Python

- Programming Language - created by Guido Van Rossum in 1991
- Runs on Interpreter - Runs code line by line

In [1]:
```
'''Multi Line Comment'''
```

Out[1]:
```
'Multi Line Comment'
```

In [2]:
```
# enumerate() -> prints index
for i in enumerate('amazon'):
    print(i)
```

```
(0, 'a')
(1, 'm')
(2, 'a')
(3, 'z')
(4, 'o')
(5, 'n')
```

In [3]:
```
# String *3 -> stringstringstring
"A"*3
```

Out[3]:
```
'AAA'
```

In [4]:
```
#To take input - returns string
age = input("Enter age")
```

```
Enter age12
```

In [5]:
```
# type() -> Find datatypes of variables
type(age)

# Convert string to integer
age=int(age)
print(type(age))
```

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

Get this output when using type() with print()

<class 'int'>

<class 'float'>

<class 'str'>

<class 'bool'>

PEDMAS

Parenthesis
Exponential -> **
Division ->
/ -> normal division
// -> Floor division
Multiplication

Addition
Subtraction

Data Structure:

Can hold data together
Ways of organizing data so that it can be used efficiently
Used to solve problems like:

1. Storage
2. Insert
3. Delete
4. Search
5. Sort
6. Traversing

Types of DS:

1. Primitive: Int,FLoat,String,Boolean
2. Non- Primitive:
   List[]
   Set {}
   Tuple ()
   Dictionary {}

String:

1. \ -> to tell the its part of the string and not end of the string if its written in single quotes.
2. \n -> for next line
3. Indexing of Strings:
   Forward Indexing -> Indexing starts from 0
   Negetive/Reverse Indexing -> Indexing starts from -1
4. len() -> gives Length of the string
5. Strings are immutable.

In [6]:
```
#Difference between break and continue:
#break - Terminates the loop when the particular condition is met.
#continue - That part of the code is skipped when the particular condition is met.

for x in range(4):
    if(x==2):
        continue
    print(x)
```

```
0
1
3
```

In [7]:
```
#break example:
for x in range(4):
    if(x==2):
        break
    print(x)
```

```
0
1
```

String Operations:

In [8]:
```
name = 'Sneha'
```

In [9]:
```
#gives Length of the string
len(name)
```

Out[9]:
```
5
```

In [10]:
```
#To get help for string functions
#help(str)
dir() # to view all attributes of an object
```

Out[10]:
```
['In',
 'Out',
 '_',
 '_1',
 '_3',
 '_9',
 '__',
 '___',
 '__builtin__',
 '__builtins__',
 '__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_dh',
 '_i',
 '_i1',
 '_i10',
 '_i2',
 '_i3',
 '_i4',
 '_i5',
 '_i6',
 '_i7',
 '_i8',
 '_i9',
 '_ih',
 '_ii',
 '_iii',
 '_oh',
 'age',
 'exit',
 'get_ipython',
 'i',
 'name',
 'quit',
 'x']
```

In [11]:
```
#String Slicing:
print(name[:]) # from start to end
print(name[-1]) # gets last element
print(name[:4])# gets all element from start to 3rd index bcz last index is excluded
print(name[-1:2:-1]) # getting elements from end till -2 position
print(name[0:]) # from zero to end

#Membership -> when a substring is available in the mentioned string/not
print('a' in name)
```

```
#Other String methods:
h="___Hi all##___"
h1="___Hi all##"
print(h.upper()) #To change all letters into uppercase
print(h.lower())#To change all letters into lowercase
print(h.strip("_")) # To strip the occurance of a substring from both ends
print(h1.lstrip("_")) # To strip the occurance of a substring from left end
print(h1.rstrip("#"))  # To strip the occurance of a substring from right end
print(h1.title()) # to get first letter of each word capitalized
print("{},{}".format(h,h1)) #To concatenate output as format string
```

```
Sneha
a
Sneh
ah
Sneha
True
___HI ALL##___
___hi all##___
Hi all##
Hi all##
___Hi all
___Hi All##
___Hi all##___,___Hi all##
```

Lists:

1. Mutable - editable - values can be edited
2. Dynamic - add/remove elements
3. Compound nature - stores values of multiple datatypes

In [12]:

```
#List Methods:
#Defining a list
list0=[]
list1 = [1,2,3]
list3=[1,2,[3,4,["hello",6]]]
print(list0)
print(list1)

#List indexing and slicing
print(list1[0:2])
print(list3[2])
print(list3[2][2][0])

#Membership:
print(2 in list3)
print(3 in list3[2])

#Concatenation: -> need to create a new list
newL= list1+["hi",[4,6]]
print(newL)

#Extend -> Changes made in the original list only
Lex=list1.extend(["hi",[4,6]])
print(Lex)
print(list1)

#Append -> takes the list and add it as an entire list.
list0.append(["hi",[4,6]])
print(list0)

#Insert
list1.insert(1,"via insert")
print(list1)
```

```
#Replace data -> checking mutability property:
print(list1)
list1[1]="updated"
print(list1)

#Delete
del list1[-1:2:-1]
print(list1)

#Pop() -> Removes last element by default
list1.pop()
print(list1)
list1.pop(0) # Removes the element at that index
print(list1)

#remove() -> removes the first occurance of the value
list1.remove('updated')
print(list1)

#Sorting
list4=[2,4,0,3,7,6,9]
print('list4 before sorting',list4)
list4.sort() #Original list is sorted
print('list4 after sorting asc',list4)
list4.sort(reverse=True)
print('list4 after sorting desc',list4)

list5=sorted(list4) #Original list is not sorted
print(list4)
print(list5)

#to prevent shallow copying:
list6=list5[:]
print(list6)
list6[0]="hello"
print(list5)
print(list6)

#List Comprehension:
lc1=[len(w) for w in list6 if w=="hello"] #[o/p for condition if condition]
print(lc1)

#Iterate over two lists:
for i,j in zip(list5,list6):
    print(i,"-",j)
```

```
[]
[1, 2, 3]
[1, 2]
[3, 4, ['hello', 6]]
hello
True
True
[1, 2, 3, 'hi', [4, 6]]
None
[1, 2, 3, 'hi', [4, 6]]
[['hi', [4, 6]]]
[1, 'via insert', 2, 3, 'hi', [4, 6]]
[1, 'via insert', 2, 3, 'hi', [4, 6]]
[1, 'updated', 2, 3, 'hi', [4, 6]]
[1, 'updated', 2]
[1, 'updated']
['updated']
[]
list4 before sorting [2, 4, 0, 3, 7, 6, 9]
```

```
list4 after sorting asc [0, 2, 3, 4, 6, 7, 9]
list4 after sorting desc [9, 7, 6, 4, 3, 2, 0]
[9, 7, 6, 4, 3, 2, 0]
[0, 2, 3, 4, 6, 7, 9]
[0, 2, 3, 4, 6, 7, 9]
[0, 2, 3, 4, 6, 7, 9]
['hello', 2, 3, 4, 6, 7, 9]
[5]
0 - hello
2 - 2
3 - 3
4 - 4
6 - 6
7 - 7
9 - 9
```

Tuples:

1. Fastest of all
2. Compound nature
3. Read-only

In [13]:

```
# Tuple methods:
#Tuple creation:
t=()# empty tuple -> python considers an an object
tup0=1, #single value tuple
tup1=("hi",5,6)
tup2=9,4,"hello"
tu=(9,7,(8,9,("oo",99)))
print(tup0)
print(tup1)
print(tup2)
print(tu)
print(len(tup1))
print(type(tu))
print(type(t))
print(dir(t)) # gives all attributes/methods that can be used

#Slicing and indexing:
print(tup1[0:2])
print(tu[2][2][0])

#Concatenation: -> creates new tuple
tup3=tup1+tup2
print(tup3)

#Methods for tuples with only numbers:
tup4=(2,5,1,0)
print(min(tup4))
print(max(tup4))
print(sum(tup4))

#Use concatenation and slicing together to perform mutation -> change 6 from tup3 to
tup5=tup3[0:2]+("changed",)+tup3[3:]
print(tup5)

#Sorting:
print('tup4 before sorting',tup4)
tup6=sorted(tup4) #Original list is sorted
print('tup4 after sorting asc',tuple(tup6)) # cant use sort() bcz immutable nature a

#packing unpacking:
#Consider a situation where we have a function that receives four arguments. We want
def fun(a, b, c, d):
```

```
    print(a, b, c, d)

my_list = [1, 2, 3, 4]

#fun(my_list) -> This doesn't work ->TypeError: fun() takes exactly 4 arguments (1 g
#fun(*my_list) -> This works -> Unpacking list into four arguments
```

```
(1,)
('hi', 5, 6)
(9, 4, 'hello')
(9, 7, (8, 9, ('oo', 99)))
3
<class 'tuple'>
<class 'tuple'>
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq_
_', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__
gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__',
'__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'in
dex']
('hi', 5)
oo
('hi', 5, 6, 9, 4, 'hello')
0
5
8
('hi', 5, 'changed', 9, 4, 'hello')
tup4 before sorting (2, 5, 1, 0)
tup4 after sorting asc (0, 1, 2, 5)
```

Sets:

1. Fast search and duplicacy.
2. Mutable.
3. unordered but sorted
4. As unordered therefore does not support indexing

In [14]:

```
#Sets methods:
set0={}
set1={2,3,5,5}
print(set0)
print(set1)
print(len(set1))

#Add elements to set:
set1.add("hi")
print(set1)

#Remove elements from set:
set1.remove(3)
print(set1)

#Set Operations:
set2={2,3,6,7,8}
print(set1|set2) # union -> print(set1.union(set2))
print(set1&set2) # intersection -> print(set1.intersection(set2))
print(set1-set2) #Difference -> removes set2 from set1 -> print(set1.difference(set2
print(set1^set2) # Symmetric Difference -> basically means from union remove the int

#Set Comprehension:
sc={x for w in set1 if w!=3}
print(sc)
```

```
{}
```

```
{2, 3, 5}
3
{'hi', 2, 3, 5}
{'hi', 2, 5}
{2, 3, 5, 'hi', 6, 7, 8}
{2}
{5, 'hi'}
{3, 'hi', 6, 7, 8, 5}
{2}
```

Dictionary:

1. Store key-value pair.
2. keys are unique and immutable and hence cant use set, list, dictionary as keys.
3. Unordered

In [15]:
```
#Dictionary Methods:
#Creating dictionary
dict1={1:"India",2:"USA"}
dict2={"datacamp":{"Deep Learning": "Python", "Machine Learning": "Pandas"},"linkedi
print(dict1)
print(dict2)
print(len(dict1))
#Accessing via keys:
print(dict1[1])

#Mutability:
dict1[1]="Changed"
print(dict1)

#Sorting:
print(sorted(dict1))

#Sorting a list of nested dictionary
#l1=[{},{}]
#l1.sort(key=lambda x:e['key']['subkey'])

# delete entry:
del dict1[1]
print(dict1)

print(dict2.values()) # to extract all values
print(dict2.keys()) # to extract all keys

#update values:
dict1.update({2:"UK",3:"nigeria"})
print(dict1)

print(dict1.items()) #return a list of dictionary items in the form of a key, value

#Dictionary Comprehension:
dc={w:len(w) for w in dict1.values()}
print(dc)
```

```
{1: 'India', 2: 'USA'}
{'datacamp': {'Deep Learning': 'Python', 'Machine Learning': 'Pandas'}, 'linkedin':
'jobs', 'nvidia': 'hardware'}
2
India
{1: 'Changed', 2: 'USA'}
[1, 2]
{2: 'USA'}
dict_values([{'Deep Learning': 'Python', 'Machine Learning': 'Pandas'}, 'jobs', 'har
dware'])
```

```
dict_keys(['datacamp', 'linkedin', 'nvidia'])
{2: 'UK', 3: 'nigeria'}
dict_items([(2, 'UK'), (3, 'nigeria')])
{'UK': 2, 'nigeria': 7}
```

Loops and Iterations:

1. For
2. While

In [16]:
```
# required for exiting the code and displaying the error
import sys
a=1
while a<4:
    if a==3:
        sys.exit("a==3")
    print(a)
    a+=1
```

```
1
2
An exception has occurred, use %tb to see the full traceback.

SystemExit: a==3
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3426: Us
erWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In [17]:
```
#FOR LOOP:
for i in range(2,7):
    print(i,end=",") #by default value is \n
print("\n")

# in order to print range -> typecast to list -> range(n) has elements from 0 to n-1
print(list(range(1,7,2)))

#Looping through dictionaries:
for k,val in dict1.items():
    print(k,val)

for k in dict1.keys():
    print(k)

#Using extra variable
l1=["hi","sneha"]
l2=[]
for i in l1:
    l2.append(i.title())
print(l2)

#without using extra variable
for i in range(len(l1)):
    l1[i]=l1[i].upper()
print(l1)
```

```
2,3,4,5,6,

[1, 3, 5]
2 UK
3 nigeria
2
3
```

```
['Hi', 'Sneha']
['HI', 'SNEHA']
```

Functions:

In [18]:
```python
# Positional Arguments -> required when calling the function
def func1 (n):
    return n*n

print(func1(3))

# Default parameters
def func2(name,age=23):
    greet="hi {}! Your age is {}".format(name,age)
    return greet

print(func2('Sneha'))

# Order of parameters matter

# Variable length parameters:
def func3(*n):
    res=sum(n)
    return res

print(func3(1,3,4))
print(func3(1,3))

##Lambda Functions: to define function in single line
lamb1=lambda x:"even" if x%2 == 0 else "odd"
print(lamb1(2))

#Lambda with list comprehension
lamb2 = lambda :[x for x in range(5) if x%2==0]
lamb2()

lamb3 =lambda x:x.upper()
print(lamb3('hello'))

# Map() -> maps a collection to another collection based on certain functionality
lm1=["hello","hi"]
lm2=map(lambda x:x.upper(),lm1)
print(list(lm2)) # need to typecast bcz map gives an object

#Filter()
lf1=[1,2,3,4,5,6,7,8]
lf2=filter(lambda x:x%3==0,lf1)
print(list(lf2))

#Reduce() -> gives single o/p
from functools import reduce
lr1=reduce(lambda x,y:x+y,lf1)
print(lr1)
```

```
9
hi Sneha! Your age is 23
8
4
even
HELLO
['HELLO', 'HI']
[3, 6]
36
```

Class and Objects:

1. Each class has a constructor -> __init\(self)

In [19]:
```
class Rect1:
    def __init__(self): #refers to newly created instance of a class
        self.length=10 # defining attributes

rec1=Rect1() # object creation
print(rec1.length) # printing the attribute value via object

#Parameterized constructor: -> to dynamically assign values during object creation
class Rect2:
    def __init__(self,length):
        self.Length=length
rec2=Rect2(20)
print(rec2.Length)


#Class and instance variables:
class Student:
    teacher="A" #Class variables -> shared variables
    classroom = "5-A" #Class variables -> shared variables
    def __init__(self,Name):
        self.name=Name #Instance Variables -> Unique Variables

s1=Student('stu1')
s2=Student('stu2')

print("Name:{},Teacher:{}".format(Student.teacher,s1.name)) # class variables can be
print("Name:{},Teacher:{}".format(s2.teacher,s2.name))

Student.teacher="B"
print("Name:{},Teacher:{}".format(Student.teacher,s1.name))

s2.teacher="C" # changed specifically for s2 only
print("Name:{},Teacher:{}".format(Student.teacher,s1.name))
print("Name:{},Teacher:{}".format(s2.teacher,s2.name))

#Methods: Class, static, instance
class cls1:
    def __init__(self,length):
        self.len=length
        self.brd=4
    def func1(self,br): # INstance methods
        return self.len+br
    @classmethod #Decorator to identify
    def func2(cls): # takes parameter 'cls' as first implicit argument just like ins
        pi=3.14
        return pi
    @staticmethod
    def func3():
        print("static")


cl1=cls1(3)
print(cl1.func1(3)) #calling instance method

print(cls1.func2()) #calling class method
cls1.func3() #calling static method


#Class inheritance and overriding: -> Reusability of code, Transitive-> if B inherit
class parent:

    def func1(self):
```

```
        pass

class child(parent): # inheriting parent class
    pi=3.14
    def __init__(self,rad):
        self.s=2
        self.rad=rad
    def func1(self): # overriding func1 method from parent
        return self.rad* child.pi*self.s

ch1=child(3)
print(ch1.func1())
```

```
10
20
Name:A,Teacher:stu1
Name:A,Teacher:stu2
Name:B,Teacher:stu1
Name:B,Teacher:stu1
Name:C,Teacher:stu2
6
3.14
static
18.84
```

NUMPY:

1. Stands for numerical Python.
2. Basic data structure for Numpy is Array.
3. Similar to list but has more functionalities and is fast.
4. Numpy is written in C and C is one of the fastest low level language, hence making numpy fast.
5. Numpy arrays are more compact than list and hence they take less storage.

List v/s Numpy:

1. Can iterate over elements of a numpy array without loop.
2. Can perform mathematical operations between numpy arrays easily.

In [20]:
```
#Numpy import
import numpy as np
#converting normal array to numpy array
nl1=[1,2,3,4]
print("\n nl1 type \n",type(nl1))
np1=np.array(nl1)
print("\n np1 type \n",type(np1))

#Mathematical operations on numpy array: - Multiplication
print(f" \n new numpy array is {np1*2} after multiplication and np1 is still the sam

#Mathematical operations on numpy array: - Addition of numpy arrays
np2=[1,0,1,1]
print(f"\n np1+np2 is {np1+np2}") #In list it concatenates but here it adds the elem

#Check length of numpy array:
print(len(np1)) #can be used to check the size in case of 1D array only because for
print(np1.size)
print(np1.shape) #can be used to check the size in case of 1D array only.

#Slicing & indexing same as lists

#Conditional Subset:
```

```
# get all even elements:
print(np1%2==0) # gives true false
print(np1[np1%2==0]) #filters out true elements

#get max/min element:
print(np1.max()) #numpy function
print(max(np1)) #list function -> both works
print(np1.min())

#get mean
print(np1.mean())

#2D ARRAYS:
#creation:
np3=np.array([[1,2,3,4.5],[False,"hello",7,8]])
np4=np.array([[1,2,3,4],[5,6,7,8]])
print(np3)
print("np4",np4)

#Get the dimension:
print(f"np3.ndim: {np3.ndim}")

#get shape: - basically 2 rows and 4 columns in case of 2D array
print(np3.shape)

#Get size or total number of elements:
print(np3.size)

#get datatype of numpy array: - gives the most common datatype depending on the elem
print(f"np3.dtype: {np3.dtype}")

#get the size of single item:
print(f"np3.itemsize: {np3.itemsize}")

#Broadcasting: -> ex.Multiply different elements of the array with different values
print(f"broadcasting example: {np4*[2,3,0,1]}")

# Axis in Numpy:
print('''
Dimensions are knows as axis
for a 2D array, axis = 0 for rows and axis = 1 for columns
''')

#Subsetting and Slicing for 2D array:
print("Subsetting and Slicing for 2D array:")
print(f" To get the second element of the first row:\n{np3[0][1]}")
print(np3[0,1]) # gives same result as np3[0][1]
print(f"To get the value of 1st column for all rows:\n{np3[:,0]}")
print("\n\n",np3[:,0:3],"\n\n") #to get all row values for first 3 columns,0,1,2
print("Subsetting",np4[np4[:,1]==2],"\n\n")

#CHanging shape of arrays:
print(np.array([0,1,2],ndmin=2)) #converted 1D array to 2D -> adds brackets
print(np.array(np3,ndmin=3)) #converted 2D array to 3D

#Creating 3D array:
np5=np.array([[[1,2,3,4],[5,6,7,8]],[[9,10,11,12],[13,14,15,16]],[[19,110,111,112],[
print(np5)
print(f"np5.ndim: {np5.ndim}")
```

```
nl1 type
<class 'list'>

np1 type
<class 'numpy.ndarray'>
```

```
 new numpy array is [2 4 6 8] after multiplication and np1 is still the same [1 2 3
4]

 np1+np2 is [2 2 4 5]
4
4
(4,)
[False  True False  True]
[2 4]
4
4
1
2.5
[['1' '2' '3' '4.5']
 ['False' 'hello' '7' '8']]
np4 [[1 2 3 4]
 [5 6 7 8]]
np3.ndim: 2
(2, 4)
8
np3.dtype: <U32
np3.itemsize: 128
broadcasting example: [[ 2  6  0  4]
 [10 18  0  8]]

Dimensions are knows as axis
for a 2D array, axis = 0 for rows and axis = 1 for columns

Subsetting and Slicing for 2D array:
 To get the second element of the first row:
2
2
To get the value of 1st column for all rows:
['1' 'False']


 [['1' '2' '3']
 ['False' 'hello' '7']]


Subsetting [[1 2 3 4]]


[[0 1 2]]
[[['1' '2' '3' '4.5']
  ['False' 'hello' '7' '8']]]
[[[  1   2   3   4]
  [  5   6   7   8]]

 [[  9  10  11  12]
  [ 13  14  15  16]]

 [[ 19 110 111 112]
  [113 114 115 116]]]
np5.ndim: 3
```

In [21]:

```
#Creating various Numpy arrays:
print(f"array of 1s: \n {np.ones([5,2])}")
print(f"\n Typecasting array of 1s to get int: \n {np.ones([5,2],dtype='int')}")
print(f"\n array of 0s: \n {np.zeros([3,2],dtype='int')}")
print(f"\n array of random integers within a particular range: \n {np.random.randint
print(f"\n array of random numbers within a particular range: \n {np.random.random([
print(f"\n array with increment of fixed step size: \n {np.arange(1,7,2)}") # by def
#reshaping the arrays:
print(f"\n Convert 1D arange elements to ndimensional \n {np.arange(1,7).reshape(2,3
print(f"\n Typecasting of np.arange: \n {np.arange(1.0,7,2)}")
print(f"\n array of fixed length: \n {np.linspace(1,7,10,dtype='int')}") #by default
print(f"\n To get an empty array with placeholders: \n {np.empty(6).reshape(3,2)}")
```

```
array of 1s:
 [[1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]]

 Typecasting array of 1s to get int:
 [[1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]]

 array of 0s:
 [[0 0]
 [0 0]
 [0 0]]

 array of random integers within a particular range:
 [3 1 1 2 3 1 1 1 3 4]

 array of random numbers within a particular range:
 [[0.39769489 0.86559152]
 [0.95982199 0.06407048]
 [0.61862463 0.85175611]]

 array with increment of fixed step size:
 [1 3 5]

 Convert 1D arange elements to ndimensional
 [[1 2 3]
 [4 5 6]]

 Typecasting of np.arange:
 [1. 3. 5.]

 array of fixed length:
 [1 1 2 3 3 4 5 5 6 7]

 To get an empty array with placeholders:
 [[0.39769489 0.86559152]
 [0.95982199 0.06407048]
 [0.61862463 0.85175611]]
```

In [22]:

```
#Mathematical Operations on Numpy Array: -> Arithmatic operations cannot be done whe

#Stacking Arrays:
nh1=np.array([1,2,3])
nh2=np.array([2,3,4])
print(f" nh1 is \n {nh1} \n and nh2 is \n {nh2}")
print(f"\n Horizontal Stacking \n {np.hstack((nh1,nh2))}") #stacks horizontally -> a
print(f"\n Vertical Stacking \n {np.vstack((nh1,nh2))}")#stacks vertically -> arrays
print(f"\n To raise all elements to a particular power \n {np.power(nh1,3)}")
print(f"\n To get absolute values using numpy function \n {np.absolute(nh1)}")
print(f"\n To get absolute values without using numpy function \n {abs(nh1)}")

#Trigo Functions:
theta=[0,30,45,60,90]
print(f"\n To get pi value \n {np.pi}")
print(f"\n To get sin values of all elements in an array \n {np.sin(theta)}")
print(f"\n To get cos values of all elements in an array \n {np.cos(theta)}")
print(f"\n To get tan values of all elements in an array \n {np.tan(theta)}")
print(f"\n To get e^element \n {np.exp(nh1)}") # e = 2.718...
print(f"\n To get 2^element \n {np.exp2([nh1])}")
print(f"\n To get n^element in int where n is any random value \n {np.exp2([nh1]).as
print(f"\n To get log values for each element in the array->log2 \n {np.log2(nh1)}")
print(f"\n To get log values for each element in the array->log10 \n {np.log10(nh1)}
```

```
print('''
Another one liner way to write the same thing as below
x=np.arange(1,7,2),
y=x*10 \n\n
First create an empty array y
and then use multiply function with out parameter
''')
y=np.empty(3)
print(f"{np.multiply(nh1,10, out=y)}") #can be used with other functions too -> shap

#Aggregates
print(f"\n To get the sum of all the values in the array \n {np.add.reduce(nh2)}") #
print(f"\n To get the multiplication value of all the values in the array \n {np.mul
print(f"\n To get the sum and store it element wise in one place too \n {np.add.accu
```

```
nh1 is
[1 2 3]
and nh2 is
[2 3 4]

Horizontal Stacking
[1 2 3 2 3 4]

Vertical Stacking
[[1 2 3]
[2 3 4]]

To raise all elements to a particular power
[ 1  8 27]

To get absolute values using numpy function
[1 2 3]

To get absolute values without using numpy function
[1 2 3]

To get pi value
3.141592653589793

To get sin values of all elements in an array
[ 0.         -0.98803162  0.85090352 -0.30481062  0.89399666]

To get cos values of all elements in an array
[ 1.          0.15425145  0.52532199 -0.95241298 -0.44807362]

To get tan values of all elements in an array
[ 0.         -6.4053312   1.61977519  0.32004039 -1.99520041]

To get e^element
[ 2.71828183  7.3890561  20.08553692]

To get 2^element
[[2. 4. 8.]]

To get n^element in int where n is any random value
[[2 4 8]]

To get log values for each element in the array->log2
[0.        1.        1.5849625]

To get log values for each element in the array->log10
[0.         0.30103    0.47712125]

Another one liner way to write the same thing as below
x=np.arange(1,7,2),
y=x*10
```

```
First create an empty array y
and then use multiply function with out parameter

[10. 20. 30.]

 To get the sum of all the values in the array
 9

 To get the multiplication value of all the values in the array
 24

 To get the sum and store it element wise in one place too
 [1 3 6]
```

In [23]:
```
#Linear Algebra Operations using numpy arrays: uses np.linalg package
print('''
Inverse of a matrix - not all matrices have inverses.
A.(A^-1) = I, where I is identity matrix
\n
Identity matrix properties:
- Always a square(same no. of rows and columns) -ex. 2x2,3x3 etc.
- diagonal(\) filled with zeros, rest all ones
''')
la1=[[7,-2],[3,7]]
la2=[[1,2],[3,4]]
print(f"\n la1: \n{la1}")
print(f"\n Inverse of a matrix: \n{np.linalg.inv(la1)}")
print(f"\n dot product to check: \n {np.around((np.dot(la1,np.linalg.inv(la1))))}")
print(f"\n Determinant: \n {(np.linalg.det(la1))}")
print(f"\n Eigen values and eigen vectors: \n {(np.linalg.eig(la1))}")
print(f"\n Matrix Multiplication: \n {(np.matmul(la1,la2))}")
print(f"\n Matrix Multiplication via dot(): \n {(np.dot(la1,la2))}")
print(f"\n Matrix Rank: \n {(np.linalg.matrix_rank(la1))}") # No. of independent row
print(f"\n Trace of a matrix - Sum of elements in the diagonal(\): \n {(np.trace(la1
print(f"\n Matrix^n where n is random number: \n {(np.linalg.matrix_power(la1,3))}")
```

```
Inverse of a matrix - not all matrices have inverses.
A.(A^-1) = I, where I is identity matrix


Identity matrix properties:
- Always a square(same no. of rows and columns) -ex. 2x2,3x3 etc.
- diagonal(\) filled with zeros, rest all ones


 la1:
[[7, -2], [3, 7]]

 Inverse of a matrix:
[[ 0.12727273  0.03636364]
 [-0.05454545  0.12727273]]

 dot product to check:
 [[ 1.  0.]
 [-0.  1.]]

 Determinant:
 55.000000000000014

 Eigen values and eigen vectors:
 (array([7.+2.44948974j, 7.-2.44948974j]), array([[0.        +0.63245553j, 0.
-0.63245553j],
       [0.77459667+0.j        , 0.77459667-0.j        ]]))

 Matrix Multiplication:
 [[ 1  6]
 [24 34]]
```

```
Matrix Multiplication via dot():
[[ 1  6]
[24 34]]

Matrix Rank:
2

Trace of a matrix - Sum of elements in the diagonal(\):
14

Matrix^n where n is random number:
[[ 217 -282]
[ 423  217]]
```

In [ ]:
```
#Numpy v/s Lists - Computational time:
import time
ct1=[i for i in range(100000)]
ct2=[i for i in range(100000)]
cnp1=np.array(ct1)
cnp2=np.array(ct2)
t0=time.time()
res=cnp1*cnp2
t1=time.time()
print(t1-t0) #by numpy array
```

In [ ]:
```
import time
t3=time.time()
ct3=[i*j for i,j in zip(ct1,ct2)]
t4=time.time()
print(t4-t3) # by lists
```

In [24]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\programdata\anaconda3\lib\site-packag
es (6.0.7)
Requirement already satisfied: entrypoints>=0.2.2 in c:\programdata\anaconda3\lib\si
te-packages (from nbconvert) (0.3)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\programdata\anaconda3\lib
\site-packages (from nbconvert) (1.4.3)
Requirement already satisfied: jupyterlab-pygments in c:\programdata\anaconda3\lib\s
ite-packages (from nbconvert) (0.1.2)
Requirement already satisfied: pygments>=2.4.1 in c:\programdata\anaconda3\lib\site-
packages (from nbconvert) (2.7.2)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\programdata\anaconda3\lib\sit
e-packages (from nbconvert) (0.8.4)
Requirement already satisfied: traitlets>=4.2 in c:\programdata\anaconda3\lib\site-p
ackages (from nbconvert) (5.0.5)
Requirement already satisfied: jinja2>=2.4 in c:\programdata\anaconda3\lib\site-pack
ages (from nbconvert) (2.11.2)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\programdata\anaconda3\li
b\site-packages (from nbconvert) (0.5.1)
Requirement already satisfied: testpath in c:\programdata\anaconda3\lib\site-package
s (from nbconvert) (0.4.4)
Requirement already satisfied: jupyter-core in c:\programdata\anaconda3\lib\site-pac
kages (from nbconvert) (4.6.3)
Requirement already satisfied: nbformat>=4.4 in c:\programdata\anaconda3\lib\site-pa
ckages (from nbconvert) (5.0.8)
Requirement already satisfied: defusedxml in c:\programdata\anaconda3\lib\site-packa
ges (from nbconvert) (0.6.0)
Requirement already satisfied: bleach in c:\programdata\anaconda3\lib\site-packages
(from nbconvert) (3.2.1)
Requirement already satisfied: ipython-genutils in c:\programdata\anaconda3\lib\site
-packages (from traitlets>=4.2->nbconvert) (0.2.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\programdata\anaconda3\lib\site
-packages (from jinja2>=2.4->nbconvert) (1.1.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\programdata\anaconda3\lib
```

```
\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (6.1.7)
Requirement already satisfied: nest-asyncio in c:\programdata\anaconda3\lib\site-pac
kages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.4.2)
Requirement already satisfied: async-generator in c:\programdata\anaconda3\lib\site-
packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.10)
Requirement already satisfied: pywin32>=1.0; sys_platform == "win32" in c:\programda
ta\anaconda3\lib\site-packages (from jupyter-core->nbconvert) (227)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\programdata\anaconda3\l
ib\site-packages (from nbformat>=4.4->nbconvert) (3.2.0)
Requirement already satisfied: webencodings in c:\programdata\anaconda3\lib\site-pac
kages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packag
es (from bleach->nbconvert) (20.4)
Requirement already satisfied: six>=1.9.0 in c:\programdata\anaconda3\lib\site-packa
ges (from bleach->nbconvert) (1.15.0)
Requirement already satisfied: pyzmq>=13 in c:\programdata\anaconda3\lib\site-packag
es (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (19.0.2)
Requirement already satisfied: tornado>=4.1 in c:\programdata\anaconda3\lib\site-pac
kages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.0.4)
Requirement already satisfied: python-dateutil>=2.1 in c:\programdata\anaconda3\lib
\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.
1)
Requirement already satisfied: attrs>=17.4.0 in c:\programdata\anaconda3\lib\site-pa
ckages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (20.3.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\programdata\anaconda3\lib\si
te-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (0.17.3)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packa
ges (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (50.3.1.post20201107)
Requirement already satisfied: pyparsing>=2.0.2 in c:\programdata\anaconda3\lib\site
-packages (from packaging->bleach->nbconvert) (2.4.7)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: