Python is a popular programming language.

It was created by Guido van Rossum, and released in 1991.

#VARIABLES

#SINGLE ASSIGNMENT  
name="MAHESH"  
age=19  
height=5.6  
weight=60  
print(name,age,height,weight)

#MULTIPLE ASSIGNMENT  
name,age,height,weight="Mahesh",19,5.6,60  
print(name,age,height,weight)

#arithmetic operations  
print(2+3)  
print(2-3)  
print(2\*3)  
print(2/3)  
print(2%3)  
  
#normal division  
print(5/2)  
#floor division  
print(5//2)  
  
#combination of arithmetic operations  
print(2+3\*10)  
print((2+3)\*10)  
  
#power  
print(2\*2\*2\*2\*2)  
print(2\*\*5)  
  
import math  
print(math.pow(2,5))  
  
import math as m  
print(m.pow(2,5))  
  
print("hello world")  
print("HELLO WORLD")  
print('HELLO WORLD')  
  
#strings can be printed using "" or ''  
print("SURYA'S LAPTOP")  
print('MAHESH"S book')  
  
#string with both " , '  
#\ backslash simply changes the meaning of the '  
print('mahesh\'s "LAPTOP" is lenovo')  
  
print(" MAHESH is \"good\" boy ")  
  
# printed many strings seperated by commas  
  
print("surya",'mahesh',"HELLO",'mowa')  
  
#strings can be concatenated simply using + sign  
  
print("surya"+'mahesh'+"HELLO"+'mowa')  
  
#printing same string multiple times using \*  
print(5\*"\nSURYA ")  
  
# \n goes to new line  
print("\n hello murugan ")  
print("\n syndicate member \n pushpa pesire")  
  
#for printing the raw string use 'r before that  
  
print(r"\n syndicate member \n pushpa pesire")  
print(r'\n syndicate member \n pushpa pesire')

#variables  
x=10  
print(" value of x = ",x)  
print(" value of x = "+str(x))  
print(" value of x = %d"%(x))  
  
print(x+5)  
print(x-1)  
print(x)  
  
name="SURYA MAHESH.kolisetty"  
print(name)  
# individual characters can be retrieved from the string  
print(name[0])  
  
#back indexing also possible  
print(name[-1])  
  
#slicing  
print(name[:3])  
print(name[1:])  
print(name[2:4])  
  
print(" strings are immutable , but subsets can be retrieved / sliced acc to our wish")  
  
print("MYNAME IS :",name)  
  
print("length of '"+name+"' is : ",len(name))

#LISTS

# Adds List Element as value of List.  
list = ['Mathematics', 'chemistry', 1997, 2000]  
list.append(20544)  
print(list)  
  
List1 = [1, 2, 3]  
List2 = [2, 3, 4, 5]  
  
# Add List2 to List1  
List1.extend(List2)  
print(List1)  
  
# Add List1 to List2 now  
List2.extend(List1)  
print(List2)  
  
list = [1, 2, 3, 4, 5]  
print(sum(list))  
  
list = [1,2, 3]  
print(sum(list))  
  
list = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]  
print(list.count(1))  
  
list = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]  
print(len(list))  
  
list = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]  
print(list.index(2))  
  
list = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]  
print(list.index(2, 2))  
  
list = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]  
  
"""index(element, start, end) : It will calculate till index end-1. """  
  
# will check from index 2 to 4.  
print("After checking in index range 2 to 4")  
print(list.index(1))  
  
# will check from index 2 to 3.  
print("After checking in index range 2 to 3")  
print(list.index(2))  
list = [1,6,2,4,3,9]  
print(min(list))  
  
list = [1,6,2,4,3,9]  
print(max(list))  
  
list = [1,6,2,4,3,9]  
  
# Reverse flag is set True  
list.sort(reverse=True)  
  
# List.sort().reverse(), reverses the sorted list  
print(list)  
  
list = [1,6,2,4,3,9]  
sorted(list)  
print(list)  
  
list = [1,6,2,4,3,9]  
print(list.pop())  
  
list = [1,6,2,4,3,9]  
del list[0]  
print(list)  
  
list = [1,6,2,4,3,9]  
list.remove(3)  
print(list)

#LIST  
  
nums=[1,4,2,3,4  
 ,66,7,8,22,1]

#list can hold any types  
print(nums)  
names=["abc","def","hello","murugan","syndicate","member","pushpa","pesire"]  
print(names)  
#list can be sliced  
print(nums[:4])  
print(nums[2:])  
print(nums[1:5])  
  
  
#list can be sliced  
  
print(names[:4])  
print(names[2:])  
print(names[1:5])  
  
#list can hold multiple types  
values=[1,2,3,"pushpa","syndicate","member",1.0,1+2j]  
print(values)  
print(values[-1])  
  
  
#list can be have another lists as members  
  
mymix=[nums,names,values]  
print(mymix)  
  
print(nums)  
nums.append(1000)  
print(nums)  
nums.insert(3,20000)  
print(nums)  
nums.remove(66)  
print(nums)  
print(nums.pop())  
print(nums)  
nums.pop(3)  
print(nums)  
print(nums.pop())  
  
del nums[5:]  
print(nums)  
  
del nums[:3]  
  
print(nums)  
#extend a list with a list  
nums.extend([1,2,3,4,5,6,7,8,9,0])  
print(min(nums))  
print(max(nums))  
print(sum(nums))  
  
nums=[6,2,5,1,3,8,9,4]  
nums.sort()  
print(nums)  
nums.sort(reverse=True)  
print(nums)  
  
l=[]  
l=nums.copy()  
print(l)  
  
print(nums.\_\_getitem\_\_(1))  
nums.\_\_setitem\_\_(1,100)  
print(nums)  
nums.sort()  
print(nums)  
  
nums.append(1)  
nums.append(1)  
nums.append(1)  
  
  
print(nums.count(1))

import random  
  
#lists are mutable in python  
a=[1,2,3]  
print(" list if ekements in a is: ",a)  
for i in a:  
 print(i)  
 # max , min  
print(max(a))  
print(min(a))  
l=[100,20,1,200,25,1000]  
print(l)  
#sort  
l.sort()  
print(l)  
s=sum(l)  
print("SUM OF ELEMENTS IN THE LIST %d"%(s))  
print("SUM OF ELEMENTS IN THE LIST "+str(s))  
l=[[100,20,1,200,25,1000],[10,20,1,200,25,1000],[100,2,1,20,25,1000]]  
print(l)  
print(l[0])  
print(l[1])  
print(l[2])  
print(l[0][2])  
print(l[2][2])  
j=0  
k=0  
for j in range(len(l)):  
 for k in range(len(l[0])):  
 print(l[j][k])  
  
print(1000 in l[0])  
b=[1,2,3,4,5,6,7,8,9]  
print(b)  
#membership in  
for i in range(len(b)):  
  
 print(b[i])  
  
#built in methods  
b.append(10)  
b.append(20)  
b.append(30)  
b.append([60,70])  
#append takes only argument single one  
print(b)  
b.extend([80,90,100])  
b.extend([110,120,130,[140,150,160]])  
print(b)  
#insert  
b.insert(3,1000)  
print(b)  
#remove  
b.remove(30)  
print(b)  
b.remove(1000)  
print(b)  
#pop  
b.pop()  
print(b)  
b.pop()  
print(b.pop(3))  
#del  
del b[2]  
print(b)  
#shuffle  
c=[1,2,3,4,5]  
random.shuffle(c)  
print(c)  
c.sort()  
print(c)  
#printing reverse  
c=[1,2,3,4,5]  
random.shuffle(c)  
print(c)  
c.sort(reverse=True)  
print(c)  
  
print(c\*2)  
c.sort()  
print(c)  
c.sort(reverse=True)  
print(b)  
  
c.append(10)  
c.append(10)  
c.append(10)  
c.append(10)  
print(c)  
#count  
print(" TOTALLY THERE ARE %d 10'S IN THE LIST"%(c.count(10)))  
#index  
print(c.index(10))  
print(c.index(2))  
  
#reverse  
c.reverse()  
print(c)

a=[1]  
  
a.append([2,3,4,5])  
print(len(a))  
print(a)  
  
#append only adds one element to the list , so a after this is with list [2,3,4,5] as second element  
  
print("Hello Mike".split())

for i,x in enumerate(['A','B','C']):  
 print(i, x)

#LIST SORTED FUNCTION

L=[1,3,2]  
print(sorted(L,reverse=False))  
print(sorted(L))  
print(sorted(L,reverse=True))  
print(L)

#list comprehensions  
d=[]  
print(d)  
d=[ele for ele in range(10)]  
print(d)  
  
d=[ele\*ele for ele in range(10)]  
print(d)  
  
d=[ele\*ele for ele in range(10) if ele%2==0]  
print(d)  
  
d=[ele\*ele for ele in range(10) if ele%2!=0]  
print(d)  
  
e=[]  
for ii in range(20):  
 e.append(ii)  
print(e)  
  
  
#printing even numbers upto 50 using list comprehensions  
  
f=[ele for ele in range(50) if ele%2==0]  
print(f)  
  
print(f.\_\_getitem\_\_(1))  
print(f.\_\_setitem\_\_(2,1000))  
print(f)  
#tuple immutable  
#() comma seperated values in paranthesis  
#multiple datatypes can be hold in the tuple  
  
t=(1,2,3,4,5)  
print(" TUPLE IS : ",t)  
#list can be assigned to tuples  
#TUPLES can be assigned to LISTS  
t=f  
e=t  
print("TUPLE T IS WITH NOW WITH LIST F ELEMENTS ",t)  
e=t  
print("LIST E IS WITH NOW WITH TUPLE ELEMENTS ",e)  
  
t2=tuple(a)  
print(t2)  
#t2[0]="BYE" not possibl in tuples  
print(t2.\_\_getitem\_\_(2))  
  
l2=list(t)  
print(l2)  
l2[0]="hello"  
print(l2)  
  
t=(10)  
print(" NOW TUPLE T IS WITH SINGLE ELEMENT LETS CHECK THE TYPE ",type(t))  
  
t=(10,)  
print(" NOW TUPLE T IS WITH SINGLE ELEMENT BUT WE FOLLOWED THE SYNTAX LETS CHECK THE TYPE ",type(t))  
print("IT's WORKING"+'COOOOOOL')

#TUPLES  
  
t1=(1,2,3)  
t2=(1,2,3)  
  
if t1==t2:  
 print("same")  
  
if (1,2,3)==(1,2,3):  
 print("same")  
  
# TUPLES COMPARISION ( INDEX BASED )  
  
#here first element of tuple1 is greater than tuple2  
if (2,2,6)>(1,2,6):  
 print("greater than")  
  
#here first element same then check next ,second element of tuple1 is greater than tuple2  
if (4,2,1)>(4,1,1):  
 print("greater than")  
  
#here first,second element same then check next ,third element of tuple1 is greater than tuple2  
if (4,1,2)>(4,1,1):  
 print("greater than")  
  
#here first element of tuple1 is less than tuple2  
  
if (0,2,6)<(1,2,6):  
 print("less than")  
  
#here first element same then check next ,second element of tuple1 is less than tuple2  
  
if (3,1,1)<(4,1,1):  
 print("less than")  
  
#here first,second element same then check next ,third element of tuple1 is less than tuple2  
  
if (4,1,0)<(4,1,1):  
 print("less than")

#LISTS  
elements=[1,"ahgsg",1.1,1+2j]  
print(type(elements))  
for i in range(len(elements)):  
 print(elements[i],end=" ")  
  
elements.append(1000)  
print(elements)  
#lists can take all mixed data types  
#one list can hold another list , tuple ,set strings all the primitive built in data types  
elements=[1,"ahgsg",1.1,1+2j,[1,"ahgsg",1.1,1+2j],(1,2,3,4,5,6),{1,2,3,4,5,6,7,8,9,0}]  
print(elements)  
  
for i in range(len(elements)):  
 print(elements[i])

#COMPREHENSIONS  
  
#list comprehensions  
import math  
  
l=list()  
print(type(l))  
l=[]  
for i in range(10):  
 l.append(i)  
  
print(l)  
  
l=[]  
l=[i for i in range(0,20,2) ]  
print(l)  
  
l=[]  
l=[i for i in range(1,20,2) ]  
print(l)  
  
l=[]  
l=[i for i in range(40) if i%4==0 ]  
print(l)  
  
  
#tuple comprehensions  
  
t=()  
print(type(t))  
  
t=tuple(i for i in range(0,20,2) )  
print(t)  
  
t=tuple(i for i in range(1,20,2) )  
print(t)  
  
t=tuple(i for i in range(0,20,2) if i%4==0 )  
print(t)  
  
  
l=list(i for i in range(0,20,2) if i%4==0 )  
print(l)  
  
s=set(i for i in range(0,20,2) if i%4==0 )  
print(s)  
  
d={i:i\*\*2 for i in range(1,20,2) if i%2==1}  
print(d)  
  
import math  
l=[ i for i in range(0,10)]  
t=tuple( i for i in range(0,10))  
s={ i for i in range(0,10)}  
d={ i:math.pow(i,2) for i in range(0,10)}  
  
  
print(l)  
print(t)  
print(s)  
print(d)

#STRINGS

#strings  
#strings are immutable  
  
s="PYTHON"  
print(s)  
  
#accesing string  
for i in range(len(s)):  
 print(s[i])  
  
print(s," TYPE --->",type(s))  
  
s1="124003143@sastra.ac.in"  
print(s1)  
print(type(s1))  
  
for i in range(len(s1)):  
 print(type(s1[i]))  
  
#i=len(s1)-1  
for i in range(len(s1)):  
 print(s1[0-i],end="")  
  
#slicing  
print("\n SLICING",s[0:])  
print("SLICING",s[:len(s)])  
print("SLICING",s1[:len(s1)])  
  
  
"""hello  
 world"""  
  
s2="""hello  
 world"""  
print(s2)  
  
#.format method  
  
print("{0} \n {1} ".format(1,"hell"))  
# 0 for first 1 for second ..so .on positiinal arguments  
  
print("{0}\n{1} ".format(10,"hell"))  
print("{0}\n{1} ".format(10,10.0))  
print("{0}\n{1}\n{2} ".format(10,10.0,1+2j))  
print("{0}\n{1}\n{2}\n{3} ".format(10,10.0,1+2j,"PESIRE"))

str1="python"  
str2="progamming"  
print(" HEY SHIV COOOL ",str1+" "+str2)  
  
  
print("MIN OF",str1,"BASED ON THE ASCII VALUE ",min(str1))  
print("MIN OF",str2,"BASED ON THE ASCII VALUE ",min(str2))  
print("MAX OF",str1,"BASED ON THE ASCII VALUE ",max(str1))  
print("MAX OF",str2,"BASED ON THE ASCII VALUE ",max(str2))  
  
#whether a chracter in the string present or not  
  
print("p"in str1)  
print("py"in str1)  
print("prog"in str1)  
print("prog"in str2)  
  
for i in range(len(str1)):  
 print(str1[i])  
  
for x in str2:  
 print(x)  
  
#methods in strings  
  
s="python"  
print(s)  
  
print(s.center(20,"\*"))  
print(str2.center(20,"\*"))  
  
  
s="welcome to python programming"  
print("o repeated ",s.count("o")," TIMES in ",s)  
  
  
print(s,"is a string starts",s.startswith("py"))  
print("STRING ENDSWITH ing :",s.endswith("ing"))  
  
print(s.capitalize())  
  
  
# we can find the frequency of each character from particular index to particular index of the string  
s="OOPSSSSSOOOOOOOOPPPPPSSSSOOOOPPPSSS"  
print(len(s))  
print(s.count("O",0,10))  
print(s.count("P",5,len(s)))  
print(s.count("S",5,10))  
  
s="PYHON IS MY LANGUAGEY PYPYPYPPYPY "  
print(" INDEX OF 'Y' IN STRING S IS ",s.find("Y"))  
print(" INDEX OF 'Y' IN STRING S WHEN SEARCHED FROM BACK IS ",s.rfind("Y"))  
print(s.find("Z"))  
  
print("index of 'P' in the string searched from the start : ",s.index("P"))  
print(len(s))  
print("index of 'P' in the string searched from the REVERSE : ",s.rindex("P"))

#REPLACE METHOD

s="AAAAAAAA"  
print(s.replace('A','B'))  
  
s="AAHAHAHAHA"  
print(s.replace('A','B'))  
  
  
s="OOAOAOAOAOAOA"  
print(s.replace('A','B'))

# to check whether the given string is alphanumeric or not  
  
s="abc123"  
#string should be with alpha / numeric / alpha numeric any of these will return true  
print(s," ia aphanumeric string : IS IT TRUE BHAVANI ? :",s.isalnum())  
s="abcdef"  
print(s," ia APLHA string only characters : IS IT TRUE BHAVANI ? :",s.isalpha())  
s="1234@#$qwee"  
print(s," ia ascii string : IS IT TRUE BHAVANI ? :",s.isascii())  
s="abc123"  
print(s," ia digit string : IS IT TRUE BHAVANI ? :",s.isdigit())  
s="123456"  
print(s," ia digit string : IS IT TRUE BHAVANI ? :",s.isdigit())  
s="HELLO MURUGAN NAANU SYNDICATE MEMBER PUSHPA PESIRE "  
print(s)  
s=" "  
print(s," is string with only space :: IS IT TRUE BHAVANI ? ",s.isspace())  
s="HELLO MURUGAN NAANU SYNDICATE MEMBER PUSHPA PESIRE "  
print(s," is string with only UPPERCASE LETTERS :: IS IT TRUE BHAVANI ? ",s.isupper())  
print(s," is string with only LOWERCASE LETTERS :: IS IT TRUE BHAVANI ? ",s.islower())  
print(s.lower())  
s=s.lower()  
print(s," is string with only LOWERCASE LETTERS :: IS IT TRUE BHAVANI ? ",s.islower())  
print(s.upper())  
s=s.lower()

print(s.capitalize())  
print(s.title())  
print(s.ljust(50,"\*"))  
print(s.rjust(50,"\*"))

s="hello man"  
s=s.title()  
print(s.ljust(20,"\*"))  
  
print(s.rjust(20,"\*"))  
  
print(s.center(20,"\*"))

# removing spaces in strings  
s=" PUSHPA PESIRE "  
print(s.strip())  
s=" PUSHPA PESIRE "  
print(s.lstrip())  
s=" PUSHPA PESIRE "  
print(s.rstrip())  
  
s=" PUSHPA PESIRE "  
print(s.replace("P","K"))  
print(s)  
s=" PUSHPA PUSHPA PUSHPA PUSHPA PUSHPA "  
print(s)  
print(s.replace("PUSHPA","PULKA"))  
print(s)  
  
#replacing the string only 2 times with the new string  
print(s.replace("PUSHPA","PULKA",2))  
  
print(s.split(" "))  
s=" PUSHPA\*PUSHPA\*PUSHPA\*PUSHPA\*PUSHPA "  
  
print(s.split("\*"))  
s=" PUSHPA PUSHPA PUSHPA PUSHPA PUSHPA "  
  
print(s.zfill(55))  
s=" PushPA PushPA PushPA PushPA PushPA "  
  
print(s.swapcase())

#TUPLES

#tuples  
  
weeks=("sunday","monday","tuesday","wednesday","thursday","friday","saturday")  
print(weeks)  
print(type(weeks))  
  
weeks="sunday","monday","tuesday","wednesday","thursday","friday","saturday"  
print(weeks)  
print(type(weeks))  
  
for i in range(len(weeks)):  
 print(weeks[i])  
  
  
weeks[2]="SURYADAY"  
  
try:  
 weeks[2]="SURYADAY"  
except:  
 print("SOMETHING WENT WRONG")  
else:  
 print("NO EXCEPTION")  
finally:  
 print("BYE BYE ")

#STRING METHODS

# Python3 program to show the  
# working of upper() function  
text = 'geeKs For geEkS'  
  
# upper() function to convert  
# string to upper case  
print("\nConverted String:")  
print(text.upper())  
  
# lower() function to convert  
# string to lower case  
print("\nConverted String:")  
print(text.lower())  
  
# converts the first character to  
# upper case and rest to lower case  
print("\nConverted String:")  
print(text.title())  
  
# original string never changes  
print("\nOriginal String")  
print(text)  
  
# Python program to demonstrate the  
# use of capitalize() function  
  
# capitalize() first letter of string  
# and make other letters lowercase  
name = "geeks FOR geeks"  
  
print(name.capitalize())  
  
# demonstration of individual words  
# capitalization to generate camel case  
name1 = "geeks"  
name2 = "for"  
name3 = "geeks"  
print(name1.capitalize() + name2.capitalize() + name3.capitalize())  
  
# First character in each word is  
# uppercase and remaining lowercases  
s = 'Geeks For Geeks'  
print(s.istitle())  
  
# First character in first  
# word is lowercase  
s = 'geeks For Geeks'  
print(s.istitle())  
  
# Third word has uppercase  
# characters at middle  
s = 'Geeks For GEEKs'  
print(s.istitle())  
  
s = '6041 Is My Number'  
print(s.istitle())  
  
# word has uppercase  
# characters at middle  
s = 'GEEKS'  
print(s.istitle())  
  
# Python code to demonstrate the working of  
# index()  
  
# initializing target string  
ch = "geeksforgeeks"  
  
# initializing argument string  
ch1 = "geeks"  
  
# using index() to find position of "geeks"  
# starting from 2nd index  
# prints 8  
pos = ch.index(ch1, 2)  
  
print("The first position of geeks after 2nd index : ", end="")  
print(pos)  
  
# Python3 code to demonstrate  
# working of isupper()  
  
# initializing string  
isupp\_str = "GEEKSFORGEEKS"  
not\_isupp = "Geeksforgeeks"  
  
# Checking which string is  
# completely uppercase  
print("Is GEEKSFORGEEKS full uppercase ? : " + str(isupp\_str.isupper()))  
print("Is Geeksforgeeks full uppercase ? : " + str(not\_isupp.isupper()))  
  
# Python code for implementation of isdigit()  
  
# checking for digit  
string = '15460'  
print(string.isdigit())  
  
string = '154ayush60'  
print(string.isdigit())  
  
# Python3 code to print  
# all encodings available  
  
from encodings.aliases import aliases  
  
# Printing list available  
print("The available encodings are : ")  
print(aliases.keys())  
  
# Python code for implementation of isspace()  
  
# checking for whitespace characters  
string = 'Geeksforgeeks'  
  
print(string.isspace())  
  
# checking if \n is a whitespace character  
string = '\n \n \n'  
  
print(string.isspace())  
  
string = 'Geeks\nfor\ngeeks'  
print(string.isspace())  
  
# Python code for implementation of isnumeric()  
  
# checking for numeric characters  
string = '123ayu456'  
print(string.isnumeric())  
  
string = '123456'  
print(string.isnumeric())  
  
# Python code for implementation of isdigit()  
  
# checking for digit  
string = '15460'  
print(string.isdigit())  
  
string = '154ayush60'  
print(string.isdigit())  
  
# Python code for implementation of isalpha()  
  
# checking for alphabets  
string = 'Ayush'  
print(string.isalpha())  
  
string = 'Ayush0212'  
print(string.isalpha())  
  
# checking if space is an alphabet  
string = 'Ayush Saxena'  
print(string.isalpha())  
  
# Python program to demonstrate the use of  
# isalnum() method  
  
# here a,b and c are characters and 1,2 and 3  
# are numbers  
string = "abc123"  
print(string.isalnum())  
  
# here a,b and c are characters and 1,2 and 3  
# are numbers but space is not a alphanumeric  
# character  
string = "abc 123"  
print(string.isalnum())  
  
word = 'geeks for geeks'  
  
# returns first occurrence of Substring  
result = word.find('geeks')  
print("Substring 'geeks' found at index:", result)  
  
result = word.find('for')  
print("Substring 'for ' found at index:", result)  
  
# How to use find()  
if (word.find('pawan') != -1):  
 print("Contains given substring ")  
else:  
 print("Doesn't contains given substring")  
  
# Python code shows the working of  
# .endswith() function  
  
text = "geeks for geeks."  
  
# returns False  
result = text.endswith('for geeks')  
print(result)  
  
# returns True  
result = text.endswith('geeks.')  
print(result)  
  
# returns True  
result = text.endswith('for geeks.')  
print(result)  
  
# returns True  
result = text.endswith('geeks for geeks.')  
print(result)  
  
# Python program to illustrate  
# string center() in python  
string = "geeks for geeks"  
  
new\_string = string.center(24)  
  
# here filchar not provided so takes space by default.  
print("After padding String is: ", new\_string)  
  
#SETS ADD FUNCTION

*"""*

*>>> s = set('HackerRank')  
>>> s.add('H')  
>>> print s  
set(['a', 'c', 'e', 'H', 'k', 'n', 'r', 'R'])  
>>> print s.add('HackerRank')  
None  
>>> print s  
set(['a', 'c', 'e', 'HackerRank', 'H', 'k', 'n', 'r', 'R'])*

*"""*

#sets  
s={}  
print(type(s))  
  
s=set()  
print(type(s))  
  
s={1,3,2,4,5,7,6,9,8}  
print(s)  
print(s)  
print(s)  
print(s)  
s.add(1)  
s.add(2)  
s.add(10)  
print(" doesnt allow duplicates.... and also insertion order not preserved")  
print(s)  
  
s.remove(1)  
print(s)  
  
s.discard(10)  
s.remove(5)  
s.discard(100)  
#s.remove(100)  
print(s)  
  
print(s.pop())  
  
print(" CLEAR METHOD CLEARS ALL THE SET ")  
s.clear()  
print(s)  
  
#print(del s)  
s={1,3,2,4,5,7,6,9,8}  
print(s)  
print(" LENGTH OF THE SET IS : ",len(s))  
print(" ELEMENTS IN THE SET ")  
for ele in s:  
 print(ele)  
  
#set objects are not subscriptable , cannot be accessed with the index since the set is unordered  
  
t=set()  
for i in range(20):  
 t.add(i)  
  
print(s," IS SUBSET OF ",t," IS IT TRUE BHAVANI ",s.issubset(t))  
  
print(t," IS SUPERSET OF ",s," IS IT TRUE BHAVANI ",t.issuperset(s))  
  
print(s.union(t))  
s={1,3,5,6,7,9}  
t={1,2,3,4,5,6}  
print(s.intersection(t))  
print(t.difference(s))  
print(s.symmetric\_difference(t))  
  
print(s.copy())  
print(s.intersection\_update(t))  
s={1,3,5,6,7,9}  
t={1,2,3,4,5,6}  
  
print(s.difference\_update(t))  
s={1,3,5,6,7,9}  
t={1,2,3,4,5,6}  
  
print(s.symmetric\_difference\_update(t))  
s={1,3,5,6,7,9}  
t={1,2,3,4,5,6}  
print(s)  
print(t)  
t=s.copy()  
print(s)  
print(t)

#dictionaries  
  
d={}  
print(type(d))  
d=dict()  
print(type(d))  
d={"1":"SURYA","2":"MAHESH","3":"GOPI","4":"PEMMA","5":"SUBBBU"}  
print(d)  
d={1:"SURYA",2:"MAHESH",3:"GOPI",4:"PEMMA",5:"SUBBBU"}  
  
i=1  
while i<len(d):  
 print(d[i])  
 i=i+1  
  
del d[2]  
print(d)  
  
#del d  
  
  
d1={6:"SATHI"}  
d2={7:"PITHI"}  
  
d.update(d1)  
print(d)  
  
d.update(d2)  
print(d)  
  
print(" DICTIONARIES USING COMPREHENSIONS")  
d={i:i\*\*2 for i in range(10)}  
print(d)  
  
import math  
d={i:math.pow(i,3) for i in range(10)}  
print(d)  
  
import math as m  
d={i:m.pow(i,4) for i in range(5)}  
print(d)  
  
d={i:i\*\*2 for i in range(10) if i%2!=0}  
print(d)  
  
d={i:i\*\*2 for i in range(10) if i%2==0}  
print(d)  
  
d={"NAME":{"FNAAME":"SURYA","LNAME":"MAHESH"},"REGNO":124003143}  
print(d["NAME"])  
  
print(d["REGNO"])  
  
d1=dict()  
d1=d.copy()  
print(d1)  
  
print(d.items())  
print(d.keys())  
print(d.values())  
  
d1={i:i\*\*2 for i in range(10) if i%2!=0}  
print(d1)  
d.update(d1)  
print(d)

#DICTIONARIES  
  
salaries = { "surya":10000000 ,"MAHESH":1000000," gopi":100100 ,"pemma": 100000}  
print(type(salaries))  
print(salaries)  
  
#UPDATE

# update() only works for iterable objects  
  
myset=set()  
myset.update([1, 2, 3, 4])  
print(myset)  
  
myset.update({1, 7, 8})  
print(myset)  
  
myset.update({1, 6}, [5, 13])  
print(myset)

# REMOVING ITEMS  
  
myset.discard(10)  
print(myset)  
myset.remove(13)  
print(myset)

#COMMON SET OPERATIONS  
  
a = {2, 4, 5, 9}  
b = {2, 4, 11, 12}  
  
print(a.union(b)) # Values which exist in a or b  
print(a.intersection(b) )# Values which exist in a and b  
  
#The union() and intersection() functions are symmetric methods:  
  
print(a.union(b) == b.union(a))  
print(a.intersection(b) == b.intersection(a))  
print(a.difference(b) == b.difference(a))

#SETS AND DICTIONARIES ARE NOT ACCESSED VIA INDEXES , CANT POSSIBLE FOR INDEXING  
try:  
 for i in range(len(salaries)):  
 print(salaries[i])  
except Exception:  
 print(Exception)  
 print("oopss something went wrong")  
 print("SETS AND DICTIONARIES ARE NOT ACCESSED VIA INDEXES , CANT POSSIBLE FOR INDEXING")  
  
  
# AND DICTIONARIES ARE ACCESSED VIA keys  
print(salaries["surya"])  
print(salaries["pemma"])  
  
print(" lets check what happens if we try to access any pair with non existing key of the dictionary")  
  
try:  
 print(salaries["mahesh"])  
except Exception as e:  
 print(e)  
 print("oopss something went wrong")  
 print("MAY BE REQUESTED KEY NOT FOUND IN THE DICTIONARY ")  
 print("oopss something went wrong")  
 print("TO AVOID THIS ERROR OR EXCEPTION U MAY USE , GET METHOD FOR ACCESSING PARTICULAR PAIR OF DICTIONARY")  
  
print(salaries.get("mahesh"))  
  
print(salaries.get("surya"))  
print(salaries.get("MAHESH"))  
print(salaries.get("pemma"))  
  
print(salaries.get("surya",1))  
print(salaries.get("MUMMA",4))  
print(salaries.get("DUMMA",20))  
print(salaries.get("KAMMA",100))  
  
print(salaries)

#functions

def display(str):  
 print("SYNDICATE MEMBER ",str," PESIRE")  
 print("YEVVA THAGGEDHELE")  
  
def show(str):  
 print("SYNDICATE MEMBER ", str, " PESIRE")  
 print("YODIKI BAYAPADEDHELE")  
a=int(input("ENTER A VALUE :"))  
s=input("ENTER UR NAME ")  
  
if a%2==0:  
 display(s)  
else:  
 show(s)  
  
def showdetails(s):  
 age=int(input("ENTER AGE :"))  
 ht=int(input("ENTER HEIGHT :"))  
 wt = int(input("ENTER WEIGHT :"))  
 det=" \nNAME :"+s+" \nAGE :"+str(age)+" \nHEIGHT :"+str(ht)+" \nWEIGHT :"+str(wt)+" \nVALUE :"+str(a)  
 return det  
  
if s=="pushpa":  
 st=showdetails(s)  
print(st)

#function with return statement and arguments

a=int(input("enter value a: "))  
b=int(input("enter value b: "))  
def add(a,b):  
 c=a+b  
 return c  
result=add(a,b)  
print("RESULT OF SUM OF ",a,b," is :",result)

a=int(input("enter value a: "))  
b=int(input("enter value b: "))  
def add(a,b):  
 c=a+b  
 return c  
result=add(a,b)  
print("RESULT OF SUM OF ",a,b," is :",result)  
  
def display(a,b):  
 print("a,b VALUES :",a,b)  
display(a,b)  
  
  
#default arguments  
c=30  
d=40  
def display(a,b,c,d=100):  
 print("a,b VALUES :",a,b,c,d)  
  
display(10,20,30)  
display(1,2,3,4)  
display(a,b,c,d)  
  
#keyword arguments  
def display(a,b,c):  
 print(a,b,c)  
display(c=300,a=100,b=200)

#passing multiple elements

def add(\*a):  
 total = 0  
 for i in a:  
 total+=i  
 return total  
  
print(add(1,2,3,4,5))

#arbitary function  
def display(\*data):  
 print(data)  
display(11,12,13,14,15,16,17,18,19,20)  
  
#global declarations  
  
def display(a,b):  
 a+=1  
 b+=1  
 print(a,b)  
display(11,12)  
  
z=100  
def show():  
 z=10  
 print("Z = ",z)  
show()  
print("Z = ",z)  
  
z=100  
def show():  
 global z  
 z+=1  
 print("Z = ",z)  
show()  
print("Z = ",z)  
  
  
#lambda functions  
add=lambda a,b:a+b  
print("SUM OF ",a,",",b,"IS : ",add(1,2))

#LEAP YEAR CHECK

def is\_leap(year):  
 leap = False  
 if year%400==0:  
 leap=True  
 elif year%100==0:  
 leap=False  
 elif year%4==0:  
 leap=True  
 return leap  
  
year = int(input())  
print(is\_leap(year))

#FUNCTIONS  
#BUILT IN FUNCTIONS  
  
print(-1)  
print(abs(-1))  
  
  
print(" returns true if all the elements in the iteratable object are true")  
l=[1,0]  
print(all(l))  
  
l=[1,2,3,4,5]  
print(all(l))  
  
print(" returns true if any of the elements in the iteratable object are true")  
l=[1,0]  
print(any(l))  
  
l=[0,0,0,0]  
print(any(l))  
  
print(ascii("a"))  
  
print(bin(10))  
print(oct(10))  
print(hex(10))  
  
print(bool(1))  
print(bool(0))  
print(bool("HELLO"))  
print(bool(""))  
  
#USERDEFINED - FUNCTIONS  
def downloadfile(url):  
 print("URL : ",url)  
 print("DOWNLOADED SUCCESSFULLY")  
  
downloadfile("www.google.com")  
  
#FUNCTIONS V/S LAMBDA FUNCTIONS  
  
def multiply(x,y):  
 print(x\*y)  
  
multiply(20,10)  
  
  
l=lambda x,y: x\*y  
  
l(10,20)  
  
  
#FUNCTIONS AND LAMBDA FUNCTIONS  
  
def cratemultiplier(x):  
 return lambda y:x\*y  
  
multiply=cratemultiplier(10)  
  
print(multiply(555))  
print(multiply(222))

#recursive functions  
#factorial  
def fact(n):  
 if n==0 or n==1:  
 return 1  
 else:  
 return n\*fact(n-1)  
print("FACTORIAL OF 5 IS :",fact(5))  
  
import math  
print("FACTORIAL OF 10 IS :",math.factorial(10))  
  
#gcd  
def gcd(a,b):  
 if a==0:  
 return b  
 elif b==0:  
 return a  
 else:  
 return gcd(b,a%b)  
   
print("GCD OF 10,100 IS :",gcd(10,100))

#TUPLES are immutable  
t=(1,2,3,4,5,6,7,8,9,10)  
print(type(t))  
print("SIZE of the tuple : ",len(t))  
print(t[0])  
  
print(" elements in tuple as follows : ")  
for i in range(len(t)):  
 print(t[i])  
  
print(" tuples are immutable ..... cant change the elements .. but can fetch the elements using index")  
#t[0]=100  
  
print(" DOESNT SUPPORT ITEM ASSIGNMENT ")  
# no replacements / changing the item values  
# iterations in tuple are fast than the list  
# when u are no way needed to change the values of the list then we can go with the tuples  
  
print(" TO ENCHANCE THE SPEED OF THE EXECUTION WE CAN USE TUPLE INSTEAD OF LISTS ")  
  
  
#SETS  
  
s={1,5,2,4,3,8,9,10}  
print(type(s))  
  
print(" SET SIZE : ",len(s))  
  
# insertion order not preserved in sets  
print(s)  
  
print(" SET ELEMENTS AS FOLLOWS : ")  
#always prints in random order  
  
s={100,123,100,100,26,1000,254,100}  
  
#set doesnt allows duplicatess  
  
print(" SET USES HASH , WE IMPROVE THE PERFORMANCE AND FETCH THE ELEMENTS AS FAST AS POSSIBLE")  
print(" as sequence or insertion order not preserved then indexing slicing not possible here")  
  
print(s)  
s.discard(1000)  
s.remove(100)  
s2=s.copy()  
print(s2)  
s3={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}  
s.update(s3)  
print(s)

#sets  
#CONCEPT

*"""  
>>> print set()  
set([])  
  
>>> print set('HackerRank')  
set(['a', 'c', 'e', 'H', 'k', 'n', 'r', 'R'])  
  
>>> print set([1,2,1,2,3,4,5,6,0,9,12,22,3])  
set([0, 1, 2, 3, 4, 5, 6, 9, 12, 22])  
  
>>> print set((1,2,3,4,5,5))  
set([1, 2, 3, 4, 5])  
  
>>> print set(set(['H','a','c','k','e','r','r','a','n','k']))  
set(['a', 'c', 'r', 'e', 'H', 'k', 'n'])  
  
>>> print set({'Hacker' : 'DOSHI', 'Rank' : 616 })  
set(['Hacker', 'Rank'])  
  
>>> print set(enumerate(['H','a','c','k','e','r','r','a','n','k']))  
set([(6, 'r'), (7, 'a'), (3, 'k'), (4, 'e'), (5, 'r'), (9, 'k'), (2, 'c'), (0, 'H'), (1, 'a'), (8, 'n')])  
  
"""*

#CREATING SETS  
  
myset = {"a","b"} # Directly assigning values to a set  
  
print((myset))  
  
myset = set("ab") # Initializing a set  
  
print((myset))  
  
myset = set(("a","b")) # Initializing a set  
  
print((myset))  
  
myset = set(['a', 'b']) # Creating a set from a list  
  
print((myset))

#SETS

s=set()  
  
print(type(s))  
  
s={1,2,3,4}  
  
print(s)

#add function

myset=set()  
  
myset.add('c')  
print(myset)  
myset.add('a') # As 'a' already exists in the set, nothing happens  
print(myset)  
myset.add((5, 4))  
print(myset)

s.add(1)  
s.add(10)  
s.add(100)  
s.add(11001)  
s.add(1200)  
  
print(s)  
  
print(1 in s)  
print( 12 in s)  
print(s.pop())  
  
print(s)

#VARIABLES IN PYTHON  
num=10  
print("ADDRESS OF ", num ," is : ", id(num))  
name="SURYA"  
print("ADDRESS OF ", name , " is : ", id(name))  
a=10  
b=a  
print("ADDRESS OF A,B", id(a),id(b))  
print(" VARIABLES WITH SAME CONTENT POSSESS SAME ADRESS OOPS...NICCCEEE effient data usage")  
print(" ADDRESS NOT DEPENDS ON THE VARIABLE .. IT DEPENDS ON THE DATA OR CONTENT IN IT ")  
print(" if we change value then address of the variable changess naiceee : )")  
  
print(" garbage collection mechanism is implicit in python ")  
print(" PYTHON DONT HAVE IMMUTABLE VARIABLES OR CONSTANTS ")

#data types  
  
#None  
#NUMERIC  
num=10  
print( num ," is OF : ",type(num)," TYPE")  
  
num=3.14242  
print( num ," is OF : ",type(num)," TYPE")  
  
num="HELLO"  
print( num ," is OF : ",type(num)," TYPE")  
  
num='SYNDCATE MEMBER PUSHPA PESIRE'  
print( num ," is OF : ",type(num)," TYPE")  
  
num= 1+ 2j  
print( num ," is OF : ",type(num)," TYPE")  
  
#convesrions  
num=2.22  
  
print(int(num))  
  
num=100  
print(float(num))  
  
num=100  
print(complex(num))  
  
  
num=100  
print(str(num))  
  
  
num=100  
print(bool(num))  
  
"""  
  
some conversions are not possible   
num='SYNDCATE MEMBER PUSHPA PESIRE'  
print(float(num))  
  
"""  
l=[1,2,3,4,5]  
print(l, " IS OF : ",type(l)," TYPE")  
  
l=(1,2,3,4,5)  
print(l, " IS OF : ",type(l)," TYPE")  
  
l={1,2,3,4,5}  
print(l, " IS OF : ",type(l)," TYPE")  
  
l="12345"  
print(l, " IS OF : ",type(l)," TYPE")  
  
print(range(10))  
  
print(" LIST OF NUMBERS < 10")  
print(list(range(10)))  
  
print(" LIST OF EVEN NUMBERS < 10")  
print(list(range(0,10,2)))  
print(" LIST OF ODD NUMBERS < 10")  
print(list(range(1,10,2)))  
  
print(type(range(10)))  
  
#dictionaries  
d={1:"a",2:"b",3:"c",4:"d",5:"e"}  
print(d)  
print(d.keys())  
print(d.values())  
d2=d.copy()  
print(d2)

#operators  
#arithmetic operators  
a=10  
b=20  
print(a,b)  
print(a+b)  
print(a-b)  
print(a\*b)  
print(a/b)  
  
#assignment operators  
a+=b  
b+=2  
print(a,b)  
a-=b  
b-=2  
print(a,b)  
  
a\*=2  
b\*=2  
print(a,b)  
  
a/=2  
b/=2  
print(a,b)  
  
#unary operators  
  
a=-a  
b=-b  
print(a,b)  
print(-a,-b)  
  
print(a>b)  
print(a<b)  
print(a==b)  
  
#logical operators  
a=10  
b=11  
  
#and  
if a>0 and b>0:  
 print(a,b," BOTH ARE POSITIVE ")  
elif a>0 and b<0:  
 print(a," is positive ",b," is negative")  
elif a<0 and b>0:  
 print(a," is negative ",b," is positive")  
else:  
 print(a, b, " BOTH ARE NEGATIVE ")  
  
#or  
if a>0 or b>0:  
 print(" OKAY")  
else:  
 print(" SORRY ")  
  
#not  
print(bool(a))  
print(bool(not a))

#conversions  
a=10  
print(a)  
print(bin(a))  
print(hex(a))  
print(oct(a))  
  
print(hex(0b1010))

print(int(str(bin(5)).replace("0b","")))  
print(int(str(hex(5)).replace("0x","")))  
print(int(str(oct(5)).replace("0o","")))

#bitwise operators  
print(12)  
#compliment  
print(" NEGATION OF 12--->",-12)  
print(" COMPLIMENT OF 12--->",~12)  
#bitwise and &operator  
print(1&0)  
print(1&1)  
print(11&10)  
print(21&5)  
  
#bitwise or | operator  
print(1|0)  
print(1|1)  
print(11|10)  
print(21|5)  
  
#bitwise xor ^ operator  
print(1^0)  
print(1^1)  
print(11^10)  
print(21^5)  
  
#bitwise left shift and right shift operator  
print(10<<2)  
print(26<<3)  
print(100>>2)  
print(1>>2)

#literal  
  
a=None  
print(type(a))

#operators  
  
x=10  
y=15  
z=20  
if(x==y and x==z):  
 print("ALL ARE SAME")  
if x<=y and x<=z:  
 if y<=z:  
 print(" x is smallest")  
if x!=y or y!=z or x!=z:  
 print("ALL ARE NOT IDENTICAL")  
  
#identity operators  
print(id(1))  
print(id(2))  
print(id(1) is id(2))  
print(id(1) is id(1))  
print(id(1) is not id(2))  
print(id(1) is id("1"))  
print(id(1) is not id(1))  
print(id(1) is id(1.0))  
print(id(1) is id(1+0j))  
  
print(type("sastra") is type(100))  
  
# membership operator  
  
print(1 in range(10))  
print(20 not in range(10))

#math functions  
  
import math  
  
print(math.sqrt(25))  
print(math.ceil(1.1))  
print(math.floor(1.9))  
print(math.pow(2,3))  
print(math.log(8,2))  
print(math.factorial(5))  
print(math.gcd(1,5))  
print(math.pi)  
print(math.e)  
  
import math as mahesh  
print(mahesh.factorial(5))

#sum of list of integers < 10  
print(sum(list(range(10))))

#USER INPUT  
  
a=int(input("ENTER VALUE FOR a : "))  
b=int(input("ENTER VALUE FOR a : "))  
print(a,b)  
  
a=float(input("ENTER VALUE FOR a : "))  
b=float(input("ENTER VALUE FOR b : "))  
print(a,b)

# IF , ELIF , ELSE  
  
a=int(input(" enter value a "))  
b=int(input(" enter value b "))  
if a%2==0:  
 print(" a IS EVEN ")  
else:  
 print(" a IS ODD ")  
if b%2==0:  
 print(" b IS EVEN ")  
else:  
 print(" b IS ODD ")  
   
   
if a%2==0 and b%2==0:  
 print(" both are even")  
elif a%2!=0 and b%2!=0:  
 print(" both are odd")  
elif a%2!=0 and b%2==0:  
 print(" a odd b even ")  
elif a%2==0 and b%2!=0:  
 print(" a even b odd")

# IF , ELIF , ELSE WE CAN EVEN USE THE CLOSE BRACKETS   
  
a=int(input(" enter value a "))  
b=int(input(" enter value b "))  
if (a%2==0):  
 print(" a IS EVEN ")  
else:  
 print(" a IS ODD ")  
if (b%2==0):  
 print(" b IS EVEN ")  
else:  
 print(" b IS ODD ")  
  
  
if (a%2==0 and b%2==0):  
 print(" both are even")  
elif (a%2!=0 and b%2!=0):  
 print(" both are odd")  
elif (a%2!=0 and b%2==0):  
 print(" a odd b even ")  
elif (a%2==0 and b%2!=0):  
 print(" a even b odd")

#LOOPS  
#LOOPS  
  
l=[]  
l=[x for x in range(10)]  
print(l)  
  
for x in l:  
 if(x%2==0):  
 if(x%4==0):  
 print(x\*(x+1))  
 elif(x%6==0):  
 print(x\*\*3)  
 else:  
 print(x\*x\*x)  
  
  
a=10  
  
while a!=0:  
 print(a)  
 a=a-1

#FOR LOOP  
i=0  
for i in range(10):  
 print("PYTHON PROGRAMMING")  
  
  
s="PUSHPA"  
  
  
for x in list(range(1,20,2)):  
 print(x,end=" ")  
  
print(" ")  
  
  
for i in s:  
 print(i,end=" ")  
print(" ")  
  
for x in tuple(range(1,20,2)):  
 print(x,end=" ")  
  
print(" ")  
  
  
for x in set(range(1,20,2)):  
 print(x,end=" ")  
  
print(" ")

#BREAK   
  
x=int(input(" HOW MANY CANDIES U WANT: "))  
limit=50  
i=0  
while i<x:  
 if i>limit:  
 print(" LIMIT REACHED ")  
 print(" OUT OF STOCK ")  
 break  
 else:  
 print("CANDY : ", i)  
 i += 1

#CONTINUE   
i=0  
a=int(input("ENTER LIMIT VALUE "))  
for i in range(a):  
 if i%3==0:  
 print(i," is divisible by 3")  
 continue  
 else:  
 print(i," is NOT divisible by 3")

#CONTINUE  
#USING WHILE LOOP  
  
a=int(input("ENTER LIMIT VALUE "))  
i=0  
while i<a:  
 if i%3==0:  
 i = i + 1  
 continue  
   
 print(i, " is NOT divisible by 3")  
 i=i+1

#PASS  
#USING WHILE LOOP  
  
a=int(input("ENTER LIMIT VALUE "))  
i=0  
while i<a:  
 if i%3==0:  
 i = i + 1  
 pass  
 print(i, " ")  
 i=i+1  
  
print("BYE")

#OOPS  
class human:  
  
 height=5.6  
 color="BLACK"  
 weight=60  
 haircolor = "WHITE"  
  
 def \_\_init\_\_(self):  
 print("INIT METHOD CALLED ")  
  
  
 def walk(self):  
 print("WALKING ")  
  
 def talk(self):  
 print("TALKING ")  
  
 def eat(self):  
 print("EATING ")  
  
#print(color)  
  
hobj=human()  
print(hobj.color)  
print(hobj.haircolor)  
hobj.talk()  
hobj.walk()  
hobj.eat()

class computer:  
 #class method  
 def configuaration(self):  
 print(" 16 GB , INTEL PROCESSOR ,1TB STORAGE ")  
  
 def message(self):  
 print(" hello murugan ,syndicate ,member ","pushpa"," pesire")  
  
  
 # class attributes  
  
  
  
comp=computer()  
a=10  
b="pushpa"  
print(type(a))  
print(type(b))  
print(type(comp))  
  
  
comp.configuaration()  
comp.message()  
  
computer.configuaration(comp)  
computer.message(comp)

#OOPS  
  
class computer:  
 #class method  
 def configuaration(self):  
 print(" 16 GB , INTEL PROCESSOR ,1TB STORAGE ")  
  
 def message(self):  
 print(" hello murugan ,syndicate ,member ","pushpa"," pesire")  
  
 def \_\_init\_\_(self):  
 print(" IN INIT METHOD , SIMILAR TO CONSTRUCTOR")  
  
  
 # class attributes  
  
  
  
comp=computer()  
comp2=computer()  
a=10  
b="pushpa"  
print(type(a))  
print(type(b))  
print(type(comp))  
  
computer.configuaration(comp)  
computer.message(comp)  
  
  
comp.configuaration()  
comp.message()  
  
  
comp2.configuaration()  
comp2.message()

#INIT METHOD  
class computer:  
 #class method  
 def configuaration(self):  
 print(self.a," GB ram ",self.b," TB storage")  
  
 def message(self):  
 print(" hello murugan ,syndicate ,member ",self.s," pesire")  
  
 def \_\_init\_\_(self,a,b,s):  
 self.a=a  
 self.b = b  
 self.s = s  
 print(" IN INIT METHOD , SIMILAR TO CONSTRUCTOR")  
  
  
 # class attributes  
  
comp=computer(1,2,"pushpa")  
comp2=computer(10,20,"PUSHPA")  
  
comp.configuaration()  
comp.message()  
  
comp2.configuaration()  
comp2.message()

#NON PARAMETERISED CONSTRUCTOR

class human:  
  
 def \_\_init\_\_(self):  
 self.color= "YELLOW"  
 self.height= 5  
 self.weight= 5  
 self.hair= " ORANGE "  
 print("INIT METHOD WITH OUT PARAMETERS CALLED ")  
  
   
  
 def display(self):  
 print(self.color)  
 print(self.height)  
 print(self.hair)  
 print(self.weight)  
  
  
 def walk(self):  
 print("WALKING ")  
  
  
 def talk(self):  
 print("TALKING ")  
  
  
 def eat(self):  
 print("EATING ")  
  
  
  
hob = human()  
  
print(hob.color)  
print(hob.hair)  
print(hob.height)  
print(hob.weight)  
  
hob.talk()  
hob.walk()  
hob.eat()  
hob.display()

#PARAMETERISED CONSTRUCTOR

class human:  
  
 def \_\_init\_\_(self, a, b, c, d):  
 self.color = a  
 self.height = b  
 self.weight = c  
 self.hair = d  
 print("INIT METHOD WITH PARAMETERS CALLED ")  
  
  
  
 def display(self):  
 print(self.color)  
 print(self.height)  
 print(self.hair)  
 print(self.weight)  
  
  
 def walk(self):  
 print("WALKING ")  
  
  
 def talk(self):  
 print("TALKING ")  
  
  
 def eat(self):  
 print("EATING ")  
  
  
hobj = human("BLACK",10,10,"WHITE")  
  
print(hobj.color)  
print(hobj.hair)  
print(hobj.height)  
print(hobj.weight)  
  
hobj.talk()  
hobj.walk()  
hobj.eat()  
hobj.display()

#CONSTRUCTOR METHOD  
  
class computer:  
 def \_\_init\_\_(self,a,b,c):  
 print(" ATTRIBUTES OF THE CLASS : ")  
 self.a = a  
 self.b = b  
 self.c = c  
 print("\n a :",self.a, "\n b :",self.b, "\n c :", self.c)  
  
 def display(self):  
 print(" ATTRIBUTES OF THE CLASS : ")  
 print("\n a :",self.a, "\n b :",self.b, "\n c :", self.c)  
  
 def getdetails(self):  
 print(" THEN ")  
 self.display()  
 print(" PROVIDE VALUES FOR THE ATTRIBUTES ")  
 self.a= input(" ENTER VALUE OF A ")  
 self.b = input(" ENTER VALUE OF B ")  
 self.c = input(" ENTER VALUE OF C ")  
 print(" NOW ")  
 self.display()  
  
  
  
comp1=computer(1,2,3)  
comp1.display()  
comp1.getdetails()  
print(id(comp1))  
  
comp2=computer(4,5,6)  
comp2.display()  
comp2.getdetails()  
print(id(comp2))

#class and objects  
  
class person:  
  
 def \_\_init\_\_(self,name,age,height,weight):  
  
 self.name=name  
 self.age=age  
 self.height=height  
 self.weight=weight  
 self.display()  
  
  
 def display(self):  
 print("DETAILS OF : ", self.name)  
 print("NAME : ", self.name)  
 print("AGE : ", self.age)  
 print("HEIGHT : ", self.height)  
 print("WEIGHT : ", self.weight)  
  
 def getdetails(self):  
 print(" provides details for the attributes of the class")  
  
 self.name=input(" ENTER NAME OF THE PERSON")  
 self.age=input(" ENTER AGE OF THE PERSON")  
 self.height=input(" ENTER HEIGHT OF THE PERSON")  
 self.weight=input(" ENTER WEIGHT OF THE PERSON")  
  
 def update(self,n,a,h,w):  
 self.name=n  
 self.age=a  
 self.height=h  
 self.weight=w  
  
  
 def check(self,other):  
  
 if self.name==other.name:  
 print(" BOTH ",self.name,other.name," WITH SAME NAME")  
 else:  
 print(" BOTH ",self.name,other.name, " WITH DIFFERENT NAME")  
  
 if self.age==other.age:  
 print(" BOTH ",self.name,other.name," WITH SAME AGE")  
 else:  
 print(" BOTH ",self.name,other.name," WITH DIFFERENT AGE")  
  
 if self.height==other.height:  
 print(" BOTH ",self.name,other.name," WITH SAME HEIGHT")  
  
 else:  
 print(" BOTH ",self.name,other.name, " WITH DIFFERENT HEIGHT")  
  
 if self.weight==other.weight:  
 print(" BOTH ",self.name,other.name," WITH SAME WEIGHT")  
  
 else:  
 print(" BOTH ",self.name,other.name," WITH DIFFERENT WEIGHT")  
  
  
p1=person("SURYA",19,5.6,60)  
  
#p1.getdetails()  
p1.display()  
p1.update("MAHESH",16,6,60)  
p1.display()  
  
p2=person("SUKU",16,6,66)  
p1.check(p2)  
  
p1.name="SANKU"  
p2.name="SHANKINI"  
  
p1.display()  
p2.display()  
  
print(" AFTER UPDATING NAMES ")  
print(p1.name)  
print(p2.name)

#instance variables , CLASS VARIABLES (STATIC VARIABLES  
print("CLASS VARIABLES ALSO CALLED AS STATIC VARIABLES ")  
class car:  
  
 #class variables  
 #inside the class outside the init  
 wheels=4  
 color="BLACK"  
  
 def \_\_init\_\_(self):  
 # INSTANCE variables  
 # inside the class AND INSIDE the init  
 self.milage=60  
 self.modelname="bmw"  
 self.cost=100000  
  
 def display(self):  
 print(" MILLAGE : ",self.milage," MODEL NAME :",self.modelname," COST :",self.cost)  
  
c1=car()  
c2=car()  
c1.display()  
c2.display()  
  
c1.modelname="HOHOHO"  
c1.cost=20000000  
c1.milage=1000  
  
c2.modelname="BENZ"  
c2.cost=5000000  
c2.milage=100  
  
c1.display()  
print(c1.wheels," WHEELS")  
print(c1.color," COLOR")  
  
  
c2.display()  
print(c2.wheels," WHEELS")  
print(c2.color," COLOR")

# METHODS IN PYTHON  
#INSTANCE METHODS  
#CLASS METHODS  
#STATIC METHODS  
  
#getters , setters  
  
class student:  
  
 school="NARAYANA SCHOOL"  
  
 def \_\_init\_\_(self,m1,m2,m3):  
 self.m1 = m1  
 self.m2 = m2  
 self.m3 = m3  
  
 def average(self):  
 avg=(self.m1+self.m2+self.m3)/3  
 print("AVERAGE : ",avg)  
  
 def display(self):  
 print(" M1 : ",self.m1," M2: ",self.m2," M3 : ",self.m3)  
  
 def setm1(self,m1):  
 self.m1=m1  
  
 def setm2(self,m2):  
 self.m2=m2  
  
 def setm3(self,m3):  
 self.m3=m3  
  
 def getm1(self):  
 return self.m1  
  
 def getm2(self):  
 return self.m2  
  
 def getm3(self):  
 return self.m3  
  
 @classmethod  
 def schoolinfo(cls):  
 print(cls.school)  
  
  
s1=student(100,100,100)  
s2=student(50,50,50)  
s3=student(35,35,35)  
  
s1.average()  
s1.display()  
print(s1.school)  
  
s2.average()  
s2.display()  
print(s2.school)  
  
s3.average()  
s3.display()  
print(s3.school)  
  
s1.setm1(70)  
s1.setm2(80)  
s1.setm3(75)  
  
print(s1.getm3())  
print(s1.getm2())  
print(s1.getm1())  
  
s1.schoolinfo()  
s2.schoolinfo()

# INNER CLASS  
#INNER CLASS OBJECT CREATING INSIDE THE INIT METHOD OF THE OUTER CLASS

class student:  
  
 def \_\_init\_\_(self,name,rollnum):  
 self.name=name  
 self.rollnum=rollnum  
 self.mylap=self.laptop()  
  
  
 def show(self):  
 print("NAME : ",self.name," ROLLNUM : ",self.rollnum)  
 self.mylap.showin()  
  
 class laptop:  
  
 def \_\_int\_\_(self):  
 self.brand="LENOVO"  
 self.processor=" AMD RYZEN 5 "  
 self.cost=45000  
  
  
 def showin(self):  
 print("BRAND NAME : ",self.brand," PROCESSOR : ",self.processor," COST : ",self.cost)  
  
  
  
s1=student("MAHESH",3143)  
s2=student("SURYA ",3144)  
  
s1.mylap.brand="HP"  
s1.mylap.cost=100000  
s1.mylap.processor="INTEL"  
  
  
s2.mylap.brand="HP"  
s2.mylap.cost=100000  
s2.mylap.processor="INTEL"  
  
  
  
s1.show()  
s2.show()

#INHERITANCE

#SINGLE LEVEL INHERITANCE  
class baseclass:  
 a=100  
 b=200  
  
 def \_\_init\_\_(self):  
 print(" BASE CLASS INIT METHOD ")  
 self.a=1000  
 self.b=2000  
  
 def display(self):  
 print("BASE CLASS DISPLAY METHOD")  
 print(self.a)  
 print(self.b)  
  
class derivedclass(baseclass):  
 c=300  
 d=400  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" DERIVED CLASS INIT METHOD ")  
 self.a=3333  
 self.b=4444  
 self.c=3000  
 self.d=4000  
  
 def display(self):  
 super().display()  
 print("DERIVED CLASS DISPLAY METHOD ")  
 print(self.c)  
 print(self.d)  
  
  
b=baseclass()  
d=derivedclass()  
b.display()  
d.display()  
  
print(b.a," - a")  
print(b.b," - b")  
  
print(d.a," - a")  
print(d.b," - b")  
print(d.c," - c")  
print(d.d," - d")

#SINGLE LEVEL INHERITANCE  
class parent:  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class child(parent):  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
print("PARENT CLASS")  
p=parent()  
p.feature1()  
p.feature2()  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()

#MULTI LEVEL INHERITANCE  
  
class parent:  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class child(parent):  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
class grandchild(child):  
 def feature5(self):  
 print("FEATURE 5 IS WORKING")  
  
 def feature6(self):  
 print("FEATURE 6 IS WORKING")  
  
  
print("PARENT CLASS")  
p=parent()  
p.feature1()  
p.feature2()  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()  
  
  
print("GRAND CHILD CLASS")  
g=grandchild()  
g.feature1()  
g.feature2()  
g.feature3()  
g.feature4()  
g.feature5()  
g.feature6()

#HEIRARCHIAL INHERITANCE  
class baseclass:  
 a=100  
 b=200  
  
 def \_\_init\_\_(self):  
 print(" BASE CLASS INIT METHOD ")  
 self.a=1000  
 self.b=2000  
  
 def display(self):  
 print("BASE CLASS DISPLAY METHOD")  
 print(self.a)  
 print(self.b)  
  
class derivedclass(baseclass):  
 c=300  
 d=400  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" DERIVED CLASS INIT METHOD ")  
 self.a=3333  
 self.b=4444  
 self.c=3000  
 self.d=4000  
  
 def display(self):  
 super().display()  
 print("DERIVED CLASS DISPLAY METHOD ")  
 print(self.c)  
 print(self.d)  
  
  
class derivedclass2(baseclass):  
 c=300  
 d=400  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" DERIVED2 CLASS INIT METHOD ")  
 self.a=3333  
 self.b=4444  
 self.c=3000  
 self.d=4000  
  
 def display(self):  
 super().display()  
 print("DERIVED2 CLASS DISPLAY METHOD ")  
 print(self.c)  
 print(self.d)  
  
  
b=baseclass()  
b.display()  
print(b.a," - a")  
print(b.b," - b")  
  
  
d=derivedclass()  
d.display()  
print(d.a," - a")  
print(d.b," - b")  
print(d.c," - c")  
print(d.d," - d")  
  
  
d2=derivedclass2()  
b.display()  
print(d2.a," - a")  
print(d2.b," - b")  
print(d2.c," - c")  
print(d2.d," - d")

#HEIRARCHIAL INHERITANCE

#HEIRARCHIAL INHERITANCE  
class baseclass:  
 a=100  
 b=200  
  
 def \_\_init\_\_(self):  
 print(" BASE CLASS INIT METHOD ")  
 self.a=1000  
 self.b=2000  
  
 def display(self):  
 print("BASE CLASS DISPLAY METHOD")  
 print(self.a)  
 print(self.b)  
  
class derivedclass(baseclass):  
 c=300  
 d=400  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" DERIVED CLASS INIT METHOD ")  
 self.a=3333  
 self.b=4444  
 self.c=3000  
 self.d=4000  
  
 def display(self):  
 super().display()  
 print("DERIVED CLASS DISPLAY METHOD ")  
 print(self.c)  
 print(self.d)  
  
  
class derivedclass2(baseclass):  
 c=300  
 d=400  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" DERIVED2 CLASS INIT METHOD ")  
 self.a=3333  
 self.b=4444  
 self.c=3000  
 self.d=4000  
  
 def display(self):  
 super().display()  
 print("DERIVED2 CLASS DISPLAY METHOD ")  
 print(self.c)  
 print(self.d)  
  
  
b=baseclass()  
b.display()  
print(b.a," - a")  
print(b.b," - b")  
  
  
d=derivedclass()  
d.display()  
print(d.a," - a")  
print(d.b," - b")  
print(d.c," - c")  
print(d.d," - d")  
  
  
d2=derivedclass2()  
b.display()  
print(d2.a," - a")  
print(d2.b," - b")  
print(d2.c," - c")  
print(d2.d," - d")

#MULTIPLE INHERITANCE  
  
class parent1:  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class parent2:  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
class child(parent1,parent2):  
 def feature5(self):  
 print("FEATURE 5 IS WORKING")  
  
 def feature6(self):  
 print("FEATURE 6 IS WORKING")  
  
  
print("PARENT1 CLASS")  
p1=parent1()  
p1.feature1()  
p1.feature2()  
  
print("PARENT2 CLASS")  
p2=parent2()  
p2.feature3()  
p2.feature4()  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()

#CONSTRUCTOR IN INHERITANCE  
  
  
#SINGLE LEVEL INHERITANCE  
class parent:  
 def \_\_init\_\_(self):  
 print(" PARENT CLASS INIT CALLED......")  
  
  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class child(parent):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" CHILD CLASS INIT CALLED.........")  
  
  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
print("PARENT CLASS")  
p=parent()  
p.feature1()  
p.feature2()  
  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()

#CONSTRUCTOR IN INHERITANCE  
  
#MULTIPLE INHERITANCE  
  
class parent1:  
 def \_\_init\_\_(self):  
 print(" PARENT 1 CLASS INIT CALLED")  
  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class parent2:  
 def \_\_init\_\_(self):  
 print(" PARENT 2 CLASS INIT CALLED")  
  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
class child(parent1,parent2):  
   
 def \_\_init\_\_(self):  
 print(" CHILD CLASS INIT CALLED")  
  
 def feature5(self):  
 print("FEATURE 5 IS WORKING")  
  
 def feature6(self):  
 print("FEATURE 6 IS WORKING")  
  
  
print("PARENT1 CLASS")  
p1=parent1()  
p1.feature1()  
p1.feature2()  
  
print("PARENT2 CLASS")  
p2=parent2()  
p2.feature3()  
p2.feature4()  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()

#CONSTRUCTOR IN INHERITANCE  
#MULTIPLE INHERITANCE  
  
class parent1:  
 def \_\_init\_\_(self):  
 print(" PARENT 1 CLASS INIT CALLED")  
  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class parent2:  
 def \_\_init\_\_(self):  
 print(" PARENT 2 CLASS INIT CALLED")  
  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
class child(parent1,parent2):  
  
 def feature5(self):  
 print("FEATURE 5 IS WORKING")  
  
 def feature6(self):  
 print("FEATURE 6 IS WORKING")  
  
  
print("PARENT1 CLASS")  
p1=parent1()  
p1.feature1()  
p1.feature2()  
  
print("PARENT2 CLASS")  
p2=parent2()  
p2.feature3()  
p2.feature4()  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()

#CONSTRUCTOR IN INHERITANCE  
  
#MULTIPLE INHERITANCE

#BIASED AND UNFAIR HERE WHEN CHILD CLASS CREATED AND SUPER().INIT

#CALLED THEN WE ARE AMBIGIOUS THAT PARENT 1 OR PARENT2 CLASS INIT CALLED ..

#BUT SYSTEM IS BIASED AND UNFAIR   
  
class parent1:  
 def \_\_init\_\_(self):  
 print(" PARENT 1 CLASS INIT CALLED")  
  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class parent2:  
 def \_\_init\_\_(self):  
 print(" PARENT 2 CLASS INIT CALLED")  
  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
class child(parent1,parent2):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" CHILD CLASS INIT CALLED")  
  
 def feature5(self):  
 print("FEATURE 5 IS WORKING")  
  
 def feature6(self):  
 print("FEATURE 6 IS WORKING")  
  
  
print("PARENT1 CLASS")  
p1=parent1()  
p1.feature1()  
p1.feature2()  
  
print("PARENT2 CLASS")  
p2=parent2()  
p2.feature3()  
p2.feature4()  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()

#CONSTRUCTOR IN INHERITANCE  
  
#MULTIPLE INHERITANCE  
#METHOD RESOLUTION ORDER  
  
class parent1:  
 def \_\_init\_\_(self):  
 print(" PARENT 1 CLASS INIT CALLED")  
   
  
 def feature1(self):  
 print("FEATURE 1 IS WORKING")  
  
 def feature2(self):  
 print("FEATURE 2 IS WORKING")  
  
class parent2:  
 def \_\_init\_\_(self):  
 print(" PARENT 2 CLASS INIT CALLED")  
  
 def feature3(self):  
 print("FEATURE 3 IS WORKING")  
  
 def feature4(self):  
 print("FEATURE 4 IS WORKING")  
  
class child(parent1,parent2):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 print(" CHILD CLASS INIT CALLED")  
  
 def feature5(self):  
 print("FEATURE 5 IS WORKING")  
  
 def feature6(self):  
 print("FEATURE 6 IS WORKING")  
  
  
print("PARENT1 CLASS")  
p1=parent1()  
p1.feature1()  
p1.feature2()  
  
print("PARENT2 CLASS")  
p2=parent2()  
p2.feature3()  
p2.feature4()  
  
print("CHILD CLASS")  
c=child()  
c.feature1()  
c.feature2()  
c.feature3()  
c.feature4()

#DUCK TYPING  
  
class pycharm:  
  
 def execute(self):  
 print("COMPILING ")  
 print("RUNNING ")  
  
class mycharm:  
  
 def execute(self):  
 print("SPELLCHECK")  
 print("CONVENTION CHECK")  
 print("COMPILING ")  
 print("RUNNING ")  
  
  
class laptop:  
  
 def code(self,myide):  
 myide.execute()  
  
  
myide=pycharm()  
lap1=laptop()  
lap1.code(myide)  
  
myide=mycharm()  
lap1=laptop()  
lap1.code(myide)

# POLYMORPHISM IN INHERITANCE  
#DUCK TYPING  
#PROVIDING THE TYPE OF THE COMPILER DYNAMICALLY AT RUN TIME

class pycharm:  
  
 def execute(self):  
 print("COMPILING IN PYCHARM")  
 print("RUNNING IN PYCHARM ")  
  
class mycharm:  
  
 def execute(self):  
 print("SPELLCHECK IN MYCHARM")  
 print("CONVENTION CHECK IN MYCHARM")  
 print("COMPILING IN MYCHARM")  
 print("RUNNING IN MYCHARM ")  
  
  
class laptop:  
  
 def code(self,myide):  
 myide.execute()  
  
  
a=int(input("\n ENTER 1 FOR PYCHARM IDE \n ENTER 2 FOR MYCHARM IDE "))  
if(a==1):  
 myide = pycharm()  
 lap1 = laptop()  
 lap1.code(myide)  
  
elif(a==2):  
 myide = mycharm()  
 lap1 = laptop()  
 lap1.code(myide)  
  
else:  
 print(" INVALID INPUT ")  
 print(" DEFAULT PROVIDING MYCHARM COMPILER ")  
 myide = pycharm()  
 lap1 = laptop()  
 lap1.code(myide)

# OPERATOR OVERLOADING

print(" INTEGERS ")  
a=10  
b=5  
  
print(a+b)  
print(int.\_\_add\_\_(a,b))  
print(a-b)  
print(int.\_\_sub\_\_(a,b))  
print(a\*b)  
print(int.\_\_mul\_\_(a,b))  
print(a/b)  
print(int.\_\_divmod\_\_(a,b))  
  
print(" FLOATING POINT VARIABLES ")  
a=1.5  
b=2.5  
  
print(a+b)  
print(float.\_\_add\_\_(a,b))  
print(a-b)  
print(float.\_\_sub\_\_(a,b))  
print(a\*b)  
print(float.\_\_mul\_\_(a,b))  
print(a/b)  
print(float.\_\_divmod\_\_(a,b))  
  
  
print(" STRING DATA TYPES ")  
a="HELLO"  
b="WORLD"

print(a+b)  
print(str.\_\_add\_\_(a,b))  
  
print(str.\_\_mul\_\_(a,5))  
print(str.\_\_mul\_\_(b,5))  
  
print(a\*5)  
print(b\*5)

#CONVRTING TO STRING TYPE

a=10  
  
print(a)  
print(a.\_\_str\_\_())

# OPERATOR OVERLOADING  
  
class student:  
  
 def \_\_init\_\_(self):  
 self.m1=0  
 self.m2=0  
  
 def \_\_init\_\_(self,m1,m2):  
 self.m1=m1  
 self.m2=m2  
  
 def \_\_add\_\_(self, other):  
 m1 = self.m1+other.m1  
 m2 = self.m2 + other.m2  
 s3=student(m1,m2)  
  
 return s3  
  
 def \_\_sub\_\_(self, other):  
 m1 = self.m1 - other.m1  
 m2 = self.m2 - other.m2  
 s3 = student(m1, m2)  
  
 return s3  
  
 def \_\_mul\_\_(self, other):  
 m1 = self.m1 \* other.m1  
 m2 = self.m2 \* other.m2  
 s3 = student(m1, m2)  
  
 return s3  
  
 def \_\_eq\_\_(self, other):  
 if(self.m1==other.m1 and self.m2==other.m2 ):  
 return True  
 else:  
 return False  
  
 def \_\_gt\_\_(self, other):  
 if (self.m1 > other.m1 and self.m2 > other.m2):  
 return True  
 else:  
 return False  
  
 def \_\_ge\_\_(self, other):  
 if (self.m1 >= other.m1 and self.m2 >= other.m2):  
 return True  
 else:  
 return False  
  
 def \_\_lt\_\_(self, other):  
 if (self.m1 < other.m1 and self.m2 < other.m2):  
 return True  
 else:  
 return False  
  
 def \_\_le\_\_(self, other):  
 if (self.m1 <= other.m1 and self.m2 <= other.m2):  
 return True  
 else:  
 return False  
  
 def \_\_str\_\_(self):  
 return '{} {}'.format(self.m1,self.m2)  
  
  
  
 def display(self):  
 print("M1 : ",self.m1)  
 print("M2 : ",self.m2)  
  
  
s1=student(100,100)  
s2=student(50,50)  
s1.display()  
s2.display()  
  
s3=s1+s2  
s3.display()  
  
s3=s1-s2  
s3.display()  
  
s3=s1\*s2  
s3.display()  
  
if(s1==s2):  
 print("both the students are with SAME marks ")  
else:  
 print("both the students are with DIFFERENT marks ")  
  
if(s1>s2):  
 print("STUDENT IS GREATER THAN STUDENT 2 ")  
else:  
 print("STUDENT IS NOT GREATER THAN STUDENT 2")  
  
if(s1>=s2):  
 print("STUDENT IS GREATER THAN EQUAL TO STUDENT 2 ")  
else:  
 print("STUDENT IS LESS THAN STUDENT 2")  
  
if(s1<s2):  
 print("STUDENT IS LESSER THAN STUDENT 2 ")  
else:  
 print("STUDENT IS NOT LESSER THAN STUDENT 2")  
  
if(s1<=s2):  
 print("STUDENT IS LESSER THAN EQUAL TO STUDENT 2 ")  
else:  
 print("STUDENT IS GREATER THAN STUDENT 2")  
  
print(s1.\_\_str\_\_())  
print(s2.\_\_str\_\_())

# METHOD OVERLOADING   
  
class student:  
  
 def \_\_init\_\_(self,m1,m2):  
 self.m1=m1  
 self.m2=m2  
  
 def display(self):  
 print(self.m1,self.m2)  
  
  
 def sum(self,a=None,b=None,c=None):  
  
 if a!=None and b!=None and c!=None:  
 sum = a + b + c  
 elif a!=None and b!=None:  
 sum = a + b  
 else:  
 sum=a  
 return sum  
  
  
s1=student(100,100)  
s1.display()  
  
print(s1.sum(10,20,30))  
print(s1.sum(200,300))  
print(s1.sum(100,200))  
print(s1.sum(100))

#METHOD OVERLOADING USING DEFAULT PARAMETERS  
#COMPILE TIME POLYMORPHISM  
  
class demo:  
  
 def add(self,a=0,b=0,c=0,d=0):  
 add=a+b+c+d  
 print("ADDITION =",add)  
  
 def sum(self, a=None, b=None, c=None):  
 if a != None and b != None and c != None:  
 sum = a + b + c  
 elif a != None and b != None:  
 sum = a + b  
 else:  
 sum = a  
 print("SUM =", sum)  
  
  
 def display(self, a=None, b=None, c=None):  
 if a != None and b != None and c != None:  
 print(a,b,c)  
 elif a != None and b != None:  
 print(a,b)  
 elif a != None:  
 print(a)  
 else:  
 print("NO PARAMETER FOUND ")  
  
  
 def show(self,a=None,b=None,c=None,d=None):  
 print("A=",a,"B=",b,"C=",c,"D=",d)  
  
  
d=demo()  
d.add()  
d.add(1)  
d.add(1,2)  
d.add(1,2,3)  
d.add(1,2,3,4)  
  
d.sum()  
d.sum(1)  
d.sum(1,2)  
d.sum(1,2,3)  
  
d.display()  
d.display("MAHESH")  
d.display("MAHESH","SURYA")  
d.display("MAHESH","SURYA","SANKU")  
  
  
d.show()  
d.show(1)  
d.show(1,2)  
d.show(1,2,3)  
d.show(1,2,3,4)

#METHOD OVERLOADING

class human:  
  
 def \_\_init\_\_(self, a, b, c, d):  
 self.color = a  
 self.height = b  
 self.weight = c  
 self.hair = d  
 print("INIT METHOD WITH PARAMETERS CALLED ")  
  
  
  
 def display(self):  
 print(self.color)  
 print(self.height)  
 print(self.hair)  
 print(self.weight)  
  
 def walk(self,name=None):  
 print(str(name)+" WALKING ")  
  
 def talk(self,name=None):  
 print(str(name)+" TALKING ")  
  
  
 def eat(self,name=None):  
 print(str(name)+ " EATING ")  
  
  
hobj = human("BLACK",10,10,"WHITE")  
  
print(hobj.color)  
print(hobj.hair)  
print(hobj.height)  
print(hobj.weight)  
  
hobj.talk()  
hobj.walk("MAHESH")  
hobj.walk()  
hobj.eat("SURYA")  
hobj.eat()  
hobj.display()

# METHOD OVERRIDING  
  
class parent:  
 def show(self):  
 print(" IN PARENT SHOW ")  
  
class child(parent):  
 pass  
  
  
  
  
  
p1=parent()  
p1.show()  
  
c1=child()  
c1.show()

#RUNTIME POLYMORPHISM  
#METHOD OVERRIDING  
  
class parent:  
 def \_\_init\_\_(self):  
 print("PARENT CLASS INIT METHOD")  
  
 def transport(self):  
 print("PARENT CLASS TRANSPORT METHOD")  
 print("CYCLE")  
 print("RIKSHAW")  
  
class child(parent):  
 def \_\_int\_\_(self):  
 super().\_\_init\_\_()  
 print("CHILD CLASS INIT METHOD")  
  
 def transport(self):  
 super().transport()  
 print("CHILD CLASS TRANSPORT METHOD")  
 print("MOTOR - CYLE")  
 print("AUTO - RIKSHAW")  
  
  
p=parent()  
p.transport()  
  
c=child()  
c.transport()

# METHOD OVERRIDING  
  
class parent:  
 def show(self):  
 print(" IN PARENT SHOW ")  
  
class child(parent):  
 def show(self):  
 print(" IN CHILD SHOW ")  
  
  
p1=parent()  
p1.show()  
  
c1=child()  
c1.show()

#ABSTRACTION

# HAVE TO IMPORT ABC,abstractmethod from abc module

#decorator @abstractmethod should be provided at each and every abstract

#methods of the abstract class

#ABSTRCT CLASS CANNOT BE INSTANTIATED

#CONCRETE CLASSES CAN BE INSTANTIATED

#CONCRETE CLASS IS CLASS THAT EXTENDS THE ABSTRACT CLASS AND OVERRIDING

#ALL THE METHODS OF THE CLASS AND PROVIDING THE IMPELEMENTATION FOR THE

#ALL THE ABSTRACT METHODS … NOW THIS CONCRETE CLASS CAN BE INSTANTIATED

# OBJECTS CAN BE CREATED FOR THIS CNCRETE METHOD

IMP:

IF U FOR GOT TO OVER RIDE PARTICULAR ABSTRACT METHOD FROM THE ABSTRACT CLASS

THEN THE EXTENDING CLASS NOT PERFECTLY CONCRETE , IT IS STILL ABSTRACT

So WHEN EXTENDING A ABSTRACT CLASS INHERIT AND OVEERRIDE ALL THE METHODS OF THE CLASS AND PROVIDE IMPLEMENTATION OF ALL THE ABSTRAT METHODS

#from abc import ABC,abstractmethod

# @abstractmethod

MUST AND SHOULDUU

from abc import ABC,abstractmethod  
  
class abstractdemo(ABC):  
 @abstractmethod  
 def houseintrest(self):  
 None  
  
 @abstractmethod  
 def vehicleintrest(self):  
 None  
  
class sbi(abstractdemo):  
 def \_\_init\_\_(self):  
 print("STATE BANK OF INDIA PVT.LTD")  
  
 def houseintrest(self):  
 print("HOUSING INTREST 10%")  
  
  
 def vehicleintrest(self):  
 print("VEHICLE INTREST 20%")  
  
class lvb(abstractdemo):  
 def \_\_init\_\_(self):  
 print("LAXMI VILAS BANK PVT.LTD")  
  
 def houseintrest(self):  
 print("HOUSING INTREST 22%")  
  
  
 def vehicleintrest(self):  
 print("VEHICLE INTREST 33%")  
  
  
#obj=abstractdemo() cannot instantiate the abstract class directly  
#OBJECTS CAN BE CREATED AND A CLASS CAN BE INSTANTIATED ONLY FROM THE CONCRETE CLASS  
  
s=sbi()  
s.vehicleintrest()  
s.houseintrest()  
  
l=lvb()  
l.vehicleintrest()  
l.houseintrest()

#DATA HIDING  
  
class demo:  
 a=10  
 b=100  
 \_\_c=1000  
  
 def display(self):  
 print("DISPLAY METHOD IN demo CLASS")  
  
 def \_\_display(self):  
 print("PRIVATE SHOW METHOD IN demo CLASS")  
 print("PRIVATE MEMBERS OF DEMO CLASS ARE : ")  
 print("c ",self.\_\_c)  
  
  
  
obj=demo()  
obj.display()  
#obj.show()  
#print(obj.c)  
print(obj.b)  
print(obj.a)

#EXCEPTION HANDLING

print("HELLO MURUGAN NAANU SYNDICATE MEMBER PUSHPA PESIRE...EXCEPTION HANDLING")  
# EXCEPTION HANDLING  
#TRY EXCEPT BLOCKS  
try:  
 a = int(input(" ENTER A VALUE FOR NUMERATOR "))  
 b = int(input(" ENTER A VALUE FOR DENOMINATOR "))  
 print("WORKING STARTED")  
 print(a / b)  
 print("WORKING ENDED ")  
   
except ZeroDivisionError as e:  
 print("YOU CANNOT DIVIDE /0",e)  
 print("WORKING ENDED WITH AN EXCEPTION")  
  
except ValueError as e:  
 print("VALUE ERROR ", e)  
 print("WORKING ENDED WITH AN EXCEPTION")  
  
except Exception as e:  
 print("SOMETHING WENT WRONG OOPS ", e)  
 print("WORKING ENDED WITH AN EXCEPTION")  
  
finally:  
 print("BYE BYE :)")  
  
""" SAME SAME  
except Exception as e:  
 print(e,"YOU CANNOT DIVIDE /0")  
   
except Exception :  
 print(e,"YOU CANNOT DIVIDE /0")  
"""

try:  
 print(10+"surya")  
except:  
 print("OOPS....Exception")  
else:  
 print("NO EXCEPTION")  
  
try:  
 print(10+"surya")  
except Exception:  
 print(Exception)  
else:  
 print("NO EXCEPTION")  
  
try:  
 print(10+"surya")  
except Exception as e:  
 print(e)  
else:  
 print("NO EXCEPTION")  
finally:  
 print("BYE BYE")

#ZeroDivisionError  
  
try:  
 print(1/0)  
except Exception:  
 print("OOPS , EXCEPTION RAISED : ",Exception)  
  
try:  
 print(1/0)  
except Exception as e:  
 print("OOPS , EXCEPTION RAISED : ", e)  
try:  
 print(1/0)  
except ZeroDivisionError as e:  
 print("OOPS , EXCEPTION RAISED : ", e)  
  
#CUSTOMISED EXCEPTION MESSAGE  
try:  
 print(1/0)  
except:  
 print(" ZERO SHOULD NOT BE IN DENOMINATOR ")  
  
#ValueError  
  
print(" IF U GIVE INPUT AS INTEGER ITS FINE ELSE , IT RAISES VALUE ERROR ")  
try:  
 a=int(input("ENTER A VALUE : "))  
except ValueError as e:  
 print("OOPS , EXCEPTION RAISED : ", e)  
  
#if we know about the type of error, then name\_of\_error as e else u can use  
# Exception as e ...  
  
print(" IF U GIVE INPUT AS INTEGER ITS FINE ELSE , IT RAISES VALUE ERROR ")  
try:  
 a=int(input("ENTER A VALUE : "))  
except Exception as e:  
 print("OOPS , EXCEPTION RAISED : ", e)  
  
print(" IF U GIVE INPUT AS FLOAT ITS FINE ELSE , IT RAISES VALUE ERROR ")  
try:  
 a=float(input("ENTER A VALUE : "))  
except ValueError as e:  
 print("OOPS , EXCEPTION RAISED : ",e)  
  
#IndexError  
  
try:  
 l=[1,2,3,4]  
 print(l[10])  
  
except IndexError as e:  
 print("OOPS , EXCEPTION RAISED : ", e)  
  
  
class demo:  
 a=10  
  
obj=demo()  
#a exist , b not exists raises exception  
try:  
 print(obj.a)  
 print(obj.b)  
except Exception as e:  
 print("OOPS , EXCEPTION RAISED : ", e)  
  
#AttributeError  
  
try:  
 print(obj.a)  
 print(obj.c)  
except AttributeError as e:  
 print("OOPS , EXCEPTION RAISED : ", e)  
  
#FileNotFoundError  
  
#for providing customised statements for exceptions  
try:  
 fo=open("surya.txt","r")  
 print(fo.read())  
 fo.close()  
 print("FILES MOWA")  
except:  
 print(" FILE NOT FOUND ERROR ")  
  
try:  
 fo=open("surya.txt","r")  
 print(fo.read())  
 fo.close()  
 print("FILES MOWA")  
except FileNotFoundError as e :  
 print(" FILE NOT FOUND ERROR EXCEPTION RAISED ",e )  
  
#NameError  
  
try:  
 a=10  
 print(z)  
except NameError as e :  
 print("EXCEPTION RAISED ",e )  
  
try:  
 a=10  
 print("a value = ",a)  
except Exception as e:  
 print(" EXCEPTION RAISED ",e)  
else:  
 print(" HOHO NO EXCEPTIONS RAISED HERE ...")  
  
#ELSE BLOCK  
#finally block  
finally:  
 print("\n\n BYE BYE BAAABU BYE BYE BYE BYE , BYE BYE BAABU...\n\n")

#EXCEPTION HANDLING

weeks=("sunday","monday","tuesday","wednesday","thursday","friday","saturday")  
print(weeks)  
print(type(weeks))  
  
try:  
 weeks[2]="SURYADAY"  
except:  
 print("SOMETHING WENT WRONG")  
else:  
 print("NO EXCEPTION")  
finally:  
 print("BYE BYE ")

class hello():  
 def run(self):  
 for i in range(5):  
 print("HELLO")  
  
  
class hii:  
 def run(self):  
 for i in range(5):  
 print("HII")  
  
  
t1=hello()  
t2=hii()  
  
t1.run()  
t2.run()

#MULTI THREADING  
  
from threading import \*  
from time import sleep  
  
class hello(Thread):  
 def run(self):  
 for i in range(50):  
 print("HELLO")  
 sleep(2)  
  
  
class hii(Thread):  
 def run(self):  
 for i in range(10):  
 print("HII")  
 sleep(2)  
  
  
t1 = hello()  
t2 = hii()  
  
t1.start()  
sleep(0.2)  
t2.start()  
  
t1.join()  
t2.join()  
print("BYE")

#COMMENTS IN PYTHON  
  
# for single line comments  
  
*"""  
 for multiline comments  
  
 """*

USERDEFINED EXCEPTIONS

age=int(input("ENTER AGE : "))  
  
try:  
 if age < 18:  
 raise Exception(" ONLY ADULTS ARE ALLOWED TO VOTE ")  
 else:  
 print(" U CAN CAST YOUR VOTE ! ")  
except Exception as e:  
 print("OOPS !",e)  
 print(" SORRY ! ")

#IMPORT MATH LIBRARY

# Import math Library  
import math  
  
# Print the value of Euler e  
print(math.e)  
  
# Print the value of pi  
print(math.pi)  
  
# Import math Library  
import math  
  
# radius of the circle  
r = 4  
  
# value of pie  
pie = math.pi  
  
# area of the circle  
print(pie \* r \* r)  
  
# Import math Library  
import math  
  
# Print the value of tau  
print(math.tau)  
  
# Import math Library  
import math  
  
# Print the positive infinity  
print(math.inf)  
  
# Print the negative infinity  
print(-math.inf)  
  
a = 2.3  
  
# returning the ceil of 2.3  
print("The ceil of 2.3 is : ", end="")  
print(math.ceil(a))  
  
# returning the floor of 2.3  
print("The floor of 2.3 is : ", end="")  
print(math.floor(a))  
  
a = 5  
  
# returning the factorial of 5  
print("The factorial of 5 is : ", end="")  
print(math.factorial(a))  
  
a = 15  
b = 5  
  
# returning the gcd of 15 and 5  
print("The gcd of 5 and 15 is : ", end="")  
print(math.gcd(b, a))  
  
a = -10  
  
# returning the absolute value.  
print("The absolute value of -10 is : ", end="")  
print(math.fabs(a))  
  
# initializing the value  
test\_int = 4  
test\_neg\_int = -3  
test\_float = 0.00  
  
# checking exp() values  
# with different numbers  
print(math.exp(test\_int))  
print(math.exp(test\_neg\_int))  
print(math.exp(test\_float))  
  
print("The value of 3\*\*4 is : ", end="")  
  
# Returns 81  
print(pow(3, 4))  
  
# returning the log of 2,3  
print("The value of log 2 with base 3 is : ", end="")  
print(math.log(2, 3))  
  
# returning the log2 of 16  
print("The value of log2 of 16 is : ", end="")  
print(math.log2(16))  
  
# returning the log10 of 10000  
print("The value of log10 of 10000 is : ", end="")  
print(math.log10(10000))  
  
# print the square root of 0  
print(math.sqrt(0))  
  
# print the square root of 4  
print(math.sqrt(4))  
  
# print the square root of 3.5  
print(math.sqrt(3.5))  
  
a = math.pi / 6  
  
# returning the value of sine of pi/6  
print("The value of sine of pi/6 is : ", end="")  
print(math.sin(a))  
  
# returning the value of cosine of pi/6  
print("The value of cosine of pi/6 is : ", end="")  
print(math.cos(a))  
  
# returning the value of tangent of pi/6  
print("The value of tangent of pi/6 is : ", end="")  
print(math.tan(a))  
  
a = math.pi / 6  
b = 30  
  
# returning the converted value from radians to degrees  
print("The converted value from radians to degrees is : ", end="")  
print(math.degrees(a))  
  
# returning the converted value from degrees to radians  
print("The converted value from degrees to radians is : ", end="")  
print(math.radians(b))  
  
# initializing argument  
gamma\_var = 6  
  
# Printing the gamma value.  
print("The gamma value of the given argument is : "  
 + str(math.gamma(gamma\_var)))

* CALENDAR MODULE

# Python program to display calendar of  
# given month of the year  
  
# import module  
import calendar  
  
yy = 2017  
mm = 11  
  
# display the calendar  
print(calendar.month(yy, mm))  
  
# using calendar to print calendar of year  
# prints calendar of 2018  
print("The calendar of year 2018 is : ")  
print(calendar.calendar(2018, 2, 1, 6))  
  
# Python code to demonstrate the working of  
# calendar() and firstweeksday()  
  
# importing calendar module for calendar operations  
import calendar  
  
# using calendar to print calendar of year  
# prints calendar of 2012  
print("The calendar of year 2012 is : ")  
print(calendar.calendar(2012, 2, 1, 6))  
  
# using firstweekday() to print starting day number  
print("The starting day number in calendar is : ", end="")  
print(calendar.firstweekday())  
  
# Python code to demonstrate the working of  
# isleap() and leapdays()  
  
# importing calendar module for calendar operations  
import calendar  
  
# using isleap() to check if year is leap or not  
if (calendar.isleap(2008)):  
 print("The year is leap")  
else:  
 print("The year is not leap")  
  
# using leapdays() to print leap days between years  
print("The leap days between 1950 and 2000 are : ", end="")  
print(calendar.leapdays(1950, 2000))  
  
# using month() to display month of specific year  
print("The month 5th of 2016 is :")  
print(calendar.month(2016, 5, 2, 1))  
  
# using month() to display month of specific year  
print("The month 5th of 2016 is :")  
print(calendar.month(2016, 5, 2, 1))  
  
# using prmonth() to print calendar of 1997  
print("The 4th month of 1997 is : ")  
calendar.prmonth(1997, 4, 2, 1)  
  
# using setfirstweekday() to set first week day number  
calendar.setfirstweekday(4)  
  
print("\r")  
  
# using firstweekday() to check the changed day  
print("The new week day number is : ", end="")  
print(calendar.firstweekday())  
  
# using weekday() to print day number of date  
print("The day number of 25 April 1997 is : ", end="")  
print(calendar.weekday(1997, 4, 25))

GETTING WEEK DAY NAME FROM THE GIVEN DATE

# MONTH DAY YEAR  
  
import calendar  
n1,n2,n3=map(int,input().split())  
print((calendar.day\_name[calendar.weekday(n3,n1,n2)]).upper())

RANDOM MODULE

# import random  
import random  
  
# prints a random value from the list  
list1 = [1, 2, 3, 4, 5, 6]  
print(random.choice(list1))  
  
random.seed(5)  
  
print(random.random())  
print(random.random())  
  
# Generates a random number between  
# a given positive range  
r1 = random.randint(5, 15)  
print("Random number between 5 and 15 is % s" % (r1))  
  
# Generates a random number between  
# two given negative range  
r2 = random.randint(-10, -2)  
print("Random number between -10 and -2 is % d" % (r2))  
  
# Python3 program to demonstrate  
# the use of random() function .  
  
# import random  
from random import random  
  
# Prints random item  
print(random())  
  
# Python3 program to demonstrate the use of  
# choice() method  
  
# import random  
import random  
  
# prints a random value from the list  
list1 = [1, 2, 3, 4, 5, 6]  
print(random.choice(list1))  
  
# prints a random item from the string  
string = "geeks"  
print(random.choice(string))  
  
# prints a random item from the tuple  
tuple1 = (1, 2, 3, 4, 5)  
print(random.choice(tuple1))  
  
# import the random module  
import random  
  
# declare a list  
sample\_list = [1, 2, 3, 4, 5]  
  
print("Original list : ")  
print(sample\_list)  
  
# first shuffle  
random.shuffle(sample\_list)  
print("\nAfter the first shuffle : ")  
print(sample\_list)  
  
# second shuffle  
random.shuffle(sample\_list)  
print("\nAfter the second shuffle : ")  
print(sample\_list)

PATTERNS

for i in range(5):  
 for j in range(i + 1):  
 print(" \* ", end=" ")  
 print(" ")  
  
print(" ")  
  
for i in range(0, 5):  
 # inner loop to handle number of columns  
 # values changing acc. to outer loop  
 for j in range(0, i + 1):  
 # printing stars  
 print(" \* ", end=" ")  
 print(" ")  
  
for i in range(5):  
 for j in range(i + 1):  
 print((j + 1), " ", end=" ")  
 print(" ")  
  
print(" ")  
print(" ")  
  
count = 0  
for i in range(5):  
 for j in range(i + 1):  
 count += 1  
 print(count, " ", end=" ")  
 print(" ")  
  
print(" ")  
  
start = 64  
for i in range(5):  
 for j in range(i + 1):  
 start += 1  
 print(chr(start), " ", end=" ")  
 print(" ")  
  
print(" ")  
print(" ")  
  
start = 64  
for i in range(5):  
 start += 1  
 for j in range(i + 1):  
 print(chr(start), " ", end=" ")  
 print(" ")  
  
print(" ")  
print(" ")  
  
start = 64  
for i in range(5):  
 start += 1  
 for j in range(i + 1):  
 print(chr(start), " ", end=" ")  
 print(" ")  
  
print(" ")  
print(" ")  
  
for i in range(5):  
 for j in range(5):  
 if (i + j) >= 4:  
 print("\* ", end=" ")  
 else:  
 print(" ", end=" ")  
  
 print(" ")  
  
print(" ")  
print(" ")  
  
for i in range(5):  
 for j in range(5):  
 if (i + j) >= 4:  
 print(j, end=" ")  
 else:  
 print(" ", end=" ")  
  
 print(" ")  
  
print(" ")  
print(" ")  
  
for i in range(5):  
 for j in range(5):  
 if (i + j) >= 4:  
 print("# ", end=" ")  
 else:  
 print(" ", end=" ")  
  
 print(" ")  
  
print(" ")  
print(" ")  
  
  
# Python 3.x code to demonstrate star pattern  
  
# Function to demonstrate printing pattern triangle  
def triangle(n):  
 # number of spaces  
 k = n - 1  
  
 # 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)  
  
rows = 6  
  
for num in range(rows):  
  
 for i in range(num):  
 print(num, end=" ") # print number  
  
 # line after each row to display pattern correctly  
  
 print(" ")  
  
rows = 6  
x = 0  
for num in range(rows):  
 for i in range(num):  
 x += 1  
 print(x, end=" ") # print number  
  
 # line after each row to display pattern correctly  
  
 print(" ")  
  
print(" \n")  
  
rows = 6  
x = -1  
for num in range(rows):  
 x += 1  
 for i in range(num):  
 print(x, end=" ") # print number  
  
 # line after each row to display pattern correctly  
  
 print(" ")  
  
print(" \n")  
  
print(" \n")  
rows = 5  
for num in range(rows, 0, -1):  
  
 for i in range(num):  
 print(num, end=" ") # print number  
  
 # line after each row to display pattern correctly  
  
 print(" ")  
  
print(" \n")  
rows = 5  
x = 0  
for num in range(rows, 0, -1):  
 x += 1  
 for i in range(num):  
 print(x, end=" ") # print number  
  
 # line after each row to display pattern correctly  
  
 print(" ")  
  
rows = 6  
start = 64  
for num in range(rows):  
  
 for i in range(num):  
 print(chr(num + 64), end=" ") # print number  
  
 # line after each row to display pattern correctly  
  
 print(" ")  
  
rows = 6  
for num in range(rows):  
  
 for i in range(num):  
 print(chr(i + 65), end=" ") # print number  
  
 # line after each row to display pattern correctly  
  
 print(" ")

rows = 5

for i in range(rows, 0, -1):

num = i

for j in range(0, i):

print(num, end=" ")

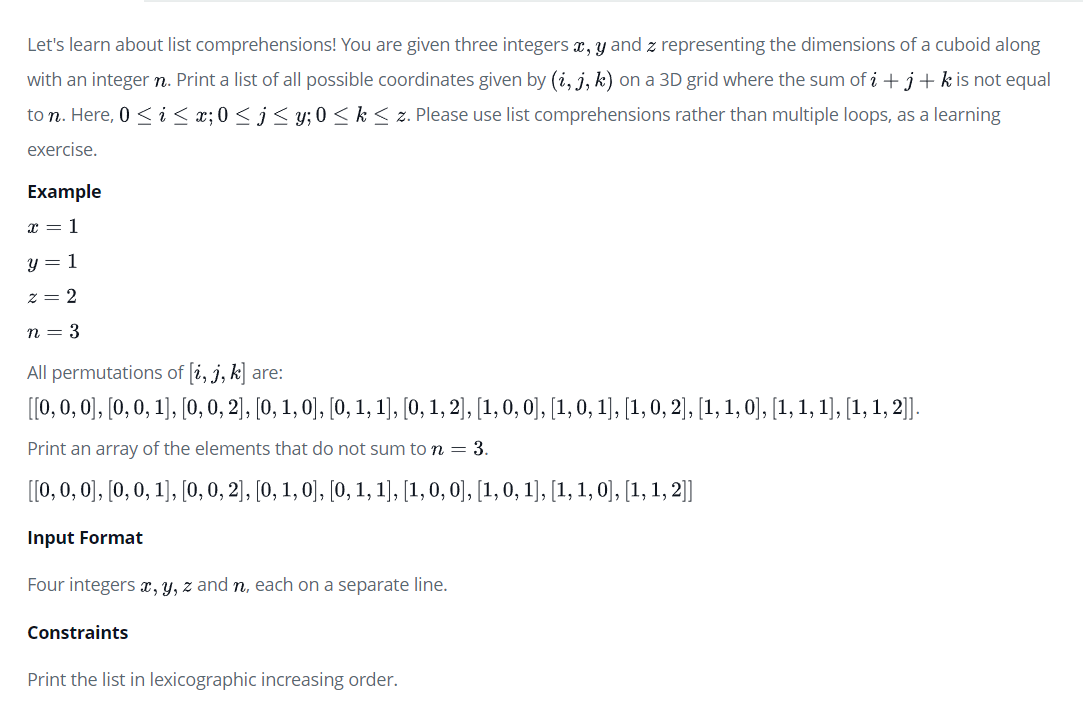
print("\n")

#PATTERNS  
  
a =int(input(" ENTER THE SIZE OF THE PATTERN: "))  
  
i=0  
while i<a:  
 j=0  
 while j<a:  
 print("# ",end=" ")  
 j+=1  
 print(" ")  
 i+=1  
  
print(" ")  
i=0  
while i<a:  
 print("\* " \* a)  
 i+=1  
  
  
print(" ")  
i=0  
for i in range(a):  
 j=0  
 for j in range(a):  
 print("% ",end=" ")  
 j+=1  
 print(" ")  
 i+=1  
  
print(" ")  
i=0  
for i in range(a):  
 j=0  
 while j<a:  
 print("& ",end=" ")  
 j+=1  
 print(" ")  
 i+=1  
  
print(" ")  
i=0  
for i in range(a+1):  
 j=0  
 for j in range(i):  
 print("% ",end=" ")  
 j+=1  
 print(" ")  
 i+=1  
  
  
print(" ")  
i=0  
for i in range(a):  
 j=0  
 for j in range(a):  
 if(i+j)>=a-1:  
 print("% ", end=" ")  
 else:  
 print(" ", end=" ")  
 j += 1  
 print(" ")  
 i += 1  
  
print(" ")  
i=0  
for i in range(a):  
 j=0  
 for j in range(a):  
 if(i+j)>=a:  
 print(" ", end=" ")  
 else:  
 print("% ", end=" ")  
 j += 1  
 print(" ")  
 i += 1  
  
  
print(" ")  
i=0  
for i in range(a+1):  
  
 j=0  
 for j in range(a + 1):  
 if j <= i:  
 print(" ", end=" ")  
 else:  
 print("% ", end=" ")  
 j += 1  
 print(" ")  
 i+=1

#PATTERNS  
#WHILE LOOP  
i=0  
while(i<10):  
 print("SURYA MAHESH")  
 i+=1  
  
#PATTERN  
"""  
\* \* \* \* \*   
\* \* \* \* \*   
\* \* \* \* \*   
\* \* \* \* \*   
\* \* \* \* \*   
  
"""  
  
i=0  
while(i<5):  
 j = 0  
 while(j<5):  
 print("\* ",end=" ")  
 j+=1  
 print(" ")  
 i+=1  
  
print(" ")  
  
"""  
$ $ $ $ $   
$ $ $ $ $   
$ $ $ $ $   
$ $ $ $ $   
$ $ $ $ $   
  
"""  
  
i=0  
while(i<5):  
 j = 0  
 while(j<5):  
 print("$ ",end=" ")  
 j+=1  
 print(" ")  
 i+=1

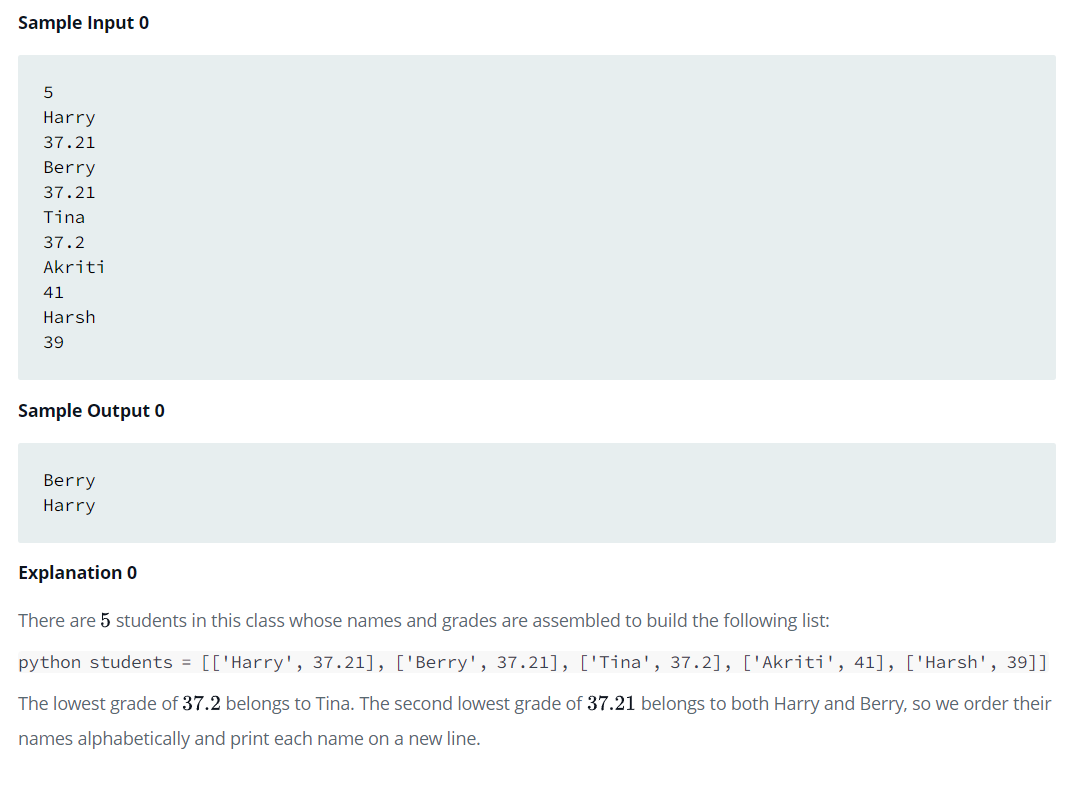
IMPORTANT THING FOR HACKER RANK PROBLEMS

a="1 2 3 4 5"  
l1=list(a.split())  
print(l1)  
  
l2=list(map(int,l1))  
print(l2)

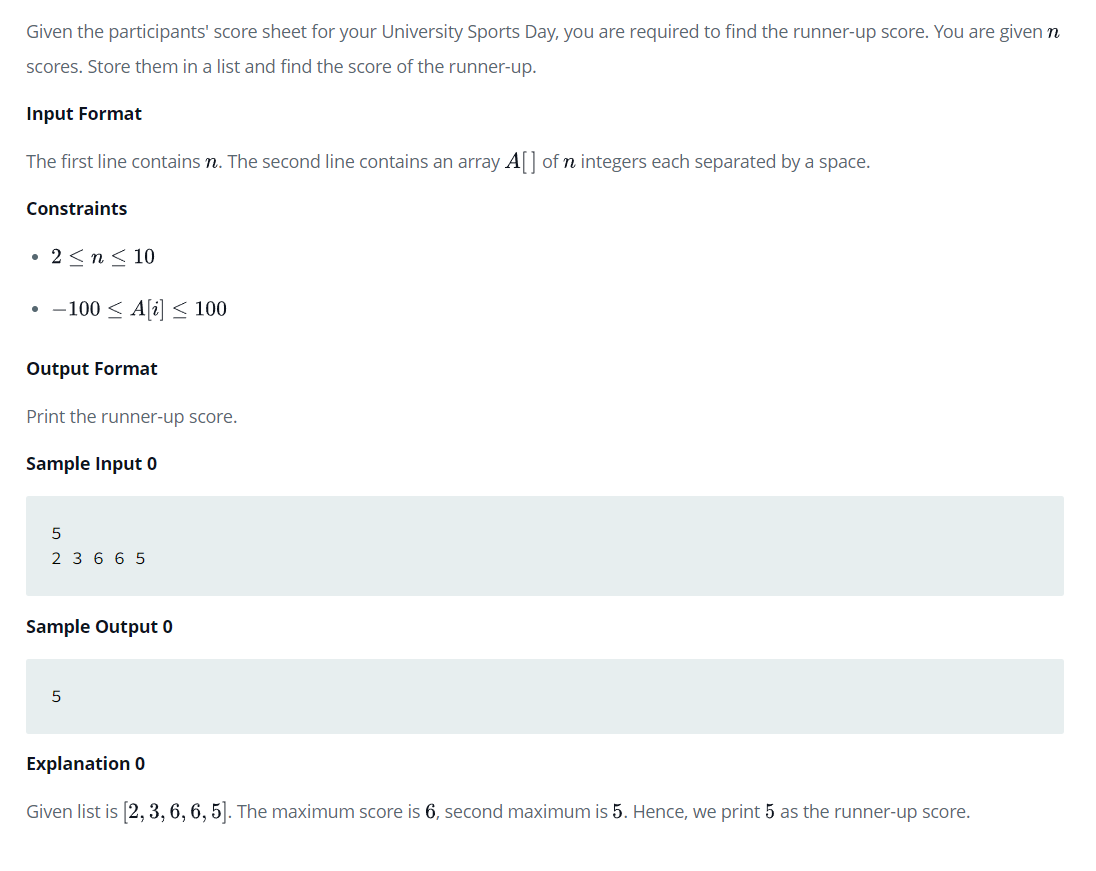


LIST COMPREHENSIONS

x = int(input())  
y = int(input())  
z = int(input())  
n = int(input())  
  
  
print([[a, b, c] for a in range(x + 1) for b in range(y + 1) for c in range(z + 1) if a + b + c != n])

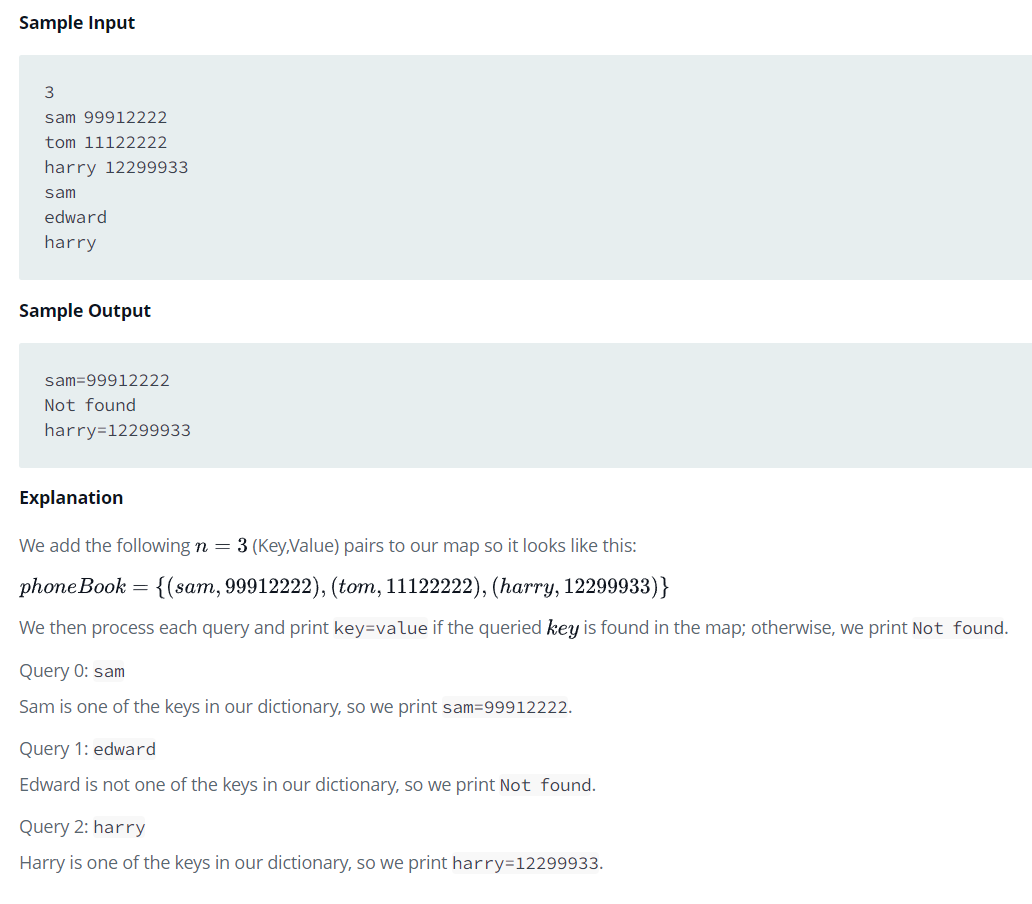


*'''  
  
  
l=[['ha', 37.21], ['be', 37.21], ['t', 37.2], ['a', 41.0], ['harsh', 39.0]]  
  
  
  
t=int(input())  
l=[]  
for i in range(t):  
 l1=[]  
 l1.append(input())  
 l1.append(float(input()))  
 l.append(l1)  
  
  
'''*t=int(input())  
l=[]  
for i in range(t):  
 l1=[]  
 l1.append(input())  
 l1.append(float(input()))  
 l.append(l1)  
  
#print(l)  
  
d=dict()  
for item in l:  
 d[item[1]]=item[0]  
  
s=set(d.keys())  
#print(s)  
l2=list(s)  
l2=sorted(l2)  
l2.remove(min(l2))  
  
req=l2[0]  
#print(l2)  
  
l3=[]  
for item in l:  
 if item[1]==req:  
 l3.append(item[0])  
 #print(item[0])  
  
  
l3=sorted(l3)  
  
for i in l3:  
 print(i)

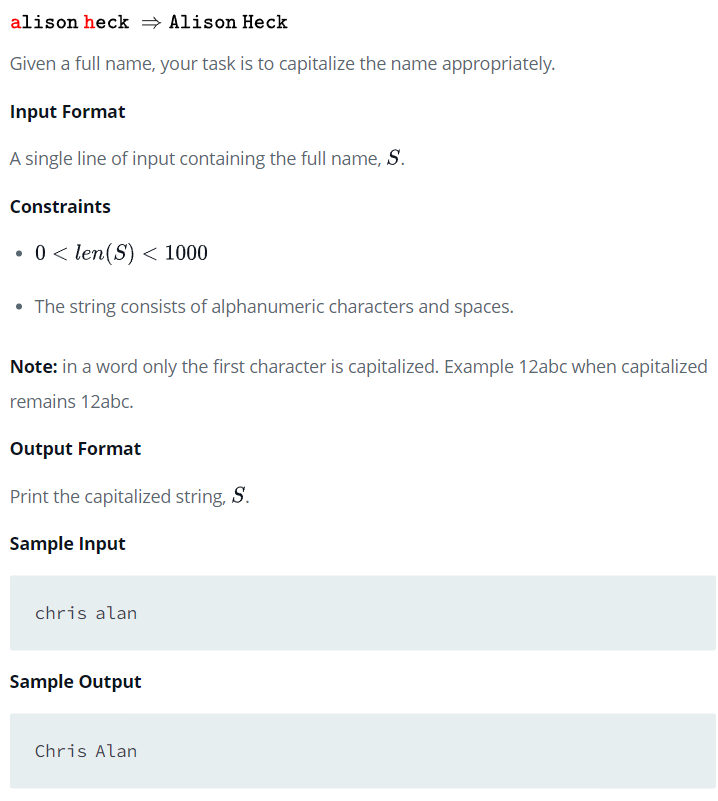


#SOLUTION -1

n = int(input())  
arr = map(int, input().split())  
print (sorted(set(arr))[-2])  
  
  
#SOLUTION -2  
  
n= int(input())  
l = []  
  
for i in range(n):  
 x=int(input())  
 l.append(x)  
  
s = set(l)  
s.remove(max(s))  
l = list(s)  
print(max(l))



#SOLUTION 1  
  
t=int(input())  
l=list()  
d=dict()  
for i in range(t):  
 l=list(input().split())  
 d[l[0]]=l[1]  
  
while True:  
 try:  
 name=input()  
 if name in d:  
 print("%s=%s"%(name,d[name]))  
 else:  
 print('Not found')  
 except:  
 break  
  
  
#SOLUTION 2  
  
n = int(input())  
name\_numbers = [input().split() for \_ in range(n)]  
phone\_book = {k: v for k, v in name\_numbers}  
while True:  
 try:  
 name = input()  
 if name in phone\_book:  
 print('%s=%s' % (name, phone\_book[name]))  
 else:  
 print('Not found')  
 except:  
 break



s1="surya mahesh"  
s2=s1.split()  
print(" ".join(i.capitalize() for i in s2))  
  
s1="surya mahesh"  
s2=s1.split()  
print("@".join(i.capitalize() for i in s2))  
  
s1="surya mahesh"  
s2=s1.split()  
print("%".join(i.capitalize() for i in s2))  
  
s1="surya mahesh"  
s2=s1.split()  
print("&".join(i.capitalize() for i in s2))

#SUBSTRING COUNT IN MAIN STRING( REPETITIONS)

def count\_substring(string, sub\_string):  
 count = 0  
 for i in range(len(string)):  
 temp = string[i:(i + len(sub\_string))]  
 if temp == sub\_string:  
 count += 1  
  
 return count

#TREES , DATA STRUCTURES , TREE TRAVERSALS

class node:  
 def \_\_init\_\_(self,data):  
 self.leftnode=None  
 self.rightnode=None  
 self.nodedata=data  
  
  
root=node(1)  
root.leftnode=node(2)  
root.rightnode=node(3)  
  
root.leftnode.leftnode=node(4)  
root.leftnode.rightnode=node(5)  
  
root.rightnode.leftnode=node(6)  
root.rightnode.rightnode=node(7)  
  
def inorder(root):  
 if root:  
 inorder(root.leftnode)  
 print(root.nodedata,end=" ")  
 inorder(root.rightnode)  
  
print("INORDER TRAVERSAL")  
inorder(root)  
print(" ")  
def preorder(root):  
 if root:  
 print(root.nodedata,end=" ")  
 preorder(root.leftnode)  
 preorder(root.rightnode)  
  
  
print("PREORDER TRAVERSAL")  
preorder(root)  
print(" ")  
  
def postorder(root):  
 if root:  
 postorder(root.leftnode)  
 postorder(root.rightnode)  
 print(root.nodedata,end=" ")  
  
print("POSTORDER TRAVERSAL")  
postorder(root)  
print(" ")

#ARRAYS ( DATA STRUCTURES)

import array as arr  
a=arr.array("i",[1,2,3,4,5,6,7,8,9])  
print(a)  
  
for item in a:  
 print(item,end=" ")  
print(" ")  
  
a.append(10)  
for item in a:  
 print(item,end=" ")  
  
print(" ")  
print(a.pop())  
  
for item in a:  
 print(item,end=" ")  
print(" ")  
  
a.extend([10,11,12,13,14,15])  
for item in a:  
 print(item,end=" ")  
print(" ")  
  
print(a.count(1))  
a.remove(5)  
for item in a:  
 print(item,end=" ")  
print(" ")  
  
a.insert(2,100)  
  
for item in a:  
 print(item,end=" ")  
print(" ")  
  
a.reverse()  
for item in a:  
 print(item,end=" ")  
print(" ")  
  
print(a.index(100))  
  
l=a.tolist()  
print(l)  
  
  
a.tounicode()  
for item in a:  
 print(item,end=" ")  
print(" ")

#STACK USING LISTS

#STACKS USING LISTS  
  
stack=[]  
def empty(stack):  
 if len(stack)==0:  
 print("STACK IS EMPTY")  
 else:  
 print("STACK IS NOT EMPTY")  
  
stack.append(1)  
stack.append(2)  
stack.append(3)  
stack.append(4)  
  
print(stack)  
empty(stack)  
try:  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
except:  
 print("STACK UNDERFLOW")  
  
empty(stack)  
print(stack)

STACK USING ARRAYS

#STACKS USING ARRAYS  
import array as arr  
  
stack=arr.array("i")  
  
def empty(stack):  
 if len(stack)==0:  
 print("STACK IS EMPTY")  
 else:  
 print("STACK IS NOT EMPTY")  
  
stack.append(1)  
stack.append(2)  
stack.append(3)  
stack.append(4)  
  
print(stack)  
empty(stack)  
try:  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
 print(stack.pop(), "POPPED OUT")  
except:  
 print("STACK UNDERFLOW")  
  
empty(stack)  
print(stack)

STACK IMPLEMENTATION USING LISTS AND CLASSES

class stack:  
 def \_\_init\_\_(self):  
 self.items=[]  
 def isempty(self):  
 if len(self.items)==0:  
 print("STACK IS EMPTY")  
 else:  
 print("STACK IS NOT EMPTY")  
 #or  
 if self.items==[]:  
 return True  
 else:  
 return False  
 def push(self,data):  
 self.items.append(data)  
 def pop(self):  
 return self.items.pop()  
  
s=stack()  
s.push(10)  
s.push(20)  
s.push(30)  
s.push(40)  
s.push(50)  
  
print(s.items)  
print(s.isempty())  
  
print(s.pop(),"POPPED OUT")  
print(s.pop(),"POPPED OUT")  
print(s.pop(),"POPPED OUT")  
  
print(s.items)  
print(s.isempty())  
print(s.pop(),"POPPED OUT")  
print(s.pop(),"POPPED OUT")  
  
print(s.items)  
print(s.isempty())

QUEUE USING (from collecions import deque )

from collections import deque  
  
q=deque()  
  
q.append(1)  
q.append("hello")  
q.append(3.14)  
  
print(q)  
  
q.appendleft(2)  
q.appendleft("hii")  
q.appendleft(2.78)  
  
print(q)  
print(q.pop())  
print(q.pop())  
print(q.pop())  
  
print(q)  
  
print(q.popleft())  
print(q.popleft())  
print(q.popleft())  
  
print(q)

QUEUE’S FROM COLLETIONS

from collections import deque  
  
queue=deque()  
  
print(queue)  
for i in range(1,6):  
 queue.append(i)  
 print(queue)  
  
for i in range(6,11):  
 queue.appendleft(i)  
 print(queue)  
  
for i in range(1,6):  
 queue.pop()  
 print(queue)  
  
for i in range(6,11):  
 queue.popleft()  
 print(queue)

LINKED LISTS IN PYTHON

# A simple Python program for traversal of a linked list  
  
# Node class  
class Node:  
  
 # Function to initialise the node object  
 def \_\_init\_\_(self, data):  
 self.data = data # Assign data  
 self.next = None # Initialize next as null  
  
  
# Linked List class contains a Node object  
class LinkedList:  
  
 # Function to initialize head  
 def \_\_init\_\_(self):  
 self.head = None  
  
 # This function prints contents of linked list  
 # starting from head  
 def printList(self):  
 temp = self.head  
 while (temp):  
 print (temp.data)  
 temp = temp.next  
  
  
# Code execution starts here  
if \_\_name\_\_=='\_\_main\_\_':  
  
 # Start with the empty list  
 llist = LinkedList()  
  
 llist.head = Node(1)  
 second = Node(2)  
 third = Node(3)  
  
 llist.head.next = second; # Link first node with second  
 second.next = third; # Link second node with the third node  
  
 llist.printList()

CUSTOM LINKED LISTS (USERDEFINED)

#CUSTOM LINKED LISTS  
  
class node:  
 def \_\_init\_\_(self,data):  
 self.data=data  
 self.next=None  
  
class linkedlist:  
 def \_\_init\_\_(self):  
 self.head=None  
  
 def show(self):  
 node=self.head  
 while node is not None:  
 print(node.data,"-->",end=" ")  
 node=node.next  
 print("None")  
 print(" ")  
 def insert(self,data):  
 newnode=node(data)  
 newnode.next=self.head  
 self.head=newnode  
  
  
  
l=linkedlist()  
node1=node(5)  
l.head=node1  
l.show()  
l.insert(4)  
l.show()  
l.insert(3)  
l.show()  
l.insert(2)  
l.show()  
l.insert(1)  
l.show()

REVERSE AN INTEGER

n=int(input("ENTER INTEGER : "))  
temp=n  
rev=0  
while n>0:  
 d=n%10  
 rev=rev\*10+d  
 n=n//10  
  
print("REVERSED INTEGER : ",rev)

TUPLE OF VALUES , LIST OF VALUES INPUTED BY USER

a=()  
l=[]  
n= int(input("ENTER LIMIT : "))  
for i in range(n):  
 item=int(input("ENTER ELEMENT VALUE: "))  
 l.append(item)  
  
  
print(l)  
a=tuple(l)  
#or a=a(l)  
#or a= tuple(l)  
print(a)

PRIME OR NOT

num=int(input("ENTER A NUMBER TO CHECK ITS PRIME OR NOT : "))  
limit=int(num/2)+1  
prime=True  
for i in range(2,limit):  
 rem=num%i  
 if rem==0:  
 prime = False  
 print(num," IS NOT PRIME NUMBER ")  
 break  
  
if prime==True:  
 print(num," IS PRIME NUMBER ")

PRIME CHECK

import math  
  
  
def primecheck(num):  
 # limit = int(num / 2) + 1  
 limit = math.ceil(math.sqrt(num))  
 result = "Prime"  
 for i in range(2, limit):  
 rem = num % i  
 if rem == 0:  
 result = "Not prime"  
 break  
 return result  
  
  
t = int(input("ENTER NUM OF TEST CASES: "))  
for i in range(t):  
 num = int(input("ENTER AN INTEGER TO CHECK ITS PRIME OR NOT : "))  
 print(primecheck(num))

PRIME OR NOT METHOD TWO

num=int(input("ENTER A NUMBER TO CHECK ITS PRIME OR NOT : "))  
# "//" is INTEGER DIVISION  
limit=num//2+1  
for i in range(2,limit):  
 rem=num%i  
 if rem==0:  
 print(num," IS NOT PRIME NUMBER ")  
 break  
  
else:  
 print(num," IS PRIME NUMBER ")

PRIME WITH LEAST TIME COMPLEXITY

for \_ in range(int(input())):  
 num = int(input())  
 i f(num == 1):  
 print("Not prime")  
 else:  
 i f(num % 2 == 0 and num > 2):  
 print("Not prime")  
 else:  
 for i in range(3, int(nu m\* \*( 1 /2) ) +1, 2):  
 if num % i == 0:  
 print("Not prime")  
 break  
 else:  
 print("Prime")

PYTHON TO CREATE A PHONE DICTIONARY

limit=int(input("ENTER LIMIT : "))  
phbook={ }  
  
# m=dict()  
phnum=None  
name=None  
  
for i in range(limit):  
 phnum=int(input("ENTER PHONE NUMBER : "))  
 name=input("ENTER NAME : ")  
 #TO ADD KEY:VALUE PAIRS INTO DICTIONARY  
 #METHOD -1  
 phbook[phnum]=name  
 #METHOD -2  
 """   
   
 contact=dict({phnum:name})  
 phbook.update(contact)  
  
 """  
  
  
print(phbook)  
  
try:  
 num = int(input("ENTER PH.NUMBER TO SEARCH IN PHONE BOOK "))  
 print("NAME OF THE PERSON OF THAT NUMGER : ", phbook[num])  
except Exception as e:  
 print("RECORDS NOT FOUND IN PHONE BOOK ",e)

SORTING LISTS

#SORTING IN ASCENDING ORDER USING BUBBLE SORT  
  
l=[4,1,3,2,5]  
print("INITIALLY LIST : ",l)  
for i in range(len(l)):  
 for j in range(len(l)-i-1):  
 if l[j]>l[j+1]:  
 temp=l[j]  
 l[j]=l[j+1]  
 l[j+1]=temp  
  
  
print("ASCENDING LIST : ",l)  
  
  
#SORTING IN ASCENDING ORDER USING SORT() BUILTIN FUNCTION  
l=[4,1,3,2,5]  
print(" INITIALLY LIST : ",l)  
l.sort()  
print("ASCENDING LIST : ",l)  
  
#SORTING IN ASCENDING ORDER USING SORTED(list) BUILTIN FUNCTION  
l=[4,1,3,2,5]  
print(" INITIALLY LIST : ",l)  
print("ASCENDING LIST : ",sorted(l))  
  
#SORTING IN DESCENDING ORDER USING BUBBLE SORT  
  
l=[4,1,3,2,5]  
print(" INITIALLY LIST : ",l)  
for i in range(len(l)):  
 for j in range(len(l)-i-1):  
 if l[j]<l[j+1]:  
 temp=l[j]  
 l[j]=l[j+1]  
 l[j+1]=temp  
  
  
print("DESCENDING LIST : ",l)  
  
  
#SORTING IN DESCENDING ORDER USING SORT() BUILTIN FUNCTION  
l=[4,1,3,2,5]  
print(" INITIALLY LIST : ",l)  
l.sort(reverse=True)  
print("DESCENDING LIST : ",l)  
  
#SORTING IN DESCENDING ORDER USING SORTED(list) BUILTIN FUNCTION  
l=[4,1,3,2,5]  
print(" INITIALLY LIST : ",l)  
print("ASCENDING LIST : ",sorted(l,reverse=True))

PATTERN

PRINT PATTERN OF PASCAL – TRAINGLE

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

n=int(input("ENTER ROWS : "))  
  
for i in range(0,n):  
 for j in range(0,n-i-1):  
 print("",end=" ")  
 for j in range(0,i+1):  
 print("\*",end=" ")  
 print()

PALINDROME CHECK

n=int(input("ENTER INTEGER : "))  
temp=n  
rev=0  
while n>0:  
 d=n%10  
 rev=rev\*10+d  
 n=n//10  
  
n=temp  
if rev==n:  
 print(n," IS PALINDROME NUMBER ")  
else:  
 print(n, " IS NOT A PALINDROME NUMBER ")

PALINDROME CHECK USING QUEUE AND STACKS CONCEPT

( from collections import deque )

append(), pop() , popleft()

import sys  
from collections import deque  
  
class Solution:  
 def \_\_init\_\_(self):  
 self.q = deque()  
  
 def pushCharacter(self, element):  
 self.q.append(element)  
  
 def enqueueCharacter(self, element):  
 self.q.append(element)  
  
 def popCharacter(self):  
 return self.q.pop()  
  
 def dequeueCharacter(self):  
 return self.q.popleft()  
  
  
# read the string s  
s = input("ENTER A STRING : ( TO CHECK IT'S PALINDROME OR NOT ) : ")  
# Create the Solution class object  
obj = Solution()  
  
l = len(s)  
# push/enqueue all the characters of string s to stack  
for i in range(l):  
 obj.pushCharacter(s[i])  
 obj.enqueueCharacter(s[i])  
  
isPalindrome = True  
'''  
pop the top character from stack  
dequeue the first character from queue  
compare both the characters  
'''  
for i in range(l // 2):  
 if obj.popCharacter() != obj.dequeueCharacter():  
 isPalindrome = False  
 break  
# finally print whether string s is palindrome or not.  
if isPalindrome:  
 print("The word, " + s + ", is a palindrome.")  
else:  
 print("The word, " + s + ", is not a palindrome.")

POWER OF X as N

x=int(input("ENTER BASE : "))  
n=int(input("ENTER POWER : "))  
print(x\*\*n)  
  
import math  
print(math.pow(x,n))  
  
  
result=1  
for i in range(n):  
 result=result\*x  
print(result)

PALINDROME CHECK , ( BY CONSIDERING INPUTED NUMBER AS STRING VIA SLICING)

num=input("ENTER ANY NUMBER : ")  
if num==num[::-1]:  
 print("PALINDROME")  
else:  
 print("NOT A PALINDROME")

PATTERN

ENTER THE TERMINATING CHARACTER: E

A

A B

A B C

A B C D

A B C D E

ch=input("ENTER THE TERMINATING CHARACTER: ")[0]

a=ord(ch)  
  
for x in range(65,a+1):  
 for y in range(65,x+1):  
 print(chr(y),end=" ")  
 print(" ")

FIBANOCCI SERIES USING FOR LOOP

print("TYPE-1")  
number=int(input("ENTER THE RANGE OF SERIES : "))  
first =0  
second=1  
  
for num in range(number):  
 if num<=1:  
 next=num  
 else:  
 next=first+second  
 first=second  
 second=next  
  
 print(next,end=" ")  
  
print(" ")  
  
print("TYPE-2")  
first =1  
second=1  
  
for num in range(number):  
 if num<=1:  
 next=num  
 else:  
 next=first+second  
 first=second  
 second=next  
  
 print(next,end=" ")

PATTERN

ENTER NUMBER OF ROWS : 5

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

n=int(input("ENTER NUMBER OF ROWS : "))  
  
for i in range(1,n+1):  
 for j in range(1,i+1):  
 print(j,end=" ")  
 print(" ")

### [any()](https://docs.python.org/2/library/functions.html#any)

This expression returns True if **any** element of the iterable is true.  
If the iterable is empty, it will return False.

**Code**

>>> any([1>0,1==0,1<0])

True

>>> any([1<0,2<1,3<2])

False

[**all()**](https://docs.python.org/2/library/functions.html#all)

This expression returns True if **all** of the elements of the iterable are true. If the iterable is empty, it will return True.

**Code**

>>> all(['a'<'b','b'<'c'])

True

>>> all(['a'<'b','c'<'b'])

False

TUPLES COMPARISION

#TUPLES  
  
t1=(1,2,3)  
t2=(1,2,3)  
  
if t1==t2:  
 print("same")  
  
if (1,2,3)==(1,2,3):  
 print("same")  
  
# TUPLES COMPARISION ( INDEX BASED )  
  
#here first element of tuple1 is greater than tuple2  
if (2,2,6)>(1,2,6):  
 print("greater than")  
  
#here first element same then check next ,second element of tuple1 is greater than tuple2  
if (4,2,1)>(4,1,1):  
 print("greater than")  
  
#here first,second element same then check next ,third element of tuple1 is greater than tuple2  
if (4,1,2)>(4,1,1):  
 print("greater than")  
  
#here first element of tuple1 is less than tuple2  
  
if (0,2,6)<(1,2,6):  
 print("less than")  
  
#here first element same then check next ,second element of tuple1 is less than tuple2  
  
if (3,1,1)<(4,1,1):  
 print("less than")  
  
#here first,second element same then check next ,third element of tuple1 is less than tuple2  
  
if (4,1,0)<(4,1,1):  
 print("less than")

INTEGER CAN BE REPRESENTED AS SUM OF SQUARES OF ANY TWO INTEGERS

*"""  
  
Given a non-negative integer c, decide whether there're two integers a and b such that a\*\*2 + b\*\*2 = c.  
  
Example 1:  
  
Input: c = 5  
Output: true  
Explanation: 1 \* 1 + 2 \* 2 = 5  
  
Example 2:  
Input: c = 3  
Output: false  
  
"""*import math  
  
def judgeSquareSum(c):  
 nums = set()  
 for i in range(int(math.sqrt(c)) + 1):  
 nums.add(i \* i)  
 if (c - (i \* i)) in nums:  
 return True  
 return False  
  
print(judgeSquareSum(5))  
print(judgeSquareSum(1))  
print(judgeSquareSum(0))  
print(judgeSquareSum(3))

ADD ARRAY FORM OF INTEGERS AND RETURN RESULT IN THE FORM OF ARRAY

USING LIST COMPREHENSIONS

# ADD ARRAY FORM OF INTEGERS  
# AND PRINT RESULT IN THE FORM OF ARRAY  
l=[1,2,3]  
k=20  
s=str(l)  
s="".join(str(i) for i in l)  
print(s)  
s="".join(str(i) for i in l)  
k+=int(s)  
l=[int(i) for i in str(k)]  
print(l)

USING LIST COMPREHENSIONS

l=[1,2,3]  
k=20  
print([int(i) for i in str(int(''.join([str(i) for i in l]))+k)])

NUMPY .

import numpy as np

print(np.\_\_version\_\_)

a=np.array([1,2,3,4,5])

print(a)

print(a[0])

a=np.array([[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]])

print(a)

print(a[0][0])

a=np.array([[[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]],[[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]]])

print(a)

print(a[0][0][0])

a=[10,20,30,40,50]

print(a)

b=np.asarray(a,dtype=float)

print(b)

b=np.asarray(a,dtype=int)

print(b)

b=np.asarray(a,dtype=str)

print(b)

a=[[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]]

b=np.asarray(a,dtype=str)

print(b)

a=[[1,2,3,4,5],[4,5,6,7,8],[9,10,11,12,13]]

b=np.asarray(a,dtype=int,order="C")

print(b)

for i in np.nditer(b):

print(i)

a=[[1,2,3,4,5],[4,5,6,7,8],[9,10,11,12,13]]

b=np.asarray(a,dtype=int,order="F")

print(b)

for i in np.nditer(b):

print(i)

#INTIALIZING array

a=np.zeros(3)

print(a)

print()

a=np.zeros([5,5])

print(a)

print()

a=np.zeros([2,3,3])

print(a)

print()

print(np.full([5,5],10))

print()

print(np.full([5,5],4))

print(np.random.rand(5,5))

print()

a=np.ones([3,3])

print(a)

print()

a=np.eye(4)

print(a)

print()

a=np.eye(3)

print(a)

print()

a=np.arange(1,11,1)

print(a)

print()

a=np.arange(10,110,10)

print(a)

print()

a=np.arange(10,110,10,dtype=float)

print(a)

print()

a=np.arange(10,70,10)

print(a)

a=a.reshape(2,3)

print(a)

print()

a=a.reshape(3,2)

print(a)

a=a.reshape(6,1)

print(a)

print()

a=a.reshape(1,6)

print(a)

print()

a=np.linspace(10,100,10)

print(a)

a=np.linspace(10,100,10,endpoint=False)

print(a)

a=np.linspace(10,100,10,endpoint=False,retstep=True)

print(a)

print()

a=np.linspace(10,100,10,endpoint=False,retstep=True,dtype=int)

print(a)

print()

a=np.logspace(10,100,10,endpoint=False,base=2)

print(a)

print()

# PROPERTIES

a=np.arange(1,10,1).reshape(3,3)

print()

print(np.size(a))

print(np.shape(a))

print(a.dtype)

#ARRAY OPERATIONS

a=np.array([[10,20,30],[40,50,60]])

print(a)

print(a.dtype)

print(np.size(a))

print(np.shape(a))

print(len(a))

#INDEXING

print(" ")

for i in range(len(a)):

for j in range(len(a[i])):

print(a[i][j],end=" ")

print("")

print(" ")

print(" ")

for i in range(len(a)):

print(a[i])

print(" ")

#SLICING

a=np.arange(10,110,10)

print(a)

print(a[2:])

print(a[:])

print(a[:5])

print(a[3:7])

#COPY

b=np.copy(a)

print(a)

print(b)

#VIEW

c=b.view()

print(c)

b[2]=0

print(a)

print(b)

print(c)

#sort

a=np.array([5,1,3,2,4])

print(np.sort(a))

a=np.array([[20,10,30],[5,4,6]])

print(np.sort(a))

a=np.array([[20,10,30],[5,4,6]])

print(np.sort(a,axis=1))

a=np.array([[20,10,30],[5,4,6]])

print(np.sort(a,axis=0))

d=np.dtype([("name","S1"),("percent","<f8")])

marks=np.array([("MAHESH",90),("SURYA",92),("SANKU",80)])

print(marks)

print(np.sort(marks))

a=np.array([1,2,3])

b=np.array([100,200,300])

print(a)

print(b)

print(np.append(a,b))

print(a)

print(b)

print(np.insert(a,2,1000))

print(np.insert(a,1,[5,55,555]))

b=np.insert(a,1,[5,55,555])

print(b)

print(np.delete(b,1))

c=np.concatenate((a,b))

print(c)

print(np.delete(c,1))

c=np.delete(c,1)

print(c)

print(c.reshape([2,4]))

a=np.array([1,2,3,4])

b=np.array([5,6,7,8])

print(np.concatenate((a,b)))

res=np.stack(a)

print(res[0])

res=np.stack((a,b))

print(res)

print(res[0])

print(np.vstack((a,b)))

print(np.hstack((a,b)))

print(np.dstack((a,b)))

x=np.hstack((a,b))

print(np.split(x,4))

y=np.arange(10,130,10)

print(y)

print(np.split(y,(2,6)))

print(y.reshape(4,3))

b=np.arange(10,130,10)

print(b)

print(np.split(b,(2,6)))

print(b.reshape(4,3))

print(np.where(b==80))

print(np.where(b==50))

print(np.where(b%20==0))

print(np.where(b%30==0))

a=np.array([1,2,3,4,5,6,7,8,9],dtype=int)

print(a)

a=np.array([1,2,3,4,5,6,7,8,9],dtype=float)

print(a)

# ARITHMETIC OPERATIONS

a=np.array([1,2,3,4,5,6,7,8,9],dtype=int)

print(a)

b=np.array([1,2,3,4,5,6,7,8,9],dtype=int)

print(b)

print(np.add(a,b))

print(np.subtract(a,b))

print(np.multiply(a,b))

print(np.divide(a,b))

print(np.exp(a))

a=np.array([1,4,9,16,25])

print(a)

print(np.sqrt(a))

print(np.array\_equal(a,b))

a=np.array([1,2,3,4,5,6,7,8,9],dtype=int)

print(a)

b=np.array([1,2,3,4,5,6,7,8,9],dtype=int)

print(b)

print(np.array\_equal(a,b))

print(np.sum(a))

print(np.min(a))

print(np.max(a))

print(np.mean(a))

print(np.median(a))

print(np.var(a))

print(np.std(a))

numpy

import numpy as np  
a=np.arange(10)  
print(a.size)  
print(a.itemsize)  
print(a.item)  
print(a.shape)  
l=a.tolist()  
print(l)  
s=a.tobytes()  
print(s)  
  
print(a.ndim)  
  
import time  
import sys  
  
print(sys.getsizeof(5)\*len(a))  
  
print(numpy.reshape(a,(2,5)))  
print(a.ndim)  
  
b=np.array([1,2,3,4,5],dtype=np.float64)  
print(b)  
  
b=np.array([1,2,3,4,5],dtype=float)  
print(b)  
  
b=np.array([1,2,3,4,5],dtype=complex)  
print(b)  
  
print(complex())  
print(float())  
  
print(a.dtype)  
print(b.dtype)  
  
#print(np.char.add(["hello","pushpa"],["murugan","syndicate","member"]))  
  
print(np.char.title("hello pushpa"))  
print(np.char.capitalize("hello pushpa"))  
print(np.char.lower("hello pushpa"))  
print(np.char.upper("hello pushpa"))  
  
print(np.char.split("hello murugan syndicate member pushpa pesire"))  
  
print(np.char.strip(["aaha"," pushpa "," vaa ","ennada ","yaarakita ","pesire"],"a"))  
print(np.char.join([":","-"],["dmy","DMY"]))  
  
print(np.char.replace("hello pushpa","hello","hii"))

numpy.reshape(a,(3,3)

import numpy  
  
change\_array = numpy.array([1,2,3,4,5,6])  
change\_array.shape = (3, 2)  
print(change\_array)  
  
"""  
#Output  
[[1 2]  
[3 4]  
[5 6]]  
"""

TRANSPOSE , FLATTEN

import numpy  
n=2  
m=3  
l1=[1,2,3,4,5,6]  
print(l1)  
arr1=numpy.reshape(l1,(m,n))  
print(arr1)  
arr1=numpy.reshape(l1,(n,m))  
print(arr1)  
print(arr1.transpose())  
print(arr1.flatten())

CONCATENATE

import numpy  
  
array\_1 = numpy.array([1,2,3])  
array\_2 = numpy.array([4,5,6])  
array\_3 = numpy.array([7,8,9])  
  
print(numpy.concatenate((array\_1, array\_2, array\_3)))  
  
array\_1 = numpy.array([[1,2,3],[0,0,0]])  
array\_2 = numpy.array([[0,0,0],[7,8,9]])  
  
print(numpy.concatenate((array\_1, array\_2), axis = 1))

USING UNDERSCORE \_

# using \_ ( under score )  
  
print([[1,2,3,4,5] for \_ in range(3)])  
print([(1,2,3,4,5) for \_ in range(3)])  
print([{1,2,3,4,5} for \_ in range(3)])  
  
l=[1,2,3,4,5]  
print([l for \_ in range(3)])  
  
t=(1,2,3,4,5)  
print([t for \_ in range(3)])  
  
s={1,2,3,4,5}  
print([s for \_ in range(3)])  
  
print([input("ENTER SPACE SEPERATED ").split() for \_ in range(3)])

ZEROES AND ONES

import numpy  
  
print(numpy.zeros((1,2))) #Default type is float  
  
print(numpy.zeros((1,2), dtype =int)) #Type changes to int  
  
print(numpy.ones((1,2))) #Default type is float  
  
print(numpy.ones((1,2), dtype =int))  
  
  
print(numpy.zeros((3,3))) #Default type is float  
  
print(numpy.zeros((2,3), dtype =int)) #Type changes to int  
  
print(numpy.ones((3,3))) #Default type is float  
  
print(numpy.ones((2,3), dtype =int))

NUMPY ZEROS AND ONES

import numpy  
  
print(numpy.zeros((3,3,3)))  
print(numpy.zeros((3,3,3),dtype=int))  
  
print(numpy.ones((3,3,3)))  
print(numpy.ones((3,3,3),dtype=int))

EYE IDENTITY

import numpy  
print(numpy.identity(3)) #3 is for dimension 3 X 3  
  
print(numpy.eye(8, 7, k = 1)) # 8 X 7 Dimensional array with first upper diagonal 1.

NUMPY ARRAY MATHEMATICS

import numpy  
  
a = numpy.array([1,2,3,4], float)  
b = numpy.array([5,6,7,8], float)  
  
print(a+b) #[ 6. 8. 10. 12.]  
print(numpy.add(a, b) ) #[ 6. 8. 10. 12.]  
  
print(a-b) #[-4. -4. -4. -4.]  
print(numpy.subtract(a, b) ) #[-4. -4. -4. -4.]  
  
print(a\*b) #[ 5. 12. 21. 32.]  
print(numpy.multiply(a, b)) #[ 5. 12. 21. 32.]  
  
print(a/b) #[ 0.2 0.33333333 0.42857143 0.5 ]  
print(numpy.divide(a, b)) #[ 0.2 0.33333333 0.42857143 0.5 ]  
  
print(a%b) #[ 1. 2. 3. 4.]  
print(numpy.mod(a, b)) #[ 1. 2. 3. 4.]  
  
print(a\*\*b) #[ 1.00000000e+00 6.40000000e+01 2.18700000e+03 6.55360000e+04]  
print(numpy.power(a, b)) #[ 1.00000000e+00 6.40000000e+01 2.18700000e+03 6.55360000e+04]

NUMPY FLOOR CEIL RINT

import numpy  
  
#floor  
#The tool floor returns the floor of the input element-wise.  
#The floor of is the largest integer where .  
  
my\_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])  
print(numpy.floor(my\_array)) #[ 1. 2. 3. 4. 5. 6. 7. 8. 9.]  
  
#ceil  
#The tool ceil returns the ceiling of the input element-wise.  
#The ceiling of is the smallest integer where .  
  
  
my\_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])  
print(numpy.ceil(my\_array)) #[ 2. 3. 4. 5. 6. 7. 8. 9. 10.]  
#rint  
#The rint tool rounds to the nearest integer of input element-wise.  
  
my\_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])  
print(numpy.rint(my\_array)) #[ 1. 2. 3. 4. 6. 7. 8. 9. 10.]

NUMPY SUM , PROD

import numpy  
  
my\_array = numpy.array([ [1, 2], [3, 4] ])  
  
print(numpy.sum(my\_array, axis = 0)) #Output : [4 6]  
print(numpy.sum(my\_array, axis = 1)) #Output : [3 7]  
print(numpy.sum(my\_array, axis = None)) #Output : 10  
print(numpy.sum(my\_array)) #Output : 10 #[ 1. 2. 3. 4. 6. 7. 8. 9. 10.]  
  
  
print(numpy.prod(my\_array, axis = 0)) #Output : [3 8]  
print(numpy.prod(my\_array, axis = 1)) #Output : [ 2 12]  
print(numpy.prod(my\_array, axis = None)) #Output : 24  
print(numpy.prod(my\_array)) #Output : 24

NUMPY MIN,MAX

import numpy  
  
my\_array = numpy.array([[2, 5],  
 [3, 7],  
 [1, 3],  
 [4, 0]])  
  
print(numpy.min(my\_array, axis = 0)) #Output : [1 0]  
print(numpy.min(my\_array, axis = 1)) #Output : [2 3 1 0]  
print(numpy.min(my\_array, axis = None)) #Output : 0  
print(numpy.min(my\_array)) #Output : 0  
  
print(numpy.max(my\_array, axis = 0)) #Output : [4 7]  
print(numpy.max(my\_array, axis = 1)) #Output : [5 7 3 4]  
print(numpy.max(my\_array, axis = None)) #Output : 7  
print(numpy.max(my\_array)) #Output : 7

NUMPY MEAN , VAR , STD ,ROUND

import numpy  
  
my\_array = numpy.array([ [1, 2], [3, 4] ])  
  
print(numpy.mean(my\_array, axis = 0)) #Output : [ 2. 3.]  
print(numpy.mean(my\_array, axis = 1)) #Output : [ 1.5 3.5]  
print(numpy.mean(my\_array, axis = None)) #Output : 2.5  
print(numpy.mean(my\_array)) #Output : 2.5  
#By default, the axis is None. Therefore, it computes the mean of the flattened array.  
  
#var  
  
#The var tool computes the arithmetic variance along the specified axis.  
  
my\_array = numpy.array([ [1, 2], [3, 4] ])  
  
print(numpy.var(my\_array, axis = 0) ) #Output : [ 1. 1.]  
print(numpy.var(my\_array, axis = 1)) #Output : [ 0.25 0.25]  
print(numpy.var(my\_array, axis = None)) #Output : 1.25  
print(numpy.var(my\_array)) #Output : 1.25  
#By default, the axis is None. Therefore, it computes the variance of the flattened array.  
  
#std  
  
#The std tool computes the arithmetic standard deviation along the specified axis.  
my\_array = numpy.array([ [1, 2], [3, 4] ])  
  
print(numpy.std(my\_array, axis = 0)) #Output : [ 1. 1.]  
print(numpy.std(my\_array, axis = 1)) #Output : [ 0.5 0.5]  
print(numpy.std(my\_array, axis = None)) #Output : 1.11803398875  
print(numpy.std(my\_array)) #Output : 1.11803398875  
  
#round function  
print(round(numpy.std(my\_array), 11))

NUMPY DOT,CROSS, MATRIX MULTIPLICATION

#dot  
  
#The dot tool returns the dot product of two arrays.  
  
import numpy  
  
A = numpy.array([ 1, 2 ])  
B = numpy.array([ 3, 4 ])  
  
print(numpy.dot(A, B)) #Output : 11  
#cross  
  
#The cross tool returns the cross product of two arrays.  
  
A = numpy.array([ 1, 2 ])  
B = numpy.array([ 3, 4 ])  
  
print(numpy.cross(A, B)) #Output : -2  
  
  
#matrix multiplication ( is dot product of two matrices)  
  
a=[[1,2],[3,4]]  
b=[[1,2],[3,4]]  
  
print(numpy.dot(a,b))

NUMPY INNER , OUTER

#inner  
  
#The inner tool returns the inner product of two arrays.  
  
import numpy  
  
A = numpy.array([0, 1])  
B = numpy.array([3, 4])  
  
print(numpy.inner(A, B)) #Output : 4  
#outer  
  
#The outer tool returns the outer product of two arrays.  
  
  
A = numpy.array([0, 1])  
B = numpy.array([3, 4])  
  
print(numpy.outer(A, B)) #Output : [[0 0]  
 # [3 4]]

Numpy polynomials

import numpy  
#poly  
  
#The poly tool returns the coefficients of a polynomial with the given sequence of roots.  
print(numpy.poly([-1, 1, 1, 10])) #Output : [ 1 -11 9 11 -10]  
#roots  
  
#The roots tool returns the roots of a polynomial with the given coefficients.  
  
print(numpy.roots([1, 0, -1])) #Output : [-1. 1.]  
#polyint  
  
#The polyint tool returns an antiderivative (indefinite integral) of a polynomial.  
  
print(numpy.polyint([1, 1, 1])) #Output : [ 0.33333333 0.5 1. 0. ]  
#polyder  
  
#The polyder tool returns the derivative of the specified order of a polynomial.  
  
print(numpy.polyder([1, 1, 1, 1])) #Output : [3 2 1]  
#polyval  
  
#The polyval tool evaluates the polynomial at specific value.  
  
print(numpy.polyval([1, -2, 0, 2], 4)) #Output : 34  
#polyfit  
  
#The polyfit tool fits a polynomial of a specified order to a set of data using a least-squares approach.  
  
print(numpy.polyfit([0,1,-1, 2, -2], [0,1,1, 4, 4], 2))  
#Output : [ 1.00000000e+00 0.00000000e+00 -3.97205465e-16]

NUMPY DETERMINANT, EIGHEN VALUES

#linalg.det  
  
#The linalg.det tool computes the determinant of an array.  
import numpy  
print(numpy.linalg.det([[1 , 2], [2, 1]])) #Output : -3.0  
  
#linalg.eig  
  
#The linalg.eig computes the eigenvalues and right eigenvectors of a square array.  
  
vals, vecs = numpy.linalg.eig([[1 , 2], [2, 1]])  
print(vals) #Output : [ 3. -1.]  
print(vecs) #Output : [[ 0.70710678 -0.70710678]  
 # [ 0.70710678 0.70710678]]  
#linalg.inv  
  
#The linalg.inv tool computes the (multiplicative) inverse of a matrix.  
  
print(numpy.linalg.inv([[1 , 2], [2, 1]])) #Output : [[-0.33333333 0.66666667]  
 # [ 0.66666667 -0.33333333]]

BUILT INS

EVAL

>>> eval("9 + 5")

14

>>> x = 2

>>> eval("x + 3")

5

>>> type(eval("len"))

<type 'builtin\_function\_or\_method'>

ARRAYS

from array import \*  
  
arr=array('i',[1,2,3,4,5])  
print(arr)  
  
arr=array('I',[1,2,3,4,5])  
print(arr)  
  
arr=array('b',[1,2,3,4,5])  
print(arr)  
  
arr=array('B',[1,2,3,4,5])  
print(arr)  
  
arr=array('h',[1,2,3,4,5])  
print(arr)  
  
arr=array('H',[1,2,3,4,5])  
print(arr)  
  
  
arr=array('L',[1,2,3,4,5])  
print(arr)  
  
arr=array('L',[1,2,3,4,5])  
print(arr)  
  
arr=array('d',[1,2,3,4,5])  
print(arr)  
  
arr=array('f',[1,2,3,4,5])  
print(arr)  
  
print(arr.buffer\_info())  
  
arr=array('i',[1,2,3,4,5])  
print(arr)  
  
for i in range(len(arr)):  
 print(arr[i],end=" ")  
print(" ")  
  
for i in range(len(arr)):  
 print(i,"->",arr[i],end=" ")  
 print(" ")  
  
for i in arr:  
 print(i,end=" ")  
print(" ")  
  
arr.reverse()  
print(arr)  
  
arr.remove(1)  
print(arr)  
  
arr.append(1)  
print(arr)  
  
arr.insert(2,1000)  
print(arr)  
  
print(arr.pop())  
print(arr)  
  
print(arr.pop(2))  
print(arr)  
  
arr.reverse()  
print(arr)  
arr.append(1)  
arr.append(1)  
arr.append(1)  
  
print(arr.count(1))  
print(arr)  
  
print(arr.index(3))  
  
l=arr.tolist()  
print(l)  
  
print("itemsize",arr.itemsize)  
  
print("len of array",len(arr))

THREADING IN PYTHON

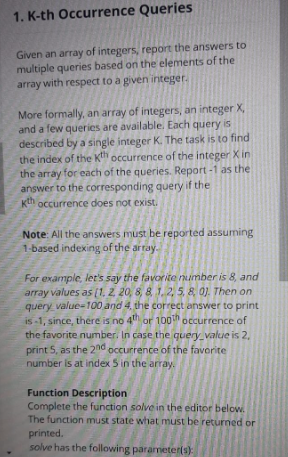
from threading import \*  
  
# FIRST WAY  
def show():  
 print("THIS IS CHILD THREAD")  
  
t=Thread(target=show())  
t.start()  
print("THIS IS PARENT THREAD")  
  
  
# SECOND WAY  
  
class Mythread(Thread):  
 def run(self):  
 for i in range(5):  
 print(" THIS IS CHILD THREAD FROM RUN METHOD")  
  
t=Mythread()  
  
t.start()  
for i in range(5):  
 print(" THIS IS parent THREAD FROM OUTSIDE")  
  
# THIRD WAY  
  
class demo:  
 def display(self):  
 for i in range(5):  
 print(" FROM DISPLAY METHOD ")  
  
obj=demo()  
t=Thread(target=obj.display())  
t.start()  
for i in range(5):  
 print(" FROM OUTSIDE OF DISPLAY METHOD OF DEMO CLASS ")  
  
  
# MULTI THREADING  
import time  
class work:  
 def num(self):  
 for i in range(5):  
 print("NUM : ",i)  
 time.sleep(0.5)  
 def double(self):  
 for i in range(5):  
 print("DOUBLE : ",i\*2)  
 time.sleep(0.5)  
  
 def square(self):  
 for i in range(5):  
 print("SQUARE : ",i\*\*2)  
 time.sleep(0.5)  
  
  
  
  
obj=work()  
t1=Thread(target=obj.num())  
t2=Thread(target=obj.double())  
t3=Thread(target=obj.square())  
  
  
t1.start()  
t2.start()  
t3.start()  
  
t1.join()  
t2.join()  
t3.join()  
  
for i in range(5):  
 print(" FROM OUTSIDE OF DISPLAY METHOD OF WORK CLASS ")

IN BUILT FUNCTIONS

<https://www.w3schools.com/python/python_ref_functions.asp>

x = abs(-7.25)  
print(x)  
x = abs(3+5j)  
print(x)  
  
mylist = [True, True, True]  
x = all(mylist)  
print(x)  
  
mylist = [0, 1, 1]  
x = all(mylist)  
print(x)  
  
  
mytuple = (0, True, False)  
x = all(mytuple)  
print(x)  
  
myset = {0, 1, 0}  
x = all(myset)  
print(x)  
  
mydict = {0 : "Apple", 1 : "Orange"}  
x = all(mydict)  
print(x)  
  
  
mylist = [False, True, False]  
x = any(mylist)  
print(x)  
  
  
mytuple = (0, 1, False)  
x = any(mytuple)  
print(x)  
  
myset = {0, 1, 0}  
x = any(myset)  
print(x)  
  
mydict = {0 : "Apple", 1 : "Orange"}  
x = any(mydict)  
print(x)  
  
x = ascii("My name is Ståle")  
print(x)  
  
x = bin(36)  
print(x)  
  
x = bool(1)  
print(x)  
  
x = bytearray(4)  
print(x)  
  
x = bytes(4)  
print(x)  
  
def x():  
 a = 5  
print(x)  
  
  
print(callable(x))  
print(x)  
  
  
x = 5  
print(callable(x))  
  
x = chr(97)  
print(x)  
  
  
x = compile('print(55)', 'test', 'eval')  
exec(x)  
  
x = compile('print(55)\nprint(88)', 'test', 'exec')  
exec(x)  
  
# DELETES ATTRIBUTES IF THE CLASS  
class Person:  
 name = "John"  
 age = 36  
 country = "Norway"  
  
delattr(Person, 'age')  
  
"""  
The getattr() function, to get the value of an attribute  
  
The hasattr() function, to check if an attribute exist  
  
The setattr() function, to set the value of an attribute  
"""  
  
x = dict(name = "John", age = 36, country = "Norway")  
print(x)  
  
class Person:  
 name = "John"  
 age = 36  
 country = "Norway"  
  
print(dir(Person))  
print(dir("hello"))  
print(dir(1))  
print(dir(1.3))  
print(dir(1+2j))  
  
#Display the quotient and the remainder of 5 divided by 2:  
#divmod(dividend, divisor)  
  
x = divmod(5, 2)  
print(x)  
  
x = 'print(55)'  
print(eval(x))  
print(eval("2+3"))  
print(eval("2-3"))  
print(eval("2\*3"))  
print(eval("2/3"))  
print(eval("2//3"))  
print(eval("2\*\*3"))  
  
"""  
The enumerate() function takes a collection (e.g. a tuple) and returns it as an enumerate object.  
  
The enumerate() function adds a counter as the key of the enumerate object.  
  
iterable An iterable object  
start A Number. Defining the start number of the enumerate object. Default 0  
  
"""  
x = ('apple', 'banana', 'cherry')  
y = enumerate(x)  
print(x)  
print(y)  
  
  
a = ("John", "Charles", "Mike")  
b = ("Jenny", "Christy", "Monica")  
  
x = zip(a, b)  
print(x)  
  
a = ("John", "Charles", "Mike")  
b = ("Jenny", "Christy", "Monica", "Vicky")  
  
x = zip(a, b)  
print(x)  
  
class Person:  
 name = "John"  
 age = 36  
 country = "norway"  
  
x = vars(Person)  
print(x)  
  
#Return the type of these objects:  
a = ('apple', 'banana', 'cherry')  
b = "Hello World"  
c = 33  
  
x = type(a)  
y = type(b)  
z = type(c)  
print(x,y,z)  
  
a = ("b", "g", "a", "d", "f", "c", "h", "e")  
x = sorted(a)  
print(x)  
  
  
a = ("h", "b", "a", "c", "f", "d", "e", "g")  
x = sorted(a, reverse=True)  
print(x)  
  
#Return the integer that represents the character "h":  
  
x = ord("h")  
  
def myfunc(n):  
 return len(n)  
  
x = map(myfunc, ('apple', 'banana', 'cherry'))  
print(x)  
#Make new fruits by sending two iterable objects into the function:  
  
def myfunc(a, b):  
 return a + b  
  
x = map(myfunc, ('apple', 'banana', 'cherry'), ('orange', 'lemon', 'pineapple'))  
print(x)

PROBLEM



K TH OCCURRENCE QUERIES

def solve(X, arr, query\_values):  
 q = query\_values  
 temp=arr  
 h = []  
 for i in q:  
 a = list(arr)  
 if arr.count(X) <i:  
 h.append(-1)  
 else:  
 for j in range(i-1):  
 h.append(a.index(X)+1)  
 return h  
  
#print(solve(9,[9,8,9,9],[7,3,7,6]))

FILE HANDLING

WRITING TO FILE

s="""puspha   
pushpa raj yevva thaggedhele..."""  
print(s)  
f=open('writefile.txt','w')  
f.write(s)  
f.close()

APPENDING A FILE

s="""pushpa ante fire anukunitiva ...  
 floweruuuu flowerehhhhh"""  
print(s)  
f=open('writefile.txt','a')  
f.write("\n")  
f.write(s)  
f.write('\n')  
f.close()

READING A FILE

s="""pushpa ante fire anukunitiva ...  
 floweruuuu flowerehhhhh"""  
print(s)  
f=open('newfile.txt','a')  
f.write("\n")  
f.write(s)  
f.write('\n')  
f.close()  
  
  
data=open('newfile.txt','r').read()  
print(data)  
  
data2=open('newfile.txt','r').readlines()  
print(data2)  
  
data3=open('newfile.txt','r').readline()  
print(data3)

GLOBAL v/s LOCAL

x=6  
def work():  
 y=5  
 print(y)  
 print(x)  
 #x+=1 returns an error cannot simply modify the global variables locally  
  
work()  
print(x)  
#print(y) since variable y is not defined , its just defined locally so return an error  
  
  
#global variables can be modified using global keyword  
x=6  
def work():  
 y=5  
  
 global x  
 print(x, y)  
 x += 1  
 print(x,y)  
  
  
work()  
  
y=x  
print(x,y)

STATISTICS MODULE

import statistics as s  
  
l = [5,3,2,9,9,7,4,3,1,8,9]  
  
x=s.mean(l)  
y=s.median(l)  
z=s.mode(l)  
p=s.stdev(l)  
q=s.variance(l)  
r=s.pvariance(l)  
m=s.fmean(l)  
n=s.geometric\_mean(l)  
o=s.harmonic\_mean(l)  
  
print(x)  
print(y)  
print(z)  
print(p)  
print(q)  
print(r)  
print(m)  
print(n)  
print(o)

IMPORT SYNTAX

l = [5,3,2,9,9,7,4,3,1,8,9]  
  
import statistics  
print(statistics.mean(l))  
  
import statistics as s  
print(s.mean(l))  
  
from statistics import mean,mode,median  
print(mean(l))  
print(mode(l))  
print(median(l))  
  
from statistics import mean as m  
print(m(l))  
  
from statistics import \*  
print(mean(l))  
print(mode(l))  
print(median(l))

LIST COMPREHENSIONS

l=[i for i in range(10)]  
print(l)  
  
s={i for i in range(10)}  
print(s)  
  
# even numbers  
l=[i for i in range(1,10) if i%2==0]  
print(l)  
  
  
# odd numbers  
l=[i\*\*2 for i in range(1,10) if i%2==0]  
print(l)  
  
l=[i\*\*2 for i in range(1,10) if i%2!=0]  
print(l)  
  
# numbers whose perfect squares those are divived by 5  
l5=[i\*\*2 for i in range(1,100) if i\*\*2 %5==0]  
print(l5)  
  
# numbers whose perfect squares those are divived by 11  
l11=[i\*\*2 for i in range(1,100) if i\*\*2 %11==0]  
print(l11)  
  
# accessing list of tuples using if condition in list comprehensions  
  
movies=[("akhanda",2021),("vasool",2020),("legend",1999),("simha",1998)]  
  
pre2k=[title for (title,year) in movies if year<2000]  
print(pre2k)  
  
post2k=[title for (title,year) in movies if year>2000]  
print(post2k)  
  
# to multiply each element of the list \*5  
l=[1,2,3,4,5,6,7,8,9,10]  
print(l)  
  
l=[5\*i for i in l]  
print(l)  
  
# possible permutation of points  
  
lx=[i for i in range(-6,6)]  
ly=[i for i in range(0,5)]  
print(lx)  
print(ly)  
  
points=[(x,y) for x in lx for y in ly]  
print(points)  
  
# permutations of persons and sections  
  
names=["pemma","subbu","gompee","surya","mahesh"]  
secs=('A','B','C','D','E','F')  
  
print(names)  
print(secs)  
  
l=[{a:b} for a in names for b in secs]  
print(l)  
  
# expression if else in list  
  
l1=[i for i in range(0,11)]  
print(l1)  
l2=[str(i)+" - EVEN " for i in l1 if i%2==0 ]  
print(l2)  
l2=[str(i)+" - ODD " for i in l1 if i%2!=0 ]  
print(l2)  
l2=[str(i)+" - EVEN " if i%2==0 else str(i)+" - ODD " for i in l1 ]  
print(l2)  
  
  
num=[100,200,300,400,500]  
  
# multiply each element by 10  
l=[x\*10 for x in num]  
print(l)  
  
# add each element by 10  
l=[x+10 for x in num]  
print(l)  
  
  
l=[(x,x\*10) for x in num]  
print(l)  
  
l=[(x,x+10) for x in num]  
print(l)  
  
# to change case of strings  
l=["surya mahesh","mahesh nanda","gompee gomu","pemma pems",'subbbu subss']  
  
u=[s.upper() for s in l]  
l=[s.lower() for s in l]  
c=[s.capitalize() for s in l]  
t=[s.title() for s in l]  
  
print(l)  
print(u)  
print(l)  
print(c)  
print(t)  
  
s="abcde12345"  
  
# extract digits from string  
dig=[x for x in s if x.isdigit()]  
print(dig)  
  
dig=[int(x) for x in s if x.isdigit()]  
print(dig)  
  
# extract alphabets from string  
alp=[x for x in s if x.isalpha()]  
print(alp)  
  
# COMPREHENSIONS ON NESTED LISTS  
  
l=[[1,2,3],[10,20,30],[100,200,300]]  
  
nums=[x[0] for x in l]  
print(nums)  
  
nums=[x[1] for x in l]  
print(nums)  
  
nums=[x[2] for x in l]  
print(nums)  
  
# COMPREHENSIONS ON FUNCTIONS  
  
def msg(s):  
 return "HELLO,"+s  
  
names=["pemma","subbu","gompee","surya","mahesh"]  
l=[msg(s) for s in names]  
print(l)  
  
  
l1=[1,2,3,4,5]  
l2=[1,2,3,4,5]  
print(l1)  
print(l2)  
print(l1+l2)  
#l1 + l2 appends list1 to list2  
  
# to add/manipualte all permutations of individual elements of two lists use comprehensions  
  
l=[x+y for x in l1 for y in l2]  
print(l)  
  
l=[x-y for x in l1 for y in l2]  
print(l)  
  
l=[x\*y for x in l1 for y in l2]  
print(l)  
  
l=[x/y for x in l1 for y in l2]  
print(l)  
  
l=[x//y for x in l1 for y in l2]  
print(l)  
  
# to add/manipualte individual elements of two lists use comprehensions  
  
l=[l1[i]+l2[i] for i in range(0,len(l1))]  
print(l)  
  
l=[l1[i]-l2[i] for i in range(0,len(l1))]  
print(l)  
  
l=[l1[i]\*l2[i] for i in range(0,len(l1))]  
print(l)  
  
l=[l1[i]/l2[i] for i in range(0,len(l1))]  
print(l)  
  
l=[l1[i]//l2[i] for i in range(0,len(l1))]  
print(l)  
  
# number of permutations such that x and y aren't same  
p=[1,2,3,4,5]  
q=[1,2,3,4,5]  
l=[(x,y) for x in p for y in q if x!=y]  
print(l)  
  
# SQUARE ROOT OF NUMBERS BELOW 10  
  
import math  
l=[math.sqrt(i) for i in range(0,11)]  
print(l)  
  
l=[int(math.sqrt(i)) for i in range(0,11)]  
print(l)

# list comprehensions with conditions  
l1=[x for x in range(20) if x<10 and x%2!=0]  
l2=[x for x in range(20) if x<10 and x%2==0]  
l3=[x for x in range(20) if x>10 and x%2!=0]  
l4=[x for x in range(20) if x>10 and x%2!=0]  
  
print(l1)  
print(l2)  
print(l3)  
print(l4)  
  
# list comprehensions with multiple ifs  
l1=[x for x in range(20) if x<10 if x%2!=0]  
print(l1)

PANDAS LIBRARY

import pandas as pd  
  
print(pd.\_\_version\_\_)  
print(pd.array([1,2,3,4,5]))  
print(pd.array([1,2,3,4,5],dtype=float))  
print(pd.array([1,2,3,4,5],dtype=str))  
print(pd.array((1,2,3,4,5)))  
  
# DataFrame using list of tuples  
  
l=[("pemma",1,100),("subbu",2,100),("gompee",3,100),("surya",4,100)]  
df=pd.DataFrame(l)  
print(l)  
print(df)  
  
  
# DataFRmae using dictionaries  
  
di={"name":["pemma","subbu","gompee","surya","pemss","subbss","gompss","mahesh"],"rollnum":[1,2,3,4,5,6,7,8],"percentage":[90,87,89,100,78,99,79,100]}  
df=pd.DataFrame(di)  
print(di)  
print(df)  
  
# bullt in functions  
  
print(df.head(2))  
print(df.head(3))  
  
print(df.tail(2))  
print(df.tail(3))  
  
print(df.describe())  
  
print(df.values)  
print(df.keys)  
  
print(df.shape)  
  
# indexing slicing  
  
print(df[0::2])  
print(df[::3])  
print(df[0:5:4])  
print(df[1::1])  
  
  
print(df["name"])  
print(df["rollnum"])  
print(df["percentage"])  
  
print(df[["name","rollnum"]])  
  
print(df[["name","percentage"]])  
  
print(df[["rollnum","percentage"]])  
  
print(df[["name","rollnum"]][0:5:2])  
print(df[["name","percentage"]][1:6:3])  
print(df[["rollnum","percentage"]][2::1])  
  
print(df.iteritems)  
  
for rec in df.iterrows():  
 print(rec)  
  
for rec in df.\_iter\_column\_arrays():  
 print(rec)  
  
for rec in df.itertuples():  
 print(rec)  
  
# UNDERSTANDING LOC[]  
  
print(df.loc[1])  
print(df.loc[2])  
  
for i in range(len(df)):  
 print(df.loc[i])  
  
  
print(df.loc[1,["name","rollnum"]])  
print(df.loc[2,["name","rollnum"]])  
  
for i in range(len(df)):  
 print(df.loc[i,["name","rollnum"]])  
  
for i in range(len(df)):  
 print(df.loc[i, ["name", "percentage"]])  
  
for i in range(len(df)):  
 print(df.loc[i, ["rollnum", "percentage"]])  
  
  
print(df.loc[0:5])  
print(df.loc[3:8])  
  
print(df.loc[0:5,"name"])  
print(df.loc[3:8,["name","rollnum"]])  
  
# ILOC  
  
print(df.iloc[1])  
print(df.iloc[1:5,0:2])  
  
print(df.iloc[0:5,1])  
print(df.iloc[3:8,[0,1]])  
  
# SORTING DataFrame  
  
print(df.sort\_values)  
print(df.sort\_index)  
  
print(df.sort\_values("percentage"))  
print(df.sort\_values("rollnum"))  
print(df.sort\_values("name"))  
  
  
print(df.sort\_values("percentage",ascending=False))  
print(df.sort\_values("rollnum",ascending=False))  
print(df.sort\_values("name",ascending=False))  
  
print(df.sort\_values(["percentage","rollnum"]))  
print(df.sort\_values(["percentage","name"]))  
print(df.sort\_values(["rollnum","name"]))  
  
# MANIPULATING DataFrame  
  
df["percentage"]=0  
print(df)  
  
# adding new column name with default value  
  
df["JEE"]=95  
print(df)  
  
df["percentage"]=92  
print(df)  
  
# adding new column name , expression  
  
df["JEE"]=df["percentage"]\*1.1  
print(df)  
  
# REMOVING COLUMN  
  
df["GRADE"]="pass"  
print(df)  
  
print(df.drop(columns="GRADE"))  
print(df)  
  
df.drop(columns="GRADE",inplace=True)  
print(df)  
  
# REMOVE DUPLICATES  
print(df.duplicated())  
  
print(df.drop\_duplicates())  
print(df)  
  
df.drop\_duplicates(inplace=True)  
print(df)

TKINTER - GUI

Empty frame

from tkinter import \*  
  
main=Tk()  
main.mainloop()

FRAME BASIC WINDOW

from tkinter import \*  
  
#obj is the instance of Tk class  
  
obj=Tk()  
  
#GUI LOGIC  
obj.mainloop()

geometry(WxH) , maxsize(W,H) , minsize(W,H)

Label(text=”HHHSHHSAJHD”)

GUI - LABEL

from tkinter import \*  
  
# surya is instance for Tk class  
surya=Tk()  
  
  
# geometry( WIDTH X HEIGHT )  
surya.geometry("500x500")  
  
# minsize ( width , height )  
surya.minsize(100,100)  
  
# maxsize ( width , height )  
surya.maxsize(700,700)  
  
# Label(text="PUSHPARAJ") - NO USER INTERACTION  
l1 = Label(text="WELCOME TO GUI")  
l1.pack()  
  
# GUI LOGIC  
surya.mainloop()

LABEL and PACKING TO WINDOW (OBJ)

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text="SURYA MAHESH",font=("Arial Bold",20))  
l1.pack()  
  
l2=Label(obj,text="SURYA MAHESH",font=("Arial 20 bold"))  
l2.pack()  
  
l3=Label(text="SURYA MAHESH",font=("Helvetica",20,"bold"))  
l3.pack()  
  
l4=Label(text="SURYA MAHESH",bg="red",fg="yellow",padx=20,pady=20,font=("comicsansns",20,"bold"))  
l4.pack()  
  
  
obj.mainloop()

LABEL USING GRID

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text="SURYA MAHESH",font=("Arial Bold",20))  
l1.grid(row=1,column=10)  
  
l2=Label(obj,text="SURYA MAHESH",font=("Arial 20 bold"))  
l2.grid(row=2,column=10)  
  
l3=Label(text="SURYA MAHESH",font=("Helvetica",20,"bold"))  
l3.grid(row=3,column=10)  
  
l4=Label(text="SURYA MAHESH",bg="red",fg="yellow",padx=20,pady=20,font=("comicsansns",20,"bold"))  
l4.grid(row=4,column=10)  
  
  
obj.mainloop()

PNG IMAGES

from tkinter import \*  
  
myframe= Tk()  
myframe.title("IMAGE - FRAME")  
myframe.geometry("500x500")  
  
myframe.maxsize(1000,1000)  
myframe.minsize(100,100)  
  
mylabel1=Label(text=" WELCOME TO GUI ")  
mylabel1.pack()  
  
mypic=PhotoImage(file="MyLogo.png")  
mylabel2=Label(image=mypic)  
mylabel2.pack()  
  
myframe.mainloop()

JPG IMAGES

from tkinter import \*  
from PIL import Image,ImageTk  
  
myframe= Tk()  
myframe.title("IMAGE - FRAME")  
myframe.geometry("500x500")  
  
myframe.maxsize(1000,1000)  
myframe.minsize(100,100)  
  
mylabel1=Label(text=" WELCOME TO GUI ")  
mylabel1.pack()  
  
image1=Image.open("Mypic.jpg")  
photo1=ImageTk.PhotoImage(image1)  
  
mylabel2=Label(image=photo1)  
mylabel2.pack()  
  
myframe.mainloop()

LABEL , PACK ATTRIBUTES

# LABEL ATTRIBUTES , PACK ATTRIBUTES   
  
from tkinter import \*  
  
frame1 = Tk()  
frame1.title(" PUSHPA GUI ")  
frame1.geometry("750x750")  
frame1.minsize(100,100)  
frame1.maxsize(1000,1000)  
  
label1=Label(text=" WELCOME TO PUSHPA WORLD ",font=15)  
label1.pack()  
  
# LABEL OPTIONS  
"""  
text = adds text  
bg = background  
fg = foreground  
font   
1) font=("comicsansns",20,"bold")  
2) font="comicsansns 20 bold"  
3) font = 10  
 we can simply add what ever label options / label attributes we need   
  
, padx ( padding in x direction ),pady - ( Padding in y direction ) ((, relief  
  
 """  
  
label2=Label(text=" PUSHPA , PUSHPA RAJ YEVVA THAGGEDHELE ",bg="red",fg="yellow",padx=20,pady=20,font=("comicsansns",20,"bold"))  
label2.pack()  
  
  
label3=Label(text=" PUSHPA ANTE FLOWER ANUKINTIVA................",bg="pink",fg="green",font=15)  
label3.pack()  
  
label4=Label(text=" FIRE UU........ FIRE EHHHH ....................... ",bg="orange",fg="blue",padx=10,pady=10,font="comicsansns 15 bold")  
label4.pack()  
  
label5=Label(text=" PUSHPA ANTEY YODIKI BAYAPADDU....",bg="red",fg="yellow",font=15,borderwidth=30,relief=SUNKEN)  
label5.pack()  
  
label6=Label(text=" SHIKAWAT SIR NA KODAKA.. ENDHEY BULLETKI BAYAPADE BRAND AAA NADHIII....",bg="magenta",fg="blue",font=15,borderwidth=30)  
label6.pack()  
  
# pack attributes ,  
# anchor = "ne" , "se" , " nw "," sw "  
# side=top, bottom , left , right  
  
  
label1.pack(anchor="nw")  
label2.pack(anchor="sw")  
label3.pack(anchor="ne")  
label4.pack(anchor="se")  
"""  
label5.pack(anchor="nw",side=BOTTOM)  
label6.pack(anchor="sw",side=TOP)  
label3.pack(anchor="ne",side=LEFT)  
label4.pack(anchor="se",side=RIGHT)  
  
"""  
  
frame1.mainloop()

FRAMES IN GUI

from tkinter import \*  
  
obj=Tk()  
obj.title("FRAME")  
obj.geometry("750x750")  
#obj.maxsize(1000,1000)  
obj.minsize(500,500)  
  
f1=Frame(obj,bg="grey",borderwidth=6)  
f1.pack(side=TOP,fill="x",padx=50,pady=30)  
l1=Label(f1,text="WELCOME TO GUI",fg="red",bg="black",font="comicsansns 20 bold")  
l1.pack()  
  
f2=Frame(obj,bg="grey",borderwidth=6)  
f2.pack(side=BOTTOM,fill="x",pady=50,padx=30)  
l2=Label(f2,text="THANK YOU !",fg="red",bg="black",font="Helvetica 20 bold")  
l2.pack()  
  
l3=Label(text="HELLO ! SURYA MAHESH.. NICE TO SEE YOU.. HOPE WE WILL MEET AGAIN ",fg="BLUE",font="comicsansns 15 bold")  
l3.pack()  
  
  
# frames are inside obj's labels inside the frame or labels should be inside obj  
obj.mainloop()

SIMPLE BUTTON AND PACK TO WINDOW

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text="CLICK THE BUTTON",font=("Arial Bold",15))  
l1.pack()  
  
b1=Button(obj,text=" PUCHUK ",bg="red",font=10)  
b1.pack()  
  
obj.mainloop()

SIMPLE BUTTON GRID

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text="CLICK THE BUTTON",font=("Arial Bold",15))  
l1.grid(row=1,column=2)  
  
b1=Button(obj,text=" PUCHUK ",bg="red",font=10)  
b1.grid(row=2,column=2)  
  
obj.mainloop()

BUTTONS IN GUI

from tkinter import \*  
  
obj=Tk()  
obj.title("BUTTON WINDOW")  
obj.geometry("750x750")  
#obj.maxsize(1000,1000)  
obj.minsize(100,100)  
  
l=Label(text="WELCOME TO GUI",font="Helvetica 15 bold")  
l.pack()  
  
f1=Frame(obj,borderwidth=5)  
f1.pack(side=LEFT, anchor="nw")  
  
def hello():  
 print("HELLO MURUGAN SYNDICATE MEMBER PUSHPA PESIRE..")  
 l1=Label(text="HELLO MURUGAN SYNDICATE MEMBER PUSHPA PESIRE..",fg="blue",font=20)  
 l1.pack()  
  
def diologue():  
 print("PUSHPA NAAM SUNKE FLOWER SAMJE KYA ? FIRE EH MEIII...")  
 l3 = Label(text="PUSHPA NAAM SUNKE FLOWER SAMJE KYA ? FIRE EH MEIII...", fg="blue", font=20)  
 l3.pack()  
  
  
def byebye():  
 print("FIRE UUU FIRE EHHH ")  
 l2 = Label(text="FIRE UUU FIRE EHHH ", fg="blue", font=20)  
 l2.pack()  
  
  
b1=Button(f1,bg="red",text="CLICK HERE",command=hello)  
b1.pack(side=LEFT)  
  
b2=Button(f1,bg="red",text="SUBMIT NOW",command=byebye)  
b2.pack(side=LEFT)  
  
b3=Button(bg="red",text="PUSHPA",command=diologue)  
b3.pack()  
  
"""  
  
b3=Button(f1,bg="red",text="CLICK HERE")  
b3.pack()  
  
b4=Button(f1,bg="red",text="SUBMIT NOW")  
b4.pack()  
  
"""  
  
  
obj.mainloop()

BUTTONS AND COMMAND

from tkinter import \*  
  
obj=Tk()  
obj.title("BUTTON WINDOW")  
obj.geometry("750x750")  
#obj.maxsize(1000,1000)  
obj.minsize(100,100)  
  
l=Label(text="WELCOME TO GUI",font="Helvetica 15 bold")  
l.pack()  
  
f1=Frame(obj,borderwidth=5)  
f1.pack(side=LEFT, anchor="nw")  
  
def hello():  
 print("HELLO MURUGAN SYNDICATE MEMBER PUSHPA PESIRE..")  
 l1=Label(text="HELLO MURUGAN SYNDICATE MEMBER PUSHPA PESIRE..",fg="magenta",font=20)  
 l1.pack()  
  
def diologue():  
 print("PUSHPA NAAM SUNKE FLOWER SAMJE KYA ? FIRE EH MEIII...")  
 l3 = Label(text="PUSHPA NAAM SUNKE FLOWER SAMJE KYA ? FIRE EH MEIII...", fg="blue", font=20)  
 l3.pack()  
  
  
def byebye():  
 print("FIRE UUU FIRE EHHH ")  
 l2 = Label(text="FIRE UUU FIRE EHHH ", fg="orange", font=20)  
 l2.pack()  
  
  
b1=Button(f1,bg="red",text="CLICK HERE",command=hello)  
b1.pack(side=BOTTOM)  
  
b2=Button(f1,bg="red",text="SUBMIT NOW",command=byebye)  
b2.pack(side=BOTTOM)  
  
  
b3=Button(f1,bg="red",text="PUSHPA",command=diologue)  
b3.pack(side=BOTTOM)  
  
  
"""  
  
b3=Button(f1,bg="red",text="CLICK HERE")  
b3.pack()  
  
b4=Button(f1,bg="red",text="SUBMIT NOW")  
b4.pack()  
  
"""  
  
  
obj.mainloop()

SIMPLY ENTRY WIDGET

from tkinter import \*  
  
master = Tk()  
master.minsize(250,250)  
Label(master, text='First Name').grid(row=0)  
Label(master, text='Last Name').grid(row=1)  
e1 = Entry(master)  
e2 = Entry(master)  
e1.grid(row=0, column=1)  
e2.grid(row=1, column=1)  
  
master.mainloop()

SIMPLE ENTRY WIDGET

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text=" WLCOME TO GUI ",font=("Arial Bold",15))  
l1.pack()  
  
l2=Label(obj,text="ENTER YOUR NAME ")  
l2.pack()  
  
t1=Entry(obj,width=15)  
t1.pack()  
  
  
obj.mainloop()

ENTRY WITH StringVar , IntVar

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text=" WLCOME TO GUI ",font=("Arial Bold",15))  
l1.pack()  
  
l2=Label(obj,text="ENTER YOUR NAME ")  
l2.pack()  
  
t1=StringVar()  
t1entry=Entry(obj,width=15,textvariable=t1)  
t1entry.pack()  
  
l3=Label(obj,text="ENTER YOUR NUMBER ")  
l3.pack()  
  
t2=IntVar()  
t2entry=Entry(obj,width=10,textvariable=t2)  
t2entry.pack()  
  
obj.mainloop()

BUTTONS

from tkinter import \*  
  
root = Tk()  
root.minsize(300,300)  
frame = Frame(root)  
frame.pack()  
  
bottomframe = Frame(root)  
bottomframe.pack(side=BOTTOM)  
redbutton = Button(frame, text='Red', fg='red')  
redbutton.pack(side=LEFT)  
greenbutton = Button(frame, text='Brown', fg='brown')  
greenbutton.pack(side=LEFT)  
bluebutton = Button(frame, text='Blue', fg='blue')  
bluebutton.pack(side=LEFT)  
blackbutton = Button(bottomframe, text='Black', fg='black')  
blackbutton.pack(side=BOTTOM)  
root.mainloop()

LABELS , ENTRY , BUTTONS (IMPORTANT)

( l.configure(…..) to change the already present labels, to get the keys l.cget(key) , b.cget(key)

T.get() for the textfield )

from tkinter import \*  
  
obj=Tk()  
obj.title("LABELS AND TEXTS")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text=" WELCOME TO GUI! ",font=("Arial Bold",15))  
l1.pack()  
  
l2=Label(obj,text="ENTER YOUR NAME ")  
l2.pack()  
  
l3=Label(obj,text="FILL ALL THE FIELDS.... ")  
l3.pack()  
  
t1=StringVar()  
t1entry=Entry(obj,width=15,textvariable=t1)  
t1entry.pack()  
  
def submit():  
 print("submitted successfully")  
 print(b1.widgetName,"PRESSED")  
 print("FONT OF LABEL IS :",l1.cget("font"))  
 print(l1.cget("text"))  
 print(t1.get())  
 print("button is : ",b1.cget("text"))  
 if len(t1.get()):  
 l2.configure(text="")  
 l3.configure(text="THANK YOU.....")  
  
 txt=l1.cget("text")+"\nHELLO !"+str(t1.get())  
 l1.configure(text=txt,fg="black",bg="red",font="Arial 15 bold")  
  
  
b1=Button(obj,text="SUBMIT",command=submit)  
b1.pack()  
  
  
obj.mainloop()

GRID ,TEXTFIELDS, LOGIN FORM

from tkinter import \*  
  
obj=Tk()  
obj.title("BUTTON WINDOW")  
obj.geometry("750x750")  
#obj.maxsize(1000,1000)  
obj.minsize(100,100)  
  
  
"""  
l=Label(obj,text="WELCOME TO GUI",font="Helvetica 15 bold")  
l.pack()  
"""  
  
  
l1=Label(obj,text="USERNAME",font="Helvetica 15 bold")  
#l1.pack()  
l1.grid()  
  
l2=Label(obj,text="PASSWORD",font="Helvetica 15 bold")  
#l2.pack()  
l2.grid(row=1)  
  
# variable classes in tkinter  
# BooleanVar , DoubleVar , IntVar , StringVar  
  
  
t1= StringVar()  
t2=StringVar()  
  
userentry = Entry(obj,textvariable=t1)  
passentry = Entry(obj,textvariable=t2)  
  
userentry.grid(row=0,column=1)  
passentry.grid(row=1,column=1)  
  
  
def done():  
 print("USERNAME : ",t1.get()," PASSWORD : ",t2.get())  
 l1=Label(text=t1.get(),bg="black",fg="red",font=20)  
 l1.grid()  
 l2 = Label(text=t2.get(),bg="black",fg="red",font=20)  
 l2.grid()  
 obj2 = Tk()  
 obj2.title("BUTTON WINDOW")  
 obj2.geometry("750x750")  
 # obj.maxsize(1000,1000)  
 obj2.minsize(100, 100)  
 l1 = Label(obj2,text="HELLO "+str(t1.get()), bg="black", fg="red", font=20)  
 l1.pack()  
 l2 = Label(obj2,text="WELCOME TO PUSHPA WORLD ", bg="black", fg="red", font=20)  
 l2.pack()  
 obj2.mainloop()  
  
  
b1=Button(text="SUBMIT",command=done,bg="red")  
b1.grid()  
obj.mainloop()

SUBMIT FORM , INTERACTIVE BUTTONS

from tkinter import \*  
  
obj=Tk()  
obj.title("BUTTON WINDOW")  
obj.geometry("750x750")  
#obj.maxsize(1000,1000)  
obj.minsize(100,100)  
  
l=Label(obj,text=" WELOCOME TO GUI ",fg="magenta",bg="black",font="Heveltica 20 bold")  
l.grid(row=0,column=3)  
  
l1=Label(obj,text=" NAME ",font="Heveltica 15 bold")  
l1.grid(row=1,column=2)  
t1=StringVar()  
t1entry=Entry(obj,textvariable=t1)  
t1entry.grid(row=1,column=3)  
  
l2=Label(obj,text=" PHONE ",font="Heveltica 15 bold")  
l2.grid(row=2,column=2)  
t2=StringVar()  
t2entry=Entry(obj,textvariable=t2)  
t2entry.grid(row=2,column=3)  
  
l3=Label(obj,text=" GENDER ",font="Heveltica 15 bold")  
l3.grid(row=3,column=2)  
t3=StringVar()  
t3entry=Entry(obj,textvariable=t3)  
t3entry.grid(row=3,column=3)  
  
l4=Label(obj,text=" EMERGENCY CONTACT ",font="Heveltica 15 bold")  
l4.grid(row=4,column=2)  
t4=StringVar()  
t4entry=Entry(obj,textvariable=t4)  
t4entry.grid(row=4,column=3)  
  
l5=Label(obj,text=" PAYMENT ",font="Heveltica 15 bold")  
l5.grid(row=5,column=2)  
t5=IntVar()  
t5entry=Entry(obj,textvariable=t5)  
t5entry.grid(row=5,column=3)  
  
def submitclicked():  
 s=""  
 s+="\n NAME : "+t1.get()+"\n PHONE : "+t2.get()+"\n GENDER : "+t3.get()+"\n EMERGENCY CONTACT: "+t4.get()+"\n PAYMENT : "+str(t5.get())  
 print(s)  
 obj2 = Tk()  
 obj2.title("BUTTON WINDOW")  
 obj2.geometry("750x750")  
 # obj.maxsize(1000,1000)  
 obj2.minsize(100, 100)  
  
 l=Label(obj2,text=s,font="Helvetica 15 bold",bg="yellow",fg="red")  
 l.grid(row=3,column=3)  
  
  
 obj2.mainloop()  
  
  
b1=Button(obj,text=" SUBMIT ",font="Heveltica 17 bold",bg="grey",command=submitclicked)  
b1.grid(row=7,column=3)  
  
  
obj.mainloop()

COMBOBOX

from tkinter import \*  
from tkinter.ttk import \*  
  
obj=Tk()  
obj.title("LABELS AND TEXTS")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text=" WELCOME TO GUI! ",font=("Arial Bold",15))  
l1.pack()  
  
cb=Combobox(obj)  
cb["values"]=(1,2,3,4,5,"pushpa",3.14,)  
cb.current(3)  
cb.pack()  
  
def submit():  
 print("currently cb selected item is : ",cb.current())  
 print(cb.get(),"selected")  
 print(b1.cget("text"),"pressed")  
 print(cb.keys())  
  
b1=Button(obj,text="SUBMIT",command=submit)  
b1.pack()  
  
def pushpa():  
 cb.set("pushpa")  
  
b2= Button(obj, text="pushpa", command=pushpa)  
b2.pack()  
  
  
obj.mainloop()

SIMPLE CHECK BUTTON WIDGET

from tkinter import \*  
from tkinter.ttk import \*  
  
obj=Tk()  
obj.title("LABELS AND TEXTS")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text=" WELCOME TO GUI! ",font=("Arial Bold",15))  
l1.pack()  
  
chk\_state=BooleanVar()  
chk\_state.set(True)  
chk=Checkbutton(obj,text=" SELECT ",var=chk\_state)  
print(chk.cget("text"))  
chk.configure(text="TRUE OR FALSE")  
print(chk.cget("text"))  
print(chk.state())  
print(chk.bbox)  
chk.pack()  
  
  
obj.mainloop()

Checkbutton

from tkinter import \*  
obj = Tk()  
obj.minsize(250,250)  
Checkbutton(obj, text='male').grid(row=0, column=0)  
Checkbutton(obj, text='female').grid(row=1, column=0)  
obj.mainloop()

Checkbutton

from tkinter import \*  
  
obj=Tk()  
obj.title("BUTTON WINDOW")  
obj.geometry("750x750")  
#obj.maxsize(1000,1000)  
obj.minsize(100,100)  
  
l=Label(obj,text=" WELOCOME TO GUI ",fg="magenta",bg="black",font="Heveltica 20 bold")  
l.grid(row=0,column=3)  
  
l1=Label(obj,text=" NAME ",font="Heveltica 15 bold")  
l1.grid(row=1,column=2)  
t1=StringVar()  
t1entry=Entry(obj,textvariable=t1)  
t1entry.grid(row=1,column=3)  
  
l2=Label(obj,text=" PHONE ",font="Heveltica 15 bold")  
l2.grid(row=2,column=2)  
t2=StringVar()  
t2entry=Entry(obj,textvariable=t2)  
t2entry.grid(row=2,column=3)  
  
l3=Label(obj,text=" GENDER ",font="Heveltica 15 bold")  
l3.grid(row=3,column=2)  
t3=StringVar()  
t3entry=Entry(obj,textvariable=t3)  
t3entry.grid(row=3,column=3)  
  
l4=Label(obj,text=" EMERGENCY CONTACT ",font="Heveltica 15 bold")  
l4.grid(row=4,column=2)  
t4=StringVar()  
t4entry=Entry(obj,textvariable=t4)  
t4entry.grid(row=4,column=3)  
  
l5=Label(obj,text=" PAYMENT ",font="Heveltica 15 bold")  
l5.grid(row=5,column=2)  
t5=IntVar()  
t5entry=Entry(obj,textvariable=t5)  
t5entry.grid(row=5,column=3)  
  
s=""  
  
def submitclicked():  
 s=""  
 s+="\n NAME : "+t1.get()+"\n PHONE : "+t2.get()+"\n GENDER : "+t3.get()+"\n EMERGENCY CONTACT: "+t4.get()+"\n PAYMENT : "+str(t5.get())  
 print(s)  
 obj2 = Tk()  
 obj2.title("BUTTON WINDOW")  
 obj2.geometry("750x750")  
 # obj.maxsize(1000,1000)  
 obj2.minsize(100, 100)  
  
 l=Label(obj2,text=s,font="Helvetica 15 bold",bg="yellow",fg="red")  
 l.grid(row=3,column=3)  
  
  
 obj2.mainloop()  
  
  
b1=Button(obj,text=" SUBMIT ",font="Heveltica 13 bold",bg="grey",command=submitclicked)  
b1.grid(row=9,column=3)  
  
def order():  
 print("FULL MEALS ORDERED ")   
  
  
cb1=Checkbutton(obj,text="MEALS ORDER ",font="Heveltica 17 bold",bg="grey",command=order)  
cb1.grid(row=7,column=2)  
  
obj.mainloop()

GET , SET METHODS AND SAVE RECORD IN A FILE

from tkinter import \*  
  
obj=Tk()  
obj.title("BUTTON WINDOW")  
obj.geometry("750x750")  
#obj.maxsize(1000,1000)  
obj.minsize(100,100)  
  
l=Label(obj,text=" WELOCOME TO GUI ",fg="magenta",bg="black",font="Heveltica 20 bold")  
l.grid(row=0,column=3)  
  
l1=Label(obj,text=" NAME ",font="Heveltica 15 bold")  
l1.grid(row=1,column=2)  
t1=StringVar()  
t1entry=Entry(obj,textvariable=t1)  
t1entry.grid(row=1,column=3)  
t1.set("ENTER YOUR NAME ")  
  
l2=Label(obj,text=" PHONE ",font="Heveltica 15 bold")  
l2.grid(row=2,column=2)  
t2=StringVar()  
t2entry=Entry(obj,textvariable=t2)  
t2entry.grid(row=2,column=3)  
t2.set("ENTER YOUR PHONE ")  
  
l3=Label(obj,text=" GENDER ",font="Heveltica 15 bold")  
l3.grid(row=3,column=2)  
t3=StringVar()  
t3entry=Entry(obj,textvariable=t3)  
t3entry.grid(row=3,column=3)  
t3.set("ENTER YOUR GENDER ")  
  
l4=Label(obj,text=" EMERGENCY CONTACT ",font="Heveltica 15 bold")  
l4.grid(row=4,column=2)  
t4=StringVar()  
t4entry=Entry(obj,textvariable=t4)  
t4entry.grid(row=4,column=3)  
t4.set("ENTER YOUR CONTACT ")  
  
l5=Label(obj,text=" PAYMENT ",font="Heveltica 15 bold")  
l5.grid(row=5,column=2)  
t5=IntVar()  
t5entry=Entry(obj,textvariable=t5)  
t5entry.grid(row=5,column=3)  
t5.set(0)  
  
s=""  
f = open("abc.txt", "a")  
  
  
def submitclicked():  
 s=""  
 s+="\n NAME : "+t1.get()+"\n PHONE : "+t2.get()+"\n GENDER : "+t3.get()+"\n EMERGENCY CONTACT: "+t4.get()+"\n PAYMENT : "+str(t5.get())  
 f.write(s)  
 print(s)  
 obj2 = Tk()  
 obj2.title("BUTTON WINDOW")  
 obj2.geometry("750x750")  
 # obj.maxsize(1000,1000)  
 obj2.minsize(100, 100)  
  
  
 l=Label(obj2,text=s,font="Helvetica 15 bold",bg="yellow",fg="red")  
 l.grid(row=3,column=3)  
  
  
  
  
 obj2.mainloop()  
  
  
b1=Button(obj,text=" SUBMIT ",font="Heveltica 13 bold",bg="grey",command=submitclicked)  
b1.grid(row=9,column=3)  
  
def order():  
 print("FULL MEALS ORDERED ")  
  
cb1=Checkbutton(obj,text="MEALS ORDER ",font="Heveltica 17 bold",bg="grey",command=order)  
cb1.grid(row=7,column=2)  
  
  
  
obj.mainloop()

RADIOBUTTON WIDGET

from tkinter import \*  
from tkinter.ttk import \*  
  
obj=Tk()  
obj.title("LABELS AND TEXTS")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text=" WELCOME TO GUI! ",font=("Arial Bold",15))  
l1.grid(row=0,column=0)  
  
def python():  
 print("python")  
def java():  
 print("java")  
def clang():  
 print("clang")  
  
#VALUES MUST BE UNIQUE SO THAT ONLY RADIOBUTTON WILL BE SELECTED EVERY TIME  
  
r1=Radiobutton(obj,text="python",value=32,command=python)  
r2=Radiobutton(obj,text="java",value=24,command=java)  
r3=Radiobutton(obj,text="clang",value=14,command=clang)  
r1.grid(row=1,column=1)  
r2.grid(row=1,column=2)  
r3.grid(row=1,column=3)  
  
obj.mainloop()

SCROLLEDRTEXT WIDGET

from tkinter import \*  
from tkinter import scrolledtext  
from tkinter.ttk import \*  
  
obj=Tk()  
obj.title("LABELS AND TEXTS")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
l1=Label(obj,text=" WELCOME TO GUI! ",font=("Arial Bold",15))  
l1.grid(row=0,column=0)  
  
txt=scrolledtext.ScrolledText(obj,width=40,height=10)  
txt.insert(INSERT," HELLO ! WELCOME TO GUI....")  
txt.grid(row=1,column=1)  
  
  
obj.mainloop()

LISTBOX WIDGET

from tkinter import \*  
  
obj = Tk()  
#obj.minsize(300,300)  
  
Lb = Listbox(obj)  
Lb.insert(1, 'Python')  
Lb.insert(2, 'Java')  
Lb.insert(3, 'C++')  
Lb.insert(4, 'Any other')  
Lb.pack()  
  
obj.mainloop()

CANVAS WIDGET

from tkinter import \*  
  
obj=Tk()  
obj.title("CANVAS WIDGET")  
#obj.geometry("500x500")  
#obj.minsize(400,400)  
#obj.maxsize(400,400)  
  
canvas\_width=800  
canvas\_height=400  
  
obj.geometry(f"{canvas\_width}x{canvas\_height}")  
#obj.geometry("{}x{}".format(canvas\_width,canvas\_height))  
can\_widget=Canvas(obj,width=canvas\_width,height=canvas\_height)  
can\_widget.pack()  
  
#LINE GOES FROM X1,Y1 TO X2,Y2  
can\_widget.create\_line(0,0,800,400,fill="red")  
can\_widget.create\_line(0,400,800,0,fill="blue")  
#can\_widget.create\_line(400,400,0,0)  
  
# RECTANGLE PARAMETERS CORNERS OF TOPLEFT TO BOTTOM RIGHT  
can\_widget.create\_rectangle(3,5,700,300,fill="magenta")  
  
# CREATE TEXT  
can\_widget.create\_text(200,200,text="HELLO WORLD !")  
  
#CREATE OVAL INSIDE GIVEN DIMENSIONS OF THE RECTANGLE  
can\_widget.create\_oval(3,5,700,300,fill="black")  
  
obj.mainloop()

EVENT HANDLING

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
# BUTTON INTERACTION WITH EVENT HANDLING BINDING  
  
b1=Button(obj,text="CLICK HERE")  
b1.pack()  
  
def clickhere(event):  
 l1=Label(obj,text="MELA POITANGA..")  
 l1.pack()  
 print("HELLO MURUGAN")  
  
b1.bind("<Button-1>",clickhere)  
  
# SIMPLE BUTTON INTERACTION WITH OUT EVENT HANDLING AND BINDING  
  
def puchuk():  
 l1=Label(obj,text="YAARIKITA PESRE..")  
 l1.pack()  
 print("HELLO PUSHPA")  
  
b2=Button(obj,text="PUCHUK HERE",command=puchuk)  
b2.pack()  
  
  
  
obj.mainloop()

EVENT HANDLING , MOUSE LEFT,MIDDLE,RIGHT

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
  
def leftclick(event):  
 l1=Label(obj,text="LEFT SIDE MELA POITANGA..")  
 l1.pack()  
 print("HELLO MURUGAN")  
def middleclick(event):  
 l1=Label(obj,text="MIDDLE LA MELA POITANGA..")  
 l1.pack()  
 print("HELLO MURUGAN")  
def rightclick(event):  
 l1=Label(obj,text=" RIGHT SIDE MELA POITANGA..")  
 l1.pack()  
 print("HELLO MURUGAN")  
  
obj.bind("<Button-1>",leftclick)  
obj.bind("<Button-2>",middleclick)  
obj.bind("<Button-3>",rightclick)  
  
obj.mainloop()

<Button-1>

from tkinter import \*  
  
obj=Tk()  
obj.title("EVENT HANDLING")  
obj.geometry("750x750")  
obj.minsize(500,500)  
#obj.maxsize(1000,1000)  
  
# BUTTON INTERACTION WITH EVENT HANDLING BINDING  
  
b1=Button(obj,text="CLICK HERE")  
b1.pack()  
  
def clickhere(event):  
 l1=Label(obj,text="MELA POITANGA..")  
 l1.pack()  
 print("HELLO MURUGAN")  
  
b1.bind("<Button-1>",clickhere)  
obj.bind("<Button-1>",clickhere)  
  
obj.mainloop()

SIMPLE CALCULATOR GUI USING EVAL()

from tkinter import \*  
  
obj=Tk()  
obj.title("SIMPLE CALCULATOR")  
#obj.geometry("750x750")  
obj.minsize(400,400)  
obj.maxsize(1000,1000)  
  
t=StringVar()  
e=Entry(obj,textvariable=t,width=20,font="Arial 15 bold")  
e.grid(row=0)  
  
def clear():  
 t.set("")  
  
clear=Button(obj,text="clear",command=clear,font="Arial 15 bold",width=20)  
clear.grid(row=1,column=0)  
  
  
def div():  
 t.set(t.get()+"/")  
  
div=Button(obj,text="/",command=div,font=",Arial 15 bold",width=20)  
div.grid(row=1,column=1)  
  
  
def equal():  
 t.set(eval(t.get()))  
  
eq=Button(obj,text="=",command=equal,font=",Arial 15 bold",width=20)  
eq.grid(row=1,column=2)  
  
def mod():  
 t.set(t.get()+"%")  
  
mo=Button(obj,text="%",command=mod,font=",Arial 15 bold",width=20)  
mo.grid(row=1,column=3)  
  
def seven():  
 t.set(t.get()+"7")  
  
sev=Button(obj,text="7",command=seven,font="Arial 15 bold",width=20)  
sev.grid(row=2,column=0)  
  
def eight():  
 t.set(t.get()+"8")  
  
eig=Button(obj,text="8",command=eight,font="Arial 15 bold",width=20)  
eig.grid(row=2,column=1)  
  
def nine():  
 t.set(t.get()+"9")  
  
nin=Button(obj,text="9",command=nine,font="Arial 15 bold",width=20)  
nin.grid(row=2,column=2)  
  
  
def mul():  
 t.set(t.get()+"\*")  
  
mul=Button(obj,text="\*",command=mul,font="Arial 15 bold",width=20)  
mul.grid(row=2,column=3)  
  
def four():  
 t.set(t.get()+"4")  
  
fou=Button(obj,text="4",command=four,font="Arial 15 bold",width=20)  
fou.grid(row=3,column=0)  
  
def five():  
 t.set(t.get()+"5")  
  
fiv=Button(obj,text="5",command=five,font="Arial 15 bold",width=20)  
fiv.grid(row=3,column=1)  
  
def six():  
 t.set(t.get()+"6")  
  
si=Button(obj,text="6",command=six,font="Arial 15 bold",width=20)  
si.grid(row=3,column=2)  
  
def sub():  
 t.set(t.get()+"-")  
  
su=Button(obj,text="-",command=sub,font="Arial 15 bold",width=20)  
su.grid(row=3,column=3)  
  
def one():  
 t.set(t.get()+"1")  
  
on=Button(obj,text="1",command=one,font="Arial 15 bold",width=20)  
on.grid(row=4,column=0)  
  
def two():  
 t.set(t.get()+"2")  
  
tw=Button(obj,text="2",command=two,font="Arial 15 bold",width=20)  
tw.grid(row=4,column=1)  
  
def three():  
 t.set(t.get()+"3")  
  
thr=Button(obj,text="3",command=three,font="Arial 15 bold",width=20)  
thr.grid(row=4,column=2)  
  
def add():  
 t.set(t.get()+"+")  
  
ad=Button(obj,text="+",command=add,font="Arial 15 bold",width=20)  
ad.grid(row=4,column=3)  
  
obj.mainloop()

DATA STRUCTURES

Stack using lists

stack=[]  
  
stack.append(10)  
stack.append(20)  
stack.append(30)  
stack.append(40)  
stack.append(50)  
  
print(stack)  
  
print("TOP IS : ",stack[-1])  
  
stack.pop()  
stack.pop()  
stack.pop()  
stack.pop()  
stack.pop()  
  
print("IS EMPTY : ",len(stack)==0)

stack using functions

stack=[]  
  
def pushit():  
 element=input(" enter element: ")  
 stack.append(element)  
 print(stack)  
  
def popit():  
 e=stack.pop()  
 print(e,"popped")  
 print(stack)  
  
def isempty():  
 if len(stack)==0:  
 print(" true , stack is empty")  
 else:  
 print(" false , stack is not empty")  
  
def peek():  
 print(stack[-1])  
  
pushit()  
pushit()  
pushit()  
pushit()  
pushit()  
  
popit()  
popit()  
popit()  
  
isempty()  
peek()

stacks using modules

# USING COLLECTIONS MODULE  
  
from collections import deque  
  
stack =deque()  
  
stack.append(10)  
stack.append(20)  
stack.append(30)  
stack.append(40)  
stack.append(50)  
  
print(stack)  
  
stack.pop()  
stack.pop()  
stack.pop()  
  
print("IS EMPTY ",not stack)  
print("IS EMPTY ",len(stack)==0)  
  
print("PEEK ELEMENT IS ",stack[-1])  
  
stack.pop()  
stack.pop()  
  
print("IS EMPTY ",not stack)  
print("IS EMPTY ",len(stack)==0)  
  
# USING QUEUE MODULE  
  
from queue import LifoQueue  
  
stack=LifoQueue()  
  
stack.put(10)  
stack.put(20)  
stack.put(30)  
stack.put(40)  
stack.put(50)  
  
print(stack.get())  
print(stack.get())  
print(stack.get())  
  
# we can set the size of the stack(lifoqueue)  
stack=LifoQueue(4)  
  
stack.put(10)  
stack.put(20)  
stack.put(30)  
stack.put(40)  
# STACK OVERFLOW THROWS ERROR AFTER 5 SECONDS  
stack.put(50,timeout=5)  
  
  
print(stack.get())  
print(stack.get())  
print(stack.get())  
print(stack.get())  
# STACK UNDEFLOW THROWS ERROR AFTER 5 SECONDS  
print(stack.get(),timeout=5)

QUEUE USING LISTS

# LISTS AS QUEUE  
  
queue=[]  
  
queue.append(10)  
queue.append(20)  
queue.append(30)  
queue.append(40)  
queue.append(50)  
print(queue)  
print("POPPED-",queue.pop(0))  
print("POPPED-",queue.pop(0))  
print("POPPED-",queue.pop(0))  
  
print("IS EMPTY : ",not queue)  
print(queue)  
  
print("POPPED-",queue.pop(0))  
print("POPPED-",queue.pop(0))  
  
print("IS EMPTY : ",not queue)  
print(queue)

QUEUE USING FUNCTIONS

queue=[]  
  
def enqueue():  
 element=input(" enter element: ")  
 queue.append(element)  
 print(queue)  
  
def dequeue():  
 e=queue.pop(0)  
 print(e,"popped")  
 print(queue)  
  
def isempty():  
 if len(queue)==0:  
 print(" true , queue is empty")  
 else:  
 print(" false , queue is not empty")  
  
def peek():  
 print(queue[-1])  
  
enqueue()  
enqueue()  
enqueue()  
enqueue()  
enqueue()  
  
dequeue()  
dequeue()  
dequeue()  
  
isempty()  
peek()

QUEUE USING MODULES

# USING COLLECTIONS MODULE  
  
from collections import deque  
# WAY 1  
queue =deque()  
  
queue.append(10)  
queue.append(20)  
queue.append(30)  
queue.append(40)  
queue.append(50)  
  
print(queue)  
  
print(queue.popleft(),"popped")  
print(queue.popleft(),"popped")  
print(queue.popleft(),"popped")  
  
print("IS EMPTY ",not queue)  
print("IS EMPTY ",len(queue)==0)  
  
print("PEEK ELEMENT IS ",queue[-1])  
  
print(queue.popleft(),"popped")  
print(queue.popleft(),"popped")  
  
print("IS EMPTY ",not queue)  
print("IS EMPTY ",len(queue)==0)  
  
# WAY 2  
queue =deque()  
  
queue.appendleft(10)  
queue.appendleft(20)  
queue.appendleft(30)  
queue.appendleft(40)  
queue.appendleft(50)  
  
print(queue)  
  
print(queue.pop(),"popped")  
print(queue.pop(),"popped")  
print(queue.pop(),"popped")  
  
print("IS EMPTY ",not queue)  
print("IS EMPTY ",len(queue)==0)  
  
print("PEEK ELEMENT IS ",queue[-1])  
  
print(queue.pop(),"popped")  
print(queue.pop(),"popped")  
  
print("IS EMPTY ",not queue)  
print("IS EMPTY ",len(queue)==0)  
  
from queue import Queue  
q=Queue()  
  
q.put(10)  
q.put(20)  
q.put(30)  
q.put(40)  
q.put(50)  
  
print(q.get(),"POPPED")  
print(q.get(),"POPPED")  
print(q.get(),"POPPED")  
print(q.get(),"POPPED")  
print(q.get(),"POPPED")

PRIORITY QUEUE USING LIST

l=[]  
  
l.append(30)  
l.append(20)  
l.append(40)  
l.append(50)  
l.append(10)  
  
print(l)  
  
l.sort()  
  
print(l)  
  
print(l.pop(0))  
print(l.pop(0))  
print(l.pop(0))  
print(l.pop(0))  
  
print(l)

PRIORITY QUEUE WITH TUPLES AS ELEMENTS USING LISTS

q=[]  
q.append((1,"HELLO"))  
q.append((3,"HELP"))  
q.append((2,"HIII"))  
q.append((4,"HELPING"))  
q.append((5,"HELL"))  
q.append((6,"HEYY"))  
  
print(q)  
q.sort()  
print(q)  
q.sort(reverse=True)  
print(q)  
  
print(q.pop(0),"POPPED")  
print(q.pop(0),"POPPED")  
print(q.pop(0),"POPPED")  
print(q.pop(0),"POPPED")  
print(q)

PRIORITY QUEUE USING MODULES

from queue import PriorityQueue  
  
q = PriorityQueue()  
q.put(50)  
q.put(10)  
q.put(20)  
q.put(30)  
q.put(40)  
q.put(40)  
  
print(q.get()," REMOVED ")  
print(q.get()," REMOVED ")  
print(q.get()," REMOVED ")  
print(q.get()," REMOVED ")  
print(q.get()," REMOVED ")

SINGLY LINKED LIST

class node:  
 def \_\_init\_\_(self,data):  
 self.data=data  
 self.link=None  
  
  
class linkedlist:  
 def \_\_init\_\_(self):  
 self.head=None  
  
 def addbegin(self,data):  
 newnode = node(data)  
 newnode.link = self.head  
 self.head = newnode  
  
 def addend(self,data):  
 newnode = node(data)  
 if self.head is None:  
 self.head=newnode  
 else:  
 temp = self.head  
 while temp.link is not None:  
 temp = temp.link  
 temp.link = newnode  
  
 def addafterx(self,data,x):  
 newnode = node(data)  
 if self.head is None:  
 print(" LINKED LIST IS EMPTY ")  
 else:  
 temp=self.head  
 while(temp is not None):  
 if temp.data==x:  
 newnode.link=temp.link  
 temp.link=newnode  
 break  
 else:  
 temp=temp.link  
 if temp is None:  
 print(x," is not in the linked list,( end of the list) ")  
  
  
 def addbeforex(self,data,x):  
 newnode = node(data)  
 if self.head is None:  
 print(" LINKED LIST IS EMPTY ")  
 else:  
 temp=self.head  
 while(temp.link is not None):  
 if temp.link.data==x:  
 newnode.link=temp.link  
 temp.link=newnode  
 break  
 else:  
 temp=temp.link  
 if temp.link is None:  
 print(x," is not in the linked list,( end of the list) ")  
  
 def insert\_empty(self,data):  
 newnode = node(data)  
 if self.head is None:  
 self.head=newnode  
 else:  
 print("LINKED LIST IS NOT EMPTY ")  
  
  
 def display(self):  
 if self.head==None:  
 print("LINKED LIST IS EMPTY")  
 else:  
 temp=self.head  
 while temp is not None:  
 print(temp.data,end=" --> ")  
 temp=temp.link  
 print("None")  
  
 def count(self):  
 if self.head is None:  
 print(" LINKED LIST IS EMPTY ")  
 else:  
 n=0  
 temp=self.head  
 while(temp is not None):  
 n+=1  
 temp=temp.link  
 print("\n",n," NODES IN THE LIST \n")  
  
 def isempty(self):  
 if self.head is None:  
 return True  
 else:  
 return False  
  
 def delete\_first(self):  
 if self.head is None:  
 print("NOT POSSIBLE TO DELETE FROM EMPTY LIST")  
 else:  
 self.head=self.head.link  
  
 def delete\_last(self):  
 if self.head is None:  
 print("NOT POSSIBLE TO DELETE FROM EMPTY LIST")  
 else:  
 temp=self.head  
  
 while temp.link.link is not None:  
 temp = temp.link  
 temp.link = None  
  
 def deletex(self,x):  
 if self.head is None:  
 print("NOT POSSIBLE TO DELETE FROM EMPTY LIST")  
 else:  
 if self.head.data == x:  
 self.head = self.head.link  
 else:  
 temp = self.head  
 while temp.link is not None:  
 if temp.link.data is x:  
 temp.link = temp.link.link  
 break  
 else:  
 temp = temp.link  
 if temp.link is None:  
 print(x," IS NOT IN THE LIST ")  
  
  
  
  
ll = linkedlist()  
print("LINKED LIST IS EMPTY : ",ll.isempty(),"\n")  
ll.display()  
  
ll.insert\_empty(100)  
ll.insert\_empty(1)  
ll.display()  
  
ll.addbegin(40)  
ll.addbegin(30)  
ll.addbegin(20)  
ll.addbegin(10)  
ll.display()  
  
ll.addend(50)  
ll.addend(60)  
ll.addend(70)  
ll.display()  
  
ll.addafterx(35,300)  
ll.addafterx(35,30)  
ll.display()  
  
ll.addbeforex(15,400)  
ll.addbeforex(15,20)  
ll.display()  
  
ll.count()  
  
ll.delete\_first()  
ll.display()  
ll.count()  
  
ll.delete\_last()  
ll.display()  
ll.count()  
  
ll.deletex(15)  
ll.display()  
ll.count()  
  
ll.deletex(35)  
ll.display()  
ll.count()  
  
ll.deletex(60)  
ll.display()  
ll.count()  
  
ll.deletex(11)  
ll.display()  
ll.count()  
  
print(" LINKED LIST IS EMPTY : ",ll.isempty())

DOUBLY LINKED LIST

class node:  
 def \_\_init\_\_(self,data):  
 self.prev=None  
 self.data=data  
 self.next=None  
  
class doublyll:  
 def \_\_init\_\_(self):  
 self.head=None  
  
 def addtoempty(self,data):  
 if self.head is None:  
 newnode=node(data)  
 self.head=newnode  
 else:  
 print("LINKED LIST IS NOT EMPTY")  
  
 def addbegin(self,data):  
 newnode = node(data)  
 if self.head is None:  
 newnode.next=None  
 self.head=newnode  
 else:  
 temp = self.head  
 temp.prev = newnode  
 newnode.next = temp  
 self.head = newnode  
 def addend(self,data):  
 newnode = node(data)  
 if self.head is None:  
 newnode.next = None  
 self.head = newnode  
 else:  
 temp = self.head  
 while temp.next is not None:  
 temp=temp.next  
 newnode.prev=temp  
 temp.next = newnode  
 newnode.next = None  
  
 def addafterx(self,data,x):  
 if self.head is None:  
 print("INSERTION NOT POSSIBLE ( LINKED LIST IS EMPTY)")  
 else:  
 done = "NO"  
 newnode=node(data)  
 temp=self.head  
 while temp.next is not None:  
 if temp.data is x:  
 nextnode=temp.next  
 newnode.next=temp.next  
 nextnode.prev=newnode  
 newnode.prev=temp  
 temp.next=newnode  
 done = "YES"  
 break  
 else:  
 temp=temp.next  
 if done == "NO":  
 print(x, " NOT FOUND ( END OF THE LIST )")  
  
  
  
 def addbeforex(self, data,x):  
 if self.head is None:  
 print("INSERTION NOT POSSIBLE ( LINKED LIST IS EMPTY)")  
 else:  
 done="NO"  
 newnode=node(data)  
 temp=self.head  
 if temp.data is x:  
 self.addbegin(data)  
 else:  
 newnode = node(data)  
 temp = self.head  
 while temp.next.next is not None:  
 if temp.next.data is x:  
 nextnode = temp.next  
 newnode.next = temp.next  
 nextnode.prev = newnode  
 newnode.prev = temp  
 temp.next = newnode  
 done="YES"  
 break  
 else:  
 temp = temp.next  
 if done == "NO":  
 print(x, " NOT FOUND ( END OF THE LIST )")  
  
 def delete\_first(self):  
 if self.head is None:  
 print("NOT POSSIBLE TO DELETE FIRST NODE, LIST IS EMPTY")  
 else:  
 temp=self.head  
 temp2=temp.next  
 temp2.prev=None  
 self.head=temp2  
  
 def delete\_last(self):  
 if self.head is None:  
 print("NOT POSSIBLE TO DELETE LAST NODE, LIST IS EMPTY")  
 else:  
 temp=self.head  
 while temp.next.next is not None:  
 temp = temp.next  
 temp.next=None  
  
 def deletex(self,x):  
 if self.head is None:  
 print("NOT POSSIBLE TO DELETE ",x," data (NODE) LIST IS EMPTY")  
 else:  
 temp = self.head  
 if temp.data is x:  
 self.delete\_first()  
 else:  
 found = False  
 while temp.next.next is not None:  
 if temp.next.data is x:  
 temp2 = temp.next.next  
 temp2.prev = temp  
 temp.next = temp2  
 found = True  
 break  
 else:  
 temp = temp.next  
 if temp.next.next is None or found is False:  
 if temp.next.data is x:  
 found = True  
 temp.next = None  
 else:  
 print(" END OF THE LIST, ", x, " NOT FOUND :(")  
  
  
  
  
 def display(self):  
 if self.head == None:  
 print("LINKED LIST IS EMPTY")  
 else:  
 print("\nNone <--", end=" ")  
 temp = self.head  
 while temp is not None:  
 print(temp.data, end=" <--> ")  
 temp = temp.next  
 print("None")  
  
 def reverse\_display(self):  
 if self.head == None:  
 print("LINKED LIST IS EMPTY")  
 else:  
 print("\nNone <--",end=" ")  
 temp = self.head  
 while temp.next is not None:  
 temp = temp.next  
 while temp is not None:  
 print(temp.data, end=" <--> ")  
 temp=temp.prev  
 print("None")  
  
 def isempty(self):  
 if self.head is None:  
 print("True, DOUBLY LINKED LIST IS EMPTY ")  
 else:  
 print("False, DOUBLY LINKED LIST IS EMPTY ")  
  
 def count(self):  
 temp=self.head  
 n=0  
 l=[]  
 while temp is not None:  
 n+=1  
 l.append(temp.data)  
 temp=temp.next  
 print(n," NODES IN THE LIST ")  
 print(l)  
  
  
dll=doublyll()  
dll.isempty()  
dll.count()  
  
print("AFTER ADDING 11 to empty LIST")  
dll.addtoempty(11)  
dll.reverse\_display()  
dll.display()  
  
print("AFTER ADDING 10,5,1 BEGINING OF THE LIST")  
dll.addbegin(10)  
dll.addbegin(5)  
dll.addbegin(1)  
dll.display()  
dll.reverse\_display()  
  
print("AFTER ADDING 20,30,40,50 ENDING OF THE LIST")  
dll.addend(20)  
dll.addend(30)  
dll.addend(40)  
dll.addend(50)  
dll.display()  
dll.reverse\_display()  
  
print("AFTER ADDING 25,35 after 20,30 NODES OF THE LIST")  
dll.addafterx(25,20)  
dll.addafterx(35,30)  
dll.display()  
dll.reverse\_display()  
  
print("AFTER ADDING 3,7 before 5,10 NODES OF THE LIST")  
dll.addbeforex(3,5)  
dll.addbeforex(7,10)  
dll.display()  
  
print(" LINKED LIST IN REVERSE DIRECTION ")  
dll.reverse\_display()  
  
dll.count()  
  
dll.display()  
dll.delete\_first()  
print(" AFTER DELETING FIRST NODE ")  
dll.display()  
print(" LINKED LIST IN REVERSE DIRECTION ")  
dll.reverse\_display()  
dll.count()  
  
dll.display()  
dll.delete\_last()  
print(" AFTER DELETING LAST NODE ")  
dll.display()  
print(" LINKED LIST IN REVERSE DIRECTION ")  
dll.reverse\_display()  
dll.count()  
  
dll.display()  
dll.deletex(35)  
print(" AFTER DELETING 35 (data) NODE ")  
dll.display()  
print(" LINKED LIST IN REVERSE DIRECTION ")  
dll.reverse\_display()  
dll.count()  
  
print(" AFTER DELETING 40 (data) NODE ")  
dll.deletex(40)  
dll.display()  
print(" LINKED LIST IN REVERSE DIRECTION ")  
dll.reverse\_display()  
dll.count()  
  
print(" TRYING TO DELETE 3 (data) NODE ")  
dll.deletex(3)  
dll.display()  
print(" LINKED LIST IN REVERSE DIRECTION ")  
dll.reverse\_display()  
dll.count()  
  
print(" TRYING TO DELETE 50 (data) NODE FROM LIST ")  
dll.deletex(50)  
dll.display()  
print(" LINKED LIST IN REVERSE DIRECTION ")  
dll.reverse\_display()  
dll.count()

CIRCULAR LINKED LIST

# Represents the node of list.  
class Node:  
 def \_\_init\_\_(self, data):  
 self.data = data;  
 self.next = None;  
  
  
class CreateList:  
 # Declaring head and tail pointer as null.  
 def \_\_init\_\_(self):  
 self.head = Node(None);  
 self.tail = Node(None);  
 self.head.next = self.tail;  
 self.tail.next = self.head;  
  
 # This function will add the new node at the end of the list.  
  
 def add(self, data):  
 newNode = Node(data);  
 # Checks if the list is empty.  
 if self.head.data is None:  
 # If list is empty, both head and tail would point to new node.  
 self.head = newNode;  
 self.tail = newNode;  
 newNode.next = self.head;  
 else:  
 # tail will point to new node.  
 self.tail.next = newNode;  
 # New node will become new tail.  
 self.tail = newNode;  
 # Since, it is circular linked list tail will point to head.  
 self.tail.next = self.head;  
  
 # Displays all the nodes in the list  
  
 def display(self):  
 current = self.head;  
 if self.head is None:  
 print("List is empty");  
 return;  
 else:  
 print("Nodes of the circular linked list: ");  
 # Prints each node by incrementing pointer.  
 print(current.data),  
 while (current.next != self.head):  
 current = current.next;  
 print(current.data),  
  
  
class CircularLinkedList:  
 cl = CreateList();  
 # Adds data to the list  
 cl.add(1);  
 cl.add(2);  
 cl.add(3);  
 cl.add(4);  
 # Displays all the nodes present in the list  
 cl.display();

IMPLEMENTATION OF BINARY SEARCH TREE ( SAMPLE PROGRAM)

class BST:  
 def \_\_init\_\_(self,key):  
 self.key=key  
 self.lchild=None  
 self.rchild=None  
  
root=BST(10)  
root.lchild=BST(5)  
root.rchild=BST(20)  
  
print(root.key)  
print(root.lchild)  
print(root.rchild)  
  
  
print(root.lchild.key)  
print(root.lchild.lchild)  
print(root.lchild.rchild)  
  
print(root.rchild.key)  
print(root.rchild.lchild)  
print(root.rchild.rchild)

IMPLEMENTATION OF BINARY SEARCH TREE

INSERTION , TRAVERSAL , SEARCH

class BST:  
 def \_\_init\_\_(self,data):  
 self.key=data  
 self.lchild=None  
 self.rchild=None  
  
 def insert(self,data):  
 if self.key is None:  
 self.key=data  
 return  
  
 elif self.key>=data:  
 if self.lchild:  
 self.lchild.insert(data)  
 else:  
 self.lchild=BST(data)  
  
 elif self.key<data:  
 if self.rchild:  
 self.rchild.insert(data)  
 else:  
 self.rchild=BST(data)  
  
 def preorder\_traversal(self):  
 print(self.key, end=" ")  
 if self.lchild:  
 self.lchild.preorder\_traversal()  
 if self.rchild:  
 self.rchild.preorder\_traversal()  
  
 def inorder\_traversal(self):  
 if self.lchild:  
 self.lchild.inorder\_traversal()  
 print(self.key,end=" ")  
 if self.rchild:  
 self.rchild.inorder\_traversal()  
  
 def postorder\_traversal(self):  
 if self.lchild:  
 self.lchild.postorder\_traversal()  
 if self.rchild:  
 self.rchild.postorder\_traversal()  
 print(self.key, end=" ")  
  
  
 def search(self,data):  
 found = False  
 if self.key is not None:  
 if self.key is data:  
 found=True  
 print(data," FOUND IN BST")  
 elif self.key>data:  
 if self.lchild:  
 self.lchild.search(data)  
 else:  
 print(data," NOT FOUND IN BST")  
  
 elif self.key<data:  
 if self.rchild:  
 self.rchild.search(data)  
 else:  
 print(data," NOT FOUND IN BST")  
 else:  
 print(data, " NOT FOUND IN BST")  
  
 def delete(self,data):  
 if self.key is None:  
 print(" BST IS EMPTY ")  
 return  
 if self.key is data:  
 if self.lchild is None:  
  
  
 elif self.key>data:  
 if self.lchild:  
 self.lchild=self.lchild.delete(data)  
 else:  
 print(data," NOT PRESENT IN THE TREE ")  
  
 elif self.key<data:  
 if self.rchild:  
 self.rchild=self.rchild.delete(data)  
 else:  
 print(data," NOT PRESENT IN THE TREE ")  
  
  
  
  
#creates an empty tree key,lchild,rchild as None  
root=BST(None)  
  
# inserting into empty tree, 10 will be very first node places as root  
# printing key,lchild,rchild of tree with single node(as root)  
root.insert(10)  
print(root.key)  
print(root.lchild)  
print(root.rchild)  
  
# list of values to create a binary search tree ,(inserting one by one)  
# duplicates allowed , will be placed in the left side of the tree  
  
l = [6,3,1,98,3,7]  
for i in l:  
 root.insert(i)  
  
# BST TRAVERSALS (PRE , IN , POST ORDER TRAVERSALS)  
root.preorder\_traversal()  
print("")  
root.inorder\_traversal()  
print("")  
root.postorder\_traversal()  
print("")  
  
# searching data in bst  
root.search(3)  
root.search(6)  
root.search(25)  
root.search(15)

S.\_\_CONTAINS\_\_(SS)

SSmy\_str **=** "Hello from AskPython"

target **=** "askpython"

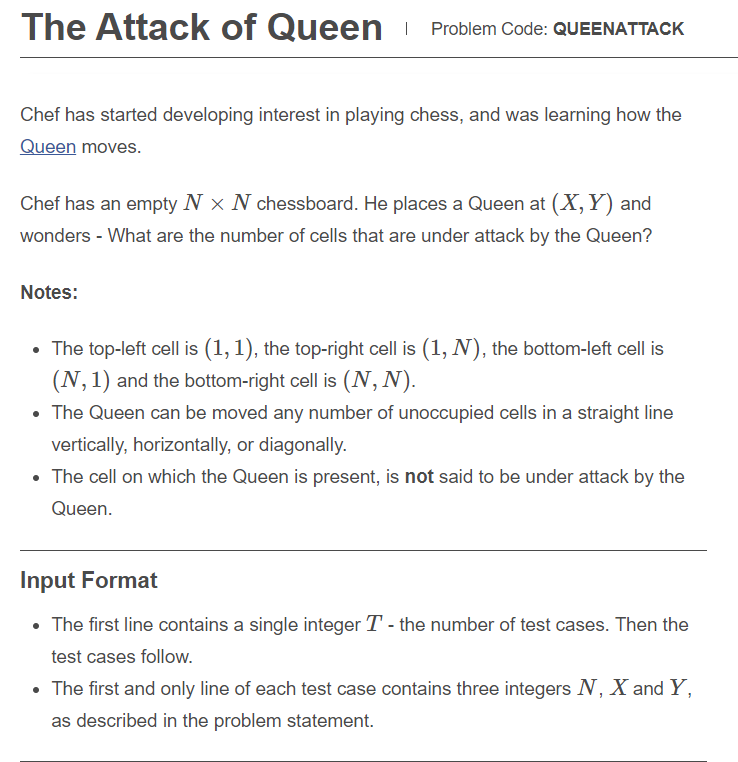
**if** (my\_str.\_\_contains\_\_(target)):

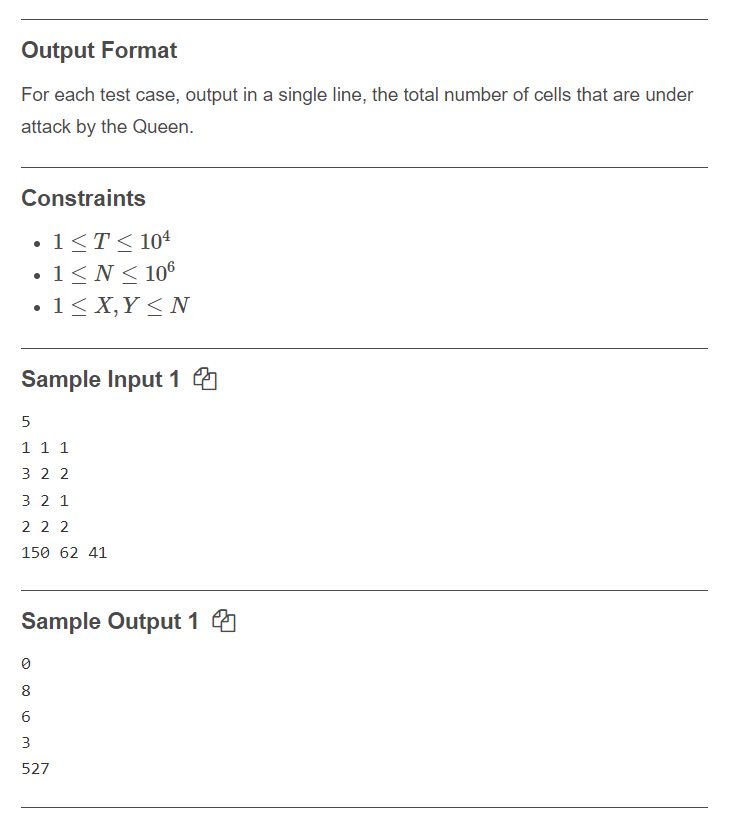
    print("String contains target!")

**else**:

    print("String does not contain target")

QUEEN ATTACK





t=int(input())

while(t):

n,x,y=map(int,input().split())

def canQueenAttack(qR, qC, oR, oC):

if qR == oR:

return (oR,oC)

if qC == oC:

return (oR,oC)

if abs(qR - oR) == abs(qC - oC):

return (oR,oC)

return (qR,qC)

if(n==1):

print(0)

else:

s=set()

for i in range(1,n+1):

for j in range(1,n+1):

s.add(canQueenAttack(x,y,i,j))

s.remove((x,y))

print(len(s))

t=t-1

……………………………………………………………………………….……………………………………………………………………………….…………………………………………………

ASSESMENT TEST

i=2  
while True:  
 if i%3==0:  
 break  
 print(i)  
 i+=2  
  
  
import re  
  
m=re.sub('morning','evening','good morning')  
print(m)  
  
  
s='welcome home'  
m=re.match(r'(.\*)(.\*?)',s)  
print(m.group())  
  
print('abcdefcdghcd'.split('cd',2))  
  
  
a=[1,2,3,4,5]  
b=[6,7,8,9]  
a.append(b)  
print(a)  
a=[1,2,3,4,5]  
b=[6,7,8,9]  
a.extend(b)  
print(a)  
  
print("Hello World"[::-1])  
  
person={"name":"jhon doe","age":83}  
  
d={}  
print(type(d))  
  
print(type(1/2))  
  
for i in range(10):  
 if i==5:  
 break  
 else:  
 print(i)  
else:  
 print("Here")  
  
  
def p(i,j):  
 if(i==0):  
 return j  
 else:  
 return p(i-1,i+j)  
  
print(p(4,7))  
  
import random  
random.randrange(3)  
  
  
def a(n):  
 if n==0:  
 return 0  
 elif n==1:  
 return 1  
 else:  
 return a(n-1)+a(n-2)  
  
for i in range(0,4):  
 print(a(i),end=" ")  
  
for i in '':  
 print(i)  
  
def foo(k):  
 k=[1]  
q=[0]  
foo(q)  
print(q)  
  
  
import math  
num=int(input("Enter a number whose factorial yoy want to find"))  
print(math.factorial(num))  
  
t=tuple([1,2,3,4,5,6,7])  
print(t)  
print(t)  
  
  
#symbol @ Decorators. The main use case of the symbol @ in Python are decorators. In Python, a decorator extends the functionality of an existing function or class.12

……………………………………………………………………………….……………………………………………………………………………….…………………………………………………

PYTHON-LEARNING

* Python can be treated in a procedural way, an object-oriented way or a functional way.
* Comments start with a #, and Python will render the rest of the line as a comment:
* Python does not really have a syntax for multi line comments.
* To add a multiline comment you could insert a # for each line
* For multiline commnents u can use 🡪 “”” “””
* """  
  This is a comment  
  written in  
  more than just one line  
  """
* Python has no command for declaring a variable.
* CASTING:
* x = str(3)    # x will be '3'  
  y = int(3)    # y will be 3  
  z = float(3)  # z will be 3.0
* String variables can be declared either by using single or double quotes
* x = 5  
  y = "John"  
  print(type(x))  
  print(type(y))
* Variable names are case-sensitive.

## **Variable Names**

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and )
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* Legal variable names:
* myvar = "John"  
  my\_var = "John"  
  \_my\_var = "John"  
  myVar = "John"  
  MYVAR = "John"  
  myvar2 = "John"
* Illegal variable names:
* 2myvar = "John"  
  my-var = "John"  
  my var = "John"

## **Many Values to Multiple Variables**

Python allows you to assign values to multiple variables in one line

x, y, z = "Orange", "Banana", "Cherry"

## **One Value to Multiple Variables**

And you can assign the same value to multiple variables in one line:

x = y = z = "Orange"

## **Unpack a Collection**

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

Unpack a list:

fruits = ["apple", "banana", "cherry"]  
x, y, z = fruits

The best way to output multiple variables in the print() function is to separate them with commas, which even support different data types:

x = 5  
y = "John"  
print(x, y)

\*\*\*

## **The global Keyword**

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

def myfunc():  
 global x  
 x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

## **Built-in Data Types**

|  |  |
| --- | --- |
| Text Type: | str |
| Numeric Types: | int, float,  complex  x = 1j+3 |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

You can get the data type of any object by using the type() function

RANGE IS ONE OF THE SEQUENCE DATA TYPE

x = range(6)  
  
#display x:  
print(x)  
  
#display the data type of x:  
print(type(x))

frozen set

x = frozenset({"apple", "banana", "cherry"})  
  
#display x:  
print(x)  
  
#display the data type of x:  
print(type(x))

## **Int**

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

## **Float**

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

## **Type Conversion**

You can convert from one type to another with the int(), float(), and complex() methods

## **Random Number**

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

Import the random module, and display a random number between 1 and 9:

import random  
  
print(random.randrange(1, 10))

# **Python Casting**

## **Specify a Variable Type**

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

* int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
* float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
* str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

## **Multiline Strings**

You can assign a multiline string to a variable by using three quotes

a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)  
  
a = '''Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.'''  
print(a)

## **Strings are Arrays**

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

Get the character at position 1 (remember that the first character has the position 0):

a = "Hello, World!"  
print(a[1])

## **Looping Through a String**

Since strings are arrays, we can loop through the characters in a string, with a for loop.

for x in "banana":  
 print(x)  
  
  
for x in "banana":  
 print(x,end="")

## **String Length**

To get the length of a string, use the len() function.

## **Check String**

To check if a certain phrase or character is present in a string, we can use the keyword in

a="Sastra Deemed University"  
print(len(a))  
  
print("substring found : ","Sastra" in a)

Use it in an if statement:

txt = "The best things in life are free!"  
if "life" in txt:  
 print("Yes, 'free' is present.")

not in

txt = "The best things in life are free!"  
print("expensive" not in txt)  
  
txt = "The best things in life are free!"  
if "expensive" not in txt:  
 print("No, 'expensive' is NOT present.")

# **Python - Slicing Strings**

## **Slicing**

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

**Note:**The first character has index 0.

## **Slice From the Start**

## **Slice To the End**

By leaving out the end index, the range will go to the end:

s="surya mahesh"  
  
#print string as it is  
print(s)  
  
#first character in string  
print(s[1])  
  
#slicing starts from index 2 and index 5 not included  
print(s[2:5])  
  
#starts from 2 and upto end  
print(s[2:])  
  
#starts from 0 index ( since not mentioned) and upto index 9 (10 not included)  
print(s[:10])  
  
#total string  
print(s[:])  
  
#starts from 0 and upto end step value 1 ( usual string)  
print(s[::1])  
  
#starts from 0 and upto end step value 2  
print(s[::2])  
  
  
# slicing negative index starts from end with step value 1  
print(s[::-1])  
  
# slicing negative index starts from end with step value 2  
print(s[::-2])  
  
# slicing negative index starts from 5 th character fom last and ends at second character from last   
print(s[-5:-2])

## **Negative Indexing**

Use negative indexes to start the slice from the end of the string:

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

b = "Hello, World!"  
print(b[-5:-2])

# **Python - Modify Strings**

s="sUrYa MaHeSh"  
  
#lowercase  
print(s.upper())  
  
#uppercase  
print(s.lower())  
  
#capitalize  
print(s.capitalize())  
  
#Whitespace is the space before and/or after the actual text, and very often you want to remove this space.  
a=" hello world "  
print(a.strip())  
  
#replaces a word of string with others  
print(a.replace("world","surya"))  
  
#The split() method splits the string into substrings if it finds instances of the separator  
a="surya,mahesh,pemma,brahmi,vikky"  
print(a.split(","))  
  
a="surya mahesh pemma brahmi vikky"  
print(a.split(" "))

concatenation of strings

a="hello"  
b="world"  
print(a,b)  
print(a+b)  
print(a+" "+b)

STRING FORMATTING

s="myself Surya age:{}"  
age=20  
print(s.format(age))  
  
s="hello! surya here class: {} age: {} weight: {} height: {}"  
age=20  
standard=15  
weight=55  
height=5.5  
print(s.format(age,standard,weight,height))  
  
quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want {} pieces of item {} for {} dollars."  
print(myorder.format(quantity, itemno, price))  
  
  
quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."  
print(myorder.format(quantity, itemno, price))

surname="Kolisetty"  
print(f"Mr.{surname}.Surya Mahesh")  
  
age=10  
height=6  
print("Hello, This is {0}.Surya Mahesh age:{1} weight:{2}".format(surname,age,height))  
  
  
friend\_name="pemma"  
greeting="how are you {name}"  
mygreeting=greeting.format(name=friend\_name)  
print(mygreeting)

txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)  
txt2 = "My name is {0}, I'm {1}".format("John",36)  
txt3 = "My name is {}, I'm {}".format("John",36)

## **Escape Character**

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

txt = "We are the so-called \"Vikings\" from the north."  
print(txt)  
  
txt = "We are the so-called 'Vikings' from the north."  
print(txt)  
  
txt = 'We are the so-called "Vikings" from the north.'  
print(txt)  
  
txt = 'We are the so-called \'Vikings\' from the north.'  
print(txt)

)

If the value is not found, the find() method returns -1, but the index() method will raise an exception:

txt = "Hello, welcome to my world."  
  
print(txt.find("q"))  
print(txt.index("q"))

The index() method finds the first occurrence of the specified value.

The index() method raises an exception if the value is not found.

The index() method is almost the same as the [find()](https://www.w3schools.com/python/ref_string_find.asp) method, the only difference is that the find() method returns -1 if the value is not found. (See example below)

NUMBER OF OPERATIONS REQUIRED SO A S TO MAKE A SET OF SIDES TO FORM A TRIANGLE

a,b,c=map(int,input().split(" "))  
if(a+b)>c and (b+c)>a and (c+a)>b:  
 print(0)  
else:  
 x=max(a,b,c)  
 print(x+x-a-b-c+1)

Booleans

Almost any value is evaluated to True if it has some sort of content.

Any string is True, except empty strings.

Any number is True, except 0.

Any list, tuple, set, and dictionary are True, except empty ones.

One more value, or object in this case, evaluates to False, and that is if you have an object that is made from a class with a \_\_len\_\_ function that returns 0 or False:

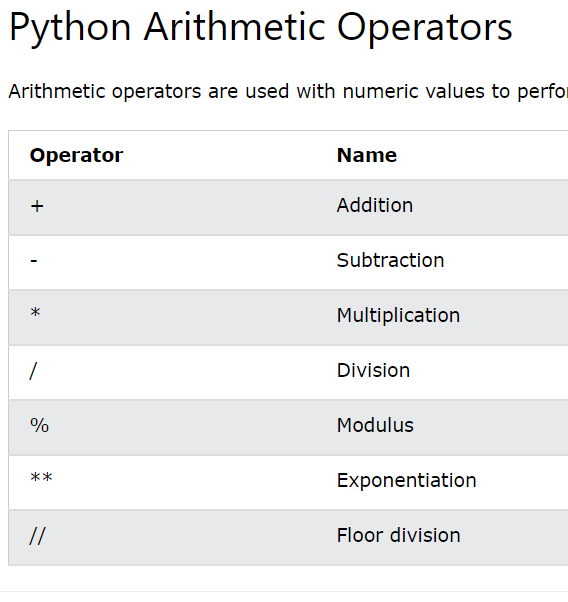
class myclass():  
  def \_\_len\_\_(self):  
    return 0  
  
myobj = myclass()  
print(bool(myobj))

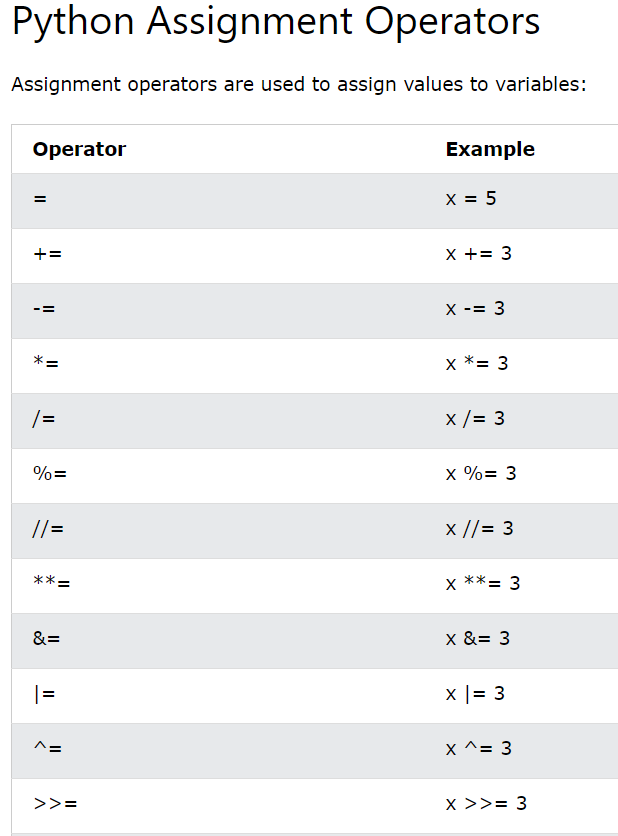
Python also has many built-in functions that return a boolean value, like the isinstance() function, which can be used to determine if an object is of a certain data type:

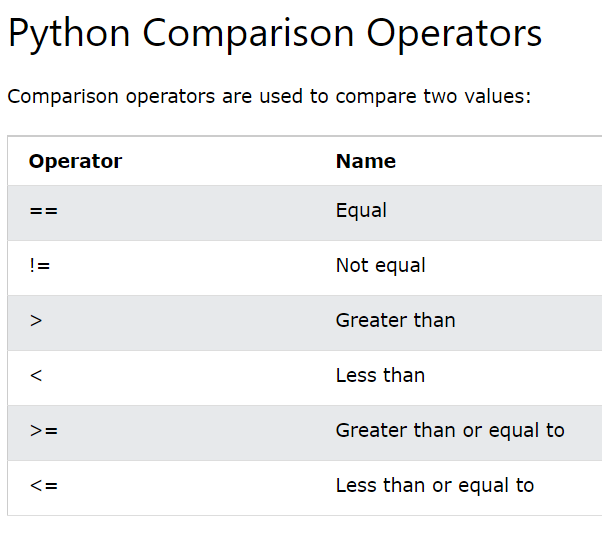
x = 200  
print(isinstance(x, int))  
  
x = 200.0  
print(isinstance(x, float))  
  
x ="sastra"  
print(isinstance(x, str))

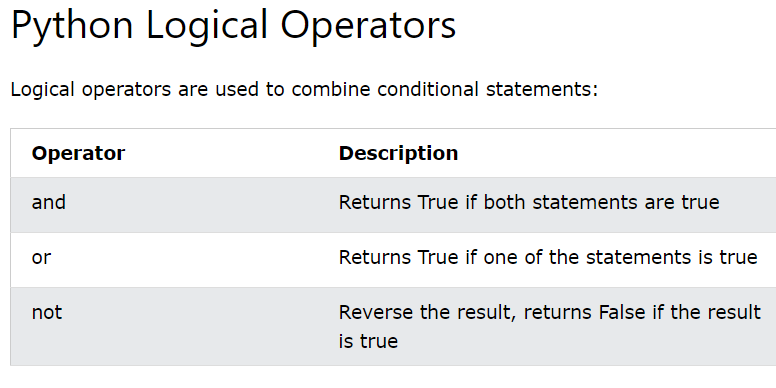
Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators



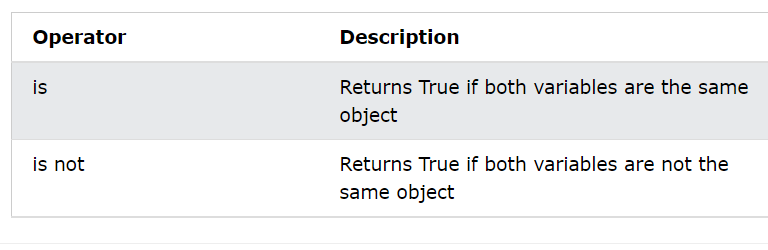






## **Python Identity Operators**

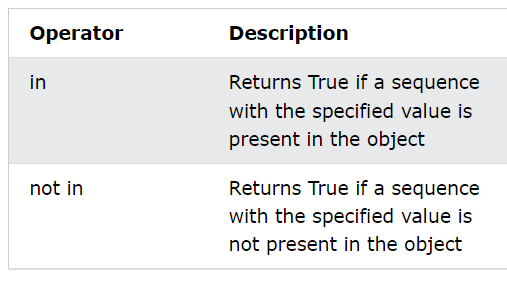
Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:



x=10  
y=10  
  
print(x is y)  
print(x is not y)

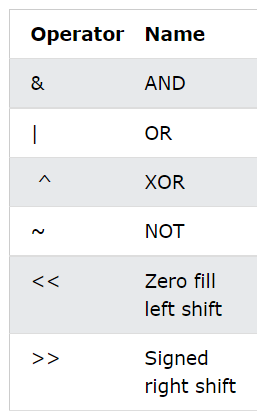
## **Python Membership Operators**

Membership operators are used to test if a sequence is presented in an object:



## **Python Bitwise Operators**

Bitwise operators are used to compare (binary) numbers:



## **List**

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets:

## **List Items**

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

## **Ordered**

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

**Note:** There are some [list methods](https://www.w3schools.com/python/python_lists_methods.asp) that will change the order, but in general: the order of the items will not change.

## **Changeable**

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

## **Allow Duplicates**

Since lists are indexed, lists can have items with the same value:

## **List Items - Data Types**

List items can be of any data type:

## **List Length**

To determine how many items a list has, use the len() function

A list can contain different data types

## **Python Collections (Arrays)**

There are four collection data types in the Python programming language:

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* [**Tuple**](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [**Set**](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.
* [**Dictionary**](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\*\* and changeable. No duplicate members.

\*Set items are unchangeable, but you can remove and/or add items whenever you like.

### Negative Indexing

Negative indexing means start from the end

-1 refers to the last item, -2 refers to the second last item etc.

## **Change Item Value**

To change the value of a specific item, refer to the index number:

## **Change a Range of Item Values**

To change the value of items within a specific range, define a list with the new values, and refer to the range of index numbers where you want to insert the new values

## **Insert Items**

To insert a new list item, without replacing any of the existing values, we can use the insert() method.

The insert() method inserts an item at the specified index

l=[1,"sanku",2.3,True,'sastra']  
print(l)  
print(l[-1])  
print(l[2:])  
print(l[:5])  
print(l[:])  
print(l[2:5])  
print(l[-5:-1])  
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[-4:-1])  
  
l[2]="surya"  
print(l)  
  
l[1:4]=[10,20,30,40,50]  
print(l)  
  
  
l[1:4]=["hallow"]  
print(l)  
  
l.insert(2,"watermelon")  
print(l)

## **Append Items**

To add an item to the end of the list, use the append() method

INSERT

To insert a list item at a specified index, use the insert() method.

The insert() method inserts an item at the specified index

EXTEND CAN BE USED TO ADD ITEMS OF ONE ITERABLE TO ANOTHER

l=[1,"sanku",2.3,True,'sastra']  
  
l.append("tinku")  
print(l)  
  
l.insert(3,"hallow")  
print(l)  
  
# extends one list with another list  
l2=[10,20,30]  
l2.extend(l)  
print(l2)  
  
# append one list with another at the end makes nested list  
l2=[10,20,30]  
l2.append(l)  
print(l2)  
  
# add iterable at the end  
thislist = ["apple", "banana", "cherry"]  
thistuple = ("kiwi", "orange")  
thislist.extend(thistuple)  
print(thislist)

## **Remove Specified Item**

The remove() method removes the specified item.

## **Remove Specified Index**

The pop() method removes the specified index.

If you do not specify the index, the pop() method removes the last item

## **Clear the List**

The clear() method empties the list.

The list still remains, but it has no content.

The del keyword can also delete the list completely.

l=[1,"sanku",2.3,True,'sastra']  
  
l.remove('sastra')  
print(l)  
  
l.pop(2)  
print(l)  
  
l.pop()  
print(l)  
  
l=[1,"sanku",2.3,True,'sastra']  
del l[1]  
print(l)  
  
l=[1,"sanku",2.3,True,'sastra']  
l.clear()  
print(l)  
  
del l  
# deletes list completely  
# print(l)

# **Python - Loop Lists**

l=[1,2,3,4,5]  
  
# printing list by looping  
for i in l:  
 print(i,end=" ")  
print()  
  
# printing list by looping the range og len of list  
for i in range(len(l)):  
 print(l[i],end=" ")  
print()  
  
# printing list using while loop  
i=0  
while i<len(l):  
 print(l[i],end=" ")  
 i+=1  
  
print()  
  
# printing list using comprehensions  
l2=[x for x in l]  
print(l2)  
  
[print(x,end=" ") for x in l]  
  
[print(x) for x in l]

LIST COMPREHENSIONS

newlist = [expression for item in iterable if condition == True]

l=[1,2,3,4,5,6,7,8,9,10]  
print(l)  
  
# all  
l2=[x for x in l]  
print(l2)  
  
# evens  
l2=[x for x in l if x%2==0]  
print(l2)  
  
# odds  
l2=[x for x in l if x%2!=0]  
print(l2)  
  
# squares of evens  
l2=[x\*\*2 for x in l if x%2==0]  
print(l2)  
  
# squares of odds  
l2=[x\*\*2 for x in l if x%2!=0]  
print(l2)  
  
l=["apple","mango","guava",'maaza','cat','dog','pilli']  
print(l)  
l2=[x for x in l if "a" in x]  
print(l2)  
  
l2=[x for x in l if "a" not in x]  
print(l2)  
  
  
l2=[x for x in l if "apple"!=x]  
print(l2)  
  
  
l2=[x for x in l if "apple"==x]  
print(l2)  
  
l2=[x for x in range(10)]  
print(l2)  
  
l2=[x for x in range(10) if x%3==0]  
print(l2)  
  
  
l2=[x for x in range(10) if x<6]  
print(l2)  
  
l2=[x for x in range(10) if x%2==0 and x%3!=0]  
print(l2)  
  
l2=[x.upper() for x in l]  
print(l2)  
  
l2=[x.capitalize() for x in l]  
print(l2)

## **Sortings**

l=[1.5,6.5,2,5.1,4.2,3,7,10.8]  
l.sort()  
print(l)  
  
  
l=[1,6,2,5,4,3,7,10]  
l.sort()  
print(l)  
  
# here sorted list comes for output but list is not sorted  
l=[1,6,2,5,4,3,7,10]  
print(sorted(l))  
  
print(l)  
  
l.sort(reverse=True)  
print(l)  
  
  
thislist = ["banana", "Orange", "Kiwi", "cherry"]  
thislist.sort()  
print(thislist)  
  
  
# reversing the order of list  
l=[1,85,2,45,36]  
print(l)  
l.reverse()  
print(l)  
  
  
# printing list elements from end  
l=[1,85,2,45,36]  
print(l[::-1])

By default the sort() method is case sensitive, resulting in all capital letters being sorted before lower case letters:

Case sensitive sorting can give an unexpected result:

thislist = ["banana", "Orange", "Kiwi", "cherry"]  
thislist.sort()  
print(thislist)

## **Copy a List**

You cannot copy a list simply by typing list2 = list1, because: list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2.

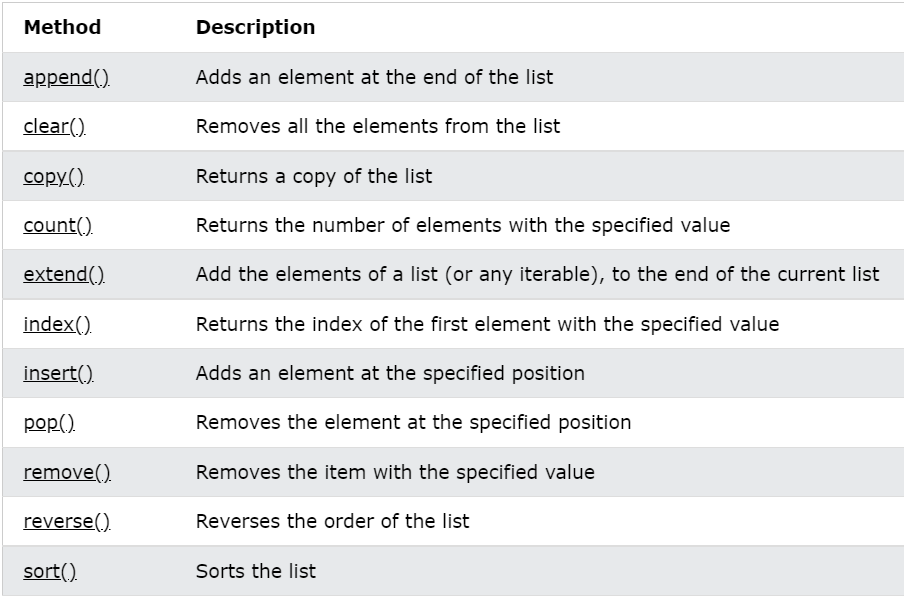
There are ways to make a copy, one way is to use the built-in List method copy().

l=[1,6,2,5,4,3,7,10]  
l2=l  
print(l)  
print(l2)  
  
#assigning one list to others changes made at one list will also causes to make changes in another list  
l[2]=100  
print(l)  
print(l2)  
  
#copy method  
l=[1,6,2,5,4,3,7,10]  
l2=l.copy()  
print(l)  
print(l2)  
  
l[2]=100  
print(l)  
print(l2)  
  
#list method  
l=[1,6,2,5,4,3,7,10]  
l2=list(l)  
print(l)  
print(l2)  
  
l[2]=100  
print(l)  
print(l2)

## **Join Two Lists**

There are several ways to join, or concatenate, two or more lists in Python.

list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
list1= list1 + list2  
print(list1)  
  
  
list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
list1.extend(list2)  
print(list1)  
  
list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
for i in list2:  
 list1.append(i)  
print(list1)



## **Tuple**

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

## **Tuple Items**

Tuple items are ordered, unchangeable, and allow duplicate values.

Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

## **Ordered**

When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.

## **Unchangeable**

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

'tuple' object does not support item assignment

## **Allow Duplicates**

Since tuples are indexed, they can have items with the same value:

## **Tuple Length**

To determine how many items a tuple has, use the len() function:

thistuple = ("apple",)  
print(type(thistuple))  
  
#NOT a tuple  
thistuple = ("apple")  
print(type(thistuple))

t=(1,2,3,4,5)  
print(t)  
  
t=("apple")  
print(type(t))  
  
t=("apple",)  
print(type(t))  
  
t="apple","mango"  
print(type(t))  
  
thistuple = ("apple", "banana", "cherry")  
print(thistuple[1])

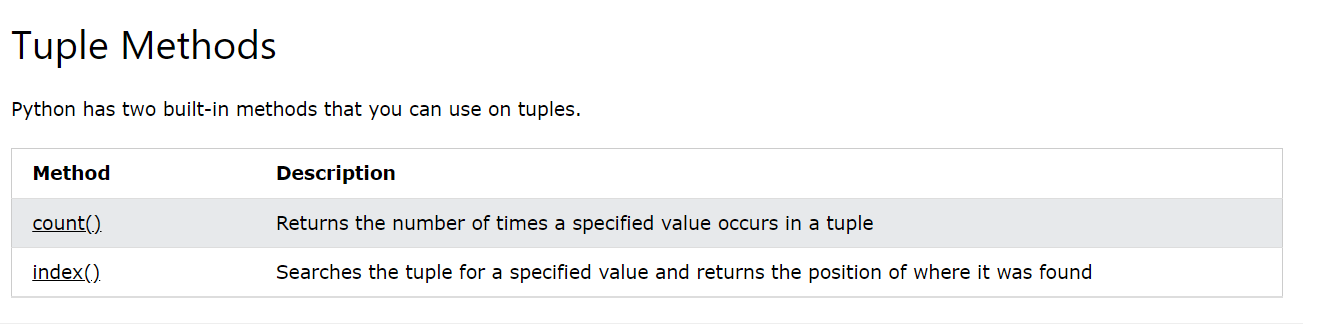
Tuples are unchangeable, meaning that you cannot change, add, or remove items once the tuple is created.

Changes can be made by converting a tuple into list and make changes and convert back the list into tuple

## **Unpacking a Tuple**

fruits = ("apple", "banana", "cherry")  
  
(green, yellow, red) = fruits  
  
print(green)  
print(yellow)  
print(red)  
  
  
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")  
  
(green, yellow, \*red) = fruits  
  
print(green)  
print(yellow)  
print(red)  
  
  
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")  
  
(green, \*yellow, red) = fruits  
  
print(green)  
print(yellow)  
print(red)  
  
  
  
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")  
  
(\*green, yellow, red) = fruits  
  
print(green)  
print(yellow)  
print(red)

tuple1 = ("a", "b" , "c")  
tuple2 = (1, 2, 3)  
  
tuple3 = tuple1 + tuple2  
print(tuple3)  
  
fruits = ("apple", "banana", "cherry")  
mytuple = fruits \* 2  
  
print(mytuple)



## **Set**

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Tuple](https://www.w3schools.com/python/python_tuples.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

A set is a collection which is unordered, unchangeable\*, and unindexed.

## **Set Items**

Set items are unordered, unchangeable, and do not allow duplicate values.

## **Unordered**

Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

## **Unchangeable**

Set items are unchangeable, meaning that we cannot change the items after the set has been created.

Once a set is created, you cannot change its items, but you can remove items and add new items.

## **Access Items**

You cannot access items in a set by referring to an index or a key.

But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}  
  
for x in thisset:  
  print(x)

Once a set is created, you cannot change its items, but you can add new items.

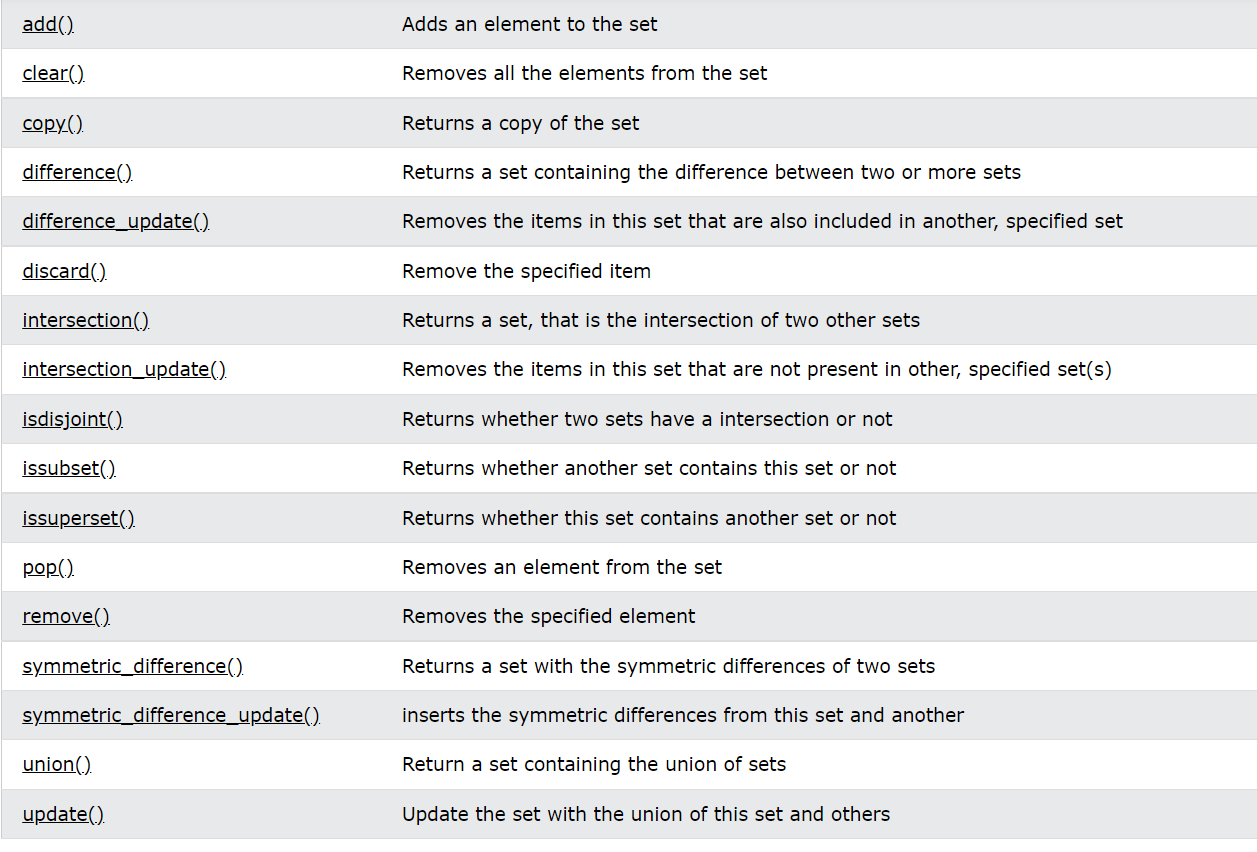
If the item to remove does not exist, remove() will raise an error.

**Note:** If the item to remove does not exist, discard() will **NOT** raise an error.

s={1,19,35,2,10,200}  
print(s)  
  
s.add(100)  
print(s)  
  
s1={1,19,35,2,10,200}  
print(s1)  
  
  
s2={11,119,351,211,101,2100}  
s1.update(s2)  
print(s2)  
print(s1)  
  
  
s.remove(1)  
print(s)  
  
s.discard(1)  
print(s)  
  
s.pop()  
print(s)  
  
s.clear()  
print(s)  
  
# deletes set permanently  
del s  
print(s)

thisset = {"apple", "banana", "cherry"}  
  
for x in thisset:  
 print(x)

s1={1,19,35,2,10,200}  
print(s1)  
  
s2={11,119,351,211,101,2100}  
print(s2)  
  
s3=s1.union(s2)  
print(s3)  
  
s1.update(s2)  
print(s2)  
  
print(s1==s3)



s1={1,19,35,2,10,200}  
print(len(s1))  
print(sum(s1))  
print(max(s1))  
print(min(s1))  
  
  
s1=[1,19,35,2,10,200]  
print(len(s1))  
print(sum(s1))  
print(max(s1))  
print(min(s1))  
  
  
s1=(1,19,35,2,10,200)  
print(len(s1))  
print(sum(s1))  
print(max(s1))  
print(min(s1))

## **Dictionary**

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

## **Dictionary Items**

Dictionary items are ordered, changeable, and does not allow duplicates.

Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

## **Accessing Items**

You can access the items of a dictionary by referring to its key name, inside square brackets

## **Change Values**

You can change the value of a specific item by referring to its key name

## **Update Dictionary**

The update() method will update the dictionary with the items from the given argument.

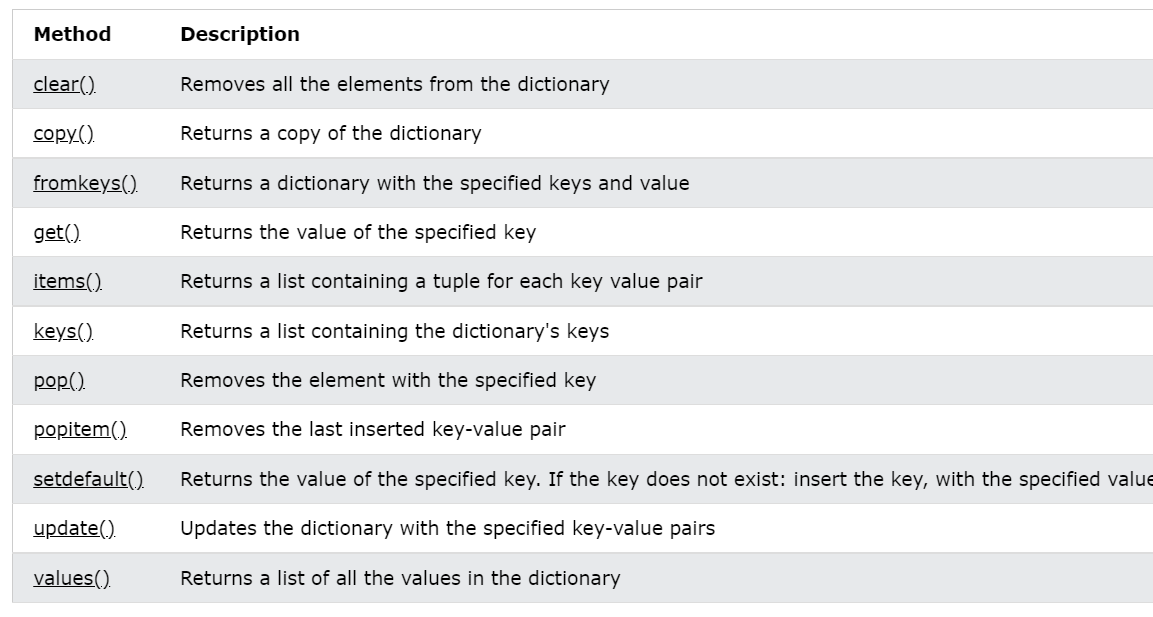
The argument must be a dictionary, or an iterable object with key:value pairs

## **Adding Items**

Adding an item to the dictionary is done by using a new index key and assigning a value to it

## **Removing Items**

There are several methods to remove items from a dictionary:



d={1:3,"sastra":"btech",1.1:True,False:0,False:"IDHI THAPPU","aha":3}  
print(len(d))  
print(type(d))  
  
  
for i in d.keys():  
 print(i,end="")  
  
print()  
  
for i in d.values():  
 print(i,end="")  
  
print()  
  
for i in d.keys():  
 print(d[i],end="")  
  
print()  
  
d2=d.copy()  
print(d2)  
  
d3=dict(d)  
print(d3)  
  
print(d.keys())  
print(d.values())  
l=d.keys()  
  
print(d.items())  
  
print(l)  
l=d.values()  
print(l)  
  
print(d["sastra"])  
print(d.get("aha"))  
print(d.get(1))  
  
d[1]=100  
print(d)  
  
d.update({1:111})  
print(d)  
  
d.pop("sastra")  
print(d)  
  
d.popitem()  
print(d)  
  
d.clear()  
print(type(d))  
  
del d  
  
d={1:"abc",1:{"a":1,"b":2},2:{"abc":123,"sastra":100}}  
print(d)

TUPLE,SET,DICTIONARY COMPREHENSIONS

s=[1,2,3,4,5]  
l=[f"My height is 5 feet {h}inches" for h in s]  
print(l)  
  
s=["apple","bat","cat"]  
l=[x.upper() for x in s]  
print(l)  
  
  
  
s=[1,2,3,4,5]  
l=(f"My height is 5 feet {h}inches" for h in s)  
print(l)  
print(type(l))  
  
s=["apple","bat","cat"]  
l={x.upper() for x in s}  
print(l)  
print(type(l))  
  
  
  
s=[1,2,3,4,5]  
l={f"My height is 5 feet {h}inches":h for h in s}  
print(l)  
print(type(l))  
  
s=["apple","bat","cat"]  
l={x.upper():x for x in s}  
print(l)  
print(type(l))

Zip Functions

keys=[1,2,3,4,5]  
values=[10,20,30,40,50]  
  
d=dict(zip(keys,values))  
print(d)  
print(type(d))  
  
keys=[1,2,3,4,5]  
values=[10,20,30,40,50]  
  
l=list(zip(keys,values))  
print(l)  
print(type(l))  
  
keys=[1,2,3,4,5]  
values=[10,20,30,40,50]  
  
s=set(zip(keys,values))  
print(s)  
print(type(s))

……………………………………Assignments……………………………………………………………….…………………………………………………

ASSIGNMENT-1

1.print true if three floating point numbers are strictly in ascending order

print("WRITE 3 FLOAT VALUES")  
x=float(input())  
y=float(input())  
z=float(input())  
if(x<y and y<z):  
 print(True)  
else:  
 print(False)

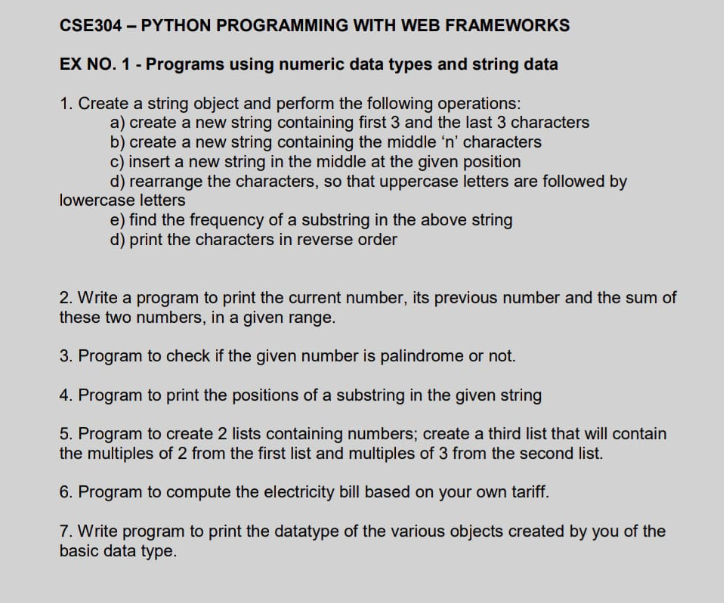
2. Write a script to convert RGB values to their equivalent CMYK values. If R, G, B values all are

zero, then CMY values are zero and K value is 1; else, use the following formulas.

print("Enter R, G, B values")  
R = float(input())  
G = float(input())  
B = float(input())  
if R == 0 and G == 0 and B == 0:  
 print("C = 0, M = 0, Y = 0, K = 1")  
else:  
 R1 = R / 255  
 G1 = G / 255  
 B1 = B / 255  
 K = 1 - max(R1, G1, B1)  
 C = (1 - R1 - K) / (1 - K)  
 M = (1 - G1 - K) / (1 - K)  
 Y = (1 - B1 - K) / (1 - K)  
 print(f"C = {C}, M = {M}, Y = {Y}, K = {K}")

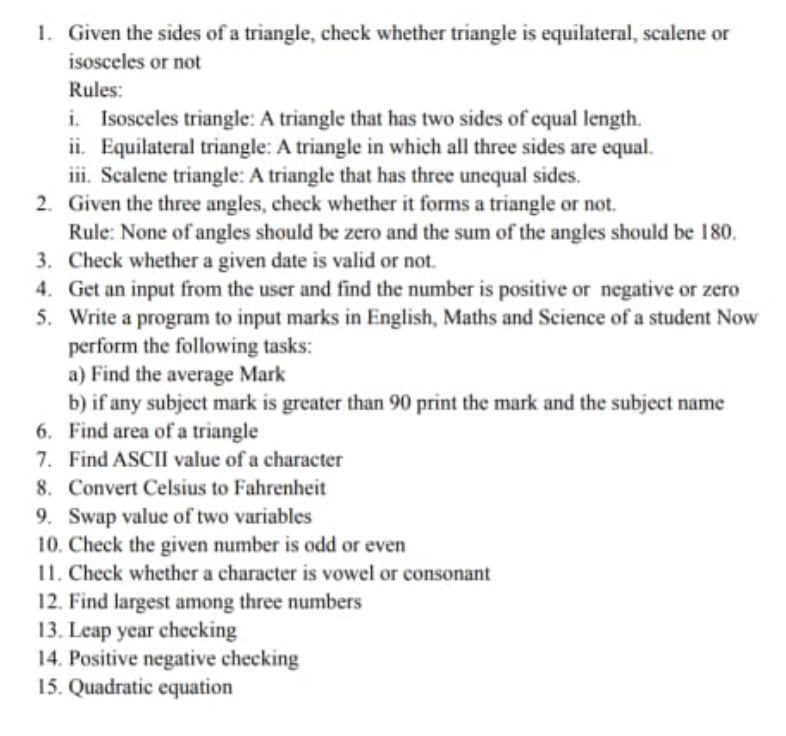
import math  
t= int(input())  
while (t):  
 n = int(input())  
 d = dict()  
 l = list(map(int, input().split()))  
 for i in l:  
 if (i not in d):  
 d[i] = 1  
 else:  
 d[i] = d[i] + 1  
 # print(d)  
 l = d.values()  
 if (max(l) > (math.ceil(n / 2))):  
 print('No')  
 else:  
 print('Yes')  
  
 t = t - 1  
  
  
  
  
import math  
print(math.factorial(5))  
  
  
  
  
# string methods  
  
s="sUrYaMaHEsH"  
print(s.capitalize())  
print(s.lower())  
print(s.upper())  
print(s.casefold())  
  
  
#places the string center  
txt = "banana"  
print(txt.center(20))  
print(txt.center(20, "\*"))  
print(txt.center(20, "$"))  
  
#count number of times a particular sub string exists in main string  
s="an apple day keeps apple a day away from apple seller"  
print(s.count("apple"))  
print(s.count("apple",1,10))  
print(s.count("apple",10,20))  
print(s.count("apple",10))  
  
#endswith  
s="surya is my world."  
print(s.endswith("."))  
print(s.endswith("world."))  
  
# whether a phrase ends with particular string  
print(s.endswith("world.",5,10))  
  
print(s.find("surya"))  
print(s.index("world"))  
  
  
print(s.find("surya",1,10))  
print(s.find("surya",1,10))  
  
txt = "Hello, welcome to my world."  
  
x = txt.index("welcome")  
  
print(x)  
  
#  
# The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).  
#  
# Example of characters that are not alphanumeric: (space)!#%&? etc.  
  
txt = "Company 12"  
  
x = txt.isalnum()  
  
print(x)  
  
# The isalpha() method returns True if all the characters are alphabet letters (a-z).  
txt = "CompanyX"  
  
x = txt.isalpha()  
  
print(x)  
  
# The isdecimal() method returns True if all the characters are decimals (0-9).  
#  
# This method is used on unicode objects.  
#  
  
  
a = "\u0030" #unicode for 0  
b = "\u0047" #unicode for G  
  
print(a.isdecimal())  
print(b.isdecimal())  
  
  
myDict = {"name": "John", "country": "Norway"}  
mySeparator = "TEST"  
  
x = mySeparator.join(myDict)  
  
print(x)  
  
  
myTuple = ("John", "Peter", "Vicky")  
  
x = "#".join(myTuple)  
  
print(x)  
  
# Make the lower case letters upper case and the upper case letters lower case:  
txt = "Hello My Name Is PETER"  
  
x = txt.swapcase()  
  
print(x)  
  
# Make the first letter in each word upper case:  
  
txt = "Welcome to my world"  
  
x = txt.title()  
  
print(x)  
  
# Remove spaces at the beginning and at the end of the string:  
  
txt = " banana "  
  
x = txt.strip()  
  
print("of all fruits", x, "is my favorite")  
  
  
  
txt = "We are the so-called \"Vikings\" from the north."  
print(txt)  
  
txt = "We are the so-called 'Vikings' from the north."  
print(txt)  
  
txt = 'We are the so-called "Vikings" from the north.'  
print(txt)  
  
txt = 'We are the so-called \'Vikings\' from the north.'  
print(txt)  
  
  
s="myself Surya age:{}"  
age=20  
print(s.format(age))  
  
s="hello! surya here class: {} age: {} weight: {} height: {}"  
age=20  
standard=15  
weight=55  
height=5.5  
print(s.format(age,standard,weight,height))  
  
quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want {} pieces of item {} for {} dollars."  
print(myorder.format(quantity, itemno, price))  
  
  
quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."  
print(myorder.format(quantity, itemno, price))  
  
  
a="hello"  
b="world"  
print(a,b)  
print(a+b)  
print(a+" "+b)  
  
s="sUrYa MaHeSh"  
  
#lowercase  
print(s.upper())  
  
#uppercase  
print(s.lower())  
  
#capitalize  
print(s.capitalize())  
  
#Whitespace is the space before and/or after the actual text, and very often you want to remove this space.  
a=" hello world "  
print(a.strip())  
  
#replaces a word of string with others  
print(a.replace("world","surya"))  
  
#The split() method splits the string into substrings if it finds instances of the separator  
a="surya,mahesh,pemma,brahmi,vikky"  
print(a.split(","))  
  
a="surya mahesh pemma brahmi vikky"  
print(a.split(" "))  
  
  
  
#print string as it is  
print(s)  
  
#first character in string  
print(s[1])  
  
#slicing starts from index 2 and index 5 not included  
print(s[2:5])  
  
#starts from 2 and upto end  
print(s[2:])  
  
#starts from 0 index ( since not mentioned) and upto index 9 (10 not included)  
print(s[:10])  
  
#total string  
print(s[:])  
  
#starts from 0 and upto end step value 1 ( usual string)  
print(s[::1])  
  
#starts from 0 and upto end step value 2  
print(s[::2])  
  
  
# slicing negative index starts from end with step value 1  
print(s[::-1])  
  
# slicing negative index starts from end with step value 2  
print(s[::-2])  
  
# slicing negative index starts from 5 th character fom last and ends at second character from last  
print(s[-5:-2])  
  
  
  
  
  
  
txt = "The best things in life are free!"  
print("expensive" not in txt)  
  
txt = "The best things in life are free!"  
if "expensive" not in txt:  
 print("No, 'expensive' is NOT present.")  
  
  
  
  
txt = "The best things in life are free!"  
if "life" in txt:  
 print("Yes, 'free' is present.")  
  
  
  
a="Sastra Deemed University"  
print(len(a))  
  
print("substring found : ","Sastra" in a)  
  
  
  
  
import random  
  
print(random.randrange(1, 10))  
  
  
a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)  
  
a = '''Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.'''  
print(a)  
  
  
for x in "banana":  
 print(x)  
  
  
for x in "banana":  
 print(x,end="")  
  
  
print()  
  
  
x = frozenset({"apple", "banana", "cherry"})  
  
#display x:  
print(x)  
  
#display the data type of x:  
print(type(x))  
  
  
  
  
x = range(6)  
  
#display x:  
print(x)  
  
#display the data type of x:  
print(type(x))  
  
  
i=2  
while True:  
 if i%3==0:  
 break  
 print(i)  
 i+=2

import re  
  
m=re.sub('morning','evening','good morning')  
print(m)  
  
  
s='welcome home'  
m=re.match(r'(.\*)(.\*?)',s)  
print(m.group())  
  
print('abcdefcdghcd'.split('cd',2))  
  
  
a=[1,2,3,4,5]  
b=[6,7,8,9]  
a.append(b)  
print(a)  
a=[1,2,3,4,5]  
b=[6,7,8,9]  
a.extend(b)  
print(a)  
  
print("Hello World"[::-1])  
  
person={"name":"jhon doe","age":83}  
  
d={}  
print(type(d))  
  
print(type(1/2))  
  
for i in range(10):  
 if i==5:  
 break  
 else:  
 print(i)  
else:  
 print("Here")  
  
  
def p(i,j):  
 if(i==0):  
 return j  
 else:  
 return p(i-1,i+j)  
  
print(p(4,7))  
  
import random  
random.randrange(3)  
  
  
def a(n):  
 if n==0:  
 return 0  
 elif n==1:  
 return 1  
 else:  
 return a(n-1)+a(n-2)  
  
for i in range(0,4):  
 print(a(i),end=" ")  
  
for i in '':  
 print(i)  
  
def foo(k):  
 k=[1]  
q=[0]  
foo(q)  
print(q)  
  
  
import math  
num=5  
print(math.factorial(num))  
  
t=tuple([1,2,3,4,5,6,7])  
print(t)  
print(t)  
  
  
#symbol @ Decorators. The main use case of the symbol @ in Python are decorators. In Python, a decorator extends the functionality of an existing function or class.12  
  
def myfunc():  
 global x  
 x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)



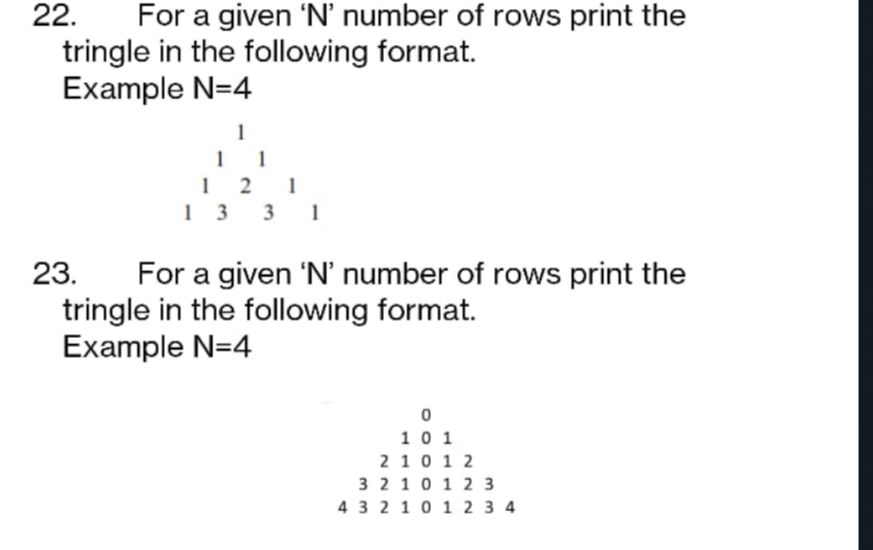
print("hello world")  
x = 10  
print(x)  
y = 0  
for i in range(x):  
 y = y + x  
 print(y, end=" ")  
  
print("")  
d = dict()  
print(d)  
print(type(d))  
  
l = [1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 1, 2, 3, 1, 2, 31, 2, 3, 1, 32, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 8, 9,  
 0, 1, 2, 3, 4, 5, 6]  
for i in l:  
 if (i in d):  
 d[i] = d[i] + 1  
 else:  
 d[i] = 1  
print(d)  
print(type(d))  
  
s = set()  
print(type(s))  
s = set(l)  
print(s)  
  
s = str()  
print(type(s))  
  
s = "Surya Mahesh KOLISETTY"  
print(s)  
s1 = str()  
s1 = s[:3] + s[len(s) - 3:]  
print(s1)  
  
s = "hello"  
print(s)  
print(len(s))  
s2 = str()  
l = len(s)  
n = 3  
d = l - n  
  
if n:  
 s2 = s[(d + 1) // 2:l - (d // 2)]  
print(s2)  
  
s = "surya mahesh kolisetty"  
s3 = "BABU"  
n = 12  
s4 = s[:n] + s3 + s[n:]  
print(s4)  
  
s = "suryaMAHESHkOlIsEttY"  
for i in s:  
 if (i.isupper()):  
 print(i, end="")  
for i in s:  
 if (i.islower()):  
 print(i, end="")  
  
print("")  
  
# frequency of the sub string in main string  
s = "friend is a friend who is not a friend but a friend is a friend"  
print(s.count('friend'))  
  
# printing string in reverse order  
print(s)  
print(s[::-1])  
  
# printing previous number and current number and sum of these two numbers  
n = 10  
for i in range(1, n):  
 print(i - 1, " ", i, " ", (i + i - 1))  
  
  
# palindrome check  
def palindromecheck(s):  
 if (s == s[::-1]):  
 print(s, 'PALINDOME NUMBER')  
 else:  
 print(s, 'NOT A PALINDROME NUMBER')  
  
  
n = 1234321  
n = str()  
palindromecheck(n)  
  
n = 71832  
n = str()  
palindromecheck()  
  
n = 7890987  
n = str()  
palindromecheck(n)  
  
# print the position of sub string in main string  
s = 'I LOVE YOU'  
print(s.index('LOVE'))  
  
l1 = [i for i in range(10)]  
l2 = [i for i in range(0, 100, 5)]  
print(l1)  
print(l2)  
  
l3 = [i for i in l1 if (i % 2 == 0)]  
l4 = [i for i in l2 if (i % 3 == 0)]  
l3 = l3 + l4  
print(l3)

basic exercise-1



# 1  
  
a=10  
b=10  
c=10  
if((a+b)>c and (b+c)>a and (c+a)>b):  
 print('Triangle exists with this measurements')  
  
 if (a == b == c):  
 print('Equilateral triangle')  
 elif ((a == b and b != c) or (b == c and c != a) or (a != b and b == c)):  
 print("Isosceles Triangle")  
 else:  
 print("Scalene Triangle")  
  
else:  
 print('Triangle not possible with this measurements')  
  
  
#2  
  
a=90  
b=30  
c=60  
  
if((a==0 or b==0 or c==0) or (a+b+c)!=180):  
 print('Triangle Not Exists')  
else:  
 print(('Triangle Exists'))  
  
  
#3  
  
d='10-05-2021'  
l=d.split("-")  
print(\*l)  
a=int(l[0])  
b=int(l[1])  
c=int(l[2])  
  
if((a>=1 and a<=31) and (b>=1 and b<=12) and (c>0)):  
 print('Date is Valid')  
else:  
 print('Date is invalid')  
  
#4  
  
n=10  
if(n>0):  
 print("positive")  
elif(n<0):  
 print("Negative")  
else:  
 print("zero")  
  
#7  
  
# ascii value of the character  
s='sastra'  
for i in s:  
 print(ord(i))  
  
  
#9  
  
# swap two variables  
a=10  
b=20  
print(a,b)  
a,b=b,a  
print(a,b)  
  
  
#11  
# check a character is a vowel or consonant  
  
ch='g'  
if(ch=='A' or ch=='a' or ch=='E' or ch =='e' or ch=='I'  
 or ch=='i' or ch=='O' or ch=='o' or ch=='U' or ch=='u'):  
 print(ch, "is a Vowel")  
else:  
 print(ch, "is a Consonant")  
  
  
#12  
# largest among three numbers  
a,b,c=10,5,20  
print(max(a,b,c))  
  
#13  
# leap year checking  
  
y=2000  
if((y%4 ==0) and (y%100 ==0) and (y%400 ==0)):  
 print(y," is a leap year")  
else:  
 print(y," Is not a leap year")

print("hello world")  
x=10  
print(x)  
y=0  
for i in range(x):  
 y=y+x  
 print(y,end=" ")  
  
print("")  
d=dict()  
print(d)  
print(type(d))  
  
  
l=[1,2,3,4,5,6,7,1,2,3,4,5,6,1,2,3,1,2,31,2,3,1,32,1,2,3,4,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6]  
for i in l:  
 if(i in d):  
 d[i]=d[i]+1  
 else:  
 d[i]=1  
print(d)  
print(type(d))  
  
s=set()  
print(type(s))  
s=set(l)  
print(s)  
  
  
s=str()  
print(type(s))  
  
s="Surya Mahesh KOLISETTY"  
print(s)  
s1=str()  
s1=s[:3]+s[len(s)-3:]  
print(s1)  
  
s="hello"  
print(s)  
print(len(s))  
s2=str()  
l=len(s)  
n=3  
d=l-n  
  
if n:  
 s2=s[(d+1)//2:l-(d//2)]  
print(s2)  
  
s="surya mahesh kolisetty"  
s3="BABU"  
n=12  
s4=s[:n]+s3+s[n:]  
print(s4)  
  
  
s="suryaMAHESHkOlIsEttY"  
for i in s:  
 if(i.isupper()):  
 print(i,end="")  
for i in s:  
 if(i.islower()):  
 print(i,end="")  
  
print("")  
  
# frequency of the sub string in main string  
s="friend is a friend who is not a friend but a friend is a friend"  
print(s.count('friend'))  
  
# printing string in reverse order  
print(s)  
print(s[::-1])  
  
  
# printing previous number and current number and sum of these two numbers  
n=10  
for i in range(1,n):  
 print(i-1," ",i," ",(i+i-1))  
  
  
# palindrome check  
def palindromecheck(s):  
 if (s == s[::-1]):  
 print(s, 'PALINDOME NUMBER')  
 else:  
 print(s, 'NOT A PALINDROME NUMBER')  
  
n=1234321  
n=str(n)  
palindromecheck(n)  
  
n=71832  
n=str(n)  
palindromecheck(n)  
  
n=7890987  
n=str(n)  
palindromecheck(n)  
  
# print the position of sub string in main string  
s='I LOVE YOU'  
print(s.index('LOVE'))  
  
  
l1=[i for i in range(10)]  
l2=[i for i in range(0,100,5)]  
print(l1)  
print(l2)  
  
l3=[i for i in l1 if(i%2==0)]  
l4=[i for i in l2 if(i%3==0)]  
l3=l3+l4  
print(l3)



23

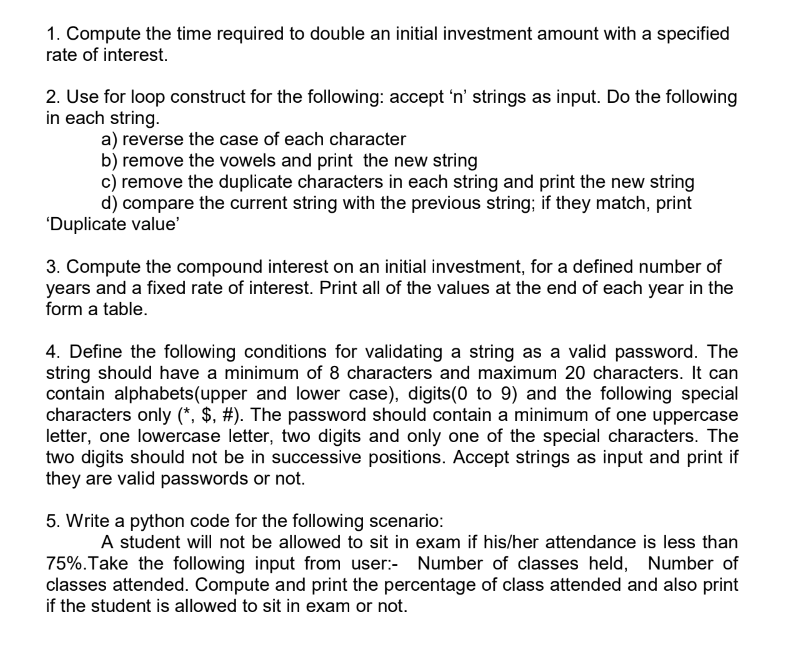
n = int(input())  
for i in range(n+1):  
 for \_ in range(n-i):  
 print(" ",end=" ")  
 x=i  
 while(x):  
 print(x,end=" ")  
 x=x-1  
 print(0,end=" ")  
 for p in range(1,i+1):  
 print(p,end=" ")  
  
 print()

eval

s="1+2"  
print(eval(s))  
  
s="2\*3"  
print(eval(s))  
  
import math  
s="math.pow(2,3)"  
print(eval(s))  
  
s="math.log(8,2)"  
print(eval(s))  
  
s="math.pow(2,3)"  
print(int(eval(s)))  
  
s="math.log(8,2)"  
print(int(eval(s))

strong password checker

# strong password will be with  
# atleast one lower case character  
# atleast one upper case character  
# atleast one digit  
# atleast one special character  
  
  
s=str(input())  
n=len(s)  
if(n>=8):  
 d=lc=uc=sp=0  
 for i in s:  
 if(i.isdigit()):  
 d+=1  
 elif(i.isalpha()):  
 if(i.islower()):  
 lc+=1  
 else:  
 uc+=1  
 else:  
 sp+=1  
 if(sp>=1 and d>=1 and uc>=1 and lc>=1):  
 print("Password accepted (Strong Password)....")  
 else:  
 print("Try again with strong password...")  
  
else:  
 print("Length of the password must be 8 characters or above...")



# 1

r = float(input("Enter the rate of interest:- "))

x = int(100 / r)

print("Time required to double an initial investment amount in simple intrest :- {} years".format(x))

# 2

n = int(input("Enter Number Of Test Cases "))

print("Enter {} Strings ".format(n))

l = []

for \_ in range(n):

l.append(input())

a = []

b = []

c = []

d = []

e = []

for i in l:

a.append(i.swapcase())

s = ""

vowels = "aeiouAEIOU"

for j in i:

if (j not in vowels):

s += j

b.append(s)

s = set(i)

ss = ""

for j in i:

if ((j in s) and (j not in ss)):

ss += j

c.append(ss)

# inputed strings

print("Your input...")

for i in l:

print(i)

# a

print("after reversing the case of each character")

for i in a:

print(i)

# b

print("after removing vowels in each string new strings are ....")

for i in b:

print(i)

# c

print("after removing the duplicate characters in each string new strings are.....")

for i in c:

print(i)

# d

print("searching fo the duplicate value")

for i in range(n):

if (l[i] == c[i]):

print("DUPLICATE VALUE FOUND......")

print(l[i], c[i])

else:

print("NOT A DUPLICATE VALUE....")

print(l[i], c[i])

# 3

# A = P(1 + r/100)^t

p = int(input("Enter Principle amount :- "))

r = float(input("Enter Rate of intrest :- "))

n = int(input("Enter Number of Years :- "))

def compundinterest(p, t, r):

a = p \* (1 + r \* 0.01) \*\* t

return a - p

ci = []

fv = []

t = 1

for i in range(n):

x = compundinterest(p, t, r)

ci.append(x)

p = p + x

fv.append(p)

# print(" END OF YEAR ", "COMPOUND INTEREST", "FINAL PRINCIPAL AMOUNT")

for i in range(1, n + 1):

print(

" END OF YEAR : {} COMPOUND INTEREST : {} Rupees FINAL PRINCIPLE AMOUNT : {} Rupees ".format(i, int(ci[i - 1]),

int(fv[i - 1])))

# 4

s = str(input("Enter Your Password..."))

n = len(s)

y = True

if (n >= 8 and n <= 20):

d = lc = uc = sp = 0

for o in range(n):

i = s[o]

q = o

if (i.isdigit()):

d += 1

if (((o - 1) >= 0 and s[o - 1].isdigit()) or (o + 1) < n and s[o + 1].isdigit()):

print("string is with successive digits...")

y = False

break

elif (i.isalpha()):

if (i.islower()):

lc += 1

else:

uc += 1

elif (i in ['\*', '$', '#']):

sp = sp + 1

else:

print("String Containing Unwanted characters")

y = False

break

if (sp == 1 and d >= 2 and uc >= 1 and lc >= 1 and y):

print("Password accepted ....")

else:

print("Try again with strong password...")

else:

print("Length of the password must be 8 characters or above...")

#5

m = int(input("Enter Number Of classes attended :- "))

n = int(input("Enter number Of Classes Held :- "))

a = (m \* 100) / n

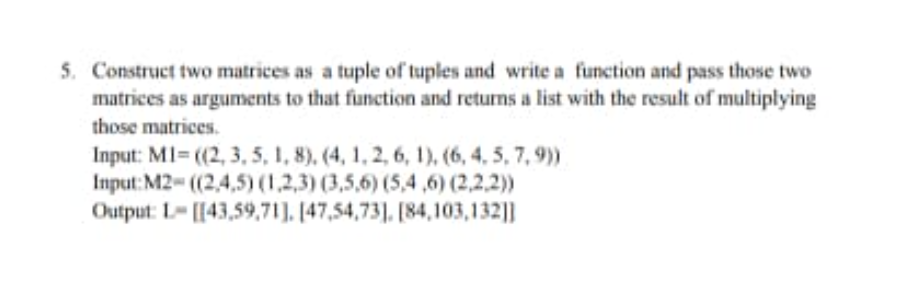
print(" Attendance percentage :- ", a)

if (a >= 75):

print("U are allowed to sit in exam")

else:

print("U are not allowed too sit in exam")



Matrix multiplication of 2d matrices

List of lists, tuple of tuples, list of tuples

m1=list()  
a=int(input("Number of rows in matrix 1:- "))  
b=int(input("Number of columns in matrix 1:- "))  
for i in range(a):  
 print("Enter "+str(b)+" space seperated integers : ")  
 t=tuple()  
 t+=tuple(map(int,input().split()))  
 m1.append(t)  
m1=tuple(m1)  
print(m1)  
  
  
m2=list()  
c=int(input("Number of rows in matrix 2:- "))  
d=int(input("Number of columns in matrix 2:- "))  
for i in range(c):  
 print("Enter "+str(d)+" space seperated integers : ")  
 t=tuple()  
 t+=tuple(map(int,input().split()))  
 m2.append(t)  
m2=tuple(m2)  
print(m2)  
  
def matrixmultiplication(m1,m2):  
 m = len(m1)  
 n = len(m1[0])  
 o = len(m2)  
 p = len(m2[0])  
 result=list()  
 if(n!=o):  
 print("matrix multiplication not possible....")  
 else:  
 for i in range(m):  
 l=list()  
 l=[0 for \_ in range(p)]  
 result.append(l)  
  
 for i in range(len(m1)):  
 for j in range(len(m2[0])):  
 for k in range(len(m2)):  
 result[i][j] += m1[i][k] \* m2[k][j]  
  
 print(result)  
matrixmultiplication(m1,m2)

default arguments

l=[1,2,3,4,5]  
a=[1,2,3,4,5]  
a.append(l)  
print(a)  
  
l=[1,2,3,4,5]  
a=[1,2,3,4,5]  
a.extend(l)  
print(a)  
  
def my\_function(\*kids):  
 print("The youngest child is " + kids[2])  
  
my\_function("Emil", "Tobias", "Linus")  
  
  
def my\_function(a,b,c=10):  
 print(a,b,c)  
  
my\_function(1,2,3)  
  
def my\_function(a,b=20,c=10):  
 print(a,b,c)  
  
my\_function(4,5,6)  
  
def my\_function(a,b=20,c=10):  
 print(a,b,c)  
  
my\_function(4)

MODULES

Calc.py

def add(a,b):  
 return a+b  
def sub(a,b):  
 return a-b  
def mul(a,b):  
 return a\*b  
def div(a,b):  
 return a/b  
def intdiv(a,b):  
 return a//b  
def mod(a,b):  
 return a%b  
def power(a,b):  
 return a\*\*b

Custom Calc module,math module,random module

import calc  
print(calc.add(1,2))  
print(calc.sub(1,2))  
print(calc.mul(1,2))  
print(calc.div(1,2))  
print(calc.intdiv(1,2))  
print(calc.power(2,3))  
  
  
import calc as c  
print(c.add(1,2))  
print(c.sub(1,2))  
print(c.mul(1,2))  
print(c.div(1,2))  
print(c.intdiv(1,2))  
print(c.power(2,3))  
  
  
from calc import add  
print(add(1,2))  
  
from calc import \*  
print(add(1,2))  
print(sub(1,2))  
print(mul(1,2))  
print(div(1,2))  
print(intdiv(1,2))  
print(power(1,2))  
  
import math  
print(math.sqrt(10))  
print(math.pow(10,2))  
print(math.factorial(5))  
print(math.gcd(1,2))  
print(math.ceil(10.1))  
print(math.floor(10.9))  
  
print(math.fabs(-10))  
print(math.fmod(3,2))  
  
  
import random  
print(random.random())  
print(random.randint(1,3))  
print(random.randrange(1,10))  
print(random.randrange(1,10,3))  
print(random.choice([1,2,"hello",1.5]))  
  
l=[1,2,3,4,5,6,7,8,9,10]  
print(l)  
random.shuffle(l)  
print(l)

datetime module

import datetime  
  
x=datetime.datetime.today()  
print(x)  
x=datetime.datetime.now()  
print(x)  
print(x.year)  
print(x.month)  
print(x.day)  
print(x.hour)  
print(x.min)  
print(x.minute)  
print(x.second)  
print(x.microsecond)  
  
x=datetime.datetime(2003,2,3)  
print(x.now())  
  
  
# format strftime  
  
print("week day short version -",x.strftime("%a"))  
print("week day full version -",x.strftime("%A"))  
print("month name short version -",x.strftime("%b"))  
print("month name full version -",x.strftime("%B"))  
print("year number short version -",x.strftime("%y"))  
print("year number full version -",x.strftime("%Y"))  
print("week day as a number -",x.strftime("%w"))  
print("month name as a number -",x.strftime("%m"))  
print("day of the month as number -",x.strftime("%d"))  
  
  
print("Hours 12 hour clock -",x.strftime("%I"))  
print("Hours 24 hour clock -",x.strftime("%H"))  
print("MINUTES -",x.strftime("%M"))  
print("SECONDS -",x.strftime("%S"))  
print("MICROSECONDS -",x.strftime("%f"))  
  
print("am/pm -",x.strftime("%p"))  
  
  
from datetime import date  
x=date(2002,8,12)  
print(x)  
  
print(date.today())  
  
x=date.today()  
print(x.year)  
print(x.month)  
print(x.day)

……………………………………………………………………………….……………

CSV FILES

# csv files  
  
# 4. Create a csv file containing the following details – FlowerName, Color, and Size. Add  
# some records into this file. Read the contents of the file and print the FlowerName of a  
# specific color.  
  
col = ['name', 'color', 'size']  
import csv  
  
with open('data.csv', 'w', newline='') as csv\_file:  
 csv\_writer = csv.writer(csv\_file)  
 csv\_writer.writerow(col)  
 n = int(input("Enter number of records wanna add into the file:- "))  
 for \_ in range(n):  
 l = []  
 l.append(input("Enter name of the flower :- "))  
 l.append(input("Enter COLOR of the flower :- "))  
 l.append(input("Enter SIZE of the flower :- "))  
 csv\_writer.writerow(l)  
  
with open('data.csv', 'r') as csv\_file:  
 csv\_rows = csv.reader(csv\_file)  
 for row in csv\_rows:  
 for i in row:  
 print(i, end=" ")  
 print()  
 c = input("Enter the color of the flower:- ")  
with open('data.csv', 'r') as csv\_file:  
 csv\_rows = csv.reader(csv\_file)  
 for row in csv\_rows:  
 l = list(row)  
 if (c in l):  
 print("Flower :- ", l[0], " is with ", l[1], " Color")

Csv files

l = [["surya", "mahesh", "kolisetty"], ["venkata", "koushik", "pemma"], ["gopi", "nath", "ch"]  
 , ["subbu", "akella", "trivikram"]]  
col = ["firstname", "lastname", "surname"]  
  
import csv  
  
with open('data.csv', 'w', newline='') as csv\_file:  
 csv\_writer = csv.writer(csv\_file)  
 csv\_writer.writerow(col)  
 csv\_writer.writerows(l)  
  
with open('data.csv', 'r') as file:  
 csv\_rows = csv.reader(file)  
 for row in csv\_rows:  
 print(row)  
  
with open('data.csv', 'r') as file:  
 csv\_rows = csv.DictReader(file)  
 for row in csv\_rows:  
 print(row)  
  
mydict = [{'passenger': '1', 'id': '1', 'survived': '1'},  
 {'passenger': '2', 'id': '2', 'survived': '2'},  
 {'passenger': '3', 'id': '3', 'survived': '3'}]  
fields = ['passenger', 'id', 'survived']  
with open('data2.csv', 'w') as file:  
 writer = csv.DictWriter(file, fieldnames=fields, delimiter='-')  
 writer.writeheader()  
 writer.writerows(mydict)  
  
with open('data2.csv', 'r') as file:  
 csv\_rows = csv.DictReader(file)  
 for row in csv\_rows:  
 print(row)

CSV DICTREADER

l = [["surya", "mahesh", "kolisetty"], ["venkata", "koushik", "pemma"], ["gopi", "nath", "ch"]  
 , ["subbu", "akella", "trivikram"]]  
col = ["firstname", "lastname", "surname"]  
  
import csv  
  
with open('data.csv', 'w', newline='') as csv\_file:  
 csv\_writer = csv.writer(csv\_file)  
 csv\_writer.writerow(col)  
 csv\_writer.writerows(l)  
  
with open('data.csv', 'r') as file:  
 csv\_rows = csv.reader(file)  
 for row in csv\_rows:  
 print(row)  
  
with open('data.csv', 'r') as file:  
 csv\_rows = csv.DictReader(file)  
 for row in csv\_rows:  
 print(row)  
  
mydict = [{'passenger': '1', 'id': '1', 'survived': '1'},  
 {'passenger': '2', 'id': '2', 'survived': '2'},  
 {'passenger': '3', 'id': '3', 'survived': '3'}]  
fields = ['passenger', 'id', 'survived']  
with open('data2.csv', 'w') as file:  
 writer = csv.DictWriter(file, fieldnames=fields, delimiter='-')  
 writer.writeheader()  
 writer.writerows(mydict)  
  
with open('data2.csv', 'r') as file:  
 csv\_rows = csv.DictReader(file)  
 for row in csv\_rows:  
 print(row)

python arrays

a=[1,2,3,4,5,6,7,8,9,10]  
print(a)  
print(\*a)  
  
# default removes last element  
a.pop()  
print(a)  
  
# default element at a particular index  
a.pop(1)  
print(a)  
  
# remove particular element in an array  
a.remove(5)  
print(a)  
  
# append a element at the end of the array  
a.append(100)  
print(a)  
  
b=a.copy()  
print(b)  
  
c=[11,12,13,14,15,16]  
print(c)  
  
# extend one list of elements with another list  
a.extend(c)  
print(a)  
  
# index of specified element in the array  
print(a.index(100))  
  
# inserts specified element at specified position  
a.insert(7,200)  
print(a)  
  
a.insert(7,200)  
print(a)  
# returns the count of the specifies element in the array  
print(a.count(200))  
  
# reverses the array of elements  
a.reverse()  
print(a)  
  
# sorts the elements in the array  
a.sort()  
print(a)  
  
print(b)  
b.reverse()  
print(b)  
print(sorted(b))  
print(b)  
  
b.sort(reverse=True)  
print(b)  
  
# clear all the elements in the array  
a.clear()  
print(a)

Exception handling

# exception handling  
# code with exceptions  
  
  
try:  
 l = [1]  
 print(l[10])  
except IndexError as e:  
 print("Index out of bound exception broo take care of it...! ",e)  
except Exception as e:  
 print("Something went wrong",e)  
else:  
 print("no exceptions broo hurray...!")  
finally:  
 print("i am from finally block :) ")  
  
  
  
print("\n##################################################################\n")  
  
# file not found error (exception)  
  
try:  
 f = open("suryafile")  
 print("U Can't Print Me Okay. If u print then u have no exceptions")  
except ArithmeticError:  
 print("Arithmetic error...........")  
except FileNotFoundError as e:  
 print("File Not Found Bro Sorry..",e)  
except FileExistsError as e:  
 print("File Exists But Error Sorry For that...")  
except Exception:  
 print("Something Went wrong Sorry..")  
else:  
 print("Hurrayyy No exceptions brooo ! Party....:) i only execute only when no errors...")  
finally:  
 print("Myself finally block broo am i looking beautiful...?")  
  
print("\n##################################################################\n")  
  
# arithmetic error exception  
try:  
 x=0  
 print(1/x)  
 print("U Can't Print Me Okay. If u print then u have no exceptions")  
except ArithmeticError as e:  
 print("Arithmetic error (Exception) brooo...........",e)  
except FileNotFoundError as e:  
 print("File Not Found Bro Sorry..",e)  
except FileExistsError as e:  
 print("File Exists But Error Sorry For that...")  
except Exception:  
 print("Something Went wrong Sorry..")  
else:  
 print("Hurrayyy No exceptions brooo ! Party....:) i only execute only when no errors...")  
finally:  
 print("Myself finally block broo am i looking beautiful...?")  
  
  
  
print("\n##################################################################\n")  
  
# code with out exceptions  
try:  
 f = open("suryafile.txt")  
 print("U Can't Print Me Okay. If u print then u have no exceptions")  
except ArithmeticError:  
 print("Arithmetic error...........")  
except FileNotFoundError as e:  
 print("File Not Found Bro Sorry..",e)  
except FileExistsError as e:  
 print("File Exists But Error Sorry For that...")  
except Exception:  
 print("Something Went wrong Sorry..")  
else:  
 print("Hurrayyy No exceptions brooo ! Party....:) i only execute only when no errors...")  
finally:  
 print("Myself finally block broo am i looking beautiful...? i will visible to u irrespective to the exceptions")  
  
  
print("\n##################################################################\n")  
  
# User Defined Exceptions  
#  
# userdefined exceptions should be declared in the form of classes that should  
# derived from the exception class using raise keyword  
  
class invaliddata(Exception):  
 pass  
  
marks=int(input("Enter invalid marks of student to raise user defined exception:- "))  
  
try:  
 if marks<0 or marks>100:  
 raise invaliddata  
except invaliddata:  
 print("Marks of the student is invalid..")  
 print("Marks should be >0 and <100 ")  
except Exception as e:  
 print("Something went wrong...",e)  
else:  
 print("No exceptions")  
  
  
print("\n##################################################################\n")  
  
# userdefined exceptions for eligibility for age for voting  
  
class voting(Exception):  
 pass  
  
age=int(input("Enter age that is not suitable for voting to see user defined exception:- "))  
  
try:  
 if(age<18 or age>100):  
 raise voting  
except voting:  
 print("U are not eligible for voting broo sorry try for next elections...")  
except Exception as e:  
 print("something went wrong... May be some Exception..",e)  
else:  
 print("hurray no exceptions......")  
finally:  
 print("From finally block....")  
  
  
  
print("\n##################################################################\n")

File handling

# file handling  
  
f=open("myfile.txt","w")  
f.write("Good morning thanjavur...\n")  
f.write("Good morning trichy...\n")  
f.write("Good morning thirumalaisamudram...\n")  
f.write("Good morning chennai...\n")  
f.close()  
  
  
f=open("myfile.txt","r")  
print(f.read())  
f.close()  
  
# open the files with with keyword the file will automatically closes the file object  
# no need of closing the file manually  
with open("myfile.txt","r")as f:  
 print(f.read())  
  
  
f=open("test.txt",'w')  
  
# to write the contents into the file  
f.write("first line\n")  
f.write("second line\n")  
f.write("third line\n")  
f.write("fourth line\n")  
f.write("fifth line\n")  
f.write("sixth line\n")  
f.write("seventh line\n")  
  
# to get the mode of the file  
print(f.mode)  
  
# to check whether a file is closed or not  
print(f.closed)  
f.close()  
  
  
f=open("test.txt","r")  
# to read the contents of the file  
file\_contents=f.readlines()  
print(file\_contents)  
f.close()  
  
  
f=open("test.txt","r")  
# to read the contents of the file  
file\_contents=f.read()  
print(file\_contents,end=" ")  
f.close()  
  
  
f=open("test.txt","r")  
# to read the contents of the file  
for line in f:  
 print(line)  
f.close()  
  
# to close the file  
f.close()  
print(f.closed)  
  
  
  
#3. Open a text file in your python code and print the content in a specific line or a range of  
#line numbers.  
  
f1=open("data.txt","w")  
s="""Hello Good Morning  
My self surya  
heyyy heellooooo  
glad to meet uuuuu  
Superrr wonderfullll"""  
  
f1.write(s)  
f1.close()  
  
#printing entire file ( multiple lines of file at a time)  
#using f.read() method  
f2=open("data.txt","r")  
print(f2.read())  
f2.close()  
  
#printing data in file line by line  
#using for loop  
f3=open("data.txt","r")  
for line in f3:  
 print(line)  
f3.close()  
  
#get the count of number of lines in file  
#using len(f.readlines() method  
f=open("data.txt","r")  
  
# f.readlines() reads entire file and returns it as a list  
n=len(f.readlines())  
print("Number of lines in 'data.txt' file :- ",n)  
f.close()  
  
#printing data in file line by line  
#using readline method  
  
f=open("data.txt","r")  
for \_ in range(n):  
 print(f.readline())  
f.close()  
  
# printing data in file at specific line  
f=open("data.txt","r")  
l=f.readlines()  
print("Total number of lines :- ",len(l))  
m=int(input("Enter specific line number to get the data :- "))  
  
if(m<=len(l)):  
 print(l[m-1])  
else:  
 print("error :input should me within th range of the file ")  
f.close()  
  
# printing data at range of line numbers  
f=open("data.txt","r")  
l=f.readlines()  
print("Total number of lines :- ",len(l))  
m=int(input("Enter specific line number to get the data :- "))  
  
if(m<=len(l)):  
 for i in range(0,m):  
 print(l[i])  
else:  
 print("error :input should me within th range of the file ")  
  
  
# 3. Open a text file in your python code and print the content in a specific line or a range of  
# line numbers.  
  
f1 = open("data.txt", "w")  
s = """Hello Good Morning  
My self surya  
heyyy heellooooo  
glad to meet uuuuu  
Superrr wonderfullll"""  
  
f1.write(s)  
f1.close()  
  
# printing entire file ( multiple lines of file at a time)  
# using f.read() method  
f2 = open("data.txt", "r")  
print(f2.read())  
f2.close()  
  
# printing data in file line by line  
# using for loop  
f3 = open("data.txt", "r")  
for line in f3:  
 print(line)  
f3.close()  
  
# get the count of number of lines in file  
# using len(f.readlines() method  
f = open("data.txt", "r")  
  
# f.readlines() reads entire file and returns it as a list  
n = len(f.readlines())  
print("Number of lines in 'data.txt' file :- ", n)  
f.close()  
  
# printing data in file line by line  
# using readline method  
  
f = open("data.txt", "r")  
for \_ in range(n):  
 print(f.readline())  
f.close()  
  
# printing data in file at specific line  
f = open("data.txt", "r")  
l = f.readlines()  
print("Total number of lines :- ", len(l))  
m = int(input("Enter specific line number to get the data :- "))  
  
if (m <= len(l)):  
 print(l[m - 1])  
else:  
 print("error :input should me within th range of the file ")  
f.close()  
  
# printing data at range of line numbers  
f = open("data.txt", "r")  
l = f.readlines()  
print("Total number of lines :- ", len(l))  
m = int(input("Enter specific line number to get the data :- "))  
  
if (m <= len(l)):  
 for i in range(0, m):  
 print(l[i])  
else:  
 print("error :input should me within th range of the file ")

Classes , objects,inheritance

#########################################################################################  
# classes and objects  
import six  
  
  
class mahesh:  
 age=19  
 sex="male"  
 weight=55  
 height=5.6  
  
obj=mahesh()  
print("Mahesh age:- ",obj.age)  
print("Mahesh sex:- ",obj.sex)  
print("Mahesh weight:- ",obj.weight)  
print("Mahesh height:- ",obj.height)  
  
  
  
# The \_\_init\_\_() Function  
# The examples above are classes and objects in their simplest form, and are not really useful in real life applications.  
#  
# To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function.  
#  
# All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.  
#  
# Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created:  
  
  
class human:  
 def \_\_init\_\_(self,name,age):  
 self.name=name  
 self.age=age  
  
h1=human("surya",20)  
print("Human name:- ",h1.name)  
print("Human age :- ",h1.age)  
  
  
# Note: The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.  
  
  
 # class with \_\_str()\_\_ function returns the string representation of the object  
  
class temp:  
  
 def \_\_init\_\_(self,x,y):  
 self.x=x  
 self.y=y  
 def \_\_str\_\_(self):  
 return "My x value :- "+str(self.x)+" My y value:- "+str(self.y)  
  
obj=temp(10,20)  
print(obj.x)  
print(obj.y)  
print(obj)  
  
# class functions  
  
class myclass:  
 def \_\_init\_\_(self,height,weight):  
 self.height=height  
 self.weight=weight  
 def \_\_str\_\_(self):  
 return "My class Height:- "+str(self.height)+" My class weight:- "+str(self.weight)  
  
 def display(self):  
 print("From Display Method of myclass class:-")  
 print("WEIGHT:- ",self.weight)  
 print("HEIGHT:- ",self.height)  
  
obj=myclass(5.5,55)  
print(obj)  
  
# calling class methods using class instances (objects)  
obj.display()  
  
# modifying class attributes  
obj.height=20  
obj.weight=200  
obj.display()  
  
# we can delete object parameters using del keyword  
  
del obj.height  
  
# we can delete objects of a class using del keyword  
del obj  
  
# creating empty classes using pass statement  
class urclass:  
 pass  
  
  
# classes and instances  
  
  
class employee:  
 # declaring the attributes of the class employee  
  
 firstname=None  
 lastname=None  
 email=None  
 pay=None  
  
 def \_\_init\_\_(self,height,weight):  
 # introducing two new attributes height and weight to employee class from class construtor  
 self.height=weight  
 self.weight=height  
  
 def display(self):  
 print(self.firstname)  
 print(self.lastname)  
 print(self.pay)  
 print(self.email)  
 print(self.height)  
 print(self.weight)  
  
  
  
emp1=employee(5,55)  
emp1.firstname="surya"  
emp1.lastname="mahesh"  
emp1.pay=100000  
emp1.email="kolisettysuryamahesh@gmail.com"  
  
print(emp1.firstname)  
print(emp1.lastname)  
print(emp1.pay)  
print(emp1.email)  
  
emp2=employee(5.5,55)  
# dipslaying the details of the object emp1 using class function  
  
emp2.display()  
  
# calling class function using classname.functionname  
employee.display(emp2)  
  
  
# class variables  
  
# \_\_dict\_\_ method of employee class  
print(emp1.\_\_dict\_\_)  
print(emp2.\_\_dict\_\_)  
  
  
# python class inheritance  
class parent:  
 def \_\_init\_\_(self,fname,lname):  
 self.fname=fname  
 self.lname=lname  
  
 def printname(self):  
 print(self.fname,self.lname)  
  
# using person clas to create an object and then execute the printname classmethod  
  
pobj=parent("Surya Mahesh","Kolisetty")  
pobj.printname()  
  
# use pass keyword if u dont want to add any other properties or methods to the class  
# use child class objects and execute the methods in parent  
class child(parent):  
 pass  
  
  
cobj=child("Venkata Koushik","Pemma")  
print(cobj.fname)  
print(cobj.lname)  
cobj.printname()  
  
# calling parent class constructor using child class constructor with parent class name  
  
class child1(parent):  
 def \_\_init\_\_(self,fname,lname):  
 parent.\_\_init\_\_(self,fname,lname)  
  
cobj1=child1("Gopi Nath","chennamsetti")  
cobj1.printname()  
  
  
# adding constructors to child class  
# child init function overrides the inheritance of the parent's  
# calling parent class constructor using child class constructor with suoer keyword  
  
  
class child1(parent):  
 def \_\_init\_\_(self,fname,lname):  
 super().\_\_init\_\_(fname,lname)  
  
cobj1=child1("Uday Kiran","Kankanala")  
cobj1.printname()  
  
# adding properties to child class other than parent class  
  
class child2(parent):  
 def \_\_init\_\_(self,fname,lname):  
 super().\_\_init\_\_(fname,lname)  
 self.age=20  
  
cobj1=child2("Thiru Malleshwara","Kamireddy")  
cobj1.printname()  
print(cobj1.age)  
  
# adding methods and constructors to the child class  
  
class child3(parent):  
 def \_\_init\_\_(self,fname,lname,age):  
 super().\_\_init\_\_(fname,lname)  
 self.age=20  
  
 def printdetails(self):  
 super().printname()  
 print(self.age)  
  
 def welcome(self):  
 print("Welcome ",self.fname," To the class Heyyyy...! ")  
  
  
cobj1=child3("Yashwant Reddy","Kanala",21)  
cobj1.printdetails()  
cobj1.welcome()

Exception handling with files

# 4. The program accepts a file containing a poem, as input. The poem should follow  
# the following rules:  
  
# a) All odd numbered lines should contain exactly 6 words and even-numbered lines  
# to contain exactly 5 words.  
# b) Odd numbered lines should contain a hyphen as the last character (this hyphen is  
# not counted in the 6 words count)  
# c) Even numbered lines should not start with uppercase letter  
# Non-compliance to the above rules are to be dealt with as follows:  
# a) Raise an exception named ‘WordCountException’ along with the actual count of  
# words present in the line  
# b) Raise an exception named ‘MissingHyphenException’ along with the line number  
# c) Raise an exception named ‘CaseMismatchException’ with a message ‘Line starts  
# with uppercase letter’  
# Write a python code to implement the above scenario. Also, print the count of each  
# type of exception raised, at the end of the execution.  
  
  
class WordCountException(Exception):  
 pass  
  
class MissingHypenException(Exception):  
 pass  
  
class CaseMismatchException(Exception):  
 pass  
  
wcount=0  
mcount=0  
ccount=0  
  
f=open("poem.txt","w")  
n=int(input("Enter Number of lines of poem:- "))  
for i in range(n):  
 s=input("Enter line "+str((i+1))+" of the poem:- ")  
 f.write(s)  
 # for writing each line of text as newline of the file using "\n"  
 f.write("\n")  
  
  
f.close()  
  
f=open("poem.txt","r")  
l=f.readlines()  
print("Total number of lines in the poem:- ",len(l))  
f.close()  
  
  
f=open("poem.txt","r")  
  
for i in range(len(l)):  
 s = f.readline()  
 try:  
 if ((i + 1) % 2 != 0):  
 arr = s.split(" ")  
 if (len(arr) != 6):  
 print((i+1)," th line in the poem is with :- ",len(arr)," words...\nExpected 6 words :( ...")  
 wcount=wcount+1  
 raise WordCountException  
  
 if(s[-2]!="-"):  
 mcount=mcount+1  
 raise MissingHypenException  
 else:  
 arr = s.split(" ")  
 if (len(arr) != 5):  
 print((i+1)," th line in the poem is with :- ",len(arr)," words...\nExpected 5 words :( ...")  
 wcount = wcount + 1  
 raise WordCountException  
 if (ord(s[0])>=65 and ord(s[0])<=90):  
 ccount=ccount+1  
 raise CaseMismatchException  
  
 except WordCountException:  
 print("Word Count Exception Raised bro...")  
 print("we expect 6 words in odd numbered line and\n5 words in even numbered line")  
 except MissingHypenException:  
 print("Broooo! Missing Hypen Exception bro ")  
 print("we Expect Hypen as last character of odd numbered line of the poem.......")  
 except CaseMismatchException:  
 print("Broooo! Case Mismatch Exception bro ")  
 print("we don't Expect Upper Case letters as start of the even numbered line of the poem.......")  
  
  
  
print(" Total Number of Exceptions Raised in the poem:- ",(mcount+wcount+ccount))  
print(" Total Number of MissingHypenExceptions Raised in the poem:- ",(mcount))  
print(" Total Number of WordCountExceptions Raised in the poem:- ",(wcount))  
print(" Total Number of CaseMismatchExceptions Raised in the poem:- ",(ccount))

Hamming distance between two bit strings and generating all the possible bit strings with the given length

# 3. The Hamming distance between two bit strings of length n is equal to the number  
# of bits in which the two strings differ. Compose a program that takes an integer k and  
# a bitstring s and writes all bit strings that have Hamming distance at most k from s.  
# Eg . if k is 2 and s = 000, then the output should be  
# 110 101 011  
  
  
s = input("Enter a Bit string:- ")  
k = int(input("Enter an integer value for k:- "))  
  
# n holds the value of the length of the bit string  
n=len(s)  
  
# l is the list that holds the bit strings of given length n  
l = []  
  
def genbin(n, bs=''):  
 if len(bs) == n:  
 l.append(bs)  
 else:  
 genbin(n, bs + '0')  
 genbin(n, bs + '1')  
  
genbin(n)  
  
def hamming\_distance(a,b):  
 count=0  
 for i in range(len(a)):  
 if(a[i]!=b[i]):  
 count=count+1  
 return count  
  
for i in l:  
 if (i!=s) and hamming\_distance(i,s)==k:  
 print(i)

COMPOSITION in Classes and objects

class dob:  
 def \_\_init\_\_(self,day,month,year):  
 self.day=day  
 self.month=month  
 self.year=year  
  
 def \_\_str\_\_(self):  
 return "Date: "+str(self.day)+"/"+str(self.month)+"/"+str(self.year)  
  
class student:  
 def \_\_init\_\_(self):  
 self.name="Surya Mahesh"  
 self.age=20  
 self.db=dob(3,2,2003)  
  
 def \_\_str\_\_(self):  
 return "My self "+self.name+" My age: "+str(self.age)+" Date Ob Birth: "+str(self.db)  
  
  
obj=student()  
print(obj)   
print(obj.\_\_dict\_\_)

Data Encapsulation

Private members, getters and setters (Accessor methods and mutator methods)

class dob:  
 def \_\_init\_\_(self,day,month,year):  
 self.day=day  
 self.month=month  
 self.year=year  
  
  
  
 def \_\_str\_\_(self):  
 return "Date: "+str(self.day)+"/"+str(self.month)+"/"+str(self.year)  
  
class person:  
 def \_\_init\_\_(self):  
 self.name="Surya Mahesh"  
 self.age=20  
 self.db=dob(3,2,2003)  
 self.\_\_salary = 100000  
 self.\_\_girlfriend="angel"  
  
 def show(self):  
 print(self.\_\_salary)  
 print(self.\_\_girlfriend)  
  
 def getsalary(self):  
 return self.\_\_salary  
  
 def getgirlfriend(self):  
 return self.\_\_girlfriend  
  
 def \_\_str\_\_(self):  
 return "My self "+self.name+" My age: "+str(self.age)+" Date Ob Birth: "+str(self.db)  
  
  
obj=person()  
print(obj)  
print(obj.\_\_dict\_\_)  
obj.show()  
print(obj.getsalary())  
print(obj.getgirlfriend())  
  
print(id(obj))

Recursive functions

# recursive functions  
  
# adding numbers in a range using recusrion  
  
def add(num):  
 if num==0:  
 return 0  
 return num+add(num-1)  
  
print("Sum of first 10 natuarl numbers:- ",add(10))  
  
  
# factorial using recursion  
  
def fact(num):  
 if num==0 or num==1:  
 return 1  
 return num\*fact(num-1)  
  
print("factorial of 5:- ",fact(5))  
  
  
# fibanocci series of n using recursion  
  
def fib(n):  
 if n==0 or n==1:  
 return n  
 else:  
 return fib(n-1)+fib(n-2)  
  
n=10  
for i in range(n+1):  
 print(fib(i),end=" ")

Datetime module

# dates and time  
  
from datetime import date,time,datetime,timedelta  
  
mybday=date(2003,2,3)  
  
print(mybday.day)  
print(mybday.month)  
print(mybday.year)  
  
print(mybday)  
  
print(date.today())  
  
print(time())  
  
print(time(10,20,30,1092))  
  
print(datetime.today())  
print(datetime.now())  
  
x=datetime.now()  
  
print(x.year)  
print(x.month)  
print(x.day)  
print(x.hour)  
print(x.minute)  
print(x.second)  
  
print(x.strftime("%A"))  
print(x.strftime("%a"))  
  
  
print(x.strftime("%B"))  
print(x.strftime("%b"))  
  
print(x.strftime("%Y"))  
print(x.strftime("%y"))  
  
  
print(x.strftime("%w"))  
print(x.strftime("%m"))  
print(x.strftime("%d"))  
  
print("hello")  
print(x-timedelta(days=2))  
print(x-timedelta(hours=2))

Exceptions

try:  
 print(1/0)  
except ZeroDivisionError:  
 print("ZDE ERROR BRO")  
except ArithmeticError:  
 print("AE ERROR BRO")  
except Exception:  
 print("Exception ra mowa")  
  
  
# ValueError Raised when there is a wrong value in a specified data type  
# ZeroDivisionError Raised when the second operator in a division is zero  
# TypeError Raised when two different types are combined  
# SyntaxError Raised when a syntax error occurs  
# RuntimeError Raised when an error occurs that do not belong to any specific expections  
# OverflowError Raised when the result of a numeric calculation is too large  
# NameError Raised when a variable does not exist  
# IndexError Raised when an index of a sequence does not exist  
# IndentationError Raised when indendation is not correct  
# FloatingPointError Raised when a floating point calculation fails  
# EOFError Raised when the input() method hits an "end of file" condition (EOF)  
# ArithmeticError Raised when an error occurs in numeric calculations  
# AssertionError Raised when an assert statement fails  
# AttributeError Raised when attribute reference or assignment fails  
# Exception Base class for all exceptions  
  
  
  
# custom exceptions  
  
class surya(Exception):  
 pass  
  
n="surya"  
  
try:  
 if (n.lower() == "surya"):  
 raise surya  
 else:  
 print("okay da")  
except surya:  
 print("surya exception raised")

dictionaries

# dictionaries  
thisdict = {  
 "brand": "Ford",  
 "electric": False,  
 "year": 1964,  
 "colors": ["red", "white", "blue"]  
}  
  
print(thisdict)  
print(len(thisdict))  
print(type(thisdict))  
print(thisdict["brand"])  
print(thisdict.get("brand"))  
  
  
for x, y in thisdict.items():  
 print(x, y)  
  
  
  
  
  
  
  
  
  
l=thisdict.keys()  
print(l)  
print(list(l))  
m=thisdict.values()  
print(m)  
print(list(m))  
  
# for adding and changing items same way  
thisdict["year"]=2003  
print(thisdict)  
  
thisdict.update({"year":3333})  
print(thisdict)  
  
thisdict.pop("year")  
print(thisdict)  
  
thisdict.popitem()  
print(thisdict)  
  
for x, y in thisdict.items():  
 print(x, y)  
  
  
dict = {  
 "brand": "Ford",  
 "model": "Mustang",  
 "year": 1964  
}  
  
mydict = dict.copy()  
print(mydict)  
print(dict)  
  
thisdict = {  
 "brand": "Ford",  
 "model": "Mustang",  
 "year": 1964  
}  
mydict = dict(thisdict)  
print(mydict)