# PROJECT REPORT

## Development Of Software In Python To Generate Various Graphical Analysis

---

**Abstract**: Graphs are used to solve many real-life problems. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, Facebook.Hence the need for development of a software that generates various graphical analysis is very essential.No matter if you want to create interactive, live or highly customized plots, python has an excellent library package. To get a little overview here are a few popular plotting libraries: Matplotlib: low level, provides lots of freedom. Matplotlib was the first Python data visualization library, many other libraries are built on top of itSome libraries like pandas and Seaborn are “wrappers” over matplotlib. They allow you to access a number of matplotlib’s methods with less code.

**Keywords** :Python, Pandas, Seaborn, Matplotlib, Data visualization

---

# INTRODUCTION

Python has already made it easy for data visualisation with two exclusive libraries for visualization, commonly known as *matplotlib and seaborn.Using this libraries we can Setup an graphical analysis system.*

*Matplotlib*: Python based plotting library offers *matplotlib* with a complete 2D support along with limited 3D graphic support. It is useful in producing publication quality figures in interactive environments across platforms. It can also be used for animations as well.

*Seaborn*: Seaborn is a library for creating informative and attractive statistical graphics in python. This library is based on matplotlib. Seaborn offers various features such as built in themes, color palettes, functions and tools to visualize univariate, bivariate, linear regression, matrices of data, statistical time series etc which lets us to build complex visualizations.

*Pandas* : It is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

METHODOLOGY

- Import Data Set:

The first step to any data science project is to import your data.Data can be in any of the popular formats - CSV, TXT, XLS/XLSX (Excel), sas7bdat (SAS), Stata, Rdata (R) etc.

In [17]:
```
import pandas as pd
import numpy as np
```

```
#Read .csv file
df = pd.read_csv (r'/home/sndphs/Desktop/python project/grade_card.csv')
df1 =df
```

| Rating_Rank | ROLL NUMBER | NAME | PHYSICS | CHEMISTRY | COMPUTER SCIENCE |
|---|---|---|---|---|---|
| 1.0 | 5 | CHRISTY | 94 | 98 | 100 |
| 2.0 | 11 | HARI | 88 | 96 | 95 |
| 3.0 | 15 | KRISHNA | 94 | 89 | 92 |
| 4.0 | 7 | ESTHER | 92 | 95 | 100 |
| 5.0 | 10 | GOPIKA | 87 | 90 | 95 |
| 6.0 | 6 | DONA | 82 | 85 | 90 |
| 7.0 | 14 | KEVIN | 81 | 89 | 78 |
| 8.0 | 1 | ABHIJITH | 83 | 87 | 80 |
| 9.0 | 20 | PARVANA | 69 | 79 | 80 |
| 10.0 | 4 | BIBIN | 75 | 73 | 80 |
| 11.0 | 19 | NEVIN | 67 | 65 | 70 |
| 12.0 | 24 | SARUN | 60 | 70 | 75 |
| 13.0 | 26 | SWATHY | 60 | 67 | 76 |
| 14.5 | 2 | ASEEM | 67 | 53 | 87 |
| 14.5 | 13 | JEFFIN | 52 | 78 | 75 |
| 16.0 | 9 | FAIHA | 64 | 72 | 69 |
| 17.0 | 25 | SREEKALA | 39 | 56 | 67 |

- Exploring the Data & Cleaning Corrupted Data

Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of data-centric python packages. *Pandas* is one of those packages and makes importing and analyzing data much easier.
Data frame objects have many useful attributes that make this easy.

- ➔ Let’s get a better look at the data using the *.head()* method, which displays the first five rows, first five index values, of every column within

a Pandas data frame object.

```
In [8]: df1.head()
```

Out[8]:

| Rating_Rank | ROLL NUMBER | NAME | PHYSICS | CHEMISTRY | COMPUTER SCIENCE | MATHEMATICS | ENGLISH | GRADE |
|---|---|---|---|---|---|---|---|---|
| 1.0 | 5 | CHRISTY | 94 | 98 | 100 | 92 | 98 | A |
| 2.0 | 11 | HARI | 88 | 96 | 95 | 86 | 95 | A |
| 3.0 | 15 | KRISHNA | 94 | 89 | 92 | 90 | 89 | A |
| 4.0 | 7 | ESTHER | 92 | 95 | 100 | 80 | 85 | A |
| 5.0 | 10 | GOPIKA | 87 | 90 | 95 | 83 | 87 | B |

➔ mean() function can be used to calculate mean/average of a given list of numbers. It returns the mean of the data set passed as parameters.

```
# Function class average marks of each subjects
def averagemarks(sdf):
    print(df1.mean(axis = 0, skipna = True))


def studentperformance(sdf):
    physicsavg = sdf.loc[:,"PHYSICS"].mean()
    chemistryavg = sdf.loc[:,"CHEMISTRY"].mean()
    compscnavg = sdf.loc[:,"COMPUTER SCIENCE"].mean()
    mathematicsavg = sdf.loc[:,"MATHEMATICS"].mean()
    englishavg = sdf.loc[:,"ENGLISH"].mean()
```

➔ Pandas set_index() is a method to set a List, Series or Data frame as index of a Data Frame.

➔ Pandas dataframe.sort_index() function sorts objects by labels along the given axis.

➔ *pop()* is an inbuilt function in Python that removes and returns last value from the list or the given index value

```
df1['Rating_Rank'] = df1['PERCENTAGE'].rank(ascending = 0)
df1 = df1.set_index('Rating_Rank')
df1 = df1.sort_index()
df1.pop('TOTAL')
df1.pop('PERCENTAGE')
#df1.pop('GRADE')
```

❖ VISUALISATION OF DATA

We can analyze  data set to visualize through different charts

➢ Line Plots

Line graphs are plots where a line is drawn to indicate a relationship between a particular set of x and y values.

syntax:plt.plot(x, y)

- Scatter Plots

Alternatively, you might want to plot quantities with 2 positions as data points.

Consider the same data as for line graph, to create scatter plots we just need to modify one line in the above code −

syntax:plt.plot(x, y,'o')

- Histogram

Histograms are very often used in science applications and it's highly likely that you will need to plot them at some point. They are very useful to plot distributions.

Syntax:

df = pd.DataFrame(data, columns = ([ ]))

df.hist()

plt.show()

- Pie chart

A Pie Chart is a circular statistical plot that can display only one series of data. The area of the chart is the total percentage of the given data.Matplotlib API has pie() function in its pyplot module which create a pie chart representing the data in an array. You can define it’s sizes, which parts should explode (distance from center), which labels it should have and which colors it should have.

Syntax:

plt.pie(sizes, explode=explode, labels=labels, colors=colors, ...)

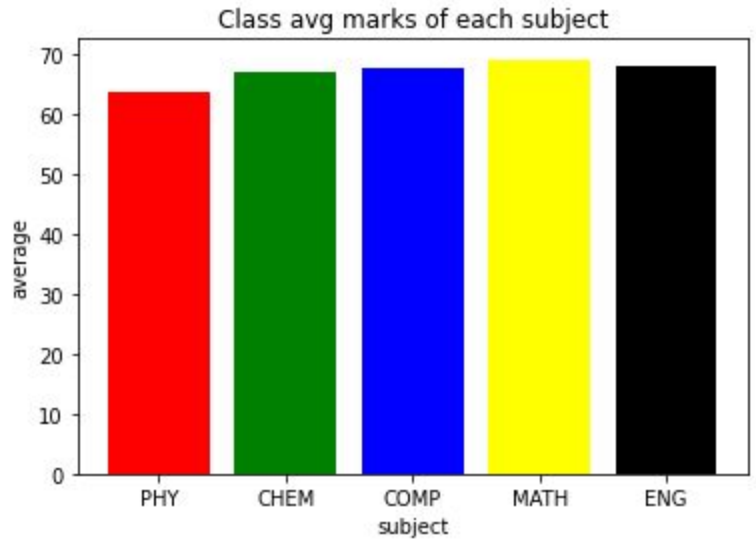
# Result Analysis

## 1.Class average marks of each subjects

Subject names are called by using a list of strings and are stored in a variable ‘sub’.Average marks for five subjects are calculated using mean() . Store it in a

variable ‘avg’.A bar graph is then plotted based on the above calculations using plt.bar() .Give a name to x-axis and y-axis using label(),title using title() and finally to view the plot use show()

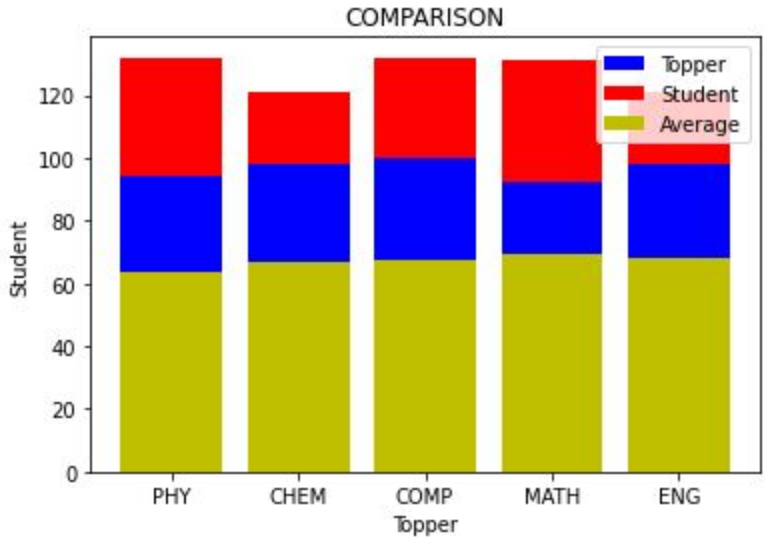
```
plt.title("Class avg marks of each subject")
plt.xlabel("subject")
plt.ylabel("average")
plt.bar(left,avg,tick_label=sub,width=0.8,color=['red','green','blue','yellow','black'])
plt.show()
```




2.Performance of the student compared with topper and average of the class

The csv data is sorted according to the rank of students.Toppers marks are extracted using iloc[] and the average is extracted from the previous question using mean().A program is written using conditional statement for the user to randomly select a student and store the marks in a variable.To compare the marks a stacked bar graph is used

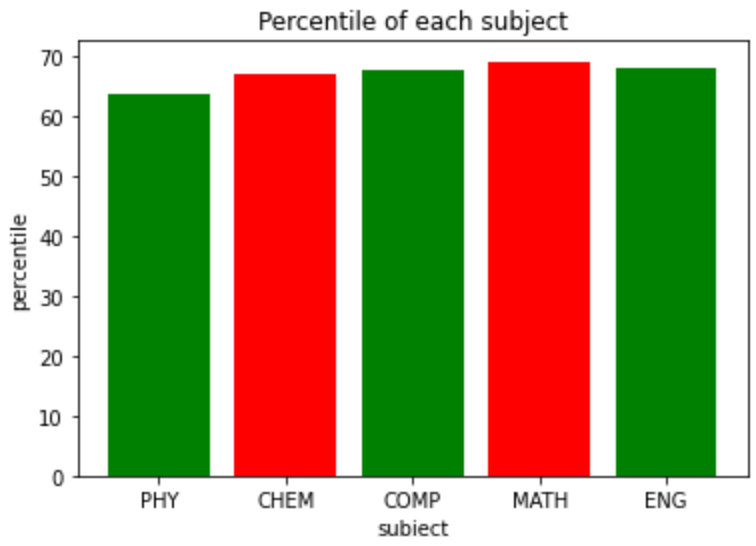
```
plt.bar(xvalues,y1,color='b', label ='Topper')
plt.bar(xvalues,y2, color='r', bottom =y1, label = 'Student')
plt.bar(xvalues,y3,color='y',label='Average')
plt.xticks(xvalues, ('PHY', 'CHEM', 'COMP', 'MATH', 'ENG'))

plt.xlabel('Topper')
plt.ylabel('Student')
plt.title('COMPARISON')
plt.legend()

plt.show()
```




3.Subject wise percentile of the student

Subject names are called by using a list of strings and are stored in a variable ‘sub’.A program is written using conditional statements for the user to randomly select a student and store the subject wise percentile of the student in a variable ’per’.A bar graph is then plotted based on the above calculations using plt.bar() .

```
sub=["PHY","CHEM","COMP","MATH","ENG"]
left=1,2,3,4,5
per=[p,q,r,s,t]
plt.title("Percentile of each subject")
plt.xlabel("subject")
plt.ylabel("percentile")
plt.bar(left,avg,tick_label=sub,width=0.8,color=['green','red'])
plt.show()
```




4.Toppers of the class

The number of toppers whose marks need to be extracted is stored in a variable “u” .It is then printed using head()

```
In [19]: T=input("enter the number of toppers you need:")
         u=int(T)
         df1.head(u)
```

```
enter the number of toppers you need:3
```

Out[19]:

| Rating_Rank | ROLL NUMBER | NAME | PHYSICS | CHEMISTRY | COMPUTER SCIENCE | MATHEMATICS | ENGLISH | GRADE |
|---|---|---|---|---|---|---|---|---|
| **1.0** | 5 | CHRISTY | 94 | 98 | 100 | 92 | 98 | A |
| **2.0** | 11 | HARI | 88 | 96 | 95 | 86 | 95 | A |
| **3.0** | 15 | KRISHNA | 94 | 89 | 92 | 90 | 89 | A |

5.Total pass percentage of the class

To find the total pass percentage of the class divide the number of students who **passed** the test by the number of students who took the test.This can be done using conditional statement

```
In [11]: failcount=0
         for index, row in df1.iterrows():
             if(row["GRADE"]=='F'):
                 failcount=failcount+1

         Percent= ((len(df1)-failcount)/len(df1))*100
         print("PASS Percentage: "+str(Percent))

         PASS Percentage: 80.76923076923077
```

CONCLUSION:

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.Python offers multiple great graphing libraries that come packed with lots of different features. In this article, we looked at Matplotlib, Pandas visualization and Seaborn.