**Chapter-1**

**Introduction:**

Student result analysis is a web-based application developed for colleges to analyse the result and keep track of students. We are able to see the individual candidate’s results separately. Student result analysis system has been designed to carry out the mark analysis process in an educational institution. The results of respective departments can be efficiently computed without much of manual involvement. Given the continuous rise in student population, tertiary institutions calculate examination result of students with the help of computer programs. the use of programs that extends capabilities beyond examination result computation is not widespread. The system provides a comprehensive solution to the demand of examination result computation as well as student information and academic record management. The system is resilient and written with the flexibility to accommodate future needs. It is expected to encourage data use in tertiary institutions. Information from such data can facilitate decision-making and improve educational standards in tertiary institutions.

**1.1 Existing System:**

Today there exist some computer program which helps user to find and store basic information such as student’s name, marks and seat number. Rest all the computational work either needs a different program or is done manually by faculties of that institution.

* O.S dependency if computer program is coded in C language.
* The use of linear search in file handling might increase time complexity.
* NO advanced feature benefit can be used such as in web application
* The current system is time consuming.
* The particular file which is to be upload to the system for data extraction  
   some of the current software are not platform independent.
* In current system, some manual work is to be done which makes it complex.
* In current system user must have some knowledge about creating and managing

**1.2 Proposed System:**

Student Result Analysis System is an student driven system that is designed in order to keep record of student data by use of technology. Now-a-day result analysis is done manually taking lots of effort and time then too desired accuracy is not achieved. Also, updation of data is very difficult as all data needs to revised again. In addition, more paperwork and documentation is required. The privileges for teachers are that they can view record of a student result and can change his/her login password. Password to teachers is initially provide.

* Information about various Students.
* Information about subjects offered in various semester
* Marks obtained by student in every semester.
* Generation of reports.
* Analysis of data based on aspects.

**1.3 Problem Definition:**

Student result analysis system is simple web-based application designed using HTML language. In this system database is designed using MySQL. Logical coding is done using java script which contains main logic of overall system functionality. To design web pages CSS is used for different styles. The system is covered under all validation checks for authentication. Data is been imported from excel sheet into database. These student data is been analysed based on various aspects like marks, grade, rank. System will be designed in such a way that it can be later updated to accept different format. For that all data will be stored in temporary storage first, and then formatted before inserting in database.

* User friendly (as faculties can easily use web-based application).
* ALL time availability. (system remains available as long as computer is well connected with network).
* Easy computation.
* Easy Storage of data.
* More efficient and Requires less effort and time.

**Chapter-2**

**Design**

**2.1 Dataflow Diagrams:**

The data flow analysis is a study of how and what data flows through the system. It is done with the help of Data Flow Diagram (DFD), which depicts the entities involved in the system, data input by the entities, processes the input data undergoes and data store where data is stored.

Data flow diagrams should be developed in a ‘top-down’ process. They show a functional perspective where each transformation represents a single function or process. They show the entire sequence of actions that take place from input being processed to the corresponding output of the system.

A data flow diagram (DFD) is a significant modelling technique for analyzing and constructing information process. Data flow diagram((DFD) is a graphical representation of the "flow" of data through an information system. DFD's can also be used for the visualization of data processing (Structured design). On a DFD, data items flow from an external data source or an internal data store to an Internal data store or an external data sink, via an internal process. A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart.

With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. A designer usually draws a context-level DFD showing the relationship between the entities inside and outside of a system as one single step. This basic DFD can be then disintegrated to a lower level diagram demonstrating smaller steps exhibiting details of the system that is being modeled. Numerous levels may be requiring to explain a compiled system. the different versions are context Diagrams (Level 0), partitioned diagrams (single process only-one level), functionally decomposed, leveled sets of data flow diagrams.

**Data Flow Diagram:**

No Yes

Load excel with Register numbers

Add regno read

Reg\_no

Get result page for reg\_no

Reg\_no>1

Parse web page extract result table

Save in dictionary

Yes

Dump list to excel sheets

Add dictionary to

List

**User Interface:**

**Login page (Checks for validity)**

**Selection page**

Student result Subject Report Grade Report Top Ten Report Report Chart

Table display by fetching data from excel sheet5

Table display by fetching data from excel sheet3

Table display by fetching data from excel sheet2

Table display by fetching data from excel sheet4

Table display by fetching data from excel sheet1

Converting into Pandas Data Frame

From Data Frame Converting into PyQt Table

Matplot lib.pyplot library

Showing PyQt Table

**Data Flow Diagram Symbols:**

Flow diagrams in general are usually designed using simple **symbols** such as a rectangle, an oval or a circle depicting a process, data stored or an external entity, arrows and decision/conditional symbols are generally used to depict the data flow from one step to another.

**DATA PROCESS**

A process shows a transformation or manipulation of data flows within the system. The symbol used is rounded rectangle or oval. A descriptive title is placed in the center of the circle. This should be a simple imperative sentence with a specific verb, for example ‘maintain customer records’ or ‘find driver’. E.g.: May be a clerk computing discounts or a combination of manual and electronic activities.

**DATA STORED**

A data store stores passively for later access. A data store responds to requests to store and access data. It does not generate any operations.

A data store allows values to be accessed in an order different from the order in which they were generated.

Input flows indicate information or operations that modify the stored data such adding or deleting elements or changing values. Output flows indicate information retrieved from the store; this information can be entire value or a computer of a value.

**EXTERNAL ENTITY**

An external entity is a source or destination of a data flow which is outside the area of study. The symbol used is a rectangle or square containing a meaningful and unique identifier. Only those entities which originate or receive data are represented on a business process diagram

E.g.: May be one customer or a number of customers with transactions(orders).

**DATA FLOW**

A data flow moves data between processes or between and data stores. As such, it represents a data value at some point within a computation and an intermediate value within a computation if the flow is internal to the diagram. This value is not changed. The names of input and output flows can indicate their roles in the computation or the type of the value they move. Data names are preferably nouns. The name of a typical piece of data, the data aspect, is written alongside the arrow

### DECISION /CONDITIONAL

https://creately.com/sites/default/files/objectdetail/thumbnails/decision.png

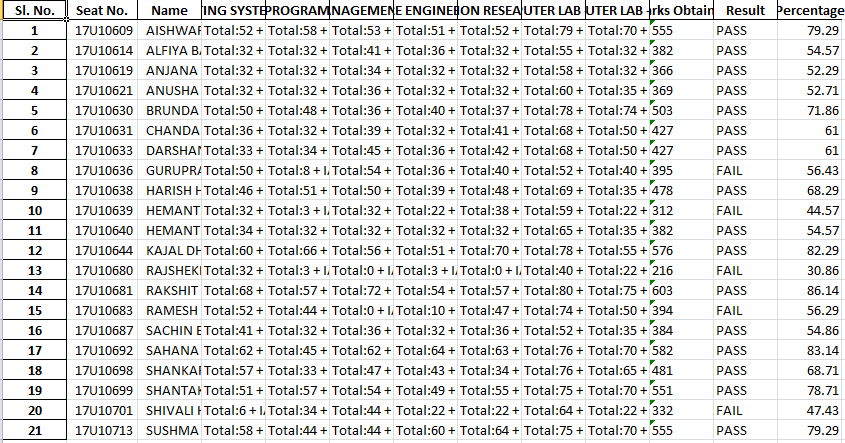
Decision shape is represented as a Diamond. This object is always used in a process flow to ask a question. And, the answer to the question determines the arrows coming out of the Diamond. This shape is quite unique with two arrows coming out of it. One from the bottom point corresponding to Yes or True and one from either the right/left point corresponding to No or False. The arrows should always be labelled to avoid confusion in the process flow.

|  |
| --- |
| **2.2 Pages**  **Login Here**  **User name**  **Password**    **Submit** |

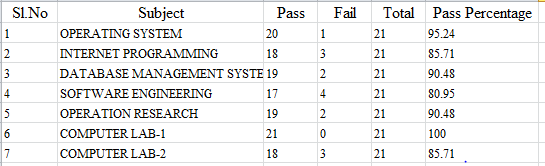
|  |
| --- |
| **GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE**  **Click on the option which you want:**  **E EE**  **Each Student Result**      **Subject Wise Report**    **Grade Wise Report**    **Top Ten Students**      **Result Chart** |

**SAMPLE USER INTERFACE**

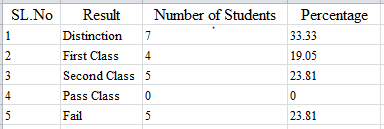
**Each Student Result:**



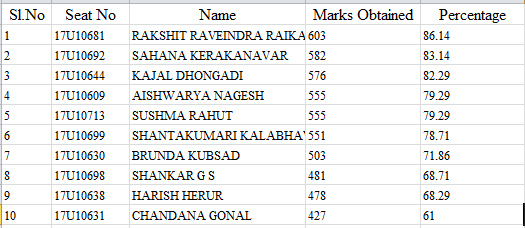
**Subject Wise Report:**



**Grade Wise Report:**



**Top Ten Students:**



**Result Chart:**

**Chapter-3**

**Requirements**

**3.Requirements:**

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified.

Requirements analysis involves all the tasks that are conducted to identify the needs of different stakeholders.

Therefore, requirements analysis means to analyse, document, validate and manage software or system requirements.

High-quality requirements are documented, actionable, measurable, testable, traceable, helps to identify business opportunities, and are defined to a facilitate system design.

**3.1 Minimum Hardware Requirement:**

* **Operating System:** Windows 7 or any equivalent OS 64 bit, Linux
* **RAM:** 4 GB or more
* **Processor:** Intel core(1m) i3-3210M
* **Web Browser:** Internet Explorer 8.0 or any equivalent browser
* **Hard Disk:** 40GB

**3.2 Software [Notes on Front end and Back end]**

* PYTHON 3.6+
* PYQT5
* Beautiful soup library
* Pandas library
* Mechanize library
* Matplotlib.py plot library

# **Front End:**

**Python 3.6+**

**Python** is a general purpose, dynamic, [high-level](https://www.javatpoint.com/classification-of-programming-languages), and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

# **PYQT5:**

[PyQt](https://riverbankcomputing.com/software/pyqt/intro)is a library that lets you use the [Qt GUI framework](https://www.qt.io/) from Python. Qt itself is written in C++. By using it from Python, you can build applications much more quickly while not sacrificing much of the speed of C++.

PyQt5 refers to the most recent version 5 of Qt. You may still find the occasional mention of (Py)Qt4 on the web, but it is old and no longer supported.

An interesting new competitor to PyQt is [Qt for Python](https://www.qt.io/qt-for-python). Its API is virtually identical. Unlike PyQt, it is licensed under the LGPL and can thus be used for free in commercial projects. It's backed by the Qt company, and thus likely the future. We use PyQt here because it is more mature. Since the API’s are so similar, you can easily switch your apps to Qt for Python later.

# **Features of PyQt:**

Here, are important features of PyQt:

PyQt consists of more than six hundred classes covering a range of features such as

Graphical User Interfaces

SQL Databases

Web toolkits

XML processing

Networking

These features can be combined to create advanced UIs as well as standalone applications. A lot of major companies across all industries use Qt. Some examples are LG, Mercedes, AMD, Panasonic, Harman, etc.

**Back End:**

1. **Beautiful Soup Library:**

Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages. Say you’ve found some webpages that display data relevant to your research, such as date or address information, but that do not provide any way of downloading the data directly. Beautiful Soup helps you pull particular content from a webpage, remove the HTML markup, and save the information. It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web. The Beautiful soup is derived from the tag soup word which is not parse and able and its unstructured in nature found in most of the website.

The [Beautiful Soup documentation](http://www.crummy.com/software/BeautifulSoup/bs4/doc/) will give you a sense of variety of things that the Beautiful Soup library will help with, from isolating titles and links, to extracting all of the text from the html tags, to altering the HTML within the document you’re working with.

1. **Pandas Library:**



Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python. This tutorial is designed for both beginners and professionals. Data analysis requires lots of processing, such as **restructuring, cleaning** or **merging**, etc. There are different tools are available for fast data processing, such as **Numpy, Scipy, Cython**, and **Panda**. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

Pandas is built on top of the **Numpy** package, means **Numpy** is required for operating the Pandas. Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., **load, manipulate, prepare, model, and analyse.**

**Key Features of Pandas**

* It has a fast and efficient Data Frame object with the default and customized indexing.
* Used for reshaping and pivoting of the data sets.
* Group by data for aggregations and transformations.
* It is used for data alignment and integration of the missing data.
* Provide the functionality of Time Series.
* Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.
* Handle multiple operations of the data sets such as subsetting, slicing, filtering, groupBy, re-ordering, and re-shaping.
* It integrates with the other libraries such as SciPy, and scikit-learn.
* Provides fast performance, and If you want to speed it, even more, you can use the **Cython**.

**Benefits of Pandas:**

The benefits of pandas over using other language are as follows:

**Data Representation:** It represents the data in a form that is suited for data analysis through its Data Frame and Series.

**Clear code:** The clear API of the Pandas allows you to focus on the core part of the code. So, it provides clear and concise code for the user.

# **Data Frames:**

A Data Frame is an essential data structure with pandas. It lets us deal with data in a tabular fashion. The rows are observations and columns are variables.  
We have the following syntax for this-

Pandas. **Data Frame** (data, index, columns, dtype, copy)

Such a data structure is-

Mutable

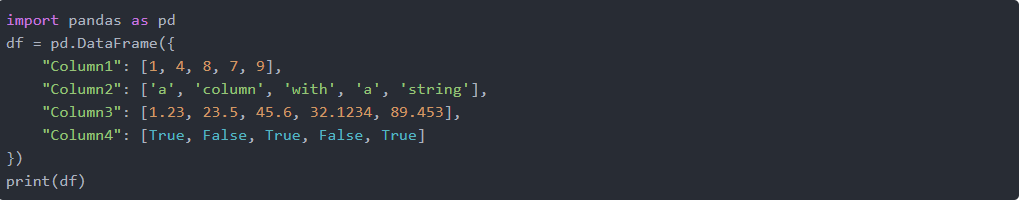
Variable columns

Labelled axes

Capable of performing arithmetic operations on columns and rows

The Pandas Data Frame can be seen as a table. It organizes data into rows and columns, making it a two-dimensional data structure. Potentially, the columns are of a different type and the size of the Data Frame is mutable, and hence can be modified.

To create a Data Frame, you can choose to start from scratch or convert other data structures like NumPy arrays into a Data Frame. Here is how you can create a Data Frame from scratch:



Output



In this example we have created a DataFrame named df. The first column of the DataFrame has integer values. The second column has a string, the third column has floating point values, while the fourth column has Boolean values.

The statement print(df) will display the contents of the Data Frame to us via the console, allowing us to inspect and verify its contents.

However, when displaying the Data Frame, you may have noticed that there is an additional column at the start of the table, with its elements beginning at 0. This column is created automatically and it marks the indexes of the rows.

To create a Data Frame, we must invoke the pd.DataFrame() method as shown in the above example.

It is possible for us to create a DataFrame from a list or even a set of lists. We only have to call the pd.DataFrame() method and then pass it the list variable as its only argument.



**Mechanize Library:**

Mechanize. Browser implements the urllib2.OpenerDirector interface. Browser objects have state, including navigation history, HTML form state, cookies, etc. The set of features and URL schemes handled by Browser objects is configurable. The library also provides an API that is mostly compatible with urllib2: your urllib2 program will likely still work if you replace “urllib2” with “mechanize” everywhere.

Features include: ftp: http: and file: URL schemes, browser history, hyperlink and HTML form support, HTTP cookies, HTTP-EQUIV and Refresh, Referer [sic] header, robots.txt, redirections, proxies, and Basic and Digest HTTP authentication. Much of the code originally derived from Perl code by Gisle Aas (libwww-perl), Johnny Lee (MSIE Cookie support) and last but not least Andy Lester (WWW: Mechanize). urllib2 was written by Jeremy Hylton.

**Chapter-4**

**Coding**

**4.1 Introduction of Programming Languages:**

**PYTHON:**

****

**Python** is a general purpose, dynamic, [high-level](https://www.javatpoint.com/classification-of-programming-languages), and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

Python's syntax and *dynamic typing* with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports *multiple programming pattern*, including object-oriented, imperative, and functional or procedural programming styles.

Python is not intended to work in a particular area, such as web programming. That is why it is known as *multipurpose* programming language because it can be used with web, enterprise, 3D CAD, etc.

We don't need to use data types to declare variable because it is *dynamically typed* so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging *fast* because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

### Python Features:

### 1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

**2) Expressive Language**

Python language is more expressive means that it is more understandable and readable.

**3) Interpreted Language**

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

**4) Cross-platform Language**

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

**5) Free and Open Source**

Python language is freely available at address. The source-code is also available. Therefore, it is open source.

**6) Object-Oriented Language**

Python supports object oriented language and concepts of classes and objects come into existence.

**7) Extensible**

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

**8) Large Standard Library**

Python has a large and broad library and prvides rich set of module and functions for rapid application development.

**9) GUI Programming Support**

Graphical user interfaces can be developed using Python.

**10) Integrated**

It can be easily integrated with languages like C, C++, JAVA etc.

**Anaconda:**

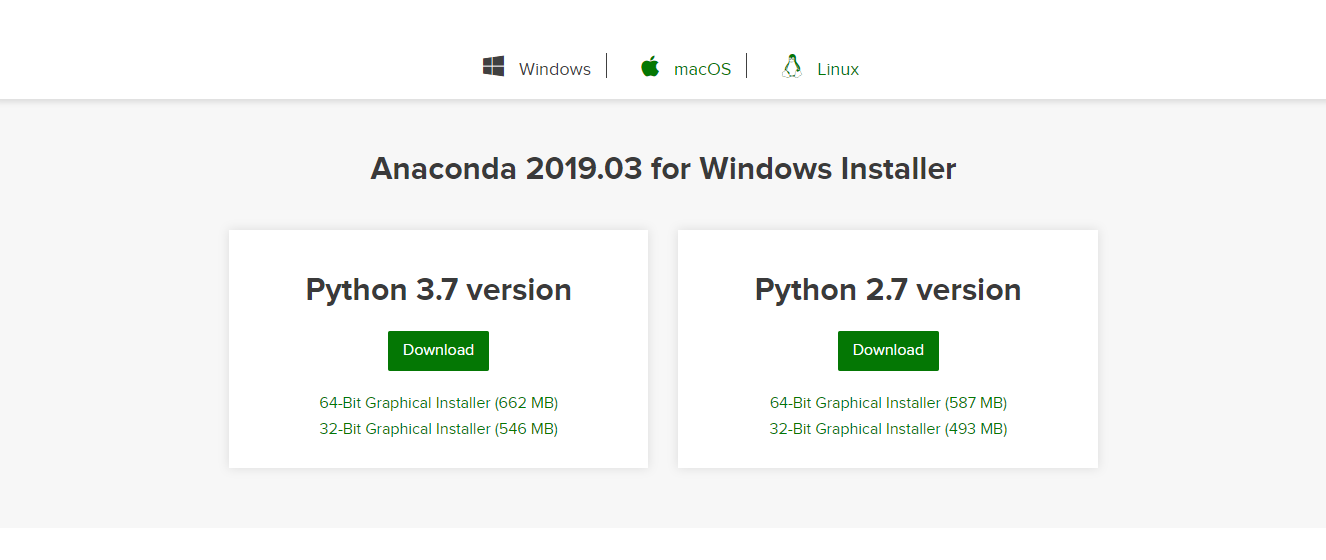
Anaconda is an open-source distribution for python and R. It is used for [data science](https://www.edureka.co/blog/how-to-learn-data-science/), [machine learning](https://www.edureka.co/blog/introduction-to-machine-learning/), [deep learning](https://www.edureka.co/blog/deep-learning-with-python/), etc. With the availability of more than 300 libraries for data science, it becomes fairly optimal for any programmer to work on anaconda for data science.



Anaconda helps in simplified package management and deployment. Anaconda comes with a wide variety of tools to easily collect data from various sources using various machine learning and AI algorithms. It helps in getting an easily manageable environment setup which can deploy any project with the click of a single button.

Now that we know what anaconda is, let’s try to understand how we can install anaconda and set up an environment to work on our systems.

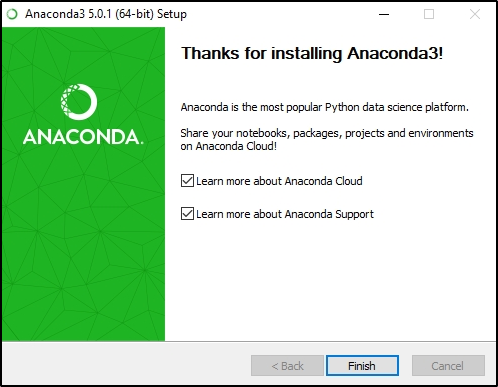
## **Installation And Setup:**

* To install anaconda, go to [https://www.anaconda.com/distribution/.](https://www.anaconda.com/distribution/.#n/)

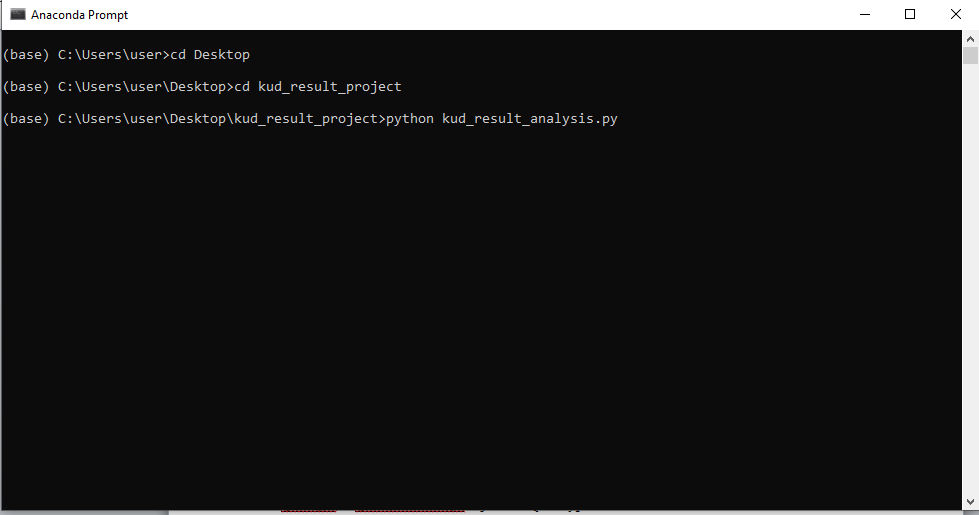
Choose a version suitable for you and click on download. Once you complete the download, open the setup.



Follow the instructions in the setup. Don’t forget to click on add anaconda to my path environment variable. After the installation is complete, you will get a window like shown in the image below.



After finishing the installation, open anaconda prompt and type [notebook](https://www.edureka.co/blog/cheatsheets/Jupyter-Notebook-Cheat-Sheet).



You will see a window like shown in the image above. Now that we know how to use anaconda for python let’s take a look at how we can install various libraries in anaconda for any project

**Qt chart for plotting graph:**

Qt Charts enables creating stylish, interactive, data centric user interfaces. Qt Charts uses the [Graphics View Framework](https://doc.qt.io/archives/qt-5.10/graphicsview.html) for ease of integration. The chart components can be used as [QWidget](https://doc.qt.io/archives/qt-5.10/qwidget.html) or [QGraphicsWidget](https://doc.qt.io/archives/qt-5.10/graphicsview.html#qgraphicswidget) objects or QML types.



The [QChart](https://doc.qt.io/archives/qt-5.10/qchart.html) class manages the graphical representation of different types of series and other chart related objects, such as legend and axes. [QChart](https://doc.qt.io/archives/qt-5.10/qchart.html) is a [QGraphicsWidget](https://doc.qt.io/archives/qt-5.10/graphicsview.html#qgraphicswidget) that can be shown in a [QGraphicsScene](https://doc.qt.io/archives/qt-5.10/qgraphicsscene.html). A simpler solution is to display a chart in a layout by using the convenience class [QChartView](https://doc.qt.io/archives/qt-5.10/qchartview.html) instead of [QChart](https://doc.qt.io/archives/qt-5.10/qchart.html). In QML, charts are displayed using the [ChartView](https://doc.qt.io/archives/qt-5.10/qml-qtcharts-chartview.html) type.

* Some chart components can also be presented as polar charts by using the [QPolarChart](https://doc.qt.io/archives/qt-5.10/qpolarchart.html) class that is a specialization of the [QChart](https://doc.qt.io/archives/qt-5.10/qchart.html) class or the [PolarChartView](https://doc.qt.io/archives/qt-5.10/qml-qtcharts-polarchartview.html) QML type that is a specialization of the [ChartView](https://doc.qt.io/archives/qt-5.10/qml-qtcharts-chartview.html) type.
* The look and feel of charts can be customized by using themes, modifying colors and properties, hiding chart components, or animating charts.
* Model mappers enable using a data model derived from the [QAbstractItemModel](https://doc.qt.io/archives/qt-5.10/qabstractitemmodel.html) class as a data source for a chart. Model mappers can be either horizontal or vertical.

**Pie Charts:**

Pie charts present data as a pie that consists of pie slices. The pie is implemented using the [QPieSeries](https://doc.qt.io/archives/qt-5.10/qpieseries.html) class or the [PieSeries](https://doc.qt.io/archives/qt-5.10/qml-qtcharts-pieseries.html) QML type and the pie slices are added using the [QPieSlice](https://doc.qt.io/archives/qt-5.10/qpieslice.html) class or the [PieSlice](https://doc.qt.io/archives/qt-5.10/qml-qtcharts-pieslice.html) QML type.

The pie can be turned into a donut by specifying a hole size between 0.0 and 1.0.

**4.2 Interface code:**

import mechanize

from bs4 import Beautiful Soup

import pandas as pd

import kudapi

import matplotlib. pyplot as plt

register numbers = ["17u10609","17u10614","17u10619","17u10621","17u10630","17u10631","17u10633",

"17u10636","17u10638","17u10639","17u10640","17u10644","17u10680","17u10681", "17u10683","17u10687","17u10692","17u10698","17u10699","17u10701","17u10713"]

def get\_web\_page ():

br = mechanize.Browser()

br.addheaders = [('User-agent','Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.6')]

br.set\_handle\_robots(False)

br.open('http://www.kud.ac.in/kud\_results.aspx?key=UG')

return br

def get\_page\_data(br,ip,i):

br.select\_form(nr=0)

br['cbo\_course'] = ["23"]

br['txt\_reg\_num'] = ip[i]

response = br.submit()

html = response.read()

soup = BeautifulSoup(html,"lxml")

return soup

def get\_results(soup,results,sbj\_results):

one\_result = {}

sbj\_result = {}

table1 = soup.findAll('table',{"width":"570"})[0]

table2 = soup.findAll('table',{"width":"570"})[1]

t1\_rows = table1.findAll("tr")[1:3]

one\_result['Seat No.']=t1\_rows[0].findAll("td")[1].string

one\_result['Name']=t1\_rows[1].findAll("td")[1].string

t2\_rows = table2.findAll("tr")[:7]

one\_result['OPERATING SYSTEMS E21'] = t2\_rows[0].findAll("td")[3].string+" + IA:"+t2\_rows[0].findAll("td")[4].string

one\_result['INTERNET PROGRAMMING E22'] = t2\_rows[1].findAll("td")[3].string+" + IA:"+t2\_rows[1].findAll("td")[4].string

one\_result['DATABASE MANAGEMENT SYSTEM E23'] = t2\_rows[2].findAll("td")[3].string+" + IA:"+t2\_rows[2].findAll("td")[4].string

one\_result['SOFTWARE ENGINEERING E24'] = t2\_rows[3].findAll("td")[3].string+" + IA:"+t2\_rows[3].findAll("td")[4].string

one\_result['OPERATION RESEARCH E25'] = t2\_rows[4].findAll("td")[3].string+" + IA:"+t2\_rows[4].findAll("td")[4].string

one\_result['COMPUTER LAB - I E31'] = t2\_rows[5].findAll("td")[3].string+" + IA:"+t2\_rows[5].findAll("td")[4].string

one\_result['COMPUTER LAB - II E32'] = t2\_rows[6].findAll("td")[3].string+" + IA:"+t2\_rows[6].findAll("td")[4].string

t2\_result = table2.findAll("tr")[7:11]

one\_result["Marks Obtained"] = t2\_result[0].findAll("td")[1].string

one\_result["Result"] = t2\_result[2].findAll("td")[1].string

one\_result["Percentage"] = round((((int(t2\_result[0].findAll("td")[1].string))/700)\*100), 2)

results. append(one\_result)

sbj\_result['OPERATING SYSTEMS'] = t2\_rows[0].findAll("td")[5].string

sbj\_result['INTERNET PROGRAMMING'] = t2\_rows[1].findAll("td")[5].string

sbj\_result['DATABASE MANAGEMENT SYSTEM'] = t2\_rows[2].findAll("td")[5].string

sbj\_result['SOFTWARE ENGINEERING'] = t2\_rows[3].findAll("td")[5].string

sbj\_result['OPERATION RESEARCH'] = t2\_rows[4].findAll("td")[5].string

sbj\_result['COMPUTER LAB - I'] = t2\_rows[5].findAll("td")[5].string

sbj\_result['COMPUTER LAB - II'] = t2\_rows[6].findAll("td")[5].string

sbj\_results.append(sbj\_result)

return results, sbj\_results

def result\_table(results,ip):

df\_result = pd.DataFrame(results,index = range(1,len(ip)+1),columns = ['Seat No.','Name','OPERATING SYSTEMS E21',

'INTERNET PROGRAMMING E22','DATABASE MANAGEMENT SYSTEM E23','SOFTWARE ENGINEERING E24',

'OPERATION RESEARCH E25','COMPUTER LAB - I E31','COMPUTER LAB - II E32','Marks Obtained','Result',"Percentage"])

print(f'\n {df\_result}')

return results, df\_result

def grade\_table(results,total\_stds):

total\_marks = []

pass\_ = fail = 0

for dictionary in results:

if dictionary['Result'] == 'PASS':

pass\_+=1

total\_marks.append(int(dictionary['Marks Obtained']))

elif dictionary['Result'] == 'FAIL':

fail+=1

distinction = first\_class = second\_class = pass\_class = 0

for mark in total\_marks:

percentage = (mark/700)\*100

if percentage >= 70:

distinction += 1

elif percentage >= 60:

first\_class += 1

elif percentage >= 50:

second\_class += 1

elif percentage >= 40:

pass\_class += 1

pass\_percentage = []

num\_of\_stds = [distinction, first\_class, second\_class, pass\_class, fail]

for stds in num\_of\_stds:

pass\_percentage.append(round(((stds/total\_stds)\*100), 2))

grade = ['Distinction', 'First Class', 'Second Class', 'Pass Class', 'Fail']

grade\_dict = {"Result":grade, "Number of Students":num\_of\_stds, "Percentage":pass\_percentage}

df\_grade = pd.DataFrame(grade\_dict, index = range(1,6))

print(df\_grade)

return df\_grade

def each\_sbj\_result(sbj\_results,total\_stds):

sbj1\_pass = sbj2\_pass = sbj3\_pass = sbj4\_pass = sbj5\_pass = sbj6\_pass = sbj7\_pass = 0

for dictionary3 in sbj\_results:

if dictionary3['OPERATING SYSTEMS'] == '\xa0\xa0PASS':sbj1\_pass += 1

if dictionary3['INTERNET PROGRAMMING'] == '\xa0\xa0PASS':sbj2\_pass += 1

if dictionary3['DATABASE MANAGEMENT SYSTEM'] == '\xa0\xa0PASS':sbj3\_pass += 1

if dictionary3['SOFTWARE ENGINEERING'] == '\xa0\xa0PASS':sbj4\_pass += 1

if dictionary3['OPERATION RESEARCH'] == '\xa0\xa0PASS':sbj5\_pass += 1

if dictionary3['COMPUTER LAB - I'] == '\xa0\xa0PASS':sbj6\_pass += 1

if dictionary3['COMPUTER LAB - II'] == '\xa0\xa0PASS':sbj7\_pass += 1

sbj\_pass = [sbj1\_pass,sbj2\_pass,sbj3\_pass,sbj4\_pass,sbj5\_pass,sbj6\_pass,sbj7\_pass]

subject = ['OPERATING SYSTEMS', 'INTERNET PROGRAMMING', 'DATABASE MANAGEMENT SYSTEMS', 'SOFTWARE ENGINEERING',

'OPERATION RESEARCH', 'COMPUTER LAB - I', 'COMPUTER LAB - II']

sbj\_fail = []

for i in range(len(sbj\_pass)):

sbj\_fail.append((total\_stds-sbj\_pass[i]))

sbj\_total = []

for i in range(len(sbj\_pass)):

sbj\_total.append((sbj\_pass[i]+sbj\_fail[i]))

pass\_percentage = []

for stds in sbj\_pass:

pass\_percentage.append(round(((stds/total\_stds)\*100), 2))

sbj\_report\_dict = {"Subject":subject, "Pass":sbj\_pass, "Fail":sbj\_fail, "Total":sbj\_total, "Pass percentage":pass\_percentage}

df\_sbj\_report = pd.DataFrame(data = sbj\_report\_dict, index = range(1,8))

print(df\_sbj\_report)

return df\_sbj\_report

def top\_10\_stds(results):

top\_10 = []

top\_stds = []

keys = ['Seat No.','Name','Marks Obtained',"Percentage"]

for dic in results:

top\_stds.append({x:dic[x] for x in keys})

top\_stds.sort(key=lambda x:x['Percentage'], reverse=True)

for i in range(10):

top\_10.append(top\_stds[i])

df\_top10 = pd.DataFrame(top\_10, index=range(1,11))

print(df\_top10)

return df\_top10

def df\_into\_excel(df\_result,df\_grade,df\_sbj\_report,df\_top10):

with pd.ExcelWriter("kud\_result4.xlsx") as writer:

df\_result.to\_excel(writer,sheet\_name = 'sheet1')

df\_grade.to\_excel(writer,sheet\_name = 'sheet2')

df\_sbj\_report.to\_excel(writer,sheet\_name = 'sheet3')

df\_top10.to\_excel(writer,sheet\_name = 'sheet4')

def result\_chart(df):

grade = ['Distinction', 'First Class', 'Second Class', 'Pass Class', 'Fail']

plot = df.plot.pie(y="Percentage", labels=grade, figsize=(6, 6), startangle=45,

autopct='%0.2f%%',wedgeprops = { 'linewidth' : 0.5, 'edgecolor' : "black" })

plt.axis('equal')

plt.tight\_layout()

plt.show()

def result\_sheet(register\_numbers):

print ("\nData submitted\nYour results are being fetched...")

ip = register\_numbers

total\_stds = len(ip)

results = []

sbj\_results = []

br = get\_web\_page()

for i in range(len(ip)):

soup = get\_page\_data(br,ip,i)

results,sbj\_results = get\_results(soup,results,sbj\_results)

results, df1 = result\_table(results,ip)

df2 = grade\_table(results,total\_stds)

df3 = each\_sbj\_result(sbj\_results,total\_stds)

df4 = top\_10\_stds(results)

df\_into\_excel(df1,df2,df3,df4)

print('\nYour results are in the file "kud\_result4.xlsx"\n')

result\_chart(df2)

if \_\_name\_\_ == "\_\_main\_\_":

result\_sheet(register\_numbers)

**Login code:**

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_LoginWindow(object):

def setupUi(self, Dialog):

Dialog.setObjectName("Dialog")

screen\_width = 1300; screen\_height = 700

Dialog.resize(screen\_width,screen\_height)

self.img =QtWidgets.QLabel(Dialog)

self.img.setObjectName("image")

self.img.setPixmap(QtGui.QPixmap("blue.jpg"))

self.img2 =QtWidgets.QLabel(Dialog)

self.img2.setGeometry(QtCore.QRect(430,-10,500,100))

self.img2.setObjectName("image2")

self.img2.setPixmap(QtGui.QPixmap("kud2.jpg"))

self.collegeLabel = QtWidgets.QLabel(Dialog)

self.usernameLabel = QtWidgets.QLabel(Dialog)

self.passwordLabel = QtWidgets.QLabel(Dialog)

self.informationLabel = QtWidgets.QLabel(Dialog)

self.usernameEdit = QtWidgets.QLineEdit(Dialog)

self.passwordEdit = QtWidgets.QLineEdit(Dialog)

self.submitButton = QtWidgets.QPushButton(Dialog)

self.collegeLabel.setGeometry(QtCore.QRect(320,70,800,50))

self.usernameLabel.setGeometry(QtCore.QRect(400,200,150,30))

self.passwordLabel.setGeometry(QtCore.QRect(400,250,150,30))

self.informationLabel.setGeometry(QtCore.QRect(550,160,400,20))

self.usernameEdit.setGeometry(QtCore.QRect(550,200,250,30))

self.passwordEdit.setGeometry(QtCore.QRect(550,250,250,30))

self.submitButton.setGeometry(QtCore.QRect(520,350,100,30))

self.passwordEdit.setEchoMode(QtWidgets.QLineEdit.Password)

font = QtGui.QFont()

font.setPointSize(13)

self.usernameEdit.setFont(font)

self.passwordEdit.setFont(font)

self.submitButton.setFont(font)

font.setPointSize(12)

self.informationLabel.setFont(font)

self.collegeLabel.setFont(QtGui.QFont("Times",16,QtGui.QFont.Bold))

self.collegeLabel.setStyleSheet("color: rgb(20,10,100)")

self.usernameLabel.setFont(QtGui.QFont("Times",16,QtGui.QFont.Bold))

self.usernameLabel.setStyleSheet("color: white")

self.passwordLabel.setFont(QtGui.QFont("Times",16,QtGui.QFont.Bold))

self.passwordLabel.setStyleSheet("color: white")

self.informationLabel.setStyleSheet("color: rgb(200,10,10)")

self.collegeLabel.setObjectName("college")

self.usernameLabel.setObjectName('username\_label')

self.passwordLabel.setObjectName('password\_label')

self.informationLabel.setObjectName('information')

self.usernameEdit.setObjectName("username")

self.passwordEdit.setObjectName("password")

self.submitButton.setObjectName('submit')

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self,Dialog):

\_translator = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translator("Dialog","BCA Vth Semester Result Analysis 2019-20"))

Dialog.setWindowIcon(QtGui.QIcon('global\_college.jpg'))

self.collegeLabel.setText(\_translator('Dialog','GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE'))

self.usernameLabel.setText(\_translator("Dialog","User Name:"))

self.passwordLabel.setText(\_translator('Dialog','Password:'))

self.submitButton.setText(\_translator('Dialog','Submit'))

def main():

import sys

app = QtWidgets.QApplication(sys.argv)

Dialog = QtWidgets.QDialog()

ui = Ui\_LoginWindow()

ui.setupUi(Dialog)

Dialog.show()

sys.exit(app.exec\_())

if \_\_name\_\_ == '\_\_main\_\_':

main()

Select result:

from PyQt5 import QtCore, QtGui, QtWidgets

class SelectionScreen(object):

def \_\_init\_\_(self):

self.setupUi

def setupUi(self,Dialog):

Dialog.setObjectName("Dialog")

screen\_width = 1300; screen\_height = 700

Dialog.resize(screen\_width,screen\_height)

self.img =QtWidgets.QLabel(Dialog)

self.img.setObjectName("image")

self.img.setPixmap(QtGui.QPixmap("blue.jpg"))

self.img2 =QtWidgets.QLabel(Dialog)

self.img2.setGeometry(QtCore.QRect(430,-10,500,100))

self.img2.setObjectName("image2")

self.img2.setPixmap(QtGui.QPixmap("kud2.jpg"))

self.collegeLabel = QtWidgets.QLabel(Dialog)

self.selectLabel = QtWidgets.QLabel(Dialog)

self.studentButton = QtWidgets.QPushButton(Dialog)

self.subjectButton = QtWidgets.QPushButton(Dialog)

self.gradeButton = QtWidgets.QPushButton(Dialog)

self.top10Button = QtWidgets.QPushButton(Dialog)

self.chartButton = QtWidgets.QPushButton(Dialog)

self.collegeLabel.setGeometry(QtCore.QRect(320,70,800,50))

self.selectLabel.setGeometry(QtCore.QRect(380,120,800,50))

self.studentButton.setGeometry(QtCore.QRect(400,200,400,50))

self.subjectButton.setGeometry(QtCore.QRect(400,300,400,50))

self.gradeButton.setGeometry(QtCore.QRect(400,400,400,50))

self.top10Button.setGeometry(QtCore.QRect(400,500,400,50))

self.chartButton.setGeometry(QtCore.QRect(400,600,400,50))

self.collegeLabel.setFont(QtGui.QFont("Times",16,QtGui.QFont.Bold))

self.collegeLabel.setStyleSheet("color: rgb(20,10,100)")

self.selectLabel.setFont(QtGui.QFont("Times",20,QtGui.QFont.Bold,True))

self.selectLabel.setStyleSheet("color: white")

font = QtGui.QFont()

font.setPointSize(16)

self.studentButton.setFont(font)

self.subjectButton.setFont(font)

self.gradeButton.setFont(font)

self.top10Button.setFont(font)

self.chartButton.setFont(font)

self.collegeLabel.setObjectName("college")

self.selectLabel.setObjectName("select")

self.studentButton.setObjectName('student')

self.subjectButton.setObjectName('subject')

self.gradeButton.setObjectName('grade')

self.top10Button.setObjectName('top 10')

self.chartButton.setObjectName('chart')

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self,Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog","BCA Vth Semester Result Analysis 2019-20"))

Dialog.setWindowIcon(QtGui.QIcon('global\_college.jpg'))

self.collegeLabel.setText(\_translate('Dialog','GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE'))

self.selectLabel.setText(\_translate('Dialog','Click on the option which you want:'))

self.studentButton.setText(\_translate("Dialog","Each Student Result"))

self.subjectButton.setText(\_translate("Dialog","Subject Wise Report"))

self.gradeButton.setText(\_translate("Dialog","Grade Wise Report"))

self.top10Button.setText(\_translate("Dialog","Top Ten Students"))

self.chartButton.setText(\_translate("Dialog","Result chart"))

def main():

import sys

app = QtWidgets.QApplication(sys.argv)

Dialog = QtWidgets.QDialog()

ui = SelectionScreen()

ui.setupUi(Dialog)

Dialog.show()

sys.exit(app.exec\_())

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Student Result:**

import sys

import pandas as pd

from PyQt5.QtCore import QAbstractTableModel, Qt, QSize

from PyQt5.QtWidgets import (QApplication, QWidget, QTableView, QHeaderView,

QVBoxLayout, QHBoxLayout, QLabel, QPushButton)

from PyQt5.QtGui import QIcon, QPixmap, QImage, QPalette, QBrush, QFont

df = pd.read\_excel('kud\_result4.xlsx',sheet\_name = 'sheet1')

class StudentResultTable(QAbstractTableModel):

def \_\_init\_\_(self,data):

QAbstractTableModel.\_\_init\_\_(self)

self.\_data = data

def rowCount(self, parent = None):

return len(self.\_data.index)

def columnCount(self, parent = None):

return len(self.\_data.columns)

def data(self, index, role=Qt.DisplayRole):

if index.isValid():

if role == Qt.DisplayRole:

return str(self.\_data.iloc[index.row(),index.column()])

return None

def headerData(self, col, orientation, role):

if orientation == Qt.Horizontal and role == Qt.DisplayRole:

return self.\_data.columns[col]

return None

class StudentResult(QWidget):

def \_\_init\_\_(self):

super(StudentResult,self).\_\_init\_\_()

self.setupUi

def setupUi(self):

self.subject\_report\_table\_model = StudentResultTable(df)

self.view = QTableView()

self.view.setModel(self.subject\_report\_table\_model)

self.view.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeToContents)

self.view.setMaximumSize(1300,380)

self.setGeometry(0,0,1300,700)

self.setWindowTitle("BCA Vth Semester Result Analysis 2019-20")

self.setWindowIcon(QIcon('global\_college.jpg'))

self.backgroundImg = QImage("blue.jpg")

self.setImage = self.backgroundImg.scaled(QSize(1300,700))

palette = QPalette()

palette.setBrush(QPalette.Window, QBrush(self.setImage))

self.setPalette(palette)

self.logoLabel = QLabel('')

self.collegeLabel = QLabel('GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE')

self.headingLabel = QLabel('Each Student Result')

self.infoLabel = QLabel("Max. marks per subject = 100 Min. marks per subject = 40 Total marks = 700")

self.logoLabel.setPixmap(QPixmap("kud2.jpg"))

self.backButton = QPushButton("Back")

self.logoLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setAlignment(Qt.AlignCenter)

self.headingLabel.setAlignment(Qt.AlignCenter)

self.infoLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setFont(QFont("Times",16,QFont.Bold))

self.collegeLabel.setStyleSheet("color: rgb(20,10,100)")

self.headingLabel.setFont(QFont("Times",20,QFont.Bold,True))

self.headingLabel.setStyleSheet("color: white")

self.infoLabel.setFont(QFont("Times",14))

self.infoLabel.setStyleSheet("color: white")

self.backButton.setMaximumSize(130,30)

self.backButton.clicked.connect(self.close)

self.hLayout = QHBoxLayout()

self.vLayout = QVBoxLayout()

self.vLayout.addWidget(self.logoLabel)

self.vLayout.addWidget(self.collegeLabel)

self.vLayout.addWidget(self.headingLabel)

self.vLayout.addWidget(self.infoLabel)

self.vLayout.addWidget(self.view)

self.hLayout.addWidget(self.backButton)

self.vLayout.addLayout(self.hLayout)

self.vLayout.setAlignment(Qt.AlignCenter)

self.vLayout.setSpacing(20)

self.setLayout(self.vLayout)

self.show()

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

result = StudentResult()

result.setupUi()

result.show()

sys.exit(app.exec\_())

**Subject report:**

import sys

import pandas as pd

from PyQt5.QtCore import QAbstractTableModel, Qt, QSize

from PyQt5.QtSql import QSqlTableModel

from PyQt5.QtWidgets import (QMainWindow, QApplication, QWidget, QAction,

QGroupBox, QCheckBox, QTableView, QTableWidgetItem,

QTabWidget, QGridLayout,QLineEdit, QFormLayout, QGroupBox,

QVBoxLayout, QHBoxLayout, QLabel, QDialog, QHeaderView, QPushButton)

from PyQt5.QtGui import QIcon, QPixmap, QImage, QPalette, QBrush, QFont

df = pd.read\_excel('kud\_result4.xlsx',sheet\_name = 'sheet3')

class SubjectReportTable(QAbstractTableModel):

def \_\_init\_\_(self,data):

QAbstractTableModel.\_\_init\_\_(self)

self.\_data = data

def rowCount(self, parent = None):

return self.\_data.shape[0]

def columnCount(self, parent = None):

return self.\_data.shape[1]

def data(self, index, role=Qt.DisplayRole):

if index.isValid():

if role == Qt.DisplayRole:

return str(self.\_data.iloc[index.row(),index.column()])

return None

def headerData(self, col, orientation, role):

if orientation == Qt.Horizontal and role == Qt.DisplayRole:

return self.\_data.columns[col]

return None

class SubjectReport(QWidget):

def \_\_init\_\_(self):

super(SubjectReport,self).\_\_init\_\_()

self.setupUi

def setupUi(self):

self.subject\_report\_table\_model = SubjectReportTable(df)

self.view = QTableView()

self.view.setModel(self.subject\_report\_table\_model)

self.view.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeToContents)

self.view.setMaximumSize(490,233)

ssself.setGeometry(0,0,1300,700)

self.setWindowTitle("BCA Vth Semester Result Analysis 2019-20")

self.setWindowIcon(QIcon('global\_college.jpg'))

self.backgroundImg = QImage("blue.jpg")

self.setImage = self.backgroundImg.scaled(QSize(1300,700))

palette = QPalette()

palette.setBrush(QPalette.Window, QBrush(self.setImage))

self.setPalette(palette)

self.logoLabel = QLabel('')

self.collegeLabel = QLabel('GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE')

self.headingLabel = QLabel('Subject Wise Report')

self.logoLabel.setPixmap(QPixmap("kud2.jpg"))

self.backButton = QPushButton("Back")

self.emptyLabel = QLabel("")

self.logoLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setAlignment(Qt.AlignTop)

self.headingLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setFont(QFont("Times",16,QFont.Bold))

self.collegeLabel.setStyleSheet("color: rgb(20,10,100)")

self.headingLabel.setFont(QFont("Times",20,QFont.Bold,True))

self.headingLabel.setStyleSheet("color: white")

self.backButton.setMaximumSize(130,30)

self.backButton.clicked.connect(self.close)

self.vLayout = QVBoxLayout()

self.h\_tableLayout = QHBoxLayout()

self.hLayout = QHBoxLayout()

self.v\_emptyLayout1 = QVBoxLayout()

self.v\_emptyLayout2 = QVBoxLayout()

self.v\_emptyLayout3 = QVBoxLayout()

self.vLayout.addWidget(self.logoLabel)

self.vLayout.addWidget(self.collegeLabel)

self.vLayout.addWidget(self.headingLabel)

self.h\_tableLayout.addWidget(self.view)

self.hLayout.addWidget(self.backButton)

self.v\_emptyLayout1.addWidget(self.emptyLabel)

self.v\_emptyLayout2.addWidget(self.emptyLabel)

self.v\_emptyLayout3.addWidget(self.emptyLabel)

self.vLayout.addLayout(self.h\_tableLayout)

self.vLayout.addLayout(self.hLayout)

self.vLayout.addLayout(self.v\_emptyLayout1)

self.vLayout.addLayout(self.v\_emptyLayout2)

self.vLayout.addLayout(self.v\_emptyLayout3)

self.vLayout.setAlignment(Qt.AlignCenter)

self.vLayout.setSpacing(20)

self.setLayout(self.vLayout)

self.show()

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

subject = SubjectReport()

subject.setupUi()

subject.show()

sys.exit(app.exec\_())

**Grade Report:**

import sys

import pandas as pd

from PyQt5.QtCore import QAbstractTableModel, Qt, QSize

from PyQt5.QtWidgets import (QApplication, QWidget, QTableView, QHeaderView,

QVBoxLayout, QHBoxLayout, QLabel, QPushButton)

from PyQt5.QtGui import QIcon, QPixmap, QImage, QPalette, QBrush, QFont

df = pd.read\_excel('kud\_result4.xlsx',sheet\_name = 'sheet2')

class GradeReportTable(QAbstractTableModel):

def \_\_init\_\_(self,data):

QAbstractTableModel.\_\_init\_\_(self)

self.\_data = data

def rowCount(self, parent = None):

return self.\_data.shape[0]

def columnCount(self, parent = None):

return self.\_data.shape[1]

def data(self, index, role=Qt.DisplayRole):

if index.isValid():

if role == Qt.DisplayRole:

return str(self.\_data.iloc[index.row(),index.column()])

return None

def headerData(self, col, orientation, role):

if orientation == Qt.Horizontal and role == Qt.DisplayRole:

return self.\_data.columns[col]

return None

class GradeReport(QWidget):

def \_\_init\_\_(self):

super(GradeReport,self).\_\_init\_\_()

def setupUi(self):

self.subject\_report\_table\_model = GradeReportTable(df)

self.view = QTableView()

self.view.setModel(self.subject\_report\_table\_model) self.view.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeToContents)

self.view.setMaximumSize(364,173)

self.setGeometry(0,0,1300,700)

self.setWindowTitle("BCA Vth Semester Result Analysis 2019-20")

self.setWindowIcon(QIcon('global\_college.jpg'))

self.backgroundImg = QImage("blue.jpg")

self.setImage = self.backgroundImg.scaled(QSize(1300,700))

palette = QPalette()

palette.setBrush(QPalette.Window, QBrush(self.setImage))

self.setPalette(palette)

self.logoLabel = QLabel('')

self.collegeLabel = QLabel('GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE')

self.headingLabel = QLabel('Grade Wise Report')

self.logoLabel.setPixmap(QPixmap("kud2.jpg"))

self.backButton = QPushButton("Back")

self.emptyLabel = QLabel("")

self.logoLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setAlignment(Qt.AlignTop)

self.headingLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setFont(QFont("Times",16,QFont.Bold))

self.collegeLabel.setStyleSheet("color: rgb(20,10,100)")

self.headingLabel.setFont(QFont("Times",20,QFont.Bold,True))

self.headingLabel.setStyleSheet("color: white")

self.backButton.setMaximumSize(130,30)

self.backButton.clicked.connect(self.close)

self.vLayout = QVBoxLayout()

self.h\_tableLayout = QHBoxLayout()

self.hLayout = QHBoxLayout()

self.v\_emptyLayout1 = QVBoxLayout()

self.v\_emptyLayout2 = QVBoxLayout()

self.v\_emptyLayout3 = QVBoxLayout()

self.vLayout.addWidget(self.logoLabel)

self.vLayout.addWidget(self.collegeLabel)

self.vLayout.addWidget(self.headingLabel)

self.h\_tableLayout.addWidget(self.view)

self.hLayout.addWidget(self.backButton)

self.v\_emptyLayout1.addWidget(self.emptyLabel)

self.v\_emptyLayout2.addWidget(self.emptyLabel)

self.v\_emptyLayout3.addWidget(self.emptyLabel)

self.vLayout.addLayout(self.h\_tableLayout)

self.vLayout.addLayout(self.hLayout)

self.vLayout.addLayout(self.v\_emptyLayout1)

self.vLayout.addLayout(self.v\_emptyLayout2)

self.vLayout.addLayout(self.v\_emptyLayout3)

self.vLayout.setAlignment(Qt.AlignCenter)

self.vLayout.setSpacing(20)

self.setLayout(self.vLayout)

self.show()

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

grade = GradeReport()

grade.setupUi()

grade.show()

sys.exit(app.exec\_())

**Top Ten Students:**

import sys

import pandas as pd

from PyQt5.QtCore import QAbstractTableModel, Qt, QSize

from PyQt5.QtWidgets import (QApplication, QWidget, QTableView, QHeaderView,

QVBoxLayout, QHBoxLayout, QLabel, QPushButton)

from PyQt5.QtGui import QIcon, QPixmap, QImage, QPalette, QBrush, QFont

df = pd.read\_excel('kud\_result4.xlsx',sheet\_name = 'sheet4')

class Top10StdsTable(QAbstractTableModel):

def \_\_init\_\_(self,data):

QAbstractTableModel.\_\_init\_\_(self)

self.\_data = data

def rowCount(self, parent = None):

return self.\_data.shape[0]

def columnCount(self, parent = None):

return self.\_data.shape[1]

def data(self, index, role=Qt.DisplayRole):

if index.isValid():

if role == Qt.DisplayRole:

return str(self.\_data.iloc[index.row(),index.column()])

return None

def headerData(self, col, orientation, role):

if orientation == Qt.Horizontal and role == Qt.DisplayRole:

return self.\_data.columns[col]

return None

class Top10Stds(QWidget):

def \_\_init\_\_(self):

super(Top10Stds,self).\_\_init\_\_()

def setupUi(self):

self.subject\_report\_table\_model = Top10StdsTable(df)

self.view = QTableView()

self.view.setModel(self.subject\_report\_table\_model)

self.view.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeToContents)

self.view.setMaximumSize(508,323)

self.setGeometry(0,0,1300,700)

self.setWindowTitle("BCA Vth Semester Result Analysis 2019-20")

self.setWindowIcon(QIcon('global\_college.jpg'))

self.backgroundImg = QImage("blue.jpg")

self.setImage = self.backgroundImg.scaled(QSize(1300,700))

palette = QPalette()

palette.setBrush(QPalette.Window, QBrush(self.setImage))

self.setPalette(palette)

self.logoLabel = QLabel('')

self.collegeLabel = QLabel('GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE')

self.headingLabel = QLabel('Top 10 Students')

self.logoLabel.setPixmap(QPixmap("kud2.jpg"))

self.backButton = QPushButton("Back")

self.emptyLabel = QLabel("")

self.logoLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setAlignment(Qt.AlignTop)

self.headingLabel.setAlignment(Qt.AlignCenter)

self.collegeLabel.setFont(QFont("Times",16,QFont.Bold))

self.collegeLabel.setStyleSheet("color: rgb(20,10,100)")

self.headingLabel.setFont(QFont("Times",20,QFont.Bold,True))

self.headingLabel.setStyleSheet("color: white")

self.backButton.setMaximumSize(130,30)

self.backButton.clicked.connect(self.close)

self.vLayout = QVBoxLayout()

self.h\_tableLayout = QHBoxLayout()

self.hLayout = QHBoxLayout()

self.v\_emptyLayout1 = QVBoxLayout()

self.v\_emptyLayout2 = QVBoxLayout()

self.vLayout.addWidget(self.logoLabel)

self.vLayout.addWidget(self.collegeLabel)

self.vLayout.addWidget(self.headingLabel)

self.h\_tableLayout.addWidget(self.view)

self.hLayout.addWidget(self.backButton)

self.v\_emptyLayout1.addWidget(self.emptyLabel)

self.v\_emptyLayout2.addWidget(self.emptyLabel)

self.vLayout.addLayout(self.h\_tableLayout)

self.vLayout.addLayout(self.hLayout)

self.vLayout.addLayout(self.v\_emptyLayout1)

self.vLayout.addLayout(self.v\_emptyLayout2)

self.vLayout.setAlignment(Qt.AlignCenter)

self.vLayout.setSpacing(20)

self.setLayout(self.vLayout)

self.show()

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

top10 = Top10Stds()

top10.setupUi()

top10.show()

sys.exit(app.exec\_())

**pie chart:**

import sys

import pandas as pd

from PyQt5.QtCore import QAbstractTableModel, Qt, QSize

from PyQt5.QtWidgets import (QApplication, QWidget, QTableView, QHeaderView,

QVBoxLayout, QHBoxLayout, QLabel, QPushButton)

from PyQt5.QtGui import QIcon, QPixmap, QImage, QPalette, QBrush, QFont

class PieChart(QWidget):

def \_\_init\_\_(self):

super(PieChart,self).\_\_init\_\_()

def setupUi(self):

self.setGeometry(0,0,1300,700)

self.setWindowTitle("BCA Vth Semester Result Analysis 2019-20")

self.setWindowIcon(QIcon('global\_college.jpg'))

self.backgroundImg = QImage("pie\_chart.png")

self.setImage = self.backgroundImg.scaled(QSize(1300,700))

palette = QPalette()

palette.setBrush(QPalette.Window, QBrush(self.setImage))

self.setPalette(palette)

self.logoLabel = QLabel('')

self.collegeLabel = QLabel('GLOBAL COLLEGE OF MANAGEMENT, IT & COMMERCE')

self.backButton = QPushButton("Back")

self.collegeLabel.setAlignment(Qt.AlignTop)

self.collegeLabel.setFont(QFont("Times",12,QFont.Bold))

self.collegeLabel.setStyleSheet("color: rgb(20,10,100)")

self.backButton.setMaximumSize(130,30)

self.backButton.clicked.connect(self.close)

self.vLayout = QVBoxLayout()

self.h\_tableLayout = QHBoxLayout()

self.hLayout = QHBoxLayout()

self.vLayout.addWidget(self.collegeLabel)

self.hLayout.addWidget(self.backButton)

self.vLayout.addStretch(1)

self.vLayout.addLayout(self.h\_tableLayout)

self.vLayout.addLayout(self.hLayout)

self.vLayout.setAlignment(Qt.AlignCenter)

self.vLayout.setSpacing(20)

self.setLayout(self.vLayout)

self.show()

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

chart = PieChart()

chart.setupUi()

chart.show()

sys.exit(app.exec\_())

**Result analysis:**

import sys

from PyQt5.QtWidgets import QApplication, QDialog, QWidget

from PyQt5.QtGui import QIcon

from kud\_login import \*

from kud\_select\_result import \*

from kud\_student\_result import \*

from kud\_subject\_report import \*

from kud\_grade\_report import \*

from kud\_top10\_stds import \*

from kud\_pieChart import \*

class StudentResultWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.select = StudentResult()

self.select.setupUi()

self.select.backButton.clicked.connect(self.selection)

def selection(self):

self.close()

self.ui = SelectionWindow()

class SubjectReportWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.select = SubjectReport()

self.select.setupUi()

self.select.backButton.clicked.connect(self.selection)

def selection(self):

self.close()

self.ui = SelectionWindow()

class GradeReportWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.select = GradeReport()

self.select.setupUi()

self.select.backButton.clicked.connect(self.selection)

def selection(self):

self.close()

self.ui = SelectionWindow()

class Top10StdsWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.select = Top10Stds()

self.select.setupUi()

self.select.backButton.clicked.connect(self.selection)

def selection(self):

self.close()

self.ui = SelectionWindow()

class PieChartWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.select = PieChart()

self.select.setupUi()

self.select.backButton.clicked.connect(self.selection)

def selection(self):

self.close()

self.ui = SelectionWindow()

class SelectionWindow(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.select = SelectionScreen()

self.select.setupUi(self)

self.show()

self.select.studentButton.clicked.connect(self.student\_result)

self.select.subjectButton.clicked.connect(self.subject\_report)

self.select.gradeButton.clicked.connect(self.grade\_report)

self.select.top10Button.clicked.connect(self.top10\_stds)

self.select.chartButton.clicked.connect(self.pie\_chart)

def student\_result(self):

self.close()

self.ui = StudentResultWindow()

def subject\_report(self):

self.close()

self.ui = SubjectReportWindow()

def grade\_report(self):

self.close()

self.ui = GradeReportWindow()

def top10\_stds(self):

self.close()

self.ui = Top10StdsWindow()

def pie\_chart(self):

self.close()

self.ui = PieChartWindow()

class LoginWindow(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.login = Ui\_LoginWindow()

self.login.setupUi(self)

self.show()

self.login.submitButton.clicked.connect(self.validate)

def validate(self):

username = self.login.usernameEdit.text()

password = self.login.passwordEdit.text()

if username == "" or password == "":

self.login.informationLabel.setText("User name and/or password required")

elif username == "globalcollege" and password == "global":

self.login.informationLabel.setText("Logged in successfully")

self.selection()

else:

self.login.informationLabel.setText("Invalid user name and/or password")

def selection(self):

self.close()

self.ui = SelectionWindow()

def main():

app = QApplication(sys.argv)

interact = LoginWindow()

sys.exit(app.exec\_())

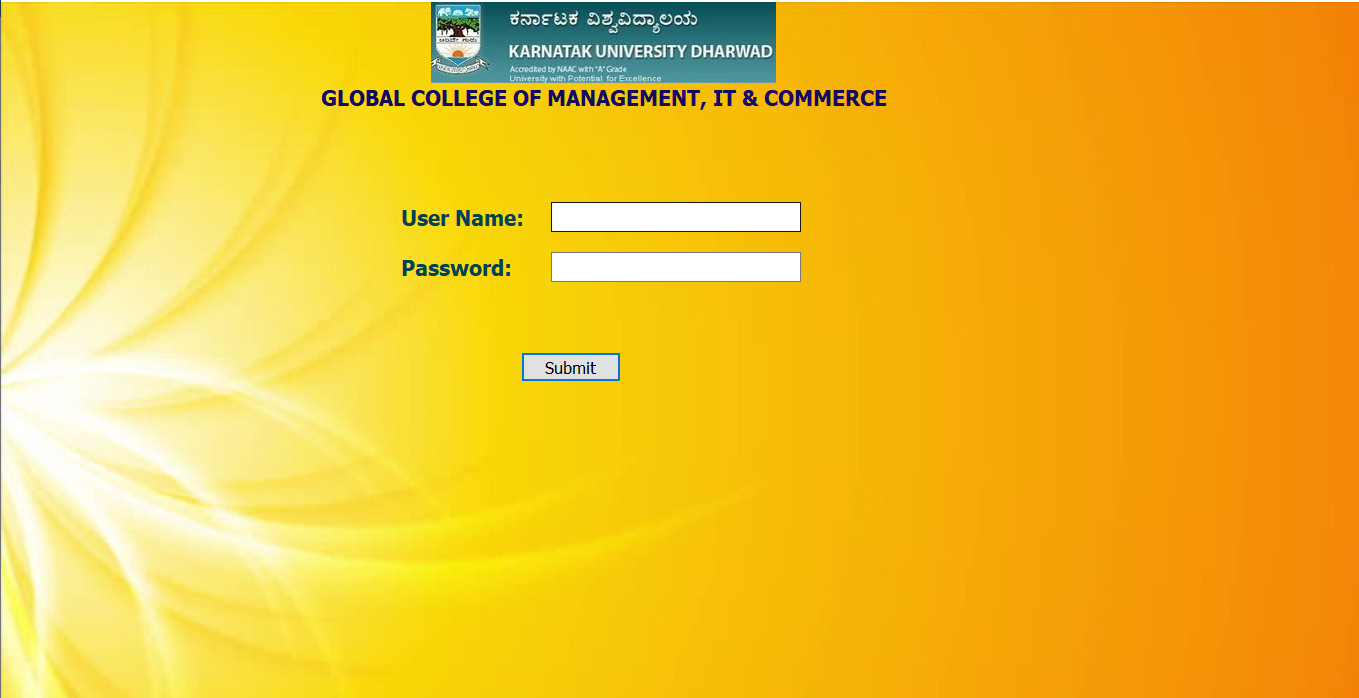
if \_\_name\_\_ == '\_\_main\_\_':

main()

**Chapter:5**

**Screen Shots**

**5.1 Login page:**

****

# **5.2 Main page:**



# **5.3 Each Student Result:**

# **C:\Users\user\Desktop\images\pic 3.PNG**

# **5.4 Subject wise report:**

# **C:\Users\user\Desktop\images\pic 4.PNG**

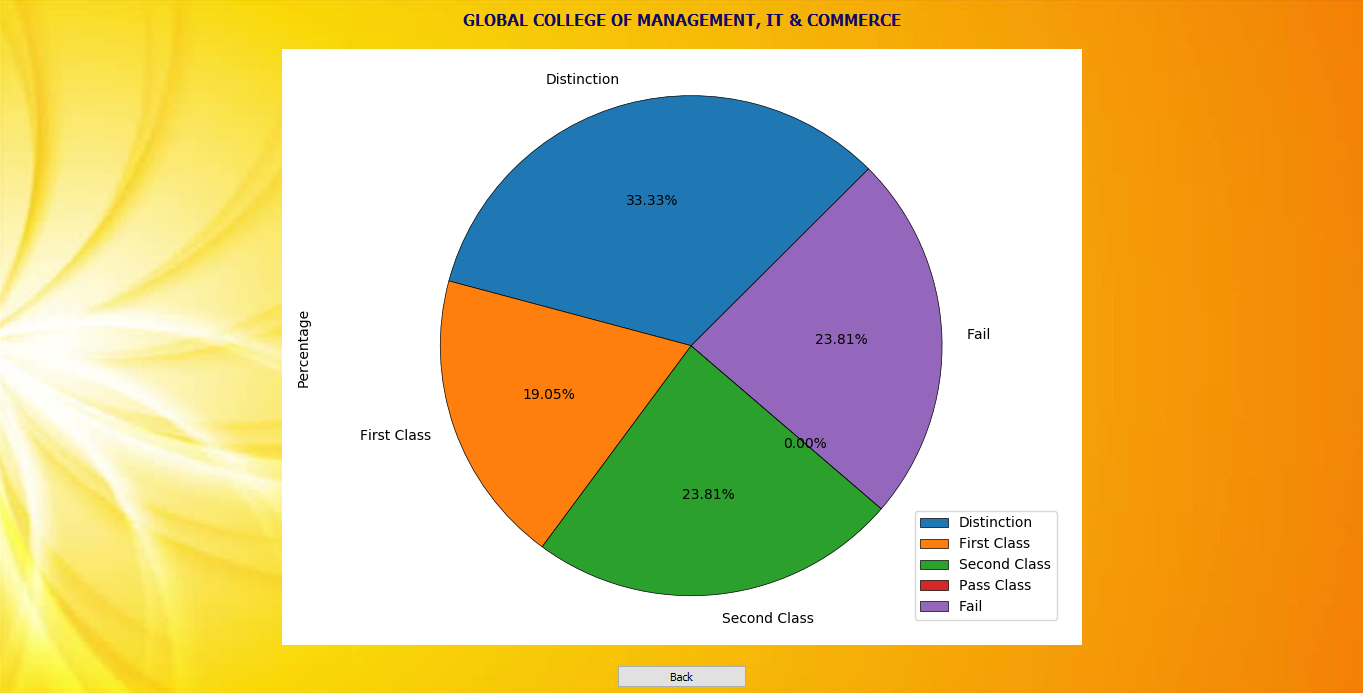
# **5.5 Grade wise report:**



**5.6 Top 10 Students:**



**5.7 Result Chart:**

****

**Chapter:**

**Conclusion**

**6.1 Outcome**

The goal of the system is achieved and difficulties are solved. The project built such that it is a user friendly. Analysis of the scoring system it shows by the grade wise result of individual subject and final result also display grade wise. Depending on its range of marks. The project can be easily used in college for college result analysis of a student. It reduces time which require for manual calculation. This system helps to calculate result fast so it optimizes the manpower.

* 1. **Future enhancement**

Previously, data used to be inserted manually to analyze result. But Currently the project supports excel(.xlsx) files for extraction of data. The future scope is that data can be fetched, parsed in other formats like doc, csv, odt, etc. Visualization can be provided to represent data in graphical format. Various representation like pie chart, graph, etc.

**Chapter: 7**

**References**

* [**https://www.python.org/doc/essays/blurb/**](https://www.python.org/doc/essays/blurb/)
* [**https://www.anaconda.com/distribution/#n/**](https://www.anaconda.com/distribution/#n/)
* [**https://pandas.pydata.org/**](https://pandas.pydata.org/)
* [**https://matplotlib.org/**](https://matplotlib.org/) **or** [**https://build-system.fman.io/pyqt5-tutorial**](https://build-system.fman.io/pyqt5-tutorial)
* [**https://programminghistorian.org/en/lessons/intro-to-beautiful-soup**](https://programminghistorian.org/en/lessons/intro-to-beautiful-soup)
* [**https://pypi.org/project/mechanize/**](https://pypi.org/project/mechanize/)