In [62]:

```
#print below pattern
n = int(10)
alphabets = "abcdefghijklmnopqrstuvwxyz"

data =[]
for i in range(n):
    data.append(alphabets[i])

items = list(range(n))
items = items[:-1]+items[::-1]

for i in items:
    start_index = i+1
    original =data[-start_index:]
    reverse = original[::-1]
    row =  "-".join(reverse+original[1:])
    column = n*4-3
    row = row.center(column, "-")
    print(row)
```

```
------------------j------------------
----------------j-i-j----------------
--------------j-i-h-i-j--------------
------------j-i-h-g-h-i-j------------
----------j-i-h-g-f-g-h-i-j----------
--------j-i-h-g-f-e-f-g-h-i-j--------
------j-i-h-g-f-e-d-e-f-g-h-i-j------
----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----
--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--
j-i-h-g-f-e-d-c-b-a-b-c-d-e-f-g-h-i-j
--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--
----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----
------j-i-h-g-f-e-d-e-f-g-h-i-j------
--------j-i-h-g-f-e-f-g-h-i-j--------
----------j-i-h-g-f-g-h-i-j----------
------------j-i-h-g-h-i-j------------
--------------j-i-h-i-j--------------
----------------j-i-j----------------
------------------j------------------
```

In [82]:

```
#Any number, say n is called an Armstrong number if it is equal to the sum of its digits, w
#For example: 153=13+53+33

n=int(153)
length= len(str(abs(n)))
sum = 0
temp = n
while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

if(sum == n):
    print('True')
else:
    print('False')
```

```
True
```

In [86]:

```
#Compute and display Fibonacci series upto n terms where n is a positive integer entered by
n=int(5)
count = 0
num1, num2 = 0,1
while count < n::
    print(num1)
    final_num = num1 + num2
    num1 = num2
    num2 = final_num
    count = count+1
```

```
0
1
1
2
3
```

In [124]:

```
#Write python code to find the sum of prime numbers from 2 to n where n is a positive integ
n = int(10)
sum = 2
for num in range(2, n + 1):
    i = 2
    for i in range(2, num):
        if (int(num % i) == 0):
            i = num
            break;
    if i is not num:
        sum += num

print(sum)
```

```
17
```

In [1]:

```
#Sanjay had m rupees and cost of each chocolate was c rupees. Shopkeeper gave away one choc
n = "15, 2, 3, 2"
number = n.split(',')
m = int(number[0]) #15
c = int(number[1]) #2
w = int(number[2]) #3
k = int(number[3]) #1

chocolates = m//c
wrapper = m//c

while(wrapper// w !=0):
    chocolates = chocolates + (wrapper//w)*k
    wrapper = (wrapper//w)*k + (wrapper%w)
print(chocolates)
```

```
17
```

In [2]:

```
#You are given a row of Lego Blocks consisting of n blocks. All the blocks given have a squ
#From the row of Lego blocks, you can only pick up either the leftmost or rightmost block.
#Print "Possible" if it is possible to stack all n cubes this way or else print "Impossible
sides = [5 ,4, 2, 1, 4 ,5]
n = len(sides)
my_stack = []
i = 0

while(i < n):
    if sides[i] >= sides[n-1]:
        my_stack.append(sides[i])
        i = i+1
    else:
        my_stack.append(sides[n-1])
        n = n - 1

flag = 0
i = 1
while i < len(my_stack):
    if(my_stack[i] > my_stack[i - 1]):
        flag = 1
    i += 1

if (not flag) :
    print ("Possible")
else :
    print ("Impossible")
```

```
Possible
```

In [3]:

```
#A pascal's triangle is a very interesting mathematical concept.
from math import factorial
n = 6
for i in range(n):
    for j in range(n-i + 1):
        print(end=" ")

    for j in range(i+1):
        # nCr = n!/((n-r)!*r!)
        print(factorial(i)//(factorial(j)*factorial(i-j)), end=" ")

    print()
```

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
```

In [4]:

```
#A pascal's triangle is a very interesting mathematical concept.The output will contain 1 l
#representing the nth row of Pascal's triangle.
from math import factorial
n=6
n = n - 1
line = [1]

for k in range(max(n ,0)):
    line.append(int(line[k]*(n-k)/(k+1)))

print(line)
```

```
[1, 5, 10, 10, 5, 1]
```

In [5]:

```
#Write a Python program to divide a given list into chunks of size k.
import ast
input_list = [[1,2,3,4,5,6,7,8,9],3]
lis=input_list[0]
k=input_list[1]

for i in range((len(lis)+ k-1) // k):
    print(lis[i*k:(i+1) *k], end ="\n")
```

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

In [7]:

```
#Given a list of numbers, find the second largest number in the list.
#Note: There might be repeated numbers in the list. If there is only one number present in

input_list = [7, 2, 0, 9, -1, 8]
my_list =[]
# Write your code here
input_list2 = list(set(input_list))
if len(input_list2)== 1:
    print('not present')
else:
    for i in input_list2:
        if i not in my_list:
            my_list.append(i)
    print(sorted(my_list)[-2])
```

```
8
```

In [8]:

```
#Given a positive integer n. Print the pattern as shown in sample outputs.
n =5
MAX =100

size = 2 * n - 1
front = 0
back = size - 1
a = [[0 for i in range(MAX)]
        for i in range(MAX)]
while (n != 0):
    for i in range(front, back + 1):
        for j in range(front, back + 1):
            if (i == front or i == back or
                j == front or j == back):
                a[i][j] = n
    front += 1
    back -= 1
    n -= 1

for i in range(size):
    for j in range(size):
        print(a[i][j], end = '')
    print()
```

```
555555555
544444445
543333345
543222345
543212345
543222345
543333345
544444445
555555555
```

In [158]:

```
#Given two strings, one of the strings will contain an extra character. Find the extra char
#The number of all the other characters in both the strings will be the same. Check the sam
#clarification.

string1 = 'abcd'
string2 = 'cedab'

if(len(string1) > len(string2)):
    string_in1 = string1
    string_in2 = string2
else:
    string_in1 = string2
    string_in2 = string1

for i in string_in1:
    if(i not in string_in2):
        print(i)
```

```
e
```

In [9]:

```
#INT if the input string is an integer and STR otherwise.
in_str ='14'

if in_str.isnumeric():
    print('INT')
else:
    print('STR')
```

```
INT
```

In [10]:

```
#While extracting data from different sources, often numeric values come in string format a
#like 1,000 or 23,321 and also sometimes with spaces in start and beginning of the string.
#For simplicity, we will consider only integer values imbedded with commas. You will take t
#print the cleaned integer without commas and spaces.
value='         3,213      '
value = value.strip()
spl1 = value.split(',')
result = int(''.join(spl1))
print(result)
```

```
3213
```

In [11]:

```
#The first element will be a string consisting of only alphabets which is taken from the di
#Output:A string denoting the password
input =['banana', 7]
password = input[0]
count = input[1]
final_pass = ""
list1=[]
for i in password:
    ch = i
    base = ord('a' if ch.islower() else 'A')
    x = chr((ord(ch) - base - count) % 26 + base)
    list1.append(x)

print("".join(list1))
```

```
utgtgt
```

In [13]:

```
#Write a program that computes the value of n+nn+nnn+nnnn+... nn...n ntimes with a given nu

n= 3
change = n
s = 0
for i in range(n):
    s += change
    change = change * 10 + n
print(s)
```

```
369
```

In [168]:

```
#You will be given a dictionary with keys as items and values as their prices. You have to
d = {'laptop': 30000, 'file': 50}
key_list = list(d.keys())
val_list = list(d.values())
cheapest_val = min(d.values())
position = val_list.index(cheapest_val)
cheapest_item_name = key_list[position]

cheapest_item_cost =str(cheapest_val)
print(cheapest_item_name +': '+cheapest_item_cost)
```

```
file: 50
```

In [173]:

```
#You will be given a list of repeated elements. You have to find the maximum distance betwe
input_arr = [1, 2, 3, 2, 5, 1, 2, 4, 6, 2, 7, 8, 6]
length = len(input_arr)
mp ={}
maxDict =0
for i in range(length):
    if input_arr[i] not in mp.keys():
        mp[input_arr[i]] = i
    else:
        maxDict = max(maxDict, i-mp[input_arr[i]])

print(maxDict)
```

```
8
```

In [184]:

```
#Your team is going for camping and you are taking a vote to decide what food to pack for d
#Everyone gets a vote and the food item that gets at least one more than half of the votes
#The input will contain a list of food items where each occurrence of an item represents on

votes =["pasta","pizza","pasta","paratha","paratha","paratha"]
num_of_votes = len(votes)
d = {}
for i in votes:
    if i not in d:
        d[i]=1
    else:
        d[i] = d[i]+1
winner = False
for val in d:
    if d[val]>num_of_votes//2:
        print(val)
        winner=True
        break
if not winner:
    print("NOTA")
```

```
NOTA
```

In [185]:

```
#Consider a nested dictionary as follows:
#{'Fruit': 1, 'Vegetable': {'Cabbage': 2, 'Cauliflower': 3}, 'Spices': 4}
#Your task is to flatten a nested dictionary and join the nested keys with the "_" characte
#{'Fruit': 1, 'Vegetable_Cabbage': 2, 'Vegetable_Cauliflower': 3, 'Spices': 4}
import ast,sys
input_str = sys.stdin.read()
input_dict = {'Fruit': 1, 'Vegetable': {'Cabbage': 2, 'Cauliflower': 3}, 'Spices': 4}
def flatten_dict(dd, separator='_', prefix=''):
    return { prefix + separator + k if prefix else k : v
             for kk, vv in dd.items()
             for k, v in flatten_dict(vv, separator, kk).items()
             } if isinstance(dd, dict) else { prefix : dd }


out1=list(flatten_dict(input_dict).keys())
out2=list(flatten_dict(input_dict).values())
out1.sort()
out2.sort()
print(out1)
print(out2)
```

```
['Fruit', 'Spices', 'Vegetable_Cabbage', 'Vegetable_Cauliflower']
[1, 2, 3, 4]
```

In [ ]: