In [1]:
```
print("hello wolrld")
```
```
hello wolrld
```

In [2]:
```
a=int(input("enter the number:"))
b=int(input("enter the number:"))
sum=a+b
print("the sum is:",sum)
```
```
enter the number:3
enter the number:5
the sum is: 8
```

In [3]:
```
x=int(input("enter the value of a"))
y=int(input("enter the value of b"))
x,y=y,x
print("after swapping:a=",x,"b=",y)
```
```
enter the value of a4
enter the value of b7
after swapping:a= 7 b= 4
```

In [4]:
```
km=float(input("enter the value in kilometers"))
cf=0.621371
m=km*cf
print(" km is equal to  miles",(km,m))
```
```
enter the value in kilometers3
 km is equal to  miles (3.0, 1.8641130000000001)
```

In [5]:
```
a=int(input("enter the value"))
if a>0:
    print("positive number")
elif a==0:
    print("zero")
else:
    print("negative number")
```
```
enter the value4
positive number
```

In [6]:
```
a=int(input("enter the year"))
if(a%4==0):
    print("leap year")
else:
    print("not a leap year")
```
```
enter the year2000
leap year
```

In [7]:
```
a=int(input("enter the low range"))
b=int(input("enter the higher range"))
for n in range(a,b+1):
    if n>1:
        for i in range(2,n):
            if(n%i)==0:
                break
            else:
                    print(n)
```
```
enter the low range1
enter the higher range4
3
```

In [11]:
```
a=int(input("enter the range:"))
n1,n2=0,1
c=0
if a<=0:
    print("enter the positive number")
elif a==1:
    print("fibonacci series upto",a,":")
    print(n1)
else:
    print("fibonacci sequence:")
    while c<a:
        print(n1)
        n=n1+n2
        n1=n2
        n2=n
        c+= 1
```
```
enter the range:4
fibonacci sequence:
0
1
1
2
```

In [ ]:

In [ ]:

In [13]:
```
a=10
b=7
c=5
if(a>b)and(a>c):
    largest = a
elif(b>a) and (b>c):
    largest = b
else:
    largest = c
print("the largest number is",largest)
```
```
the largest number is 10
```

In [3]:
```
a=int(input("enter a number"))
sum =0
temp=a
while(temp>0):
    digit =temp % 10
    sum+=digit**3
    temp//=10
if a==sum:
        print(a,"is an armstrong number")
else:
        print(a,"is not an armstrong number")
```
```
enter a number153
153 is an armstrong number
```

In [12]:
```
a=10
if a<0:
    print("enter a number")
else:
        sum=0
        while(a>0):
            sum+=a
            a-=1
print("the sum is",sum)
```
```
the sum is 55
```

In [6]:
```
num_rows = int(input("Enter the number of rows"));
k = 1
for i in range(0, num_rows):
    for j in range(0, k):
        print("* ", end="")
    k = k + 1
    print()
```
```
Enter the number of rows5
*
* *
* * *
* * * *
* * * * *
```

In [13]:
```
test_str = "GeeksForGeeks"
new_str = ""

for i in range(len(test_str)):
    if i != 2:
        new_str = new_str + test_str[i]
print ("The string after removal of i'th character : " + new_str)
```
```
The string after removal of i'th character : GeksForGeeks
```

In [9]:
```
str = "Engineering"
print ("Original string: " + str)
a= str[:2] +  str[3:]
print ("String after removal of character: " + a)
```
```
Original string: Engineering
String after removal of character: Enineering
```

In [14]:
```
a=[1,34,56,32,78,54,5]
for i in a:
    if (i%5==0):
        print("the number is divisible by 5")
    else:
        print("the number is not divisible by 5")
```
```
the number is not divisible by 5
the number is not divisible by 5
the number is not divisible by 5
the number is not divisible by 5
the number is not divisible by 5
the number is not divisible by 5
the number is divisible by 5
```

In [31]:
```
a_str="hihihihihihihihi"
a_sub="hi"
print("the original value is ",a_str )
print("the original value is ",a_sub )
f=a_str.count(a_sub)
print(f)
```
```
the original value is  hihihihihihihihi
the original value is  hi
8
```

In [2]:
```
rows=int(input("enter the number of rows:"))
for i in range(rows+1):
    for j in range(i):
        print(i,end=" ")
    print(" ")
```
```
enter the number of rows:5

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

In [3]:
```
def palindrome(n):
    temp=n
    rev=0
    while(n>0):
        a=n%10
        rev=rev*10+a
        n=n//10
    if(temp==rev):
            print("it is a palindrome")
    else:
            print("it is not a palindrome")
palindrome(525)
```
```
it is a palindrome
```

In [4]:
```
list=[1,2,3,4,5,6,7]
print("initial list")
print(list)
list.pop(0)
print(list)
list.pop(5)
print(list)
list.insert(0,7)
list.insert(6,1)
print(list)
```
```
initial list
[1, 2, 3, 4, 5, 6, 7]
[2, 3, 4, 5, 6, 7]
[2, 3, 4, 5, 6]
[7, 2, 3, 4, 5, 6, 1]
```

In [12]:
```
def swapPositions(list,pos1,pos2):
    list[pos1],list[pos2]=list[pos2],list[pos1]
    return list
List=[23,65,19,90]
pos,pos2=1,3
print(swapPositions(List,pos-1, pos2-1))
```
```
[19, 65, 23, 90]
```

In [2]:
```
list=[10,20,30]
n=len(list)
print("the length of list is :",n)
```
```
the length of list is : 3
```

In [5]:
```
def maximum(a,b):
    if a>=b:
        return a
    else:
        return b
a=2
b=5
print(maximum(a,b))
```
```
5
```

In [7]:
```
def minimum(a,b):
    if a<=b:
        return ac
    else:
        return b
a=2
b=5
print(minimum(a,b))
```
```
2
```

In [2]:
```
a=input("enter the value")
print("palindrome checking")
if a==a[::-1]:
  print("it is a palindrome")
else:
    print("not a palindrome")
print("symmetrical checking")
half=len(a)//2
if a[half:]==a[:half]:
    print ("it is symmetrical")
else:
    print("it is not symmetrical")
```
```
enter the valueabab
palindrome checking
not a palindrome
symmetrical checking
it is symmetrical
```

In [3]:
```
a=input("enter a value")
print(a[::-1])
```
```
enter a valuebuddy
yddub
```

In [1]:
```
s=input("enter a string:")
i=int(input("enter index value:"))
s1=s[:i]+s[i+1:]
print(s1)
```
```
enter a string:hello
enter index value:2
helo
```

In [2]:
```
s=input("enter the string:")
print("length of string:",len(s))
c=0
for i in s:
    c+=1
print("length of string:",c)
```
```
enter the string:hello
length of string: 5
length of string: 5
```

In [3]:
```
a=input("enter the string")
s=a.split(" ")
for i in s:
    if len (i)%2==0:
        print(i)
```
```
enter the stringhi hi hello
hi
hi
```

In [4]:
```
t=(1,2,3,4,5)
print(len(t))
```
```
5
```

In [5]:
```
t=(1,2,3,4,5)
print("maximum value=",max(t))
print("minimum value=",min(t))
```
```
maximum value= 5
minimum value= 1
```

In [6]:
```
t=(1,2,3,4,5)
print("sum of elements in the tuple:",sum(t))
```
```
sum of elements in the tuple: 15
```

In [9]:
```
tmat=((1,2,3),(4,5,6),(7,8,9))
for row in tmat:
    s=sum(row)
    print("row sum:",s)
```
```
row sum: 6
row sum: 15
row sum: 24
```

In [1]:
```
# Creating a list
myList = [6, 2, 5 ,1, 4]

# Creating list of tuples
tupleList = []
for val in myList:
    myTuple = (val, (val*val*val))
    tupleList.append(myTuple)

# print the result
print("The list of Tuples is " , str(tupleList))
```
```
The list of Tuples is  [(6, 216), (2, 8), (5, 125), (1, 1), (4, 64)]
```

In [2]:
```
myDict = {'ravi': 10, 'rajnish': 9,
                'sanjeev': 15, 'yash': 2, 'suraj': 32}

myKeys = list(myDict.keys())
myKeys.sort()
sorted_dict = {i: myDict[i] for i in myKeys}

print(sorted_dict)
```
```
{'rajnish': 9, 'ravi': 10, 'sanjeev': 15, 'suraj': 32, 'yash': 2}
```

In [3]:
```
# Python3 Program to find sum of
# all items in a Dictionary

# Function to print sum


def returnSum(myDict):

        list = []
        for i in myDict:
                list.append(myDict[i])
        final = sum(list)

        return final


# Driver Function
dict = {'a': 100, 'b': 200, 'c': 300}
print("Sum :", returnSum(dict))
```
```
Sum : 600
```

In [6]:
```
my_dict = {}
my_dict["apple"] = 2
my_dict["banana"] = 4
my_dict["orange"] = 3
size = len(my_dict)
print(size)
```
```
3
```

In [7]:
```
import sys

# sample Sets
Set1 = {"A", 1, "B", 2, "C", 3}
Set2 = {"Geek1", "Raju", "Geek2", "Nikhil", "Geek3", "Deepanshu"}
Set3 = {(1, "Lion"), ( 2, "Tiger"), (3, "Fox")}

# print the sizes of sample Sets
print("Size of Set1: " + str(sys.getsizeof(Set1)) + "bytes")
print("Size of Set2: " + str(sys.getsizeof(Set2)) + "bytes")
print("Size of Set3: " + str(sys.getsizeof(Set3)) + "bytes")
```
```
Size of Set1: 472bytes
Size of Set2: 472bytes
Size of Set3: 216bytes
```

In [9]:
```
# Creating a set using string
test_set = set("geeks")

# Iterating using for loop
for val in test_set:
        print(val)
```
```
e
g
s
k
```

In [10]:
```
# Python code to get the maximum element from a set
def MAX(sets):
        return (max(sets))

# Driver Code
sets = set([8, 16, 24, 1, 25, 3, 10, 65, 55])
print(MAX(sets))
```
```
65
```

In [11]:
```
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```
```
{'apple', 'cherry'}
```

In [12]:
```
# Python program to check
# if two lists have at-least
# one element common
# using set and property

def common_member(a, b):
        a_set = set(a)
        b_set = set(b)
        if (a_set & b_set):
                return True
        else:
                return False


a = [1, 2, 3, 4, 5]
b = [5, 6, 7, 8, 9]
print(common_member(a, b))

a =[1, 2, 3, 4, 5]
b =[6, 7, 8, 9]
print(common_member(a, b))
```
```
True
False
```

In [1]:
```
test_list=[[5,8,9],[2,0,9],[7,6,5],[8,3,6]]
print("the original list: "+str(test_list))
res={test_list[0][ele]:test_list[ele+1] for ele in range(len(test_list)-1)}
print("the assigned matrix :"+ str(res))
```
```
the original list: [[5, 8, 9], [2, 0, 9], [7, 6, 5], [8, 3, 6]]
the assigned matrix :{5: [2, 0, 9], 8: [7, 6, 5], 9: [8, 3, 6]}
```

In [3]:
```
import numpy as np
A=np.array([[1,2],[3,4]])
B=np.array([[4,5],[6,7]])
print("printing elements of first matrix")
print(A)
print("printing elements of second matrix")
print(B)
print("addition of two matrix")
print(np.add(A,B))
```
```
printing elements of first matrix
[[1 2]
 [3 4]]
printing elements of second matrix
[[4 5]
 [6 7]]
addition of two matrix
[[ 5  7]
 [ 9 11]]
```

In [1]:
```
# Python3 code to demonstrate working of
# Assigning Subsequent Rows to Matrix first row elements
# Using dictionary comprehension

# initializing list
test_list = [[5, 8, 9], [2, 0, 9], [5, 4, 2], [2, 3, 9]]

# printing original list
print("The original list : " + str(test_list))

# pairing each 1st col with next rows in Matrix
res = {test_list[0][ele] : test_list[ele + 1] for ele in range(len(test_list) - 1)}

# printing result
print("The Assigned Matrix : " + str(res))
```
```
The original list : [[5, 8, 9], [2, 0, 9], [5, 4, 2], [2, 3, 9]]
The Assigned Matrix : {5: [2, 0, 9], 8: [5, 4, 2], 9: [2, 3, 9]}
```

In [2]:
```
# importing numpy as np
import numpy as np


# creating first matrix
A = np.array([[1, 2], [3, 4]])

# creating second matrix
B = np.array([[4, 5], [6, 7]])

print("Printing elements of first matrix")
print(A)
print("Printing elements of second matrix")
print(B)

# adding two matrix
print("Addition of two matrix")
print(np.add(A, B))
```
```
Printing elements of first matrix
[[1 2]
 [3 4]]
Printing elements of second matrix
[[4 5]
 [6 7]]
Addition of two matrix
[[ 5  7]
 [ 9 11]]
```

In [3]:
```
from itertools import groupby
test_list = [1, 3, 5, 1, 3, 2, 5, 4, 2]

# printing original list
print("The original list : " + str(test_list))

# Group similar elements into Matrix
# Using list comprehension + groupby()
res = [list(val) for key, val in groupby(sorted(test_list))]

# printing result
print("Matrix after grouping : " + str(res))
```
```
The original list : [1, 3, 5, 1, 3, 2, 5, 4, 2]
Matrix after grouping : [[1, 1], [2, 2], [3, 3], [4], [5, 5]]
```

In [4]:
```
test_list = [[('Gfg', 3), ('is', 3)], [('best', 1)], [('for', 5), ('geeks', 1)]]

# printing original list
print("The original list is : " + str(test_list))

# initializing Custom eles
cus_eles = [6, 7, 8]

# Row-wise element Addition in Tuple Matrix
# Using enumerate() + list comprehension
res = [[sub + (cus_eles[idx], ) for sub in val] for idx, val in enumerate(test_list)]

# printing result
print("The matrix after row elements addition : " + str(res))
```
```
The original list is : [[('Gfg', 3), ('is', 3)], [('best', 1)], [('for', 5), ('geeks', 1)]]
The matrix after row elements addition : [[('Gfg', 3, 6), ('is', 3, 6)], [('best', 1, 7)], [('for', 5, 8), ('geeks', 1, 8)]]
```

In [5]:
```
import itertools


def sub_mat_even(n):

    temp = itertools.count(1)

    # create a 2d array ranging
    # from 1 to n^2
    l = [[next(temp)for i in range(n)]for i in range(n)]

    # If found even we reverse the alternate
    # row elements to get all diagonal elements
    # as all even or all odd
    if n%2 == 0:
        for i in range(0,len(l)):
            if i%2 == 1:
                l[i][:] = l[i][::-1]

    # Printing the array formed
    for i in range(n):
        for j in range(n):
            print(l[i][j],end=" ")
        print()

n = 4
sub_mat_even(n)
```
```
1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13
```

In [1]:
```
# explicit function
def fun(a, b):
        return a**b

# import required modules
import inspect

# use signature()
print(inspect.signature(fun))
```
```
(a, b)
```

In [3]:
```
def GFG(name, num):
    print("Hello from ", name + ', ' + num)


GFG("geeks for geeks", "25")
```
```
Hello from  geeks for geeks, 25
```

In [4]:
```
def power(N, P):

    # If power is 0 then return 1
    # if condition is true
    # only then it will enter it,
    # otherwise not
    if P == 0:
        return 1

    # Recurrence relation
    return (N*power(N, P-1))


# Driver code
if __name__ == '__main__':
    N = 5
    P = 2

    print(power(N, P))
```
```
25
```

In [5]:
```
print(sorted([1,26,3,9]))

print(sorted("Geeks foR gEEks".split(), key=str.lower))
```
```
[1, 3, 9, 26]
['foR', 'Geeks', 'gEEks']
```

In [6]:
```

def printKwargs(**kwargs):
    print(kwargs)


# driver code
if __name__ == "__main__":
    printKwargs(Argument_1='gfg', Argument_2='GFG')
```
```
{'Argument_1': 'gfg', 'Argument_2': 'GFG'}
```

In [ ]: