# PYTHON DATASET ANALYSIS

# AN INTERNSHIP REPORT

**Submitted by**

**G. NAVEEN 320126510023**

# In fulfilment of the Summer Internship

**COMPUTER SCIENCE & ENGINEERING**



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCE

**(Affiliated to Andhra University)**

# SANGIVALASA, VISAKHAPATNAM – 531162 (2020-2024)

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING.**

# ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Affiliated to Andhra University) SANGIVALASA, VISAKHAPATNAM – 531162

2019-2023



# BONAFIDE CERTIFICATE

This is to certify that this internship report **“Python Dataset Analysis”** is the bonafide work of **G. Naveen (320126510023)** of III/IV CSE carried out the project work under my supervision.

## Dr. P E S N Krishna Prasad, M.Tech,Ph.D

(Professor & HOD) Dept. of CSE

ANITS

## Reviewer: Class Teacher:

**Dr. A. ROHINI, Ph.D. K. AMARAVATHI, M. Tech, (Ph.D.)**

(Associate Professor) (Assistant Professor)

Dept. of CSE Dept. of CSE

ANITS ANITS

# ACKNOWLEDGMENT

An endeavor over a long period can be successful with the advice and support of many well- wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We owe our tributes to Dr. P E S N Krishna Prasad, Head of the Department, Computer Science & Engineering, ANITS, for his valuable support and guidance during the period of internship.

We express our warm and sincere thanks for the encouragement, untiring guidance and the confidence they had shown in us. We are immensely indebted for their valuable guidance throughout our internship.

We also thank all the staff members of the CSE department for their valuable advice. We also thank supporting staff for providing resources as and when required.

**320126510023 G. NAVEEN**

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE-NO** |
| 1. | ABSTRACT | 1 |
| 2. | INTRODUCTION TO PYTHON | 2 |
| 3. | ABOUT THE PROJECT | 3 |
| 3. | PYTHON NUMPY | 4 |
| 4. | PYTHON PANDAS | 8 |
| 5. | PYTHON MATPLOTLIB | 11 |
| 6. | DATASETS | 15 |
| 7. | NOTEBOOKS | 16 |
| 8. | CONCLUSION | 31 |
| 9. | REFERENCES | 31 |

# ABSTRACT

Python is high-level, general programming language which was designed by Guido van Rossum. Python is a free and open-source programming language which easy to code and which is easy to understand. Python is a portable and integrated language which also GUI programming support. Python can also support object-oriented programming concepts.

Python programs can do a lot more work in fewer lines of code than many other languages would require. Python is a highly-interpreted language. Python is a platform independent language.

Python can support multiple datatypes like lists, tuples, strings, sets, dictionaries, etc. Python has numerous machine learning packages (like SciPy, scikit learn), statistical packages(like NumPy, pandas), visualization packages(like matplotlib, seaborn, plotly) and it has some more packages like OpenCV, TensorFlow, pytorch which are used in image recognition and other deep learning tasks.

One of the main reasons why Python is widely used in the scientific and research communities is because of its ease of use and simple syntax which makes it easy to adapt for people who do not have an engineering background. It is also more suited for quick prototyping. According to engineers coming from academia and industry, deep learning frameworks available with Python APIs, in addition to the scientific packages have made Python incredibly productive and versatile. There has been a lot of evolution in deep learning Python frameworks and it’s rapidly upgrading.

In terms of application areas, ML scientists prefer Python as well. When it comes to areas like building fraud detection algorithms and network security, developers leaned towards Java, while for applications like natural language processing (NLP) and sentiment analysis, developers opted for Python, because it provides large collection of libraries that help to solve complex business problem easily, build strong system and data application.

# INTRODUCTION TO PYTHON

* Most of the famous apps on our mobile devices are created using python language.
  + The photos that we upload on Instagram
  + Favorite movies that we watch on Netflix
  + Favorite hip hop songs that we listen on Spotify
  + Uber to book a ride from one place to another place
* Python is easy to learn
* Python can be used to create games and apps on our own
* Python can be installed easily and effortlessly
* Python can help us to improve our critical thinking skills
* Many real time problems can be solved using python
* Python is popular, widely spread and most often used language in these days

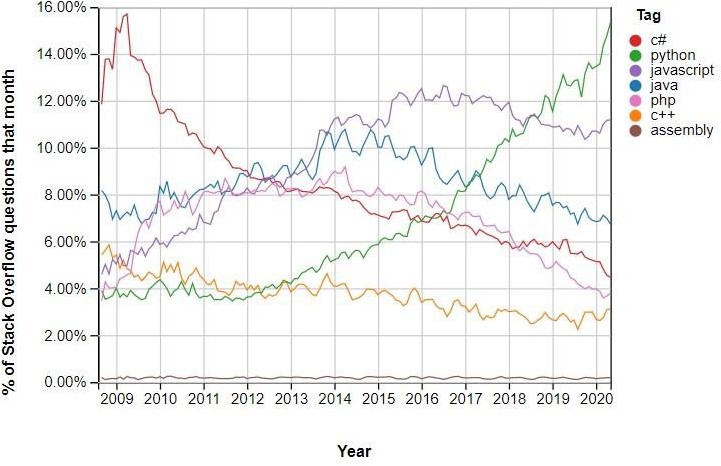


Fig-1

The above Fig-1 shows the incredible growth of python by Stack Overflow. It shows that there is abnormal change of questions related to python being asked by users in each year.

# Python Numpy

**Numpy** is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

**Array in Numpy**

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array.A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as **ndarray**. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.

**Creating a Numpy Array**

Arrays in Numpy can be created by multiple ways, with various number of Ranks, defining the size of the Array. Arrays can also be created with the use of various data types such as lists, tuples, etc. The type of the resultant array is deduced from the type of the elements in the sequences.  
**Note:** Type of array can be explicitly defined while creating the array.

|  |
| --- |
| # Python program for  # Creation of Arrays  import numpy as np    # Creating a rank 1 Array  arr = np.array([1, 2, 3])  print("Array with Rank 1: \n",arr)    # Creating a rank 2 Array  arr = np.array([[1, 2, 3],                  [4, 5, 6]])  print("Array with Rank 2: \n", arr)    # Creating an array from tuple  arr = np.array((1, 3, 2))  print("\nArray created using "        "passed tuple:\n", arr) |

**Output:**

Array with Rank 1:

[1 2 3]

Array with Rank 2:

[[1 2 3]

[4 5 6]]

Array created using passed tuple:

[1 3 2]

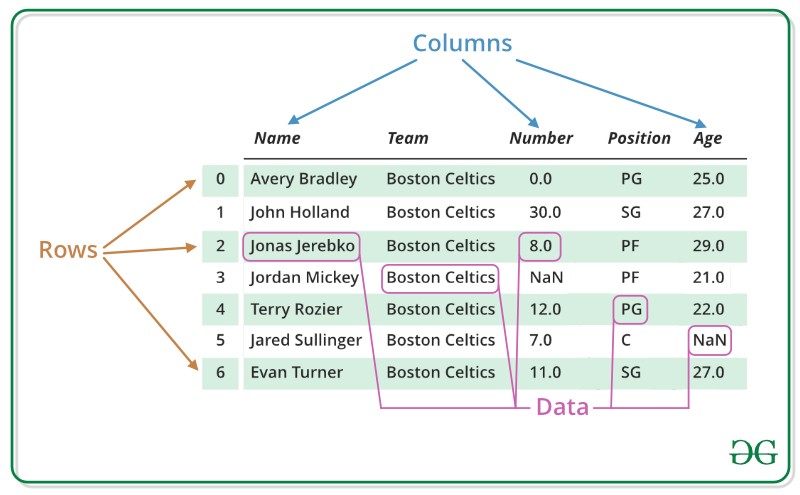
**Accessing the array Index**

In a numpy array, indexing or accessing the array index can be done in multiple ways. To print a range of an array, slicing is done. Slicing of an array is defining a range in a new array which is used to print a range of elements from the original array. Since, sliced array holds a range of elements of the original array, modifying content with the help of sliced array modifies the original array content.

|  |  |  |
| --- | --- | --- |
| # Python program to demonstrate  # Indexing in NumPy array  import NumPy as np    # Initial Array  arr = np.array([[-1, 2, 0, 4],                  [4, -0.5, 6, 0],                  [2.6, 0, 7, 8],                  [3, -7, 4, 2.0]])  print("Initial Array: ")  print(arr)    # Printing a range of Array  # with the use of slicing method  sliced\_arr = arr[:2, ::2]  print ("Array with first 2 rows and"      " alternate columns(0 and 2):\n", sliced\_arr)    # Printing elements at  # specific Indices  Index\_arr = arr[[1, 1, 0, 3],                  [3, 2, 1, 0]]  print ("\nElements at indices (1, 3), "      "(1, 2), (0, 1), (3, 0):\n", Index\_arr)  **Output:**  Initial Array:  [[-1. 2. 0. 4.]  [ 4. -0.5 6. 0.]  [ 2.6 0. 7. 8.]  [ 3. -7. 4. 2.]]  Array with first 2 rows and alternate columns (0 and 2):  [[-1. 0.]  [ 4. 6.]]  Elements at indices (1, 3), (1, 2), (0, 1), (3, 0):  [ 0. 54. 2. 3.]  **Basic Array Operations**  In NumPy, arrays allow a wide range of operations which can be performed on a particular array or a combination of Arrays. These operations include some basic Mathematical operation as well as Unary and Binary operations.   |  | | --- | | # Python program to demonstrate  # basic operations on single array  import numpy as np    # Defining Array 1  a = np.array([[1, 2],                [3, 4]])    # Defining Array 2  b = np.array([[4, 3],                [2, 1]])    # Adding 1 to every element  print ("Adding 1 to every element:", a + 1)    # Subtracting 2 from each element  print ("\nSubtracting 2 from each element:", b - 2)    # sum of array elements  # Performing Unary operations  print ("\nSum of all array "         "elements: ", a.sum())    # Adding two arrays  # Performing Binary operations  print ("\nArray sum:\n", a + b) |   **Output:**  Adding 1 to every element:  [[2 3]  [4 5]]  Subtracting 2 from each element:  [[ 2 1]  [ 0 -1]]  Sum of all array elements: 10  Array sum:  [[5 5]  [5 5]]  **Math Operations on Datatype array**  In NumPy arrays, basic mathematical operations are performed element-wise on the array. These operations are applied both as operator overloads and as functions. Many useful functions are provided in NumPy for performing computations on Arrays such as **sum**: for addition of Array elements, **T**: for Transpose of elements, etc.   |  | | --- | | # Python Program to create  # a data type object  import numpy as np    # First Array  arr1 = np.array([[4, 7], [2, 6]],                   dtype = np.float64)    # Second Array  arr2 = np.array([[3, 6], [2, 8]],                   dtype = np.float64)    # Addition of two Arrays  Sum = np.add(arr1, arr2)  print("Addition of Two Arrays: ")  print(Sum)    # Addition of all Array elements  # using predefined sum method  Sum1 = np.sum(arr1)  print("\nAddition of Array elements: ")  print(Sum1)    # Square root of Array  Sqrt = np.sqrt(arr1)  print("\nSquare root of Array1 elements: ")  print(Sqrt) |   **Output:**  Addition of Two Arrays:  [[ 7. 13.]  [ 4. 14.]]  Addition of Array elements:  19.0  Square root of Array1 elements:  [[2. 2.64575131]  [1.41421356 2.44948974]] |

**Pandas Data Frame**

**Pandas Data Frame** is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas Data Frame consists of three principal components, the **data**, **rows**, and **columns**.



**Creating a Pandas Data Frame**

In the real world, a Pandas Data Frame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, and Excel file. Pandas Data Frame can be created from the lists, dictionary, and from a list of dictionaries etc. Data frame can be created in different ways here are some ways by which we create a data frame:

[**Creating a data frame using List**](https://www.geeksforgeeks.org/create-a-pandas-dataframe-from-lists/)**:**

Data Frame can be created using a single list or a list of lists.

|  |  |
| --- | --- |
| # import pandas as pd  **import** pandas as pd    # list of strings  lst **=** ['Geeks', 'For', 'Geeks', 'is',              'portal', 'for', 'Geeks']    # Calling DataFrame constructor on list  df **=** pd.DataFrame(lst)  print(df)  **Output:**    [**Creating Data Frame from dict of ndarray/lists**](https://www.geeksforgeeks.org/python-create-a-pandas-dataframe-from-a-dict-of-equal-length-lists/)**:**  To create Data Frame from dict of narray/list, all the narray must be of same length. If index is passed then the length index should be equal to the length of arrays. If no index is passed, then by default, index will be range(n) where n is the array length.   |  | | --- | | # Python code demonstrate creating  # Data Frame from dict narray / lists  # By default addresses.    **import** pandas as pd    # Initialize data of lists.  data **=** {'Name':['Tom', 'nick', 'krish', 'jack'],          'Age’: [20, 21, 19, 18]}    # Create DataFrame  df **=** pd.DataFrame(data)    # Print the output.  print(df) |   **Output:** |

# Python Introduction to Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

**Importing matplotlib:**

from matplotlib import pyplot as plt

or

import matplotlib.pyplot as plt

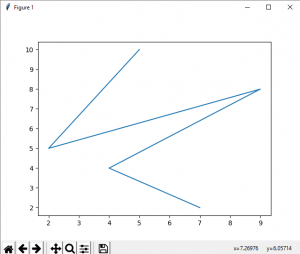
#### **Basic plots in Matplotlib:**

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They’re typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.

**Line plot:**

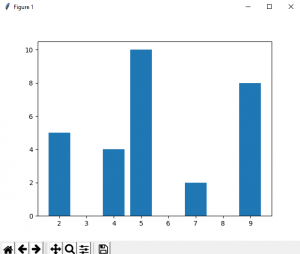
|  |
| --- |
| # Importing matplotlib module  **from** matplotlib **import** pyplot as plt    # x-axis values  x **=** [5, 2, 9, 4, 7]    # Y-axis values  y **=** [10, 5, 8, 4, 2]    # Function to plot  plt.plot(x,y)    # function to show the plot  plt.show() |

Output :



**Bar plot:**

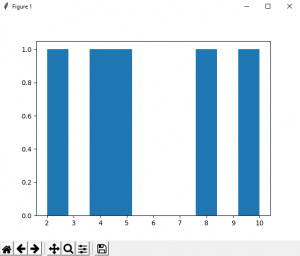
|  |
| --- |
| # importing matplotlib module  **from** matplotlib **import** pyplot as plt    # x-axis values  x **=** [5, 2, 9, 4, 7]    # Y-axis values  y **=** [10, 5, 8, 4, 2]    # Function to plot the bar  plt.bar(x,y)    # function to show the plot  plt.show() |

Output:  
  
 

**Histogram:**

|  |
| --- |
| # Importing matplotlib module  **from** matplotlib **import** pyplot as plt    # Y-axis values  y **=** [10, 5, 8, 4, 2]    # Function to plot histogram  plt.hist(y)    # Function to show the plot  plt.show() |

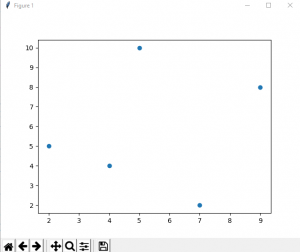
Output :



**Scatter Plot:**

|  |
| --- |
| # importing matplotlib module  **from** matplotlib **import** pyplot as plt    # x-axis values  x **=** [5, 2, 9, 4, 7]    # Y-axis values  y **=** [10, 5, 8, 4, 2]    # Function to plot scatter  plt.scatter(x, y)    # function to show the plot  plt.show() |

Output :



**Datasets**

**A Dataset** is a set or collection of data. This set is normally presented in a tabular pattern. Every column describes a particular variable. And each row corresponds to a given member of the data set, as per the given question. This is a part of [data management](https://byjus.com/maths/data-management/). Data sets describe values for each variable for unknown quantities such as height, weight, temperature, volume, etc., of an object or values of random numbers. The values in this set are known as a **datum**. The data set consists of data of one or more members corresponding to each row. In this article, let us learn the definition of the dataset, different types of datasets, properties, and so on with many solved examples.

**Types of Datasets**

In Statistics, we have different types of data sets available for different types of information. They are:

* Numerical data sets
* Bivariate data sets
* Multivariate data sets
* Categorical data sets
* Correlation data sets

## Properties of Dataset

Before performing any statistical analysis, it is essential to understand the nature of the data. We can use different Exploratory Data Analysis (EDA techniques), which helps to identify the properties of data, so that the appropriate statistical methods can be applied on the data. With the help of EDA techniques, we can check the following properties of the dataset.

* Centre of data
* Skewness of data
* Spread among the data members
* Presence of outliers
* Correlation among the data
* Type of probability distribution that the data follows