```
In [1]: print("Hello world")
```

```
Hello world
```

heading1

heading2

heading3

-this is sentence this is bold type this is italic

this is bold and italic

data types in pyt

- numeric type - int ,float complex
- text type- string
- seq type -list , tuple ,range
- maping type -dictionary
- set type - frozen set
- boolean type -bool
- type casting -changing data type (inbuild type)

Operators in python

arthematic operators

- +, - ,, /, %, *,// #### assignment operators
- +=, -=, =, /=, *=, %=, //= #### comparision operators
- ,<, >=, <=, ==, !=

  logical operators
- and , or ,not #### identity operators
- is , is not #### membership operators
- in , not in #### bitwise operators
- & ,|, ^, ~, >>, <<

In [2]:
```
a="20"
a1=34.0
a2=5j
a3=50
print(type(a))
print(type(a1))
print(type(a2))
print(type(a3))
```

```
<class 'str'>
<class 'float'>
<class 'complex'>
<class 'int'>
```

In [3]:
```
b=int(a)
print(b)
print(type(b))
```

```
20
<class 'int'>
```

In [ ]:
```
n=int(input())
n2=input()
print(n)
print(type(n))
print(n2)
print(type(n2))
```

In [ ]:
```
b2=str(a2)
print(b2)
print(type(b2))
```

In [ ]:
```
#
x=100
y=2
print(x+y)
print(x-y)
print(x*y)
print(x/y)
print(x%y)
print(x//y) # floor division
print(x**y) # exponent
```

In [ ]:
```
a1= 500
a2= 3
a1+=a2
print(a1)
a1-=a2
print(a1)
a1*=a2
print(a1)
a1**=a2
print(a1)
a1%=a2
print(a1)
a1//=a2
print(a1)
a1/=a2
print(a1,end=" this is ")
a1=a2
```

In [ ]:
```
x="hai"
y="hello"
z="hai"
x1=45
x2=67
print(x==y)
print(x==z)
print(x!=y)
print(y!=z)
print(x1>x2)
print(x1<x2)
print(x1<=x2)
print(x1>=x2)
```

In [ ]:
```
n1= 100
n2= 300
n3= 500
print(n1<n2 and n2<n3) #true and true
print(n1>n2 and n2<n3) #false and true
print(n1>n2 and n2>n3) #false and false
print(n1<n2 and n2>n3) #true and false
```

In [ ]:
```
n1= 100
n2= 300
n3= 500
print(n1<n2 or n2<n3) #true and true
print(n1>n2 or n2<n3) #false and true
print(n1>n2 or n2>n3) #false and false
print(n1<n2 or n2>n3) #true and false
```

In [ ]:
```
not True
print(not x1>100) #45>100 is false
```

In [ ]:
```
x is z
```

In [ ]:
```
x is not z
```

In [ ]:
```
python_lab= "chay"
"chay" in python_lab
```

In [ ]:
```
python_lab= "chay"
"chay" not in python_lab
```

In [ ]:
```
a1= 1
a2= 2
#4&8  0100  &  1000 0000
print(a1&a2)
```

Date 19.08.22

- ### IF Statements

In [ ]:
```

```

In [ ]:
```
# find the given number is even or not
n=int(input())
if n%2 ==0:
    print(n,"is even")
```

In [ ]:
```
n = int(input())
if n%2!=0:
    print(n,"is odd")
else:
    print(n,"is even")
```

In [ ]:
```
x,y,z=10,9,11
if x<=y and x<=z:
    print(x,"is smaller")
elif y<=x and y<=z:
    print(y,"is smaller")
else:
    print(z,"is smaller")
```

In [ ]:
```
n1=input()
if 'N' in n1:
    if n1 == "NSK":
         print("yes it's my name")
    else:
        print("No its not my name")
else:
    print("Close Program")
```

- ### **Iterators**
  - for loop
  - while loop

Syntax:

for var in range(inclusive range,exclusive range): for var in Datastructure: statements

In [18]:
```
for i in range(20):
    print(i,end=" ")
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

In [17]:
```
for i in range(1,21):
    if i%2==0:
        print(i)
```

```
2
4
6
8
10
12
14
16
18
20
```

In [ ]:
```
for i in range(0,21,3):
    print(i,end=" ")
```

In [ ]:
```
for i in range(20,0,-1): #exclusive range, inclusive range, increment for reve
rse order
    print(i,end=" ")
```

In [ ]:
```
#sum of natural numbers 1 to n
n = int(input())
s=0
for i in range(1,n+1):
    s+=i
print(s)
```

In [ ]:
```
for i in "i am happy":
    if i=="p":
        print(i)
```

- While Syntax:

while condition: { statement 1 statement 2 } can create infinite loop

In [ ]:
```
n=1
while n<=10:
    print(n,end=" ")
    n+=1
```

In [1]:
```
n=10
while n>=1:
    print(n,end=" ")
    n-=1
```

```
10 9 8 7 6 5 4 3 2 1
```

In [ ]:
```
# break in while loop
n=1
while n<=10:
    if n==6:
        break
    print(n,end=" ")
    n+=1
#break is used to come out of the control
```

In [8]:
```
#continue - to skip a particular iteration
n=0
while n<=10:
    n+=1
    if n==6:
        continue
    print(n,end=" ")
```

```
1 2 3 4 5 7 8 9 10 11
```

In [ ]:
```
for i in range(1,20):
    if i>5:
        pass
    print(i,end=" ")

#pass used instead of statement then no errors.pass is a null statement, won't
return anything.
#To define empty class or empty function we use pass keyword
```

In [ ]:
```
# Q.NO 1:
# Problem Statment : Write a Python program to find the distance between teo p
oints (x1,y1) and (x2,y2).
x1=int(input ("enter x1 value : "))
x2=int(input ("enter x2 value : "))
y1=int(input ("enter y1 value : "))
y2=int(input ("enter y2 value : "))
result=((((x2 - x1 )**2) + ((y2-y1)**2))**0.5)
print(result)
```

In [3]:
```
# Q.NO 2:
# Write a Python program to input Percentage. Calculate percentage and grade a
ccording to following:
# Percentage >= 90% : Grade A
# Percentage >= 80% : Grade B
# Percentage >= 70% : Grade C
# Percentage >= 60% : Grade D
# Percentage >= 40% : Grade E
# Percentage < 40% : Grade F
math= eval(input("Enter marks secured in Math: "))
science= eval(input("Enter marks secured in Science: "))
social= eval(input("Enter marks secured in Social: "))
hindi= eval(input("Enter marks secured in Hindi: "))
per = (((math+science+social+hindi)/400)*100)
print("Percentage = ",per)
if per>=90:
    print("Grade A")
elif per>=80:
    print ("Grade B")
elif per>=70:
    print ("Grade C")
elif per>=60:
    print ("Grade D")
elif per>=40:
    print ("Grade E")
else:
    print ("Grade F")
```

```
Enter marks secured in Math: 34
Enter marks secured in Science: 34
Enter marks secured in Social: 23
Enter marks secured in Hindi: 76
Percentage =  41.75
Grade E
```

In [7]:
```
# Q.NO 3:
# Problem Statement: Write a Python program to find maximum between three numb
ers.
a=int(input("Enter value of a :"))
b=int(input("Enter value of b :"))
c=int(input("Enter value of c :"))
if(a>b):
    if(a>c):
        print("A is maximum ",a)
elif b>c:
    print("B is maximum ",b)
else:
    print("C is maximum ",c)
```

```
Enter value of a :45
Enter value of b :34
Enter value of c :23
A is maximum  45
```

In [1]:
```
# Q.NO 4:
# Problem Statement: Write a Python program that computes the real roots of a
quadratic function. Your program
# should begin by prompting the user for the values of a, b and c. Then it sho
uld display a message
# indicating the nature of real roots, along with the values of the real roots
(if any)
a=int(input("Enter value of a :"))
b=int(input("Enter value of b :"))
c=int(input("Enter value of c :"))
nature =(b**2)-(4*a*c)
r1=(-b+(nature)**0.5)/(2*a)
r2=(-b-(nature)**0.5)/(2*a)
if(nature>0):
    print("Roots are unequal and real")
    print("Roots = ",(r1,r2))
elif nature<0:
    print("Roots are imaginary and unreal")
else:
    print("Roots are equal and real")
    r=-b/(2*a)
    print("There is exactly one root",r)
```

```
Enter value of a :5
Enter value of b :3
Enter value of c :6
Roots are imaginary and unreal
```

In [5]:
```
# Q.NO 5:
# Problem Statement: Write a program to input angles of a triangle and check w
hether triangle is valid or not.
# Also,validate the angles entered by the user. (Sum of the three angles of tr
iangle is 180degree)
A = int(input("Enter the angle 'A' of the triangle: "))
B = int(input("Enter the angle 'B' of the triangle: "))
C = int(input("Enter the angle 'C' of the triangle: "))
sum_ofangles = A+B+C
if((sum_ofangles == 180) and ((A and B and C)!=0) and ((A and B and C)<90):
    print(" The Triangle is VALID, since the sum of the angles is : ",sum_ofan
gles)
    print(" It is an Acute angled triangle ")
elif ((sum_ofangles == 180) and ((A and B and C)!=0) and ((A and B and C)==9
0)):
    print(" The Triangle is VALID, since the sum of the angles is : ",sum_ofan
gles)
    print(" It is a Right angled triangle ")
elif((sum_ofangles == 180) and ((A and B and C)!=0) and ((A and B and C)>90)):
    print(" The Triangle is VALID, since the sum of the angles is : ",sum_ofan
gles)
    print(" It is an Obtuse angled triangle ")
else:
    print(" The Triangle is INVALID, since the sum of the angles is : ",sum_of
angles)
```

```
  Input In [5]
    print(" The Triangle is VALID, since the sum of the angles is : ",sum_ofa
ngles)
    ^
SyntaxError: invalid syntax
```

In [10]:
```
# Q.NO 6:
# Problem Statement: Write a program to input basic salary of an employee and
calculate its
# Gross salary according to following:
# Basic Salary <= 10000 : HRA = 20%, DA = 80%
# Basic Salary <= 20000 : HRA = 25%, DA = 90%
# Basic Salary > 20000 : HRA = 30%, DA = 95%
BS = int(input("Enter the Basic Salary : "))
if (BS<=10000):
    HRA = (BS*20)/100
    DA = (BS*80)/100
    GS = BS+HRA+DA
    print("Gross Salary = ",(GS))
elif BS<=20000:
    HRA = (BS*25)/100
    DA = (BS*90)/100
    GS = BS+HRA+DA
    print("Gross Salary = ",(GS))
elif BS>20000:
    HRA = (BS*30)/100
    DA = (BS*95)/100
    GS = BS+HRA+DA
    print("Gross Salary = ",(GS))
```

```
Enter the Basic Salary : 5000
Gross Salary =  10000.0
```

In [18]:
```
print(print(print()))
```

```
None
None
```

In [19]:
```
_=44
print(_)
#dummy variable only in python
```

```
44
```

In [13]:
```
# Q.NO 15
# Write a Python program to print the sum of the series 1/2+1/3+1/4+ ... +1/N.
Where N is
# natural number.

n=int(input("Enter the n value "))
sum=0
for i in range(2,n+1):
    sum = sum+(1/i)
print("Sum of series = ",sum)
```

```
Enter the n value 2
Sum of series =  0.5
```

In [14]:
```
#Q.NO 16
# Write a Python program that prompts user to enter numbers. The process will
repeat until user
# enters 0. Finally, the program prints sum of the numbers entered by the user
n=1
sum=0

while (n!=0):
    n=int(input("Enter the value of n "))
    sum=sum+n
print("Sum = ",sum)
```

```
Enter the value of n 8
Enter the value of n 6
Enter the value of n 10
Enter the value of n 45
Enter the value of n 0
Sum =  69
```

In [20]:
```
#Q.NO 17
# Write a Python program to print all the numbers from 1 to 1000 that are not
divisible by
# 2, 3, 5, 7, 11, 13, 17 and 19.
n=int(input("Enter the value of n "))
for i in range (1,n+1):
    if(i%2!=0 and i%3!=0 and i%5!=0 and i%7!=0 and i%11!=0 and i%13!=0 and i%1
7!=0 and i%19!=0):
        print(i,end=" ")
```

```
Enter the value of n 50
1 23 29 31 37 41 43 47
```

In [25]:
```
# Q.No 18
# Write a Python program to find HCF (GCD) of two numbers.

n1=int(input("Enter first number : "))
n2=int(input("Enter second number : "))
if n1>n2:
    small_num=n2
else:
    small_num=n1
for i in range(1,small_num+1):
    if n1%i==0 and n2%i==0:
        HCF = i
print("The HCF of ",n1, "and", n2, "is :",HCF)
```

```
Enter first number : 60
Enter second number : 45
The HCF of  60 and 45 is : 15
```

In [28]:
```
#Q.No 19
# Write a Python program to check whether a number is Armstrong number or not.

num=int(input("Enter number : "))
sum=0
n=num
while(n!=0):
    i=n%10
    sum+=(i*i*i) # sum+=i**3
    n=n//10 # num//=10
if sum==num:
    print(" ",num,"Is a Armstrong number"  )
else:
    print(" ",num,"Is not a Armstrong number"  )
```

```
Enter number : 666
  666 Is not a Armstrong number
```

In [30]:
```
# Q.NO 20
# Write a Python program to swap first and last digits of a number.

num=int(input("Enter number : "))
n=num
count=0
while(n>0):
    n//=10 # floor division
    count=count+1
count=count-1
last=num%10
first=num//(10**count)
print(first,last)
new=(num//10)*10+first # (2001//10)*10+first
print(new)
rev=(new%(10**count))+(last*(10**count))#(2002%(10**3))+(last*(10**3))
print(rev)
```

```
Enter number : 98435
9 5
98439
58439
```

In [31]:
```
# Q.NO 21
# Write a Python program for printing prime numbers up to N. (N>100).
n=int(input("Enter n : "))
for i in range (1,n+1):
    count=0
    for j in range (1,i+1):
        if(i%j==0):
            count+=1
    if(count==2):
        print(i,end=" ")
```

```
Enter n : 100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

In [44]:
```
# Q.NO 22
# Write a Python program to construct the following pattern, using a nested fo
r loop.
# *
# * *
# * * *
# * * * *
# * * * * *
# * * * *
# * * *
# * *
# *

for i in range(1,7):
    for j in range(1,i):
        print("*",end=" ")
    print()
for i in range(5,0,-1):
    for j in range(1,i):
        print("*",end=" ")
    print()
```

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

In [46]:
```
# Q.NO 23
# Write a Python program to print following matrix.
# 1 0 1 0
# 0 1 0 1
# 1 0 1 0
# 0 1 0 1
for i in range(1,5):
    for j in range(1,5):
        if(i+j)%2==0:
            print("1\t",end="")
        else:
            print("0\t",end="")
    print()
```

```
1       0       1       0
0       1       0       1
1       0       1       0
0       1       0       1
```

In [47]:
```
# write factorial of given number
n=int(input("Enter a number : "))
fact=1;
for i in range (1,n+1):
    fact=fact*i
print(fact)
```

```
Enter a number : 5
120
```

Functions in python

- inbuilt functions
- user defined functions
- block of code that will execute when it is called
- functions with return values and without arguments
- functions without return values and with arguments

In [ ]:
```
- To define a function we need to use def keyword
# syntax: def Funcname(a,b,c): - formal parameters, variables
#            statement 1
#            statement 2
#            return output
# fucntioncalling : Funcname(1,2,3) - arguments, value to parameters
```

In [7]:
```
# define a function to print given number is even or not- this function have a
rguments but no return values
def iseven(n):
    if n%2==0:
        print(n,"is even")
    else:
        print(n,"is not even")
# n=int(input("Enter num: "))
# iseven(n)
```

```
Enter num: 1
1 is not even
```

In [3]:
```
# defining a function without any arguments
def wish():
    print("hello everyone. Good morning")
wish()
```

```
hello everyone. Good morning
```

In [11]:
```
#fucntion is called keyword argument, without value also it will run.
def wish(name=None):
    if name!=None:
        print("Hello",name,",Good Morning")
    else:
        print("Hello everyone. Good morning")
wish("Chimmy")
```

```
Hello Chimmy ,Good Morning
```

3 types of arguments

- arbitary argument
- keyword argument
- arbitary keyword argument

In [17]:
```
# arbitary argument
# *k is used to access the element
def fruits(*k):
    print("Second tastiest fruit is",k[2])
fruits("pineapple","apple","mango","banana")
```

```
Second tastiest fruit is mango
```

In [22]:
```
#keyword arguments
def add(a,b,c):
    return a+b+c
add(a=3,b=5,c=7) # u can also write add(3,b=5,c=7) default argument
```

Out[22]: 15

In [24]:
```
#arbitary keyword arguments, no need to declare the variables before, by apply
ing we can apply required no.of arguments
def multiply(**values):
    return values["x"]*values["y"]*values["z"]
multiply(x=5,y=6,z=10)
```

Out[24]: 300

In [27]:
```
# write a function to find the given number is prime number or not
def isprime(n):
    count=0
    for i in range(2,n):
        if n%i==0:
            count+=1
    if count==0:
        return True
    else:
        return False

isprime(16)
```

Out[27]:
```
False
```

In [12]:
```
def isprime(n):
    count=0
    for i in range(2,n):
        if n%i==0:
            count+=1
    if count==0:
        return True
    else:
        return False
def primesin(a,b):
    for j in range(a,b):
        if isprime(j):
            print(j,end=" ")
primesin(a=1,b=100)
```

```
1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

In [5]:
```
# Q.NO 24
# Define a function to find sum of all odd numbers between 1 to n.
def odd(n):
    sum=0
   # n=int(input("Enter the value of n:"))
    for i in range (n):
        if(i%2!=0):
            sum=sum+i
    print(" ",sum)
n=int(input("Enter the value of n:"))
odd(n)
```

```
Enter the value of n:9
  16
```

In [7]:
```
# Q.NO 25
# Define a function to check whether a number is palindrome or not.

def palindrome(n):
    temp=n
    digit=0
    while temp>0:
        digit=digit*10+temp%10
        #r=temp%10
        temp=temp//10
    if digit==n:
        print(" ",n,"is a palindrome")
    else:
        print(" ",n,"is not a palindrome")
palindrome(373)
palindrome(123)
```

```
 373 is a palindrome
 123 is not a palindrome
```

In [8]:
```
#Q.NO 26
# Define a function to calculate the area of a circle using the formula.
def area_of_circle(r):
    area = 3.14*r*r
    return "the area of circle with radius {} is {}".format(r,area)
r=int(input("Enter radius : "))
area_of_circle(r)
```

```
Enter radius : 4
```

Out[8]:
```
'the area of circle with radius 4 is 50.24'
```

In [10]:
```
## Q.NO 27
# Define a function to check whether number is perfect or not
def perfect(n):
    sum=0
    for i in range(1,n):
        if(n%i==0):
            sum=sum+i
    if(sum==n):
        print(n,"is perfect")
    else:
        print(n,"is not perfect")

n=int(input("Enter the number : "))
perfect(n)
```

```
Enter the number : 2
2 is not perfect
```

In [14]:
```
# Q.NO 28
# Define a function to print multiplication table of any number.
def multi(n):
    for i in range(1,11):
        print(n,"X",i,"=",n*i)
n=int(input("Enter number : "))
multi(n)
```

```
Enter number : 3
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
```

In [17]:
```
# Q.NO 29
# Define a function to print table of a number. Using this function display ta
ble of numbers from
# 1 to 10.
def table(n):
    for j in range(1,n+1):
        print("The multiplication table of ",j,"is")
        multi(j)
table(2)
```

```
The multiplication table of  1 is
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
1 X 10 = 10
The multiplication table of  2 is
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20
```

In [19]:
```
# Q.NO 30
# Define a recursive function to find power of a number.
def power(base,exp):
    if exp==0:
        return 1
    else:
        return pow(base,exp)
        #power(base,exp-1)*base
power(2,2)
```

Out[19]: 9

In [15]:
```
# Q.NO 31
# Define a recursive function count number of digits in a number.
def digitcount(n):
    if n<=0:
        return 0
    else:
        count = digitcount(n//10)+1
    return count
digitcount(234567)
```

Out[15]: 6

In [22]:
```
#Q.NO 32
# Write a recursive function to find the sum of 1^5 + 2^5 + ……..+n^5
def sum_of_series(n):
    if n==0:
        return 0
    else:
        return sum_of_series(n-1)+n**5
print("Sum of series = ")
sum_of_series(2)
```

```
Sum of series =
```

Out[22]: 33

In [26]:
```
#Q.NO 33
# Write a python program to find the factorial value of a number using recursi
on.
def find_factorial(n):
    fac=1
    if n==1:
        return 1
    else:
        return n*find_factorial(n-1)
print("Factorial = ",find_factorial(5))
#find_factorial(5)
```

```
Factorial =  120
```

In [29]:
```
#Q.NO 34
# Write a python program to implement Tower of Hanoi using recursive function.
def Towers_of_Hanoi(n,source,auxiliary,target):
    if (n==1):
        print("Move Disk 1 from ",source,"to ",target)
    else:

        Towers_of_Hanoi(n-1,source,auxiliary,target)
        print ("Move Disk",n,"from ",source,"to ",auxiliary)
        Towers_of_Hanoi(n-1,target,auxiliary,source)

n=int(input("Enter number of disks : "))

Towers_of_Hanoi(n,'A','B','C')
```

```
Enter number of disks : 3
Move Disk 1 from  A to  C
Move Disk 2 from  A to  B
Move Disk 1 from  C to  A
Move Disk 3 from  A to  B
Move Disk 1 from  C to  A
Move Disk 2 from  C to  B
Move Disk 1 from  A to  C
```

In [9]:
```
#35.
# Write function for finding factors (n) and use factors function to check whe
ther given number
# n is prime or not.
def factors(n):
    for i in range (1,n+1):
        if n%i==0:
            yield
def isprime(n1):
    count=0
    for i in factors(n1):
        count+=1
    if count>2:
        print(n1,"is not prime")
    else:
        print(n1,"is prime")
isprime(556)
```

```
556 is not prime
```

In [18]:
```
# 36. Write a python program for printing Fibonacci series
# a. Write recursive approach implementation
def fibonacci(n):
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return(fibonacci(n-2)+fibonacci(n-1))
for i in range(int(input())):
    print(fibonacci(i),end=" ")
```

```
8
0 1 1 2 3 5 8 13
```

In [13]:
```
# b. Write Iterative Implementation
nterms = int(input("Enter number of terms: "))
n1, n2 = 0, 1
count = 0

if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)
       nth = n1 + n2
       n1 = n2
       n2 = nth
       count += 1
```

```
Enter number of terms: 8
Fibonacci sequence:
0
1
1
2
3
5
8
13
```

Strings in Python

static declaration

my_string = " " string = ' ' ' Hello ' ' ' string = " " " Hello" " "

for dynamic we can use int keyword

in order to access the characters in the string, we can use indexing like a[0],a[1],

there are 2 types of indexing - positive indexing starts from 0,a[0]

negative indexing starts from -1, a[-1], to access the characters from reverse.

we can get bunch of characters by slicing , str[1:5], str[5:-2] [starting char : last character]

str[1:5] - slicing 2nd to 5th character str[5,:-2] - slicing 6th to 2nd character

str = programiz

str[0]=p

str[-1]=z

str[1:5]=rogr

str[5:-2]=am

to reverse the string >>str[::-1]

string concatenation

str1+str2 = HelloWorld

str1*3 = string will be printed 3 times = Hello Hello Hello (string multiplied by number will print multiple times)

methods of string - print(dir(str))

in order to know abt any method of the string use help function - help(str.method name)

- casefold() - to chnage uppers case to lower case and vice versa
- center()-pads string with specified character
- count()- which characters repeats how many times
- encode()- returns encoded string

- endswith() - string ends eith ehich character
- find() - returns index
- index() - tells whichc char in which index
- isalpha()- checks if all chars are alpha
- is num () - checks if all chars are nums
- isdecimal() -checks if all chars are decimals
- split,lower,join,upper, #### We can print emojis in python #### str ="\U0001F917" - change the nubers from F for different emojis #### print(str)

In [5]:
```
print(dir(str))
## the words without underscore are methods of python
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getne
wargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__',
'__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__
reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr_
_', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'c
enter', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'forma
t_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'is
identifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'is
upper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'remove
prefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition',
'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase',
'title', 'translate', 'upper', 'zfill']
```

In [64]:
```
help(str.split)
str1 ="python"
str1.split()
```

```
Help on built-in function split:

split(sep=None, maxsplit=-1) method of builtins.str instance
    Return a list of the words in the string, using sep as the delimiter stri
ng.

    sep
      The delimiter according which to split the string.
      None (the default value) means split according to any whitespace,
      and discard empty strings from the result.
    maxsplit
      Maximum number of splits to do.
      -1 (the default value) means no limit.
```

Out[64]:
```
['python']
```

In [50]:
```
#str = "PYTHOn hie"
#str.casefold() #casefold - to change upper to lower only
#str.swapcase() # swapcase - to change upper to lower and viceversa
#str.title() # title - changes the first char of every word to upper case
#str.capitalize() #- changes the first char to upper case of only the first wo
rd
#str.center(1) # - takes on parameter and levaes space
#str.count() # - will tell the no.of occurences in a word.
#str ="\U0001F917"
#print(str)
str =" wert to " # checks for upper and returns true if upper
#str.isupper()
#str.istitle() # returns true if first char of word is capital
#str.split(" ")
#str.join("no")
#str.isascii()
#str.isupper()
#str.isidentifier() #Return True if the string is a valid Python identifier, F
alse otherwise.
#str.isalpha() #Return True if the string is an alphabetic string, False other
wise.
#str.isspace() #Return True if the string is a whitespace string, False otherw
ise
#str.isprintable() #Return True if the string is printable, False otherwise.
#str.strip() #Return a copy of the string with leading and trailing whitespace
removed.
#str.index("e") #returns the index of the characters of the string
#print(" ",str*3) #prints 3 times
```

```
   wert to  wert to  wert to
```

In [4]:
```
list=[2,'apple',3,99,'sarayu']
list.append(35)
print(list)
```

```
[2, 'apple', 3, 99, 'sarayu', 35]
```

In [83]:
```
test ="\U0001F554"
test3 ="ikr, but we are I'm having fun with emojis"
print(test+test3)
```

```
🕔ikr, but we are having fun with emojis
```

In [56]:
```
#### Positive Indexing - Exclusive get characters from postion 2 to position 5
b = "Hello, World!"
print(b[2:6])
print(b[:5]) # (missed the strating) by default it starts from index 0 to posi
ton 5
print(b[2:]) # (missed the last) by default it will print until last
```

```
llo,
Hello
llo, World!
```

In [63]:
```
#### Negative Indexing - inclusive
b ="Hello, World!"
print(b[-5:-2])
print(b[::-1]) # to reverse string (-1 then ending to starting)
print(b[::-2]) # reversing and printing alternatively
print(b[::-3]) # reversing and skips 2 characters.
```

```
orl
!dlroW ,olleH
!lo olH
!r lH
```

In [67]:
```
# "::" so prints in reverse
for i in range(10,0,-1):
    print(i,end=" ")
str1 = "Python"
print("\n")
print(str1[len(str)::-1])
print(str1[::]) # no numbers mentioned so prints whole string.
print(str1[len(str1):0:-1])
```

```
10 9 8 7 6 5 4 3 2 1

nohtyP
Python
nohty
```

In [86]:
```
# Q.NO 54
# Write a program that counts up the number of vowels contained in the string
S. Valid vowels
# are: 'a', 'e', 'i', 'o', and 'u'. For example, if s = 'azcbobobegghakl', you
r program should print:
# number of vowels 5
s = 'azcbobobegghakl'
vowels ="aeiouAEIOU"
for i in s:
    if i in vowels:
        count+=1
print("No.of Vowels in given string : ",count)
```

```
No.of Vowels in given string :  10
```

In [6]:
```
# Q.NO 55
# Assume s is a string of lower-case characters. Write a program that prints t
he number of times
# the string 'bob' occurs in s. For example, if s = 'azcbobobegghakl', then yo
ur program should
# print Number of times bob occurs is 2.
s=input("Enter any string: ")
s1=s.lower()
bob=0
for i in range(len(s1)):
    if(s1[i:i+3]=='bob'):
        bob=bob+1
print("The string you've entered is: ",s1.lower())
print("The number of times 'bob' has occured in",s1,"is: ",bob)
```

```
Enter any string: bobby
The string you've entered is:  bobby
The number of times 'bob' has occured in bobby is:  1
```

In [9]:
```
# Q.NO 56
# Write a Python program that finds whether a given character is present in a
string or not. In
# case if it is present then it prints the index at which it is present. Do no
t use built-in find
# functions to search the character.
def findme(inputstring,s):
    for i in range(len(inputstring)):
        if inputstring[i]==s:
            break
    print("The index at which the given character is present is: ",i)
inputstring=input("Enter the string: ")
s= input("Enter the character to find: ")
findme(inputstring,s)
```

```
Enter the string: sarayu
Enter the character to find: a
The index at which the given character is present is:  1
```

In [14]:
```
# Q.NO 57
# Write a Python program that counts the occurrence of a character in a strin
g.
# Do not use built-in function.
string=input("Enter input string: ")
char=input("Enter character to count: ")
count=0
for i in string:
    if i==char:
        count+=1
print("The occurrence of a character in a string is: ",count)
```

```
Enter input string: guggaga
Enter character to count: g
The occurrence of a character in a string is:  4
```

In [2]:
```
# Q.NO 58
# Write a python program for following:
# a. Take a input string with spaces, split it into list of words
# b. From the list of words, create dictionary with keys (only unique words) a
nd values
# (length of the word)

s=input()
wl=s.split()
dic={}
for i in wl:
    if i not in dic:
        dic[i]=len(i)
dic
```

```
i am not cloud because i am not kavya
```

Out[2]:
```
{'i': 1, 'am': 2, 'not': 3, 'cloud': 5, 'because': 7, 'kavya': 5}
```

In [2]:
```
# Q.NO 59
# Write a python program to count number of vowels, spaces and to find longest
word in a given
# input string. (Take input string with spaces).
stringinput =input("Enter input string: ")
vowels ="aeiouAEIOU"
vcount=0
scount=0
for j in stringinput:
    if j in vowels:
        vcount+=1
    elif j.isspace():
        scount+=1
print(vcount,scount)
w1=stringinput.split()
max1=w1[1]
for i in range(1)
```

```
  Input In [2]
    for i in range(1)
                     ^
SyntaxError: invalid syntax
```

LISTS in Python

List is sequence of, {heterogenous data}. if data is homogenous then it's a array.

list will allow different types of elements in in.

List is mutable(we can chnage elements depending on position, append and delete)

In [5]:
```
#### Two ways to declare a list
l1=[]
l2=list()
print(type(l1))
print(type(l2))
```

```
<class 'list'>
<class 'list'>
```

In [3]:
```
a=[12,23.5,'u',45,9.34,"hello"]
for i in a:
    print(i)
b=iter(a) # function
```

```
12
23.5
u
45
9.34
hello
```

print(next(b)) #calling the function

keep on running it, it will show the elements one by on of the list, when the elements are over

then it shows stopiteration

In [19]:
```
a[0]
a[-1]
```

Out[19]:
```
'hello'
```

In [20]:
```
a[0:3] # slicing the list
print(a[::-1]) # reverse items
```

```
['hello', 9.34, 45, 'u', 23.5, 12]
```

In [21]:
```
#print(dir(list)) # just like methods in string, we have methods in list
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
'__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__get
attribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '_
_init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__
mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__re
versed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'i
nsert', 'pop', 'remove', 'reverse', 'sort']
```

In [41]:
```
#a=[12,23.5,'u',45,9.34,"hello"]
a.append('9.34') # to add elements at end of the list
a
print(a)
```

```
[12, 23.5, 'u', 45, 9.34, 'hello', 12, 12, 'u', 'u', '9.34']
```

In [16]:
```
print(a)
```

```
[12, 23.5, 'u', 45, 9.34, 'hello', 9, 9, 9, 9, 9, [...], 'a', 'hope']
```

In [30]:
```
#a.clear() # clears the elements of the list
#a.copy() # returns the copy of the list
```

Out[30]:
```
[]
```

In [68]:
```
help(list.sort)
```

```
Help on method_descriptor:

sort(self, /, *, key=None, reverse=False)
    Sort the list in ascending order and return None.

    The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
    order of two equal elements is maintained).

    If a key function is given, apply it once to each list item and sort the
m,
    ascending or descending, according to their function values.

    The reverse flag can be set to sort in descending order.
```

In [35]:
```
a=[12,23.5,'u',45,9.34,"hello",12,12]
a.count(12) # returns the no.of times an element repeats
```

Out[35]:
```
3
```

In [49]:
```
a=[12,23.5,'u',45,9.34,"hello",12]
a.extend('u')
```

In [50]:
```
print(a)
```

```
[12, 23.5, 'u', 45, 9.34, 'hello', 12, 'u']
```

In [52]:
```
a.index(23.5) #prints the index of the elements in the list
```

Out[52]:
```
1
```

In [55]:
```
a.insert(0,10)  #inserts elements in front of the index specified
```

In [56]:
```
print(a)
```

```
[10, 12, 23.5, 'u', 45, 9.34, 'hello', 12, 'u']
```

In [61]:
```
a.pop()# pops out the last element
```

Out[61]: 12

In [62]:
```
print(a)
```

```
[10, 12, 23.5, 'u', 45, 9.34, 'hello']
```

In [64]:
```
a.remove(45)# removes any elements which is specified from the list
```

In [65]:
```
print(a)
```

```
[10, 12, 23.5, 'u', 9.34, 'hello']
```

In [66]:
```
a.reverse()# reverses the list
```

In [67]:
```
print(a)
```

```
['hello', 9.34, 'u', 23.5, 12, 10]
```

In [70]:
```
a=[12,23,45,9,34,12] # only applicable to integers and sorts the elements in a
scending order
a.sort()
```

In [71]:
```
print(a)
```

```
[9, 12, 12, 23, 34, 45]
```

Dictionaries in Python

Python dictionary is an ordered collection of items.

Each item of a dictionary has a key and value pair

Dictionaries are optimized to retrieve values when the key is known

Also a mutable sequence type

In [76]:
```
# creating empty dictionary
d={} # method 1
print(type(d))
d1=dict() # method 2
print(type(d1))
```

```
<class 'dict'>
<class 'dict'>
```

In [81]:
```
d1={"a":1,"b":2, "c":[1,2,3]}
print(d1)
```

```
{'a': 1, 'b': 2, 'c': [1, 2, 3]}
```

In [82]:
```
d1["a"] # calling element in list, return value of it
d1["c"]
```

Out[82]:
```
[1, 2, 3]
```

In [85]:
```
d1["d"]="apple" # to add a new key and value pair
print(d1)
```

```
{'a': 1, 'b': 2, 'c': [1, 2, 3], 'd': 'apple'}
```

In [86]:
```
d1["c"]="harry" # assigning new value of key, it overwrites the old key value
print(d1)
```

```
{'a': 1, 'b': 2, 'c': 'harry', 'd': 'apple'}
```

In [89]:
```
a= zip([1,2,3],[5,6,7])  # 2 data structures, but should have same number of i
ndexes
for i in a: #loop, then returns tuples
    print(i)
```

```
(1, 5)
(2, 6)
(3, 7)
```

In [91]:
```
dic={}
a= zip([1,2,3],[5,6,7])
for i in a:
    dic[i[0]]=i[1]
dic
```

Out[91]:
```
{1: 5, 2: 6, 3: 7}
```

In [24]:
```
Dict = {1:'Geeks', 2: 'For', 3:{'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}
print(Dict[3]['B']) # nested dictionary
```

```
To
```

In [7]:
```
n_l=[1,[3,4,5],6,9]
n_l[1][0] #nested list
n_l[1][-1] #last element of list
```

Out[7]: 3

In [23]:
```
n=[76,90,54,3,2]
max1=n[0]
for i in n:
    if(max1<i):
        max1 = i
    else:
        pass
print(max1)
```

```
90
```

Tuples

Tuple is used to store multiple items in a single variable

We can use indexing to retrieve an element

Inmutable - can't change the element once it's assigned

Ordered collection of data within the parentheses

Can convert tuple to list, list to tuple

Compared to list, tuple is faster

In [5]:
```
# Creating a tuple
t=()
t1=tuple()
print(type(t))
print(type(t1))
```

```
<class 'tuple'>
<class 'tuple'>
```

In [8]:
```
l =[1,2,3] #converting list to tuple
l=tuple(l)
type(l)
```

Out[8]: tuple

In [9]:
```
l[0] # accessing element using indexing
```

Out[9]: 1

In [10]:
```
l[-1] #accesing the last element
```

Out[10]: 3

In [17]:
```
#n_l=[1,[3,4,5],6,9]
n_l[1]=23
print(n_l)
#l[1]=23 # cannot change
```

```
[1, 23, 6, 9]
```

In [16]:
```
print(dir(tuple))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_su
bclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '_
_new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr_
_', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

In [47]:
```
#help(tuple.count)
n_l=[1,1,3,4,5,6,1] # returns the number of occurences of the element
n_l.count(1)
```

Out[47]: 3

In [42]:
```
help(tuple.count)
```

```
Help on method_descriptor:

count(self, value, /)
    Return number of occurrences of value.
```

In [40]:
```
l =[1,2,3] # returns the index of the element
t1=tuple(l)
t1.index(2)
```

Out[40]: 1

In [ ]:
```
# 42. Write a Python program to create a list of each digit is a element in a
list from a number.
# Example: Input: 5467, Output: [5,4,6,7]
```

SETS in Python

Sets are a collection of data. Represented in { }

Sets allows heterogenous data

Sets are unordered and unindexed

Mutable

Sets are followed by basic data structure called hash table

Sets doesnot allow duplicate data

In [1]:
```
# Creating a set
s=set()
print(type(s))
```

```
<class 'set'>
```

In [5]:
```
set1 ={1,2,3,4,5}
print(f"The type of {set1} is {type(set1)}")
set2 ={3.4, 'python'}
print(f"The type of {set2} is {type(set2)}")
```

```
The type of {1, 2, 3, 4, 5} is <class 'set'>
The type of {'python', 3.4} is <class 'set'>
```

In [3]:
```
#doesnot allow duplicates
set_1 = {1,3,'a',4,'a',6,8} # Repeated items will print only once
print(set_1)
```

```
{1, 3, 'a', 4, 6, 8}
```

In [6]:
```
#unhashable elements - error
set1={1,[3,'b',5.0],6.7,'t'}
set2={4.6,{5,6,'t'},7}
set3={3.2,{1:'s',2:'r'},7,'y'}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [6], in <cell line: 2>()
      1 #unhashable elements
----> 2 set1={1,[3,'b',5.0],6.7,'t'}
      3 set2={4.6,{5,6,'t'},7}
      4 set3={3.2,{1:'s',2:'r'},7,'y'}

TypeError: unhashable type: 'list'
```

In [7]:
```
#creating a set with tuple
set4={1,3.6,(4,'t',7.8)}
set4
```

Out[7]:
```
{(4, 't', 7.8), 1, 3.6}
```

In [8]:
```
#creating an empty set
var={}
print(f"The type of {var} is {type(var)}")
```

```
The type of {} is <class 'dict'>
```

In [9]:
```
#creating set with set function
set1=set()
print(f"The type of {set1} is {type(set1)}")
bool(set1)
```

```
The type of set() is <class 'set'>
```

Out[9]:
```
False
```

In [10]:
```
#example of creating set with set function
set1=set("PythonGeeks") #creating a set from a string
print(set1)
set2=set([1,2,3,4,1,2]) #creating a set from a list
set2
set3=set(('a','b',4,5.6))#creating a set from a tuple
set3
```

```
{'e', 'n', 'G', 'o', 'h', 'y', 'P', 't', 's', 'k'}
```

Out[10]:
```
{4, 5.6, 'a', 'b'}
```

In [11]:
```
# creatinf set with string
set1={'abab'}
set1
set2=set('abab')
set2
```

Out[11]:
```
{'a', 'b'}
```

In [14]:
```
#accessing the entire set
set1={1,2,'r','y',6.9}
set1

#set2={1,4,2,1,3,2}
#set2
```

Out[14]:
```
{1, 2, 6.9, 'r', 'y'}
```

In [15]:
```
#accessing element of set using indexing
set1={5.6,'g',8,4,'k'} # error- indexing is not allowed
set1[4]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [15], in <cell line: 3>()
      1 #accessing element using index
      2 set1={5.6,'g',8,4,'k'}
----> 3 set1[4]

TypeError: 'set' object is not subscriptable
```

In [1]:
```
# Q.42
# Write a Python program to create a list of each digit is a element in a list
from a number.
# Example: Input: 5467, Output: [5,4,6,7]

s=input()
l=[]
for i in s:
    l.append(int(i))
print("The list of digits from a given :",l)
```

```
5467
The list of digits from a given : [5, 4, 6, 7]
```

In [3]:
```
# Q.NO 43
# Write a Python program to form a number from a given list of digits Example:
Input:
# [5, 4, 6, 7], Output: 5467
n=list(map(int,input("Enter numbers :").split()))
print(n)
temp=0
for i in n:
    temp=(temp*10+i)
print("number from a given list of digits :",temp)
```

```
Enter numbers :5467
[5467]
number from a given list of digits : 5467
```

In [4]:
```
# Q.NO 44
# Write a Python program to find the second smallest number and second largest
in a list.
l=[10,20,30,40,50]
l.sort()
print("""The second smallest number and second largest in a list are :""",l
[1],l[-2])
```

```
The second smallest number and second largest in a list are : 20 40
```

In [5]:
```
# Q.N0 45
# Write a python program to create dictionary of index is the key and correspo
nding
# prime number as value up to 100. Output: {1:2, 2:3, 3:5, 4:7, 5:11, 6:13, 7:
17, 8:19
# ....... and soon }
d={}
n=0
i=2
while n<100:
    count=0
    for j in range(1,i):
        if i%j==0:
            count+=1
    if count==1:
        n+=1
        d[n]=i
    i=i+1
print(d)
```

```
{1: 2, 2: 3, 3: 5, 4: 7, 5: 11, 6: 13, 7: 17, 8: 19, 9: 23, 10: 29, 11: 31, 1
2: 37, 13: 41, 14: 43, 15: 47, 16: 53, 17: 59, 18: 61, 19: 67, 20: 71, 21: 7
3, 22: 79, 23: 83, 24: 89, 25: 97, 26: 101, 27: 103, 28: 107, 29: 109, 30: 11
3, 31: 127, 32: 131, 33: 137, 34: 139, 35: 149, 36: 151, 37: 157, 38: 163, 3
9: 167, 40: 173, 41: 179, 42: 181, 43: 191, 44: 193, 45: 197, 46: 199, 47: 21
1, 48: 223, 49: 227, 50: 229, 51: 233, 52: 239, 53: 241, 54: 251, 55: 257, 5
6: 263, 57: 269, 58: 271, 59: 277, 60: 281, 61: 283, 62: 293, 63: 307, 64: 31
1, 65: 313, 66: 317, 67: 331, 68: 337, 69: 347, 70: 349, 71: 353, 72: 359, 7
3: 367, 74: 373, 75: 379, 76: 383, 77: 389, 78: 397, 79: 401, 80: 409, 81: 41
9, 82: 421, 83: 431, 84: 433, 85: 439, 86: 443, 87: 449, 88: 457, 89: 461, 9
0: 463, 91: 467, 92: 479, 93: 487, 94: 491, 95: 499, 96: 503, 97: 509, 98: 52
1, 99: 523, 100: 541}
```

In [6]:
```
# Q.NO 46
# Write a Python program to find the smallest value and largest value in a dic
tionary.
D1={1:200,2:3000,3:100,5:20}
l=list(D1.values())
l.sort()
print(f"The smallest value and the largest value in a dictionary are :{l[0]},
{l[-1]}")
```

```
The smallest value and the largest value in a dictionary are :20, 3000
```

In [8]:
```
# Q.NO 47
# Write a Python script to generate and print a dictionary that contains a num
ber (between
# 1 and n) in the form (x, x*x).
# Sample Dictionary ( n = 5) :
# Expected Output : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
n=int(input("n = "))
d={}
for i in range(1,n+1):
    d[i]=i*i
print(d)
```

```
n = 5
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [9]:
```
# Q.NO 48
# Write a Python program to convert a list of characters into a string. Exampl
e: Input:
# ['s','t','r','i','n','g'], Output: string.
strlist=list(input().split())
print(strlist)
s="".join(strlist)
print(s)
```

```
s t r i n h
['s', 't', 'r', 'i', 'n', 'h']
strinh
```

In [10]:
```
# Q.NO 49
# Write a Python program to combine two dictionary adding values for common ke
ys.
# d1 = {'a': 10, 'b': 20, 'c':30}
# d2 = {'a': 30, 'b': 20, 'd':40}
# Sample output: {'a': 40, 'b': 40, 'd': 40, 'c': 30}
d1 = {'a': 10, 'b': 20, 'c': 30}
d2 = {'a': 30, 'b': 20, 'd': 40}
d={}
for i,j in zip(d1.keys(),d2.keys()):
    if i in d2:
        if i not in d:
            d[i]=d1[i]+d2[i]
    else:
        d[i]=d1[i]
    if j in d1:
        if j not in d:
            d[j]=d1[j]+d2[j]
    else:
        d[j]=d2[j]
d
```

Out[10]:
```
{'a': 40, 'b': 40, 'c': 30, 'd': 40}
```

In [18]:
```
# Q.NO 50
# Write a program to print index at which a particular value exists. If the va
lue exists a
# multiple location in the list, then print all the indices. Also, count the n
umber of times
# the value is repeated in the list.
in_list=list(map(int,input().split()))
ele=int(input())
count=0
for i in range (len(in_list)):
    if in_list[i]==ele:
        count+=1
        print("Given Value is present at index :")
        print(i)
print("Count of given element :",count)
```

```
123
2
Count of given element : 0
```

In [17]:
```
# Q.NO 51
# Write a program to remove all duplicate elements in a list.
l=list(map(int,input().split()))
print(l)
u=[]
for i in l:
    if i not in u:
        u.append(i)
print("unique list :",u)
```

```
1454
[1454]
unique list : [1454]
```

In [15]:
```
# Q.NO 52
# Write a program to create a list of numbers in the range 1 to 10. Then delet
e all the odd
# numbers from the list and print the final list.
l=[]
for i in range(1,11):
    l.append(i)
for i in l:
    if i%2!=0:
        i=l.remove(i)
print(l)
```

```
[2, 4, 6, 8, 10]
```

File Handling

- open
- do operations in that file
- close #### Methods used
- open() - have to close using close() method
- with open() - no need to use close method,
- read() - to read the data in a file
- write() - write the data in a file
- close() - to close the opened file
- append() - to append the data in already existing file
- readlines() - #### Modes
- read mode - 'r'
- write mode - 'w'
- append mode - 'a'

In [10]:
```
f = open("Text file 10-11-22") # f is a variable to store the opening of file
fh=f.read() # fh to store data in file
print(fh)
f.close()
```

```
Hello, this is the first text file in python lab
So, I need to fill in up some data in the text file
Therefore I am typing this shit all over here dot
```

In [21]:
```
with open("Text file 10-11-22","r") as f: # if we used with open, we also need to mention the mode
    fdata = f.read(500) # we passed how many digits it needs to read
    print(fdata)
```

```
Now I am using write methodI am using the append now
```

In [18]:
```
with open("Text file 10-11-22","w") as f: # it over writes the data in the file.
    fdata = f.write("Now I am using write method")
    print(fdata)
```

```
27
```

In [22]:
```
with open("Text file 10-11-22","a") as f:
    fdata = f.write("\nI am using the append now")
    print(fdata)
```

```
26
```

In [27]:
```
with open("Text file 10-11-22","r") as r: # it reads the data, line by line
    data= r.readlines()
    for i in data:
        print(i)
```

Now I am using write method

I am using the append now

In [37]:
```
# 37. Write a Python program to copy the content of one file to other file.
f= open("Q.NO. 37")
fr= f.read()
with open("Q.NO 37 (2)","w") as f:
    fdata = f.write(fr)
print(fdata)
f.close()
```

55

In [36]:
```
# 38. Write a Python program to count the number of words in the above txt fil
e.
count=0
with open("Q.NO 37 (2)","r") as f:
    for line in f:
        words = line.split()
        count+=len(words)
    print("number of words:")
    print(count)
```

number of words:
11

In [38]:
```
# 39. Write a Python program to number of characters without space in the abov
e txt file.
count=0
with open("Q.NO 37 (2)","r") as f:
    fdata = f.read()
    count=0
    for i in fdata:
        if not i.isspace():
            count+=1
    print(count)
```

44

In [44]:
```
# Q.NO 40
# Write a program that reads data from a file and print the no of vowels and c
onstants in the file.
with open("Q.NO 40","r") as f:
    fdata=f.read()
    count_v=0
    count_c=0
    vow="AEIOUaeiou"
    for i in fdata:
        if i.isalpha():
            if i in vow:
                count_v+=1
            else:
                count_c+=1
    print(count_v,count_c)
```

```
7 12
```

In [6]:
```
# 41. Write a python program that accept file Name as input from the user. Ope
n the file and count
# the number of times a character appears in the file.
fileName=input()
letter=input()
file = open(fileName, 'r')
text = file.read()
print(text.count(letter))
```

```
Q.NO 41
i
3
```

Object Oriented Programming

- Resuable data
- Bottom up approach
- access modifiers
- mores secure
- Object can move freely within member function
- supports inheritance

Classes and Objects

A class is a collection of objects or you say it is a blueprint of objs defining the common attributes and behaviour. Class definition: class class1( ): "class- keyword class1- name of the class"

Creating an obj and class in python -

In [9]:
```
class employee():
    def __init__(self,name,age,id,salary):
        self.name =  name
        self.age = age
        self.salary = salary
        self.id = id
emp1 = employee("harshith",22,1000,1234)
emp2 = employee("arjun",23,2000,2234)
print(emp1.__dict__)
```

```
{'name': 'harshith', 'age': 22, 'salary': 1234, 'id': 1000}
```

Inheritance

- Transfering of characteristics from parent to child class without any modificaton.
- The new class is called the derived/child class.
- single(a->b),multilevel(a->b: b->c), hybrid,herirarchical(a->(b and c)), multiple ((a and b)->c)

Single inheritance

- derive charecteristsics from a single parent class
- Example :

In [11]:
```
class employee1():
    def __init__(self,name,age,salary):
        self.name =  name
        self.age  = age
        self.salary = salary
class childemployee(employee1):
    def __init__(self,name,age,salary,id):
        self.name = name
        self.age = age
        self.salary = salary
        self.id = id
emp1 = employee1('harshith',22,1000)
print(emp1.age)
```

```
22
```

In [13]:
```
#The same example, just created an obj with child class

class employee1():
    def __init__(self,name,age,salary):
        self.name =  name
        self.age  = age
        self.salary = salary
class childemployee(employee1):
    pass
emp1 = childemployee('harshith',22,1000)
print(emp1.age)
```

```
22
```

In [14]:
```
print("\U0001f921")
```

```
🤡
```

In [15]:
```
class employee1():
    def __init__(self,name,age,salary):
        self.name =  name
        self.age  = age
        self.salary = salary
class childemployee1(employee1):
    pass
class childemployee2(childemployee1):
    pass
emp1 = childemployee2('harshith',22,1000)
print(emp1.age)
```

```
22
```

In [ ]:
```
### Hierarchical
#### enables more than one derived class to inherite properities from a parent classs
- example:
```

In [16]:
```
class A:
    def classA():
        print("iam from class a")
class B(A):
    def classB():
        print("iam from class b")
class C(A):
    def classC():
        print("iam from class c")

obj=C
obj1=B
obj.classA()
obj1.classB()
```

```
iam from class a
iam from class b
```

In [17]:
```
## Hybrid Inheritance

class A:
    def classA():
        print("I am from class A")
class B:
    def classB():
        print("I am from class B")
class C(A,B):
    def classC():
        print("I am from class C")
class D(C):
    def classD():
        print("I am from class D")
class E(D):
    def classE():
        print("I am from class E")
obj=E
obj.classA()
obj.classB()
obj.classC()
obj.classD()
obj.classE()
```

```
I am from class A
I am from class B
I am from class C
I am from class D
I am from class E
```

Polymorphism - 2 types

1) Complie time :

- compile time - static polymorphism which resolves during the compilation time
- method overloading
- example:

In [1]:
```
class employee1():
    def name(self):
        print("Harshith is his name")
    def salary(self):
        print("3000 is his salary")
    def age(self):
        print("22 is his age")
class employee2():
    def name(self):
        print("Rahul is his name")
    def salary(self):
        print("4000 is his salary")
    def age(self):
        print("23 is his age")

def func(obj):
    obj.name()
    obj.salary()
    obj.age()
obj_emp1=employee1()
obj_emp2=employee2()
func(obj_emp1)
func(obj_emp2)
```

```
Harshith is his name
3000 is his salary
22 is his age
Rahul is his name
4000 is his salary
23 is his age
```

2) Runtime Polymorphism

- dynamic polymorphism where it gets resolved into the runtime.
- method overriding
- example:

In [2]:
```
# over riding of variables
class parent:
    name = 'Anna'
class child(parent):
    name = 'David'

c=child()
print(c.name)
```

```
David
```

In [4]:
```
# over riding of methods
class parent:
    def career(self):
        return "Engineer"
class child(parent):
    def career(self):
        return "Photographer"

c=child()
print(c.career())
```

```
Photographer
```

In [3]:
```
# without riding
class parent:
    name = 'Anna'
class child(parent):
    pass

c=child()
print(c.name)
```

```
Anna
```

Encapsulation

- process of binding the data (persons cannot access the data),
- for binding the data we need to use private modifier, we represent it with _
- we can access private data with in the class only

In [11]:
```
#private attribute - double underscore only
class A:
    __a=10
    def disp(self):
        print(self.__a)
obj =A()
obj.disp()
```

```
10
```

In [12]:
```
obj.__a # error why? becuase its a private attribute we cannot access it outsi
de
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [12], in <cell line: 1>()
----> 1 obj.__a

AttributeError: 'A' object has no attribute '__a'
```

In [14]:
```
class Myclass:
    def __disp(self):
        print('this is a private method')
    def disp2(self):
        print('This is calling pribvate method')
        self.__disp()

obj = Myclass()
obj.disp2()
```

```
This is calling pribvate method
this is a private method
```

In [16]:
```
class Employee():
    __id =111

    def set_eid(self,eid):
        self.__id = eid
    def get_eid(self):
        return self.__id

e = Employee()
print(e.get_eid())
# print(e.__id) - error
```

```
111
```

In [22]:
```
#protected variable
class student:
    __r=15
    def __init__(self,name):
        self._name=name
    def name(self):
        return self._name
    def setname(self,newname):
        self._name = newname
```

In [23]:
```
obj= student("honey")
```

Abstraction

- highilighting the services and hiding the implementation in child class
- just shows declaration
- special kind of class for which we cannot create a object

In [28]:
```
from abc import ABC, abstractmethod
class A (ABC):
    @abstractmethod # to specify this method is abstract
    def disp(self): #only declaration
        pass
class B(A):
    def disp(self):
        print("Good")
b= B()
b.disp()
```

```
Good
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: