WEEK-1

1)Exceptions handling and user defined exception(s)

i) Value error

```
import math
print(math.sqrt(-9))
```

output:

```
ValueError                    Traceback (most recent call last)
<ipython-input-2-b48664711132> in <module>()
     1 import math
----> 2 print(math.sqrt(-9))

ValueError: math domain error
```

After correction:

```
import math
print(math.sqrt(9))
```

output:

3

ii)Index error

```
L1=[1,2,3,4]
print(L1[4])
```

output:

```
IndexError                    Traceback (most recent call last)
<ipython-input-5-6f58254488d8> in <module>()
     1 L1=[1,2,3,4]
----> 2 print(L1[4])

IndexError: list index out of range
```

After correction:

```
L1=[1,2,3,4]
print(L1[0])
```

output: 1

iii) Type error

```
print('2'+23)
```

output:

```
TypeError                     Traceback (most recent call last)
<ipython-input-14-6975f1a96fbc> in <module>()
```

```
----> 1 print('2'+23)

TypeError: can only concatenate str (not "int") to str
```

After correction:

```
print('ece '+ '2')
output: ece2
```

iv)Name Error

```
a=5
print(a+b)
a=8
b=9
print(a/b)
```

output:

```
NameError                                 Traceback (most recent call last)
<ipython-input-8-ce7df440588b> in <module>()
     1 a=5
----> 2 print(a+b)

NameError: name 'b' is not defined
```

After correction:

```
a=4
b=3
print(a+b)
```

output:7

v)ZeroDivisionError

```
a=7
b=0
print(a/b)
```

output:

```
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-17-bfcc22604807> in <module>()
     1 a=7
     2 b=0
----> 3 print(a/b)

ZeroDivisionError: division by zero
```

After correction:

```
a=8
b=4
print(a/b)
```

output: 2.0

B) Write a python program to create user defined exceptions.

```
class MyError(Exception):
  def __init__(self,value):
    self.value = value
  def __str__(self):
    return(repr(self.value))
try:
  raise(MyError(7*1))
except MyError as error:
   print("error occured at",error.value)
```

output:

```
error occured at 7
```

C) Write a python program to understand the use of else and finally block with try block.

```
try:
  a=int(input("enter a: "))
  b=int(input("enter b: "))
  c=a/b
  print("a/b=",c)
except Exception:
  print("can't divide by zero")
  print(Exception)
else:
  print("else block")
finally:
  print("end of the program")
```

output:

```
enter a: 8
enter b: 0
can't divide by zero
<class 'Exception'>
end of the program
```

D) Write a python program that uses raise and exception class throw an exception.

```
try:
  age=int(input("enter age: "))
  if(age<18):
    raise ValueError
    print("don't have right to vote")
  else:
    print("has right to vote")
except ValueError:
  print("don't have right to vote")
```

output:

```
enter age: 24
has right to vote
```

WEEK-2

Modules and Packages:

a)Write a python program to create a module and import the module in another python program.

```
#swapping.py
def swap(a,b):
    a=a+b
    b=a-b
    a=a-b
    print("numbers after swapping are",a,b)
```

```
#swap.py
import swapping
a=23
b=42
print("numbers before swapping are :",a,b)
swapping.swap(a,b)
```

Output: swap.py

```
numbers before swapping are : 23 42
numbers after swapping are 42 23
```

B)Write a python program to import all objects from a module, specific object from module and provide custom import name to the import name to the imported object from the module.

```
# a simple module calculator.py
def min(x,y):
```

```
    if(x<y):
        return x
    else:
        return y
def max(x,y):
    if(x>y):
        return x
    else:
        return y
def square(x):
    return(x*x)
def mod(x,y):
    return(x%y)
def cube(x):
    return(x*x*x)
```

```
#specific.py
from calculator import square,cube
print(square(4))
print(cube(7))
```

Output: specific.py

```
16
49
#all.py
from calculator import *
print(max(4,7))
print(min(23,45))
```

Output: all.py

```
7
23
```

```
#customname.py
Import  calculator as c
a=c.square(25)
b=c.mod(23,11)
print(a)
print(b)
```

Output: customname.py

```
625
1
```

C)Create a python package having at least two modules in it.

```
#module 1
def hello(name):
   print("Hello",name)

#module2.py
def message():
   print("GAYATRI VIDYA PARISHADH COLLEGE!!")

#file.py
from package import module1 as m1
from package import module2 as m2
m1.hello('world')
m2.message()
```

Output: file.py

Hello divya
GAYATRI VIDYA PARISHADH COLLEGE!!"

D)Create a python package having at least one sub package in it.

```
#module_m
def swap(a,b):
   print("before swapping ",a,b)
   t=a
   a=b
   b=t
   print("after swapping",a,b)

#f2.py
from package.sub_package import module_m as m
m.swap(7,9)
```

Output: f2**.py**

before swapping 7 9
after swapping 9 7

WEEK-3

Numpy Library:

A)Create a numpy array from list,tuple with float type.

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]], dtype='float')
b=np.array(((1,2,3),(4,5,6)),dtype='float')
print(type(a))
print(type(b))
print(a)
print(b)
```

Output:

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
[[1. 2. 3.]
 [4. 5. 6.]]
[[1. 2. 3.]
 [4. 5. 6.]]
```

B)Python program to demonstrate slicing,integer,and Boolean array indexing;

```
import numpy as np
print("slicing")
arr=np.array([[1.2,3.4,5.6],[2.3,-4.7,8.3],[-1,2,-3]])
a= arr[:2,:2]
f= arr[:2,::2]
print(a)
print(f)
e= arr[1:,1::]
print(e)
print(" integer indexing")
b=arr[[2,1,0],[1,2,0]]
print(b)
print("boolean indexing")
cond1 = arr>0
c=arr[cond1]
print(c)
cond2 = arr<0
d=arr[cond2]
print(d)
```

Output:

```
slicing
[[ 1.2  3.4]
 [ 2.3 -4.7]]
[[1.2 5.6]
 [2.3 8.3]]
[[-4.7  8.3]
 [ 2.  -3. ]]
 integer indexing
[2.  8.3 1.2]
boolean indexing
[1.2 3.4 5.6 2.3 8.3 2. ]
[-4.7 -1.  -3. ]
```

C)Write a python program to find min, max, sum, cumulative sum of array.

```
import numpy as np
a=np.array([[1.2,3.4,5.6],[2.3,4.7,8.3]])
print("maximum element in the array:",a.max())
print("maximum element in the array row wise:",a.max(axis = 1))
print("minimum element in the array column-wise:",a.min(axis =0))
print("minimum element in the array:",a.min())
print("sum of all elements in the array: ",a.sum())
print("sum of all elements in the array: ",a.cumsum(axis = 1))
```

Output:

```
maximum element in the array: 8.3
maximum element in the array row wise: [5.6 8.3]
minimum element in the array column-wise: [1.2 3.4 5.6]
minimum element in the array: 1.2
sum of all elements in the array:  25.5
sum of all elements in the array:  [[ 1.2  4.6 10.2]
 [ 2.3  7.  15.3]]
```

D)Write a python program to demonstrate use of ndim, shape, size, dtype.

```
import numpy as np
a=np.array([[1.2,3.4,5.6],[2.3,4.7,8.3]])
b=np.array([[2,1,0],[1,2,0]])
print("number of dimensions :",a.ndim)
print("shape of array",a.shape)
print("size of array a : ",a.size)
print("data type of array a:",a.dtype)
print("data type of array b:",b.dtype)
```

Output:

number of dimensions : 2
shape of array (2, 3)
size of array a : 6
data type of array a: float64
data type of array b: int64

WEEK-4

Numpy Library: Linear algebra

A)Write a python program to find rank,determinant,and trace of an array.

```
import numpy as np
X=[[1,2,3],[4,5,6],[7,8,9]]
Y=[[0,2,4],[1,3,6],[1,2,3]]
x=np.array(X)
y=np.array(Y)
print(x)
print(y)
r=np.linalg.matrix_rank (x)
d=np.linalg.det(x)
t=x.trace()
rank=np.linalg.matrix_rank(y)
det=np.linalg.det(y)
trace=y.trace()
print("Rank is ",r)
print("Determinant is ",d)
print("Trace is ",t)
print("Rank is ",rank)
print("Determinant is ",det)
print("Trace is ",trace)
```

output:

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[0 2 4]
 [1 3 6]
 [1 2 3]]
Rank is  2
Determinant is  0.0
Trace is  15
Rank is  3
Determinant is  2.0
```

```
Trace is  6
```

B)Write a python program to find eigen values of matrices

```
import numpy as np
A=[[1,2,3],[4,5,6],[7,8,5]]
a=np.array(A)
b,c=np.linalg.eig(a)
print(b)
#print(c)
```

output:

```
[14.0500928  -0.31191836 -2.73817444]
```

C)Write a python program to find matrix and vector products(dot,inner,outer product),matrix exponentiation.

```
import numpy as np
a=np.array([[1,2],[3,4]])
b=np.array([[5,6],[7,8]])
x=np.array((1,2,3))
y=np.array((4,5,6))
print("dot product is ",np.dot(a,b))
print("inner product is ",np.inner(a,b))
print("exponential product is ",np.exp(a))
print("dot product is ",np.dot(x,y))
print("inner product is ",np.inner(x,y))
```

output:

```
dot product is  [[19 22]
 [43 50]]
inner product is  [[17 23]
 [39 53]]
exponential product is  [[ 2.71828183  7.3890561 ]
 [20.08553692 54.59815003]]
dot product is  32
inner product is  32
```

D)Write a python program to solve a linear matrix equation,or system of linear scalar equation

```
import numpy as np
A=[[1,2],[3,4]]
B=[[5,6],[7,8]]
```

```
a=np.array(A)

b=np.array(B)

C=np.linalg.solve(a,b)

print(C)
```

output:

```
[[-3. -4.]
 [ 4.  5.]]
```

WEEK-5

Numpy Advanced

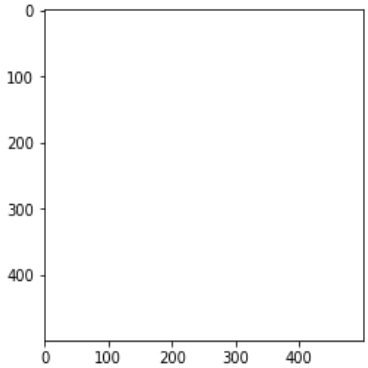
(A) Create a white image using NumPy in Python

```
import numpy as np

import matplotlib.pyplot as plt

Whiteimage=np.full((500,500,3),255,dtype=np.uint8)

print(Whiteimage)

plt.imshow(Whiteimage)
```

output:

```
[[[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]]

 [[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]]
```

```
[[255 255 255]
 [255 255 255]
 [255 255 255]
 ...
 [255 255 255]
 [255 255 255]
 [255 255 255]]

...

[[255 255 255]
 [255 255 255]
 [255 255 255]
 ...
 [255 255 255]
 [255 255 255]
 [255 255 255]]

[[255 255 255]
 [255 255 255]
 [255 255 255]
 ...
 [255 255 255]
 [255 255 255]
 [255 255 255]]

[[255 255 255]
 [255 255 255]
 [255 255 255]
 ...
 [255 255 255]
 [255 255 255]
 [255 255 255]]]
<matplotlib.image.AxesImage at 0x7f70179d3790>
```

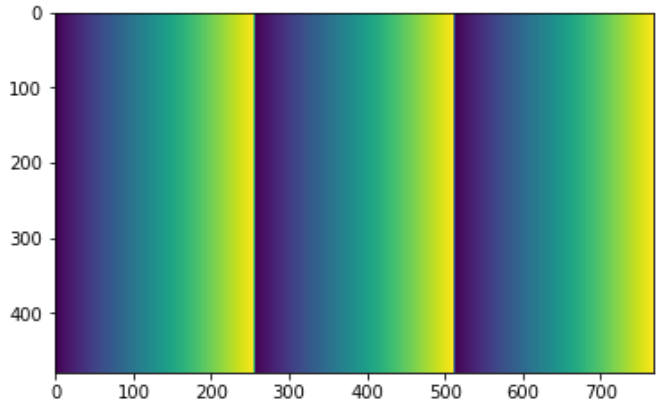


(B) Convert a NumPy array to an image and Convert images to NumPy array?

```
import matplotlib.pyplot as plt

import numpy as np

from PIL import Image as img

a=np.arange(0,368640,1,dtype=np.uint8)

pic=a.reshape(480,768)

IM=img.fromarray(pic,'L')

IM.save('week5.png')

plt.imshow(IM)

image=img.open('week5.png')

imagearray=np.array(image)

print(type(imagearray))

print(np.shape(imagearray))

print(imagearray)
```

output:

```
<class 'numpy.ndarray'>
(480, 768)
[[  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]
 ...
 [  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]]
```



(C)perform sorting,searching,counting using numpy methods

```
import numpy as lab

a=lab.array([8,5,3,0,8,0,6,5,6,9,9,5])

b=a.reshape(4,3)

print(a)

print(b)

print("searching")

print(lab.where(a==5)) #searching even numbers

print(b.sort())

print(lab.searchsorted(a,[3])) #applying searchsorted on 1d array

print("sorting")

print(lab.sort(b)) #sorting the 2d array

print("counting")

print(lab.count_nonzero(b % 2 == 1, axis=0)) #counting odd numbers

print(lab.count_nonzero(b % 2 == 0, axis=1)) #counting even numbers
```

output:

```
[8 5 3 0 8 0 6 5 6 9 9 5]

[[8 5 3]
```

```
 [0 8 0]

 [6 5 6]

 [9 9 5]]

searching

(array([ 1,  7, 11]),)

None

[5]

sorting

[[3 5 8]

 [0 0 8]

 [5 6 6]

 [5 9 9]]

counting

[3 2 1]

[1 3 2 0]
```

(D)write a program to demonstrate the use of reshape() method

```
import numpy as lab

a=lab.array([8,5,3,0,8,0,6,5,6,9,9,5])

print("original array\n",a)

print("2d array with 4 rows and 3 columns\n",a.reshape(6,2))

print("2d array with 6 columns and 2 rows\n",a.reshape(2,6))

print("3d array with 6 rows, 2 columns and 1 layers\n",a.reshape(2,2,3))
```

output:

original array
[8 5 3 0 8 0 6 5 6 9 9 5]
2d array with 4 rows and 3 columns
[[8 5]

[3 0]
[8 0]
[6 5]
[6 9]
[9 5]]
2d array with 6 columns and 2 rows
[[8 5 3 0 8 0]
[6 5 6 9 9 5]]
3d array with 6 rows, 2 columns and 1 layers
[[[8 5 3]
[0 8 0]]

[[6 5 6]
[9 9 5]]]