-th term where n is prov

# change this value for a different result
nterms = 10

# uncomment to take input from the user
#nterms = int(input("How many terms? "))

# first two terms
n1 = 0
n2 = 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence upto",nterms,":")
   while count < nterms:
       print(n1,end=' , ')
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

```
Fibonacci sequence upto 10 :
0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 ,
```

In [28]:
```
#26
inp=input("Enter a number:")                    # input from user
sum=0
if inp>0:
    for i in str(inp):
        fact=1
if int(i)!=0:
    for j in range(1,int(i)+1):
        fact=fact*j                 # factorial calculation
    sum=sum+fact
if sum==inp:                              # checking if strong number
        print("Given number is strong number")
else:
        print("Given number is not strong number")
```

```
Enter a number:145->

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call last
)
<ipython-input-28-df35582e8ede> in <module>()
      1 inp=input("Enter a number:")                    # input from user
      2 sum=0
----> 3 if inp>0:
      4     for i in str(inp):
      5             fact=1

TypeError: '>' not supported between instances of 'str' and 'int'
```

In [34]:
```
inp=input("Enter a number:")                    # input from user
sum=0
if inp>0:
    for i in str(inp):
        fact=1
        if int(i)!=0:
    for j in range(1,int(i)+1):
            fact=fact*j                 # factorial calculation
    sum=sum+fact
    if sum==inp:                              # checking if strong number
        print ("Given number is strong number")
    else:
        print ("Given number is not strong number" )
else:
    print ("Given number is not strong number")
```

```
  File "<ipython-input-34-c3306cee22c8>", line 7
    for j in range(1,int(i)+1):
    ^
IndentationError: expected an indented block
```

In [35]:
```
a = 0
while a < 10:
    a = a + 1
    if a > 5:
        print (a,">",5)
    elif a <= 7:
        print (a,"<=",7)
    else:
        print ("Neither test was true")
```

```
1 <= 7
2 <= 7
3 <= 7
4 <= 7
5 <= 7
6 > 5
7 > 5
8 > 5
9 > 5
10 > 5
```

In [41]:
```
#22
in_put=int(input())
if in_put <= 10:
    print("less then 10")
else:
    print("not ok")
```

```
6
less then 10
```

In [43]:
```
#29
def pypart(n):

    # outer loop to handle number of rows
    # n in this case
    for i in range(0, n):

        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):

            # printing stars
            print("* ",end="")

        # ending line after each row
        print("\r")

# Driver Code
n = 5
pypart(n)
```

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

In [44]:
```
#30
def pypart2(n):

    # number of spaces
    k = 2*n - 2

    # outer loop to handle number of rows
    for i in range(0, n):

        # inner loop to handle number spaces
        # values changing acc. to requirement
        for j in range(0, k):
            print(end=" ")

        # decrementing k after each loop
        k = k - 2

        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):

            # printing stars
            print("* ", end="")

        # ending line after each row
        print("\r")

# Driver Code
n = 5
pypart2(n)
```

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

In [45]:
```
#31
def triangle(n):

    # number of spaces
    k = 2*n - 2

    # outer loop to handle number of rows
    for i in range(0, n):

        # inner loop to handle number spaces
        # values changing acc. to requirement
        for j in range(0, k):
            print(end=" ")

        # decrementing k after each loop
        k = k - 1

        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):

            # printing stars
            print("* ", end="")

        # ending line after each row
        print("\r")

# Driver Code
n = 5
triangle(n)
```

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

In [ ]: