RAMNIRANJAN JHUNJHUNWALA COLLEGE GHATKOPAR (W), MUMBAI - 400 086

DEPARTMENT OF INFORMATION TECHNOLOGY

2023 - 2024

T.Y. B. SC.(I.T.) SEM VI
RJSUITP604
GEOGRAPHIC INFORMATION SYSTEM

NAME : SHIVAM PARMESH PRAJAPATI
ROLL NO. :6472

Hindi Vidya Prachar Samiti's

Ramniranjan Jhunjhunwala College of Arts, Science & Commerce

(Empowered Autonomous College)

Affiliated to

UNIVERSITY OF MUMBAI

*This is to certify that Mr. **Prajapati Shivam Parmesh Tara Devi**, Roll No. **6472** of TY BSc IT class has completed the Case Study of **Principles of Geographic Information System**, in partial fulfillment of the Requirements for the award of the degree of Bachelor of Science (**Information Technology**) during the academic year 2023- 2024.*

College seal

Sign of Co-ordinator

INDEX

Sr. No.	Details	Date
1.	NumPy, Pandas, Matplotlib and Seaborn Basics.	15/07/24
2.	Collecting and loading structured and unstructured data.	
3.	Using Data Wrangling processes: Data discovery, data pre-processing, data validation etc. for various types of data.	
4.	Basic utility design, Data auditing and Exploratory Data Analysis.	
5.	Retrieve Superstep.	
6.	Access Superstep.	
7.	Processing Data.	
8.	Transforming Data: Using Machine Learning Algorithms.	
9.	Organising and Generating data.	
10.	Data Visualization.	

Practical 1
NumPy, Pandas, Matplotlib and Seaborn Basics.

☑ NumPy

➔ **NumPy Documentation :-** https://numpy.org/doc/stable/
➔ **JupyterLite :-** https://jupyter.org/try-jupyter/lab/

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Ndarray Basic Operations

```
In [1]: import numpy as np
        a = np.arange(15).reshape(3, 5)
        a
Out[1]: array([[ 0,  1,  2,  3,  4],
               [ 5,  6,  7,  8,  9],
               [10, 11, 12, 13, 14]])

In [2]: a.shape
Out[2]: (3, 5)

In [4]: a.ndim
Out[4]: 2

In [5]: a.dtype.name
Out[5]: 'int32'

In [8]: a.itemsize
Out[8]: 4

In [9]: a.size
Out[9]: 15

In [10]: type(a)
Out[10]: numpy.ndarray
```

Array creation

```
In [11]: b = np.array([1,2,3])

In [12]: type(b)
Out[12]: numpy.ndarray

In [13]: b.ndim
Out[13]: 1

In [14]: b
Out[14]: array([1, 2, 3])
```

Changing the array dimensions

```
In [17]: d = c.reshape(4,3)
         d
Out[17]: array([[1, 2, 5],
                [8, 4, 5],
                [6, 9, 7],
                [8, 5, 6]])

In [18]: d.shape
Out[18]: (4, 3)

In [19]: d.ndim
Out[19]: 2
```

Array using tuple

```
In [20]: #Using a tuple to cretae a NUM Array
         e = np.array((7,8,5,'w',6))
         e
Out[20]: array(['7', '8', '5', 'w', '6'], dtype='<U11')

In [22]: e
Out[22]: array(['7', '8', '5', 'w', '6'], dtype='<U11')
```

2D array

```
In [25]: f = np.array([[1,2,3],[4,5,6]])
         f
Out[25]: array([[1, 2, 3],
                [4, 5, 6]])

In [26]: f.ndim
Out[26]: 2
```

3D array

```
In [27]: g = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
         g

Out[27]: array([[[ 1,  2,  3],
                 [ 4,  5,  6]],

                [[ 7,  8,  9],
                 [10, 11, 12]]])

In [28]: g.ndim

Out[28]: 3

In [30]: h = g.reshape(2,6)
         h

Out[30]: array([[ 1,  2,  3,  4,  5,  6],
                [ 7,  8,  9, 10, 11, 12]])

In [31]: h.ndim

Out[31]: 2
```

Accessing array elements

```
In [34]: print(h[1,5])

         12

In [35]: i = g[0,1,1] + g[1,1,1]
         i

Out[35]: 16
```

Multidimensional array

```
In [36]: j = np.array([[[10,20,30]],[[40,50,60]],[[10,15,60]],[[20,25,15]]])
         j

Out[36]: array([[[10, 20, 30]],

                [[40, 50, 60]],

                [[10, 15, 60]],

                [[20, 25, 15]]])
```

Slicing an array

```
In [38]: # Slicing An Array

         k = np.array([1,2,3,4,5,6,7])
         k
Out[38]: array([1, 2, 3, 4, 5, 6, 7])

In [39]: k[1:5]
Out[39]: array([2, 3, 4, 5])

In [40]: k[0:6]
Out[40]: array([1, 2, 3, 4, 5, 6])

In [41]: k[1:]
Out[41]: array([2, 3, 4, 5, 6, 7])

In [42]: k[:5]
Out[42]: array([1, 2, 3, 4, 5])

In [43]: k[1:5:2]
Out[43]: array([2, 4])

In [44]: k[:4:2]
Out[44]: array([1, 3])

In [45]: k[2::2]
Out[45]: array([3, 5, 7])

In [46]: k[::3]
Out[46]: array([1, 4, 7])
```

Creating identity and random matrix

```
In [53]: np.ones((3,3))

Out[53]: array([[1., 1., 1.],
                [1., 1., 1.],
                [1., 1., 1.]])

In [54]: # 3 set 2 row 4 column
         np.ones((3,2,4))

Out[54]: array([[[1., 1., 1., 1.],
                 [1., 1., 1., 1.]],

                [[1., 1., 1., 1.],
                 [1., 1., 1., 1.]],

                [[1., 1., 1., 1.],
                 [1., 1., 1., 1.]]])

In [55]: np.empty((2,3))

Out[55]: array([[1.24038692e-311, 1.52768399e+301, 1.24038694e-311],
                [1.24041184e-311, 6.56971658e+047, 1.24038693e-311]])

In [58]: np.arange(10,30,5)

Out[58]: array([10, 15, 20, 25])

In [59]: # random value from 0 t0 2 9 value
         np.linspace(0,2,9)

Out[59]: array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  ])
```

Array in Trigonometric

```
In [62]: from numpy import pi
         x = np.linspace(0,2*pi,100)
         y = np.sin(x)

In [63]: y

Out[63]: array([ 0.00000000e+00,  6.34239197e-02,  1.26592454e-01,  1.89251244e-01,
                 2.51147987e-01,  3.12033446e-01,  3.71662456e-01,  4.29794912e-01,
                 4.86196736e-01,  5.40640817e-01,  5.92907929e-01,  6.42787610e-01,
                 6.90079011e-01,  7.34591709e-01,  7.76146464e-01,  8.14575952e-01,
                 8.49725430e-01,  8.81453363e-01,  9.09631995e-01,  9.34147860e-01,
                 9.54902241e-01,  9.71811568e-01,  9.84807753e-01,  9.93838464e-01,
                 9.98867339e-01,  9.99874128e-01,  9.96854776e-01,  9.89821442e-01,
                 9.78802446e-01,  9.63842159e-01,  9.45000819e-01,  9.22354294e-01,
                 8.95993774e-01,  8.66025404e-01,  8.32569855e-01,  7.95761841e-01,
                 7.55749574e-01,  7.12694171e-01,  6.66769001e-01,  6.18158986e-01,
                 5.67059864e-01,  5.13677392e-01,  4.58226522e-01,  4.00930535e-01,
                 3.42020143e-01,  2.81732557e-01,  2.20310533e-01,  1.58001396e-01,
                 9.50560433e-02,  3.17279335e-02, -3.17279335e-02, -9.50560433e-02,
                -1.58001396e-01, -2.20310533e-01, -2.81732557e-01, -3.42020143e-01,
                -4.00930535e-01, -4.58226522e-01, -5.13677392e-01, -5.67059864e-01,
                -6.18158986e-01, -6.66769001e-01, -7.12694171e-01, -7.55749574e-01,
                -7.95761841e-01, -8.32569855e-01, -8.66025404e-01, -8.95993774e-01,
                -9.22354294e-01, -9.45000819e-01, -9.63842159e-01, -9.78802446e-01,
                -9.89821442e-01, -9.96854776e-01, -9.99874128e-01, -9.98867339e-01,
                -9.93838464e-01, -9.84807753e-01, -9.71811568e-01, -9.54902241e-01,
                -9.34147860e-01, -9.09631995e-01, -8.81453363e-01, -8.49725430e-01,
                -8.14575952e-01, -7.76146464e-01, -7.34591709e-01, -6.90079011e-01,
                -6.42787610e-01, -5.92907929e-01, -5.40640817e-01, -4.86196736e-01,
                -4.29794912e-01, -3.71662456e-01, -3.12033446e-01, -2.51147987e-01,
                -1.89251244e-01, -1.26592454e-01, -6.34239197e-02, -2.44929360e-16])
```

For more details :- https://numpy.org/doc/stable/user/quickstart.html

NumPy Functions and methods overview

Array cretation

arange, array, copy, empty, linspace, identity, ones, zero

Conversion
ndarray.astype, atleast_1d, atleast_2d, atleast_3d, mat

Manipulation
array_split, concatenate, digonal, ndarray.item, reshape, resize, transpose

Basic Statics
cov, mean, std, var

Operation
sum, cumsum, prod, put, compress

Basic algebra
cross, dot