In [63]:

```
# Create a variable
name='Akash'
# Print the value
print(name)
'''This is a perfect
example of multi-line comments'''
def multiply(a,b):
    """Multiplies the value of a and b"""
    return a*b
# Print the docstring of multiply function
print(multiply.__doc__)
multiply(25,89)
```

```
Akash
Multiplies the value of a and b
```

Out[63]:

```
2225
```

In [1]:

```
print(type(None))
str1="Welcome"
print(str1)
print(type(str1))#String data type is immutable
a=5
print(a)
print(type(a))#Int data type is immutable
b=6.5
print(b)
print(type(b))#Float data type is immutable
c=4+5j
print(c)
print(type(c))#Complex data type is immutable
d=True
print(d)
print(type(d))#Boolean data types are immutable
list1 = [10, -20 , 15.5, 'vijay', "mary" ]
print(list1)
print(type(list1))#List data type is mutable
byte1=bytes([1,3,5,7,9])
print(byte1)
print(type(byte1))#Bytes data type is immutable
ba1=bytearray([10,20,30,40,50])
print(ba1)
print(type(ba1))#Bytes array data type is mutable
tup=('apple','banana','cherry')
print(tup)
print(type(tup))#Tuple data type is immutable
dict1={"a":1,"b":"Henry","c":"Class 8"}
print(dict1)
print(type(dict1))#Dictionary data type is mutable but the keys in it must be of an immu
set1={1,2,3,4,5}# OR We can write it as set1=set([1,2,3,4,5])
print(set1)
print(type(set1))#Set data type is mutable
fset1=frozenset([1,2,3,4,5])
print(fset1)
print(type(fset1))#Frozen set Data type is immutable
```

```
<class 'NoneType'>
Welcome
<class 'str'>
5
<class 'int'>
6.5
<class 'float'>
(4+5j)
<class 'complex'>
True
<class 'bool'>
[10, -20, 15.5, 'vijay', 'mary']
<class 'list'>
b'\x01\x03\x05\x07\t'
<class 'bytes'>
bytearray(b'\n\x14\x1e(2')
<class 'bytearray'>
('apple', 'banana', 'cherry')
<class 'tuple'>
{'a': 1, 'b': 'Henry', 'c': 'Class 8'}
<class 'dict'>
{1, 2, 3, 4, 5}
<class 'set'>
frozenset({1, 2, 3, 4, 5})
<class 'frozenset'>
```

In [3]:

```
'''Text data types'''
word="Hello World"
print(word)
print(word[0])#Accessing characters in a string using their indices
print(len(word))#To find the length of the string
print(word.count('l'))#To count how many times the letter "l" is present in the string
print(word.find("H"))#Tofind the letter "H" in the string
# Slicing
print (word[0])        #get one char of the word
print (word[0:1])   #get one char of the word (same as above)
print (word[0:3])    #get the first three char
print (word[:3])   #get the first three char
print (word[-3:])     #get the last three char
print (word[3:])      #get all but the three first char
print (word[:-3])        #get all but the three last character
print(word[:])                # a copy of the whole list
print(word.split(" "))#To split the string on whitespace
print(word.startswith("H"))
print(word.endswith("d"))
print(word.startswith("h"))
print(word.upper())
print(word.lower())
print(word.title())
print(word.capitalize())
print(word.swapcase())
print(':'.join(word))
print(''.join(reversed(word)))
print(' '.join(word))
print(word.replace("Hello","Goodbye"))

#Strip
str1="   xyz   "
print(str1)
print(str1.strip())
print(str1.lstrip())
print(str1.rstrip())

# Partition
txt="I could eat chocolates all day"
print(txt.partition("chocolates"))

# Testing
print(word.isalnum())           #check if all char are alphanumeric
print(word.isalpha())              #check if all char in the string are alphabetic
print(word.isdigit())              #test if string contains digits
print(word.istitle())               #test if string contains title words
print(word.isupper())               #test if string contains upper case
print(word.islower())               #test if string contains lower case
print(word.isspace())               #test if string contains spaces
print(word.endswith('d'))       #test if string endswith a d
print(word.startswith('H'))     #test if string startswith H
```

```
Hello World
H
11
3
0
H
H
Hel
Hel
rld
lo World
Hello Wo
Hello World
['Hello', 'World']
True
True
False
HELLO WORLD
hello world
Hello World
Hello world
hELLO wORLD
H:e:l:l:o: :W:o:r:l:d
dlroW olleH
H e l l o   W o r l d
Goodbye World
   xyz
xyz
xyz
   xyz
('I could eat ', 'chocolates', ' all day')
False
False
False
True
False
False
False
True
True
```

In [4]:

```
'''Integers and floats as Booleans'''
a=0
print(not a)
b=None
print(not b)
c=[]
print(not c)
d=10
print(not d)
e=0
print(bool(e))
f=1
print(bool(f))
g=-9.7
print(bool(g))
```

```
True
True
True
False
False
True
True
```

In [5]:

```
'''List Data type'''
a = ['one', 'two', 'three', 'four', 'five']
print(len(a))
print(a[0])
print(a[1])
# List slicing
print(a[1:])
print(a[-1])
print(a[-2:])
print(a[:-2])
print(a[:])
#List split
b="one;two;three;four;five;1;2"
c=b.split(";")
print(c)
#List append
a.append('six')
print(a)
d=['seven','eight']
a.append(d)
print(a)
#List extend
a.extend(["Nine","Ten"])
print(a)
a.extend("Numbers")
print(a)
# List insert
a.insert(0,"Numbers")
print(a)

e=[1,23,4,56,78,97,56,25,23,65,25,36,87,66]

# List remove
e.remove(56)
print(e)
del e[6]
print(e)
# List reverse
print(e[::-1])
'''   OR    '''
e.reverse()
print(e)
# List sorting
print(sorted(e))
print(sorted(e,reverse=True))
# List pop
fruits = ['apple','banana','cherry','orange']
fruits.pop()
print(fruits)
# List clear
fruits.clear()
print(fruits)
# List count
points = [1,4,2,9,7,8,9,3,1]
x = points.count(9)
print(x)
# List copy
p=[1,2,3,4,5]
y=p.copy()
```

```
print(y)
# List constructor
thislist = list(("apple","banana","cherry"))
print(thislist)
```

```
5
one
two
['two', 'three', 'four', 'five']
five
['four', 'five']
['one', 'two', 'three']
['one', 'two', 'three', 'four', 'five']
['one', 'two', 'three', 'four', 'five', '1', '2']
['one', 'two', 'three', 'four', 'five', 'six']
['one', 'two', 'three', 'four', 'five', 'six', ['seven', 'eight']]
['one', 'two', 'three', 'four', 'five', 'six', ['seven', 'eight'], 'Nine',
'Ten']
['one', 'two', 'three', 'four', 'five', 'six', ['seven', 'eight'], 'Nine',
'Ten', 'N', 'u', 'm', 'b', 'e', 'r', 's']
['Numbers', 'one', 'two', 'three', 'four', 'five', 'six', ['seven', 'eigh
t'], 'Nine', 'Ten', 'N', 'u', 'm', 'b', 'e', 'r', 's']
[1, 23, 4, 78, 97, 56, 25, 23, 65, 25, 36, 87, 66]
[1, 23, 4, 78, 97, 56, 23, 65, 25, 36, 87, 66]
[66, 87, 36, 25, 65, 23, 56, 97, 78, 4, 23, 1]
[66, 87, 36, 25, 65, 23, 56, 97, 78, 4, 23, 1]
[1, 4, 23, 23, 25, 36, 56, 65, 66, 78, 87, 97]
[97, 87, 78, 66, 65, 56, 36, 25, 23, 23, 4, 1]
['apple', 'banana', 'cherry']
[]
2
[1, 2, 3, 4, 5]
['apple', 'banana', 'cherry']
```

In [6]:

```
'''Bytes data type'''#It can store an array of numbers from 0 to 255 excluding negative
elements=[10,20,30,40,50]
x=bytes(elements)
for i in x:
    print(i)
print(x[0])
```

```
10
20
30
40
50
10
```

In [7]:

```
'''Byte array data type'''
a=[10,20,30,40,50]
x=bytearray(a)
print(a)
print(x[0])
x[0]=99
x[1]=85
for i in x:
    print(i)
```

```
[10, 20, 30, 40, 50]
10
99
85
30
40
50
```

In [8]:

```
'''Tuple data type'''
a=(5,2,1,3,5,3,7,5,4)
print(a[0])
print(a[1:5])
print(a[:-2])
print(a[-2:])
print(a[:])
print(len(a))
print(5 in a)
print(27 in a)
b=(3,8,6,7,2,3,5,1,4)
print(max(a))
print(min(a))
import operator as op
print(op.eq(a,b))
print(op.lt(a,b))
print(op.gt(a,b))
print(a.index(7))
print(a.count(5))
```

```
5
(2, 1, 3, 5)
(5, 2, 1, 3, 5, 3, 7)
(5, 4)
(5, 2, 1, 3, 5, 3, 7, 5, 4)
9
True
False
7
1
False
False
True
6
3
```

In [10]:

```
'''Dictionary data type'''
a = {"brand":"Ford","model":"Mustang","year":1964}
print(a)
print(a["model"])
print(a.get("brand"))
a["year"]=2018
print(a)
print(len(a))
a["color"]="Red"
print(a)
for x in a:
    print(x)
for x in a.values():
    print(x)
for x in a.items():
    print(x)
a.pop("model")
print(a)
a.popitem()
print(a)
del a["year"]
print(a)
a.clear()
print(a)
b={"Name":"Rohan","Branch":"CSE","Section":"Zeta"}
print(b.items())
print(b.keys())
print(b.values())
print(b.setdefault("Name"))
b.update({"Roll_No.":300})
print(b)
c=dict(brand="Ford",Model="Mustang",Year=2018)
print(c)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Mustang
Ford
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
3
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018, 'color': 'Red'}
brand
model
year
color
Ford
Mustang
2018
Red
('brand', 'Ford')
('model', 'Mustang')
('year', 2018)
('color', 'Red')
{'brand': 'Ford', 'year': 2018, 'color': 'Red'}
{'brand': 'Ford', 'year': 2018}
{'brand': 'Ford'}
{}
dict_items([('Name', 'Rohan'), ('Branch', 'CSE'), ('Section', 'Zeta')])
dict_keys(['Name', 'Branch', 'Section'])
dict_values(['Rohan', 'CSE', 'Zeta'])
Rohan
{'Name': 'Rohan', 'Branch': 'CSE', 'Section': 'Zeta', 'Roll_No.': 300}
{'brand': 'Ford', 'Model': 'Mustang', 'Year': 2018}
```

In [40]:

```
'''Set data type'''
a=set([1,2,3,4,5])
print(a)
b=set(['apple','banana','cherry'])
b.add("durian")
print(b)
b.update(["eggplant","flax_seed"])
print(b)
print(len(b))
b.pop()
print(b)
b.remove("eggplant")
print(b)
b.discard("cherry")#Here if the item doesn't exist then it doesn;t raise an error
print(b)
print(b)
b.clear()
print(b)
del b
c=set(("apple","banana","cherry"))
print(c)

x ={"apple","banana","cherry"}
y ={"google","microsoft","apple"}
print(x.difference(y))
x.difference_update(y)
print(x)

p={1,2,3,4,5,6}
q={5,6,7,8,9,10}
print(p.intersection(q))
print(p.union(q))
p.intersection_update(q)
print(p)

c={"apple","banana","cherry"}
d={"google","microsoft","facebook"}
print(c.isdisjoint(d))

m={"a","b","c"}
n={"f","e","d","c","b","a"}
print(m.issubset(n))
print(m.issuperset(n))

s={"apple","banana","cherry"}
t={"google","microsoft","apple"}
print(s.symmetric_difference(t))
s.symmetric_difference_update(t)
print(s)
```

```
{1, 2, 3, 4, 5}
{'apple', 'durian', 'banana', 'cherry'}
{'cherry', 'apple', 'durian', 'flax_seed', 'eggplant', 'banana'}
6
{'apple', 'durian', 'flax_seed', 'eggplant', 'banana'}
{'apple', 'durian', 'flax_seed', 'banana'}
{'apple', 'durian', 'flax_seed', 'banana'}
{'apple', 'durian', 'flax_seed', 'banana'}
set()
{'apple', 'banana', 'cherry'}
{'banana', 'cherry'}
{'banana', 'cherry'}
{5, 6}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{5, 6}
True
True
False
{'cherry', 'google', 'banana', 'microsoft'}
{'cherry', 'google', 'microsoft', 'banana'}
```

In [46]:

```
#Command-line arguments
import sys
Sum=int(sys.argv[1]) + int(sys.argv[2])
print("Sum=",Sum)
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_6024\3563351939.py in <module>
      1 #Command-line arguments
      2 import sys
----> 3 Sum=int(sys.argv[1]) + int(sys.argv[2])
      4 print("Sum=",Sum)

ValueError: invalid literal for int() with base 10: '-f'
```

In [47]:

```
#Arithmetic operators
a=5
b=10
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
print(a//b)
print(a**b)
```

```
15
-5
50
0.5
5
0
9765625
```

In [48]:

```
#Comparison or Relational operators
a=5
b=15
print(a<b)
print(a>b)
print(a==b)
print(a!=b)
print(a>=b)
print(a<=b)
```

```
True
False
False
True
False
True
```

In [49]:

```
# Logical or Boolean operators
x=True
y=False
print(x and y)
print(x or y)
print(not x)
print(not y)
```

```
False
True
False
True
```

In [50]:

```
# Bitwise operators
a=10
b=4
print(a&b)
print(a|b)
print(a^b)
print(~a)
print(a>>2)
print(b<<1)
```

```
0
14
14
-11
2
8
```

In [60]:

```
# Assignment operators
x=int(input("Enter a number: "))
x+=5
print(x)
x-=5
print(x)
x*=5
print(x)
x/=5
print(x)
x%=5
print(x)
x//=5
print(x)
x**=5
print(x)
```

```
Enter a number: 123456
123461
123456
617280
123456.0
1.0
0.0
0.0
```

In [56]:

```
# Identity operators
x=5
y=5
print(x is y)
print(x is not y)
```

```
True
False
```

In [61]:

```
# Membership operators
x="Hello"
print('h' in x)
print('h' not in x)
```

```
False
True
```

In [64]:

```
# Order of precedence
a,b,c,d,e,r=1,2,3,4,5,2
print(r//a+b*c-d/e)
'''Working
[2//1+2*3-4/5]
[2//1+6-4/5]
[2//1+6-0.8]
[2+6-0.8]
[8-0.8]
[7.2]'''
print(r//((a+b)*((c-d)/e)))
'''Working
[2//((1+2)*((3-4)/5))]
[2//(3*(-1/5))]
[2//(3*-0.2)]
[2//-0.6]
[-4.0]'''
```

```
7.2
-4.0
```

Out[64]:

```
'Working\n[2//((1+2)*((3-4)/5))]\n[2//(3*(-1/5))]\n[2//(3*-0.2)]\n[2//-0.
6]\n[-4.0]'
```

In [87]:

```
'''Output statements'''
# The print() statement
print("Hello")
print("This is the \nFirst line")#Here "\n" indicates new line
print("This is the \tFirst line")#Here "\t" indicates tab space
'''To escape the effect of escape sequence, we should add one more
‘\’ (backslash) before the escape sequence character.
For example, \\n displays "\n" and \\t diplays "\t"'''
print("This is the \\nFirst line")
print("Hi"*3)#We can use repetition operator (*) to repeat the strings
'''When we use ‘+’ on strings, it will join one string with another string.
Hence ‘+’ is called concatenation (joining) operator when used on strings'''
print("City name="+"Hyderabad")
'''The ‘+’ operator in the preceding statement joined the two strings without any space
We can also write the preceding statement by separating the strings using ‘,’ '''
print("City name=","Hyderabad")
a,b=2,4
print(a)
print(a,b)#Here the values in the output are separated by a space by default
print(a,b,sep=",")
print(a,b,sep=":")
'''‘sep’ represents separator. The format is sep=”characters” which are
used to separate the alues in the output.'''
print(a,b,sep="----")
'''We can ask the print’ () function not to throw the cursor into the
next line but display the output in the same line.
This is done using ‘end’ attribute. The way one can use it is end=”characters”
which indicates the ending characters for the line.'''
print("hello",end="")
print("Dear",end="")
print("How are you?",end="")
'''
If we use end=’\t’ then the output will be displayed in the same line but tab space will
'''
print()
print("hello",end="\t")
print("Dear",end="\t")
print("How are you?",end="\t")
'''
If we use end=’\n’ then the output will be displayed in the separate line.
So, ‘\n’ is the default value for ‘end’ attribute.
'''
print()
print("hello",end="\n")
print("Dear",end="\n")
print("How are you?",end="\n")
# Formatted string
'''
The output displayed by the print () function can be formatted as we like.
The special operator ‘%’ (percent) can be used for this purpose.
It joins a string with a variable or value in the following format:
{print (“formatted string” % (variables list))}
In the formatted string we can use %i or %d to represent decimal integer numbers.
We can use %f to represent float values. similarly we can use %s to represent strings.
'''
x=10
print("value=%i"%x)
a,b=10,15
print("a=%i b=%d"%(a,b))
```

```
'''
When we use %20s, it will allot 20 spaces and the string is displayed right aligned in t
 To align the strings towards left side in the spaces, we can use %-20s'''
name="Rahul"
print("Hi %s"%name)
print("Hi (%20s)"%name)
print("Hi (%-20s)"%name)
# We can use %c to display a single character
name="Linda"
print("Hi %c, %c"%(name[0],name[1]))
'''
To display floating point values, we can use %f in the formatted string.
 If we use %8.2f, then the float value is displayed in 8 spaces and within these spaces,
 a decimal point and next 2 fraction digits.
When we use %.3f, then the float value is displayed with 3 fraction digits after the dec
However the digits before decimal point will be displayed as they are.
'''
num=123.456789
print("The value is: %f"%num)
print("The value is: %8.2f"%num)
print("The value is: %.3f"%num)
'''
Inside the formatted string, we can use replacement field which is denoted by a
pair of curly braces { } .
we can mention names or indexes in these replacement fields.
These names or indexes represent the order of the values.
After the formatted string, we should write member operator and then format() method
where we should mention the values to be displayed. Consider the general format given be
print('format string with replacement fields'.format(values))
'''
p,q,r=1,2,3
print("num1={0}".format(p))
print("num1={0}, num2={1}, num3={2}".format(p,q,r))
```

```
Hello
This is the
First line
This is the     First line
This is the \nFirst line
HiHiHi
City name=Hyderabad
City name= Hyderabad
2
2 4
2,4
2:4
2----4
helloDearHow are you?
hello   Dear    How are you?
hello
Dear
How are you?
value=10
a=10 b=15
Hi Rahul
Hi (               Rahul)
Hi (Rahul               )
Hi L, i
The value is: 123.456789
The value is:   123.46
The value is: 123.457
num1=1
num1=1, num2=2, num3=3
```

In [3]:

```
'''Input statements'''
a=input()
b=input("Enter a number: ")
c=int(b)
print(c)
```

```
Raj Kumar
Enter a number: 15
15
```

In [4]:

```
# Entry control statements(if....elif....else statements)
a=int(input("Enter a number: "))
if a%2==0:
    print(f"{a} is an even number")
else:
    print(f"{a} is an odd number")
```

```
Enter a number: 16326864
16326864 is an even number
```

In [64]:

```
# Nested loop
command=""
started=False
while True:
    command=input("> ").lower()
    if command=="start":
        if started:
            print("Car is already started!")
        else:
            started=True
            print("Car started...")
    elif command=="stop":
        if not started:
            print("Car is already stopped!")
        else:
            started=False
            print("Car stopped")
    elif command=="help":
        print("""
start-To start the car
stop-To stop the car
quit-To quit
""")
    elif command=="quit":
        print("You have quit!")
        break
    else:
        print("Sorry,I don't understand that")
```

```
> amruth
Sorry,I don't understand that
> help

start-To start the car
stop-To stop the car
quit-To quit

> start
Car started...
> start
Car is already started!
> nWHEDSGT M
Sorry,I don't understand that
> stop
Car stopped
> stop
Car is already stopped!
> jeadg
Sorry,I don't understand that
> quit
You have quit!
```

In [77]:

```
# While loop
a=1
while a<6:
    print(a)
    a=a+1
```

```
1
2
3
4
5
```

In [78]:

```
# Nested loops
for i in range(0,3):
    for j in range(0,3):
        print((i,j))
```

```
(0, 0)
(0, 1)
(0, 2)
(1, 0)
(1, 1)
(1, 2)
(2, 0)
(2, 1)
(2, 2)
```

In [5]:

```
'''Loop control statements'''
# break statement
'''The break statement terminates the loop statement and transfers the execution to the
statement immediately following the loop'''
for letter in "python":
    if letter=="h":
        break
    print(f"Current letter: {letter}")
print()
# continue statement
'''The continue statement cause the loop to skip the remainder of its body and retest it
prior to reiterating'''
x=0
while x<10:
    x+=1
    if x>5:
        continue
    print("x=",x)
print("Out of loop")
print()
# pass statement
'''The pass statement is used when a statement is required syntactically but you do not
want any command or code to execute'''
num=[1,2,3,-4,-5,-6,-7,-8,9]
for i in num:
    if i>0:
        pass
    else:
        print(i)
print()
# Assert statement
'''The assert statement is used to check if a particular condition is fulfilled or not
the syntax is as follows:
      {assert expression,message}'''
a=int(input("Enter a number greater than 0: "))
assert a>0,"Wrong input entered"
print("You entered: ",a)
```

```
Current letter: p
Current letter: y
Current letter: t

x= 1
x= 2
x= 3
x= 4
x= 5
Out of loop

-4
-5
-6
-7
-8

Enter a number greater than 0: -1
```

```
---------------------------------------------------------------------------
-
AssertionError                            Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_18624\448689506.py in <module>
     34       {assert expression,message}'''
     35 a=int(input("Enter a number greater than 0: "))
---> 36 assert a>0,"Wrong input entered"
     37 print("You entered: ",a)

AssertionError: Wrong input entered
```

In [24]:

```
"""Arrays"""
# Creating an array
from array import *
a=array('i',[10,20,30,40,50])
for i in a:
    print(i)
print()
# Accessing array element
print(a[0])
print(a[1])
print()
# Insertion operation
a.insert(1,60)
for i in a:
    print(i)
print()
# Append
a.append(70)
print(a)
# Extend
a.extend([80,90,100])
print(a)
print()
# Concatenation of two arrays
odd=array('i',[1,3,5])
even=array('i',[2,4,6])
numbers=array('i')
numbers=odd+even
print(numbers)
# Remove
a.remove(60)
print(a)
# Deletion operation
del a[8]
print(a)
# Search operation
print(a.index(40))
# Update operation
a[5]=60
print(a)
# Changing or adding elements
a[7]=70
print(a)
a[2:5]=array('i',[50,60,70])
print(a)
# Slicing of an array
print(a[2:5])
print(a[:-5])
print(a[5:])
print(a[:])
```

```
10
20
30
40
50

10
20

10
60
20
30
40
50

array('i', [10, 60, 20, 30, 40, 50, 70])
array('i', [10, 60, 20, 30, 40, 50, 70, 80, 90, 100])

array('i', [1, 3, 5, 2, 4, 6])
array('i', [10, 20, 30, 40, 50, 70, 80, 90, 100])
array('i', [10, 20, 30, 40, 50, 70, 80, 90])
3
array('i', [10, 20, 30, 40, 50, 60, 80, 90])
array('i', [10, 20, 30, 40, 50, 60, 80, 70])
array('i', [10, 20, 50, 60, 70, 60, 80, 70])
array('i', [50, 60, 70])
array('i', [10, 20, 50])
array('i', [60, 80, 70])
array('i', [10, 20, 50, 60, 70, 60, 80, 70])
```

In [33]:

```
# Two dimensional array
from array import*
a=[[11,12,5,2],[15,5,10],[10,8,12,5],[12,15,8,6]]
print(a[0])
print(a[1][2])
print(a)
# Array end to line print
for r in a:
    for c in r:
        print(c,end=" ")
print()
# Inserting values in two dimensional arrays
a.insert(2,[0,5,11,13,6])
for r in a:
    for c in r:
        print(c,end=" ")
print()
# Updating values in two dimensional arrays
a[2]=[11,9]
a[0][3]=7
for r in a:
    for c in r:
        print(c,end=" ")
```

```
[11, 12, 5, 2]
10
[[11, 12, 5, 2], [15, 5, 10], [10, 8, 12, 5], [12, 15, 8, 6]]
11 12 5 2 15 5 10 10 8 12 5 12 15 8 6
11 12 5 2 15 5 10 0 5 11 13 6 10 8 12 5 12 15 8 6
11 12 5 7 15 5 10 11 9 10 8 12 5 12 15 8 6
```

In [34]:

```
# Matrix Addition
a=[[12,7,3],[4,5,6],[7,8,9]]
b=[[5,8,1],[6,7,3],[4,5,9]]
result=[[0,0,0],[0,0,0],[0,0,0]]
for i in range(len(a)):
    for j in range(len(a[0])):
        result[i][j]=a[i][j]+b[i][j]
for r in result:
    print(r)
```

```
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
```

In [35]:

```
# Transpose of a matrix
a=[[12,7],[4,5,],[3,8]]
result=[[0,0,0],[0,0,0]]
for i in range(len(a)):
    for j in range(len(a[0])):
        result[j][i]=a[i][j]
for r in result:
    print(r)
```

```
[12, 4, 3]
[7, 5, 8]
```

In [6]:

```
# Matrix multiplication
a=[[1,2,3],[4,5,6],[7,8,9]]
b=[[1,1,1],[1,1,1],[1,1,1]]
result=[[0,0,0],[0,0,0],[0,0,0]]
for i in range(len(a)):
    for j in range(len(b[0])):
        for k in range(len(b)):
            result[i][j]+=a[i][k]*b[k][j]
for r in result:
    print(r)
```

```
[6, 6, 6]
[15, 15, 15]
[24, 24, 24]
```

In [39]:

```
# Creating a numpy array
import numpy as np
a=np.array([1,2,3])
print(a)
print(type(a))
```

```
[1 2 3]
<class 'numpy.ndarray'>
```

In [40]:

```
# Comparing memory use of numpy array and list
import numpy as np
import time
import sys
a=np.arange(10)
print(f"Size of numpy array: {a.size*a.itemsize}")
b=range(0,10)
print(f"Size of list: {sys.getsizeof(1)*len(b)}")
```

```
Size of numpy array: 40
Size of list: 280
```

In [41]:

```
# Convenience of numpy array
import numpy as np
a=np.array([1,2,3])
b=np.array([4,5,6])
print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

```
[5 7 9]
[-3 -3 -3]
[ 4 10 18]
[0.25 0.4  0.5 ]
```

In [38]:

```
# Arrays of integers,floats and complex numbers
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
print(a)
b=np.array([[1.1,2,3],[4,5,6]])
print(b)
c=np.array([[1,2,3],[4,5,6]],dtype=complex)
print(c)
```

```
[[1 2 3]
 [4 5 6]]
[[1.1 2.  3. ]
 [4.  5.  6. ]]
[[1.+0.j 2.+0.j 3.+0.j]
 [4.+0.j 5.+0.j 6.+0.j]]
```

In [50]:

```
# Arrays of zeros and ones
import numpy as np
a=np.zeros((2,3))
print(a)
b=np.ones((1,5),dtype=int)
print(b)
# Create an array with random values
from numpy import random
c=random.randint(100,size=(4,2))
print(c)
# Creating an empty array
d=np.empty((3,2))
print(d)
# Creating a full array
e=np.full((2,2),7)
print(e)
# Create an array of evenly-spaced values
f=np.arange(10,25,5)
print(f)
g=np.linspace(0,2,9)
print(g)
# Creating an identity matrix
i=np.identity(3,dtype=int)
print(i)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
[[1 1 1 1 1]]
[[18 93]
 [26 82]
 [ 5 52]
 [66 15]]
[[0. 0.]
 [0. 0.]
 [0. 0.]]
[[7 7]
 [7 7]]
[10 15 20]
[0.   0.25 0.5  0.75 1.   1.25 1.5  1.75 2.  ]
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

In [43]:

```
# Using arange() and reshape()
import numpy as np
a=np.arange(4)
print(a)
b=np.arange(12).reshape(2,6)
print(b)
```

```
[0 1 2 3]
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]
```

In [53]:

```
# Copying from one array to another
import numpy as np
a=np.full([2,3,2],5)
print(a)
b=np.full([2,3,2],3)
print(b)
np.copyto(b,a)
print(b)
```

```
[[[5 5]
  [5 5]
  [5 5]]

 [[5 5]
  [5 5]
  [5 5]]]
[[[3 3]
  [3 3]
  [3 3]]

 [[3 3]
  [3 3]
  [3 3]]]
[[[5 5]
  [5 5]
  [5 5]]

 [[5 5]
  [5 5]
  [5 5]]]
```

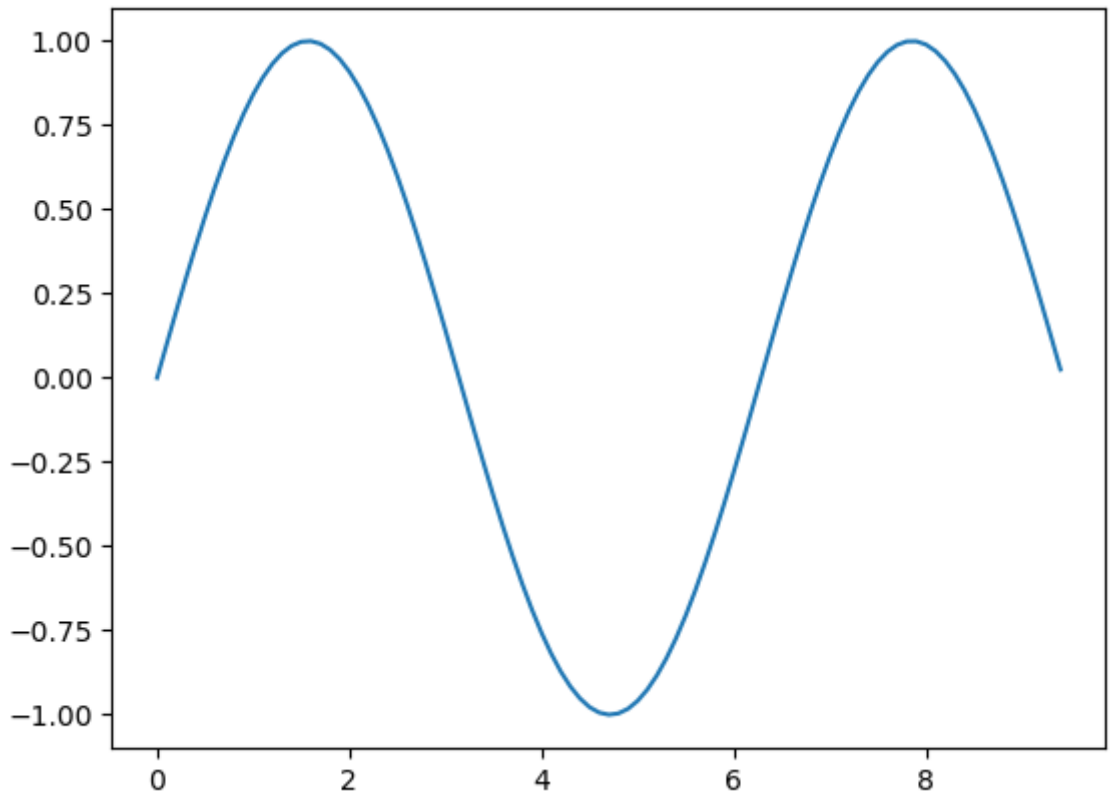
In [60]:

```
import numpy as np
# Matrix addition
a=np.array([[2,4],[5,-6]])
b=np.array([[9,-3],[3,6]])
print(a+b)
# Matrix multiplication
c=np.array([[3,6,7],[5,-3,0]])
d=np.array([[1,1],[2,1],[3,-3]])
print(a.dot(b))
# Transpose of a matrix
e=np.array([[1,1],[2,1],[3,-3]])
print(e.transpose())
# Access matrix elements, rows and columns
f=np.array([2,4,6,8,10])
print(f[0])
print(f[2])
print(f[-1])
# Access elements of a two dimensional array
g=np.array([[1,4,5,12],[-5,8,9,0],[-6,7,11,19]])
print(g[0][0])
print(g[1][2])
print(g[-1][-1])
```

```
[[11  1]
 [ 8  0]]
[[ 30  18]
 [ 27 -51]]
[[ 1  2  3]
 [ 1  1 -3]]
2
6
10
1
9
19
```

In [62]:

```
# Python numpy special functions
import numpy as np
import matplotlib.pyplot as plt
x= np.arange(0,3*np.pi,0.1)
y=np.sin(x)
plt.plot(x,y)
plt.show()
```



In [79]:

```
# Formal and Actual argumments
'''
When a function is defined, it may have some parameters. These parameters are useful to
from outside of the function. They are called 'formal arguments'. When we call the funct
pass data or values to the function. These values are called 'actual arguments'. In the
'a' and 'b' are formal arguments and 'x' and 'y' are actual arguments.
'''
def sum(a, b): #a, b are formal arguments
    c= a+b
    print(c)
x=10
y=15
sum(x, y)
```

25

In [80]:

```
# Default Parameter value
def my_function(country = "Norway"):
    print("I am from " + country)
my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

In [7]:

```
# Required Arguments
def sum (a,b):
    return a+b
a = int(input("Enter a: "))
b = int(input("Enter b: "))
print("Sum = ",sum(a,b))
```

```
Enter a: 10
Enter b: 20
Sum =  30
```

In [83]:

```
# Keyword Arguments
def student(firstname, lastname):
    print(firstname, lastname)
student(firstname='Mani', lastname='Pal')
student(lastname='Pal', firstname='Mani')
```

```
Mani Pal
Mani Pal
```

In [84]:

```
# Arbitrary or Variable-length arguments
def sum_function(*x):
    total=0
    for i in x:
        total+=i
    return total
print(sum_function(1,2,3,4,5))
```

```
15
```

In [86]:

```
# Arbitrary keyword arguments
def second(**kwargs):
    print(f"The value at second position is: {kwargs['a2']}")
second(a1=1,a2=2,a3=3)
```

```
The value at second position is: 2
```

In [87]:

```
# retuurn values
def my_function(a):
    return 5*a
print(my_function(3))
print(my_function(5))
```

```
15
25
```

In [1]:

```
# Global snd Local variable
y=7
def func4():
    global y
    y+=1
    print(y)
func4()


def print_message():
    message = "hello !! I am going to print a message."
# the variable message is local to the function itself
    print(message)
print_message()
```

```
8
hello !! I am going to print a message.
```

In [4]:

```
# Recursive functions [Factorial of a given number]
a=int(input("Enter a number: "))
def fact(a):
    if a>=1:
        return a*fact(a-1)
    elif a==0:
        return 1
print(fact(a))
```

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

In [5]:

```
# Lambda functions
cube=lambda x:x**3
print(cube(7))
```

```
343
```

In [6]:

```
# filter() method
a=[1,5,4,6,8,11,3,12]
b=list(filter(lambda x:x%2==0,a))
print(b)
```

```
[4, 6, 8, 12]
```

In [7]:

```
# map() method
a=[1,5,4,6,8,11,3,12]
b=list(map(lambda x:x*2,a))
print(b)
```

```
[2, 10, 8, 12, 16, 22, 6, 24]
```

In [8]:

```
# Class and Object
class Rectangle():
    def __init__(self,length,breadth):
        self.length=length
        self.breadth=breadth
    def area(self):
        return self.length*self.breadth
obj1=Rectangle(4,5)
print(obj1.area())
obj2=Rectangle(10,20)
print(obj2.area())
```

```
20
200
```

In [9]:

```
#Single inheritance
class ParentClass:
    def feature_1(self):
        print("feature_1 from ParentClass is running...")
    def feature_2(self):
        print("feature_2 from ParentClass is running...")
class ChildClass(ParentClass):
    def feature_3(self):
        print("feature_3 from ChildClass is running...")
obj=ChildClass()
obj.feature_1()
obj.feature_2()
obj.feature_3()
```

```
feature_1 from ParentClass is running...
feature_2 from ParentClass is running...
feature_3 from ChildClass is running...
```

In [10]:

```
# Multiple inheritance
class ParentClass1:
    def feature_1(self):
        print("feature_1 from ParentClass1 is running...")
class ParentClass2:
    def feature_2(self):
        print("feature_2 from ParentClass2 is running...")
class ChildClass(ParentClass1,ParentClass2):
    def feature_3(self):
        print("feature_3 from ChildClass is running...")
obj=ChildClass()
obj.feature_1()
obj.feature_2()
obj.feature_3()
```

```
feature_1 from ParentClass1 is running...
feature_2 from ParentClass2 is running...
feature_3 from ChildClass is running...
```

In [11]:

```
#Multi-level inheritance
class ParentClass:
    def feature_1(self):
        print("feature_1 from ParentClass is running...")
class ChildClass1(ParentClass):
    def feature_2(self):
        print("feature_2 from ChildClass1 is running...")
class ChildClass2(ChildClass1):
    def feature_3(self):
        print("feature_3 from ChildClass2 is running...")
obj=ChildClass2()
obj.feature_1()
obj.feature_2()
obj.feature_3()
```

```
feature_1 from ParentClass is running...
feature_2 from ChildClass1 is running...
feature_3 from ChildClass2 is running...
```

In [12]:

```
# Hierarchial inheritance
class ParentClass:
    def feature_1(self):
        print("feature_1 from ParentClass is running...")
class ChildClass1(ParentClass):
    def feature_2(self):
        print("feature_2 from ChildClass1 is running...")
class ChildClass2(ParentClass):
    def feature_3(self):
        print("feature_3 from ChildClass2 is running...")
obj1=ChildClass1()
obj1.feature_1()
obj1.feature_2()
obj2=ChildClass2()
obj2.feature_1()
obj2.feature_3()
```

```
feature_1 from ParentClass is running...
feature_2 from ChildClass1 is running...
feature_1 from ParentClass is running...
feature_3 from ChildClass2 is running...
```

In [13]:

```
# Hybrid inheritance
class ParentClass:
    def feature_1(self):
        print("feature_1 from ParentClass is running...")
class ChildClass1(ParentClass):
    def feature_2(self):
        print("feature_2 from ChildClass1 is running...")
class ChildClass2(ParentClass):
    def feature_3(self):
        print("feature_3 from ChildClass2 is running...")
class ChildClass3(ChildClass1,ChildClass2):
    def feature_4(self):
        print("feature_4 from ChildClass3 is running...")
obj=ChildClass3()
obj.feature_1()
obj.feature_2()
obj.feature_3()
obj.feature_4()
```

```
feature_1 from ParentClass is running...
feature_2 from ChildClass1 is running...
feature_3 from ChildClass2 is running...
feature_4 from ChildClass3 is running...
```

In [8]:

```
# Constructor
class Human:
    def __init__(self):
        self.legs=2
        self.arms=2
bob=Human()
print(bob.legs)
```

```
2
```

In [9]:

```
# Method overriding
class Square:
    side=5
    def calc_area(self):
        return self.side*self.side
class Triangle:
    base=5
    height=4
    def calc_area(self):
        return 0.5*self.base*self.height
sq=Square()
tri=Triangle()
print("Area of Square:",sq.calc_area())
print("Area of Triangle:",tri.calc_area())
```

```
Area of Square: 25
Area of Triangle: 10.0
```

In [15]:

```
class Error(Exception):
    pass
class ValueTooSmallError(Error):
    pass
class ValueTooLargeError(Error):
    pass
number=10
while True:
    try:
        a=int(input("Enter a number: "))
        if a<number:
            raise ValueTooSmallError
        elif a>number:
            raise ValueTooLargeError
        break
    except ValueTooSmallError:
        print("This value is too small try again!")
        print()
    except ValueTooLargeError:
        print("This value is too large,try again!")
        print()
    else:
        print("Congratulations! You guessed the number")
```

```
Enter a number: 5
This value is too small try again!

Enter a number: 8
This value is too small try again!

Enter a number: 55
This value is too large,try again!

Enter a number: 654
This value is too large,try again!

Enter a number: 2
This value is too small try again!

Enter a number: -1
This value is too small try again!

Enter a number: 10
```

In [17]:

```
# Handle ZeroDivisionError exception
a=3
try:
    b=a/(a-3)
except ZeroDivisionError:
    print("Division by zero raised")
else:
    print("Job done")
finally:
    print("Task completed")
```

```
Division by zero raised
Task completed
```

In [1]:

```
# Different ways of creating a new file and writing data into a file
f=open('C:/Users/subha/OneDrive/Desktop/file.txt','r')
file_contents=f.read()
print(file_contents)
f.close()
f=open('C:/Users/subha/OneDrive/Desktop/file.txt','w')
f.write("This is some sample data\n")
f.write("This is another line of data")
print("Data written into the file:")
f.close()
f=open('C:/Users/subha/OneDrive/Desktop/file.txt','r')
modified_file_contents=f.read()
print(modified_file_contents)
```

```
Hi, I am Alex

Data written into the file:
This is some sample data
This is another line of data
```

In [5]:

```
# Python program to know whether a file exists or not, if it is existed display the cont
import os.path
path='C:/Users/subha/OneDrive/Desktop/file.txt'
check_file=os.path.isfile(path)
print(check_file)
f=open('C:/Users/subha/OneDrive/Desktop/file.txt','r')
file_contents=f.read()
print(file_contents)
```

```
True
This is some sample data
This is another line of data
```