**Student Name:**

**Roll No:**

**Section:**

Experiment No. 01

*Lab 01 – Introduction to Programming Fundamentals.*

**Lab Objectives:**

1. Introduction to Programming

2. Introduction to Anaconda Framework

3. Introduction to High Level Programming

4. Familiarization with different IDEs

**Introduction to Programming**

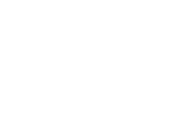
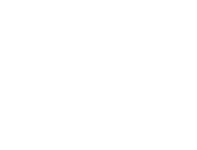
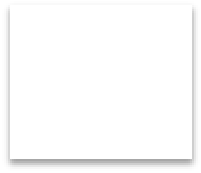
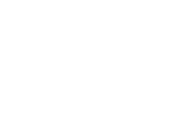
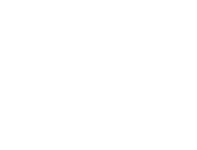
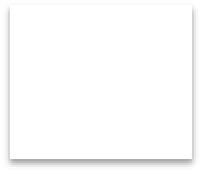
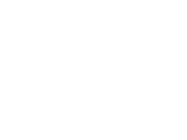
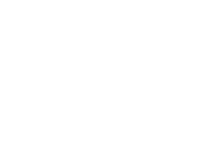
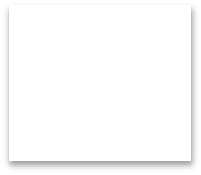
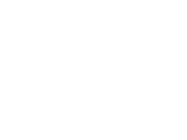
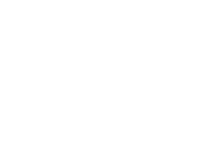
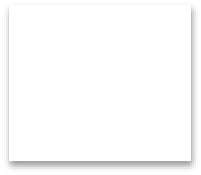
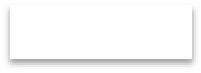
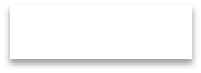
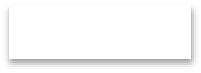
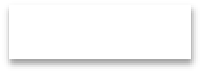
**1. Background:**

Communicating with a computer involves speaking the language that computer understands, which immediately rules out English as the language of communication with computer. The classical method of learning English is to first learn the alphabets used in the language, then learn to combine these alphabets to form words. Instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables and keywords are constructed, and finally how are these combined to form an instruction. A group of instructions would be combined later on to form a program. This is illustrated in the Figure 1.1.

Steps in learning English Language:

Alphabets Words Sentences Paragraphs

Steps in learning Programming Language:



Alphabets, Digits, Special Symbols

Constants, Variables, Key Words

Instruction Programs

*Figure 1- Similarity between English language and any Programming language.*

**Student Name:**

**Roll No:**

**Section:**

**2. Introduction to Anaconda Framework**

Anaconda is a complete, open source data science package with a community of over 6 million users. It is easy to download and install, and it is supported on Linux, MacOS, and Windows.

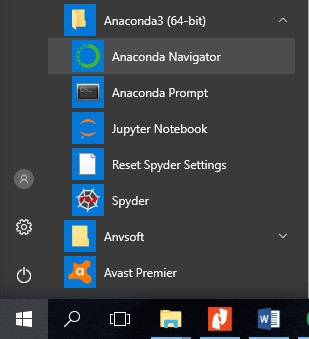
The distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, so it eliminates the need to learn to install each library independently. As Anaconda's website says, "The Python and R conda packages in the Anaconda Repository are curated and compiled in our secure environment so you get optimized binaries that

'just work' on your system."

As you know Anaconda Navigator, a desktop graphical user interface (GUI) system that includes links to all the applications included with the distribution including RStudio, iPython, Jupyter Notebook, JupyterLab, Spyder, Glue, and Orange. The default environment is Python 3.6, but you can also easily install Python 3.5, Python 2.7, or R. The documentation is incredibly detailed and there is an excellent community of users for additional support.

To work in Anaconda Distribution Environment. Click on Windows Icon -> Search for Anaconda3

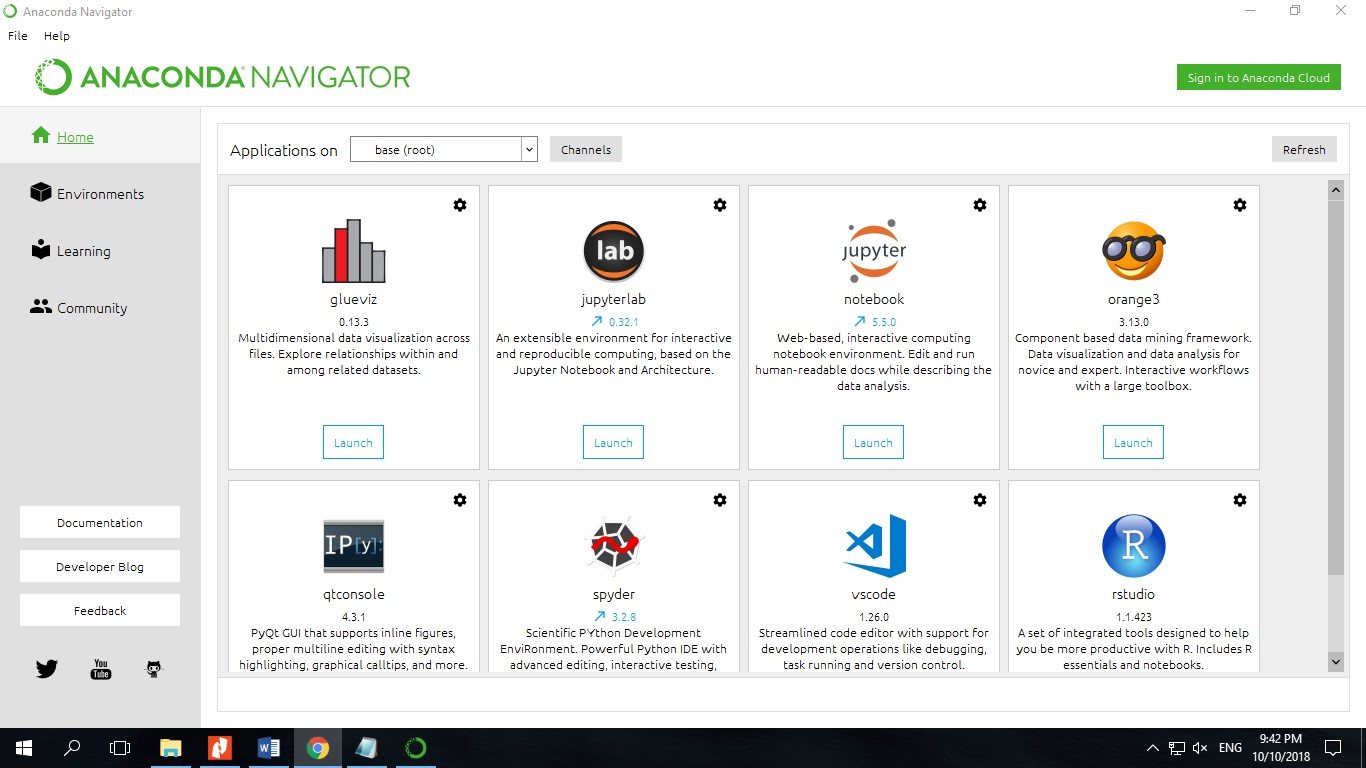
-> Click on Anaconda Navigator.



*Figure 2 - Start Menu*

After you click on the Anaconda Navigator it will take some time depending on your computer configuration and then open the Anaconda Navigator Windows as shown in figure

**Student Name:**



**Roll No:**

**Section:**

*Figure 3 – Anaconda Navigator*

Here we can see that Anaconda Distribution is full of tools such as Jupyter Notebook, Orange 3

,rstudio, vscode, sypder etc. All these tools will be use later in your upcoming semesters. But for you in current semester we will use Jupyter Notebook. So let's start with Jupyter Notebook.

**3. Introduction Jupyter Notebook**

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. The intuitive workflow promotes iterative and rapid development, making notebooks an increasingly popular choice at the heart of contemporary data science, analysis, and increasingly science at large. Best of all, as part of the open source Project Jupyter, they are completely free.

The Jupyter project is the successor to the earlier IPython Notebook, which was first published as a prototype in 2010. Although it is possible to use many different programming languages within Jupyter Notebooks, this article will focus on Python as it is the most common use case.

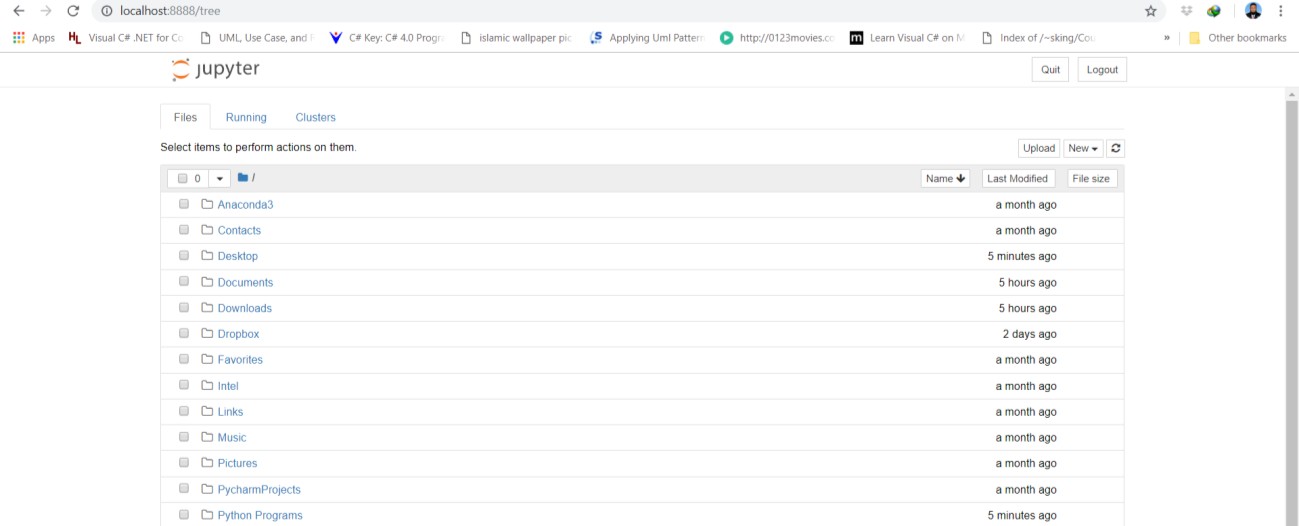
**Student Name:**

**3.1 Running Jupyter**

**Roll No:**

**Section:**

On Windows, you can run Jupyter via the shortcut Anaconda adds to your start menu, which will open a new tab in your default web browser that should look something like the following screenshot.



*Figure 4 - Jupyter*

**3.2 Jupyter Control Panel**

This isn't a notebook just yet, but don't panic! There's not much to it. This is the Notebook Dashboard, specifically designed for managing your Jupyter Notebooks. Think of it as the launchpad for exploring, editing and creating your notebooks.

Be aware that the dashboard will give you access only to the files and sub-folders contained within Jupyter's start-up directory; however, the start-up directory can be changed. It is also possible to start the dashboard on any system via the command prompt (or terminal on Unix systems) by entering the command jupyter notebook; in this case, the current working directory will be the start-up directory.

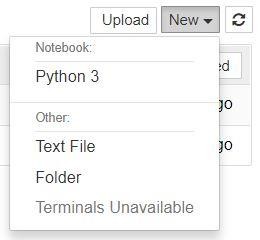
The astute reader may have noticed that the URL for the dashboard is something like http://localhost:8888/tree. Localhost is not a website, but indicates that the content is being served from your local machine: your own computer. Jupyter's Notebooks and dashboard are web apps, and Jupyter starts up a local Python server to serve these apps to your web browser, making it essentially platform independent and opening the door to easier sharing on the web.

**Student Name:**

**Roll No:**

**Section:**

The dashboard's interface is mostly self-explanatory — though we will come back to it briefly later. So what are we waiting for? Browse to the folder in which you would like to create your first notebook, click the "New" drop-down button in the top-right and select "Python 3" (or the version of your choice).



*Figure 5 - Drop down option of New*

**3.3 New Notebook Menu**

Hey presto, here we are! Your first Jupyter Notebook will open in new tab — each notebook uses its own tab because you can open multiple notebooks simultaneously. If you switch back to the dashboard, you will see the new file Untitled.ipynb and you should see some green text that tells you your notebook is running.

**3.4 What is an ipynb File?**

It will be useful to understand what this file really is. Each .ipynb file is a text file that describes the contents of your notebook in a format called JSON. Each cell and its contents, including image attachments that have been converted into strings of text, is listed therein along with some metadata. You can edit this yourself — if you know what you are doing! — by selecting "Edit > Edit Notebook Metadata" from the menu bar in the notebook.

**Student Name:**

**Roll No:**

**Section:**

You can also view the contents of your notebook files by selecting "Edit" from the controls on the

dashboard, but the keyword here is "can"; there's no reason other than curiosity to do so unless you really know what you are doing.

**3.4 The Notebook Interface**

Now that you have an open notebook in front of you, its interface will hopefully not look entirely alien; after all, Jupyter is essentially just an advanced word processor. Why not take a look around? Check out the menus to get a feel for it, especially take a few moments to scroll down the list of commands in the command palette, which is the small button with the keyboard icon (or Ctrl + Shift + P).



*Figure 6 - Notebook Interface*

**3.5 New Jupyter Notebook**

There are two fairly prominent terms that you should notice, which are probably new to you: cells and kernels are key both to understanding Jupyter and to what makes it more than just a word processor. Fortunately, these concepts are not difficult to understand.

A kernel is a "computational engine" that executes the code contained in a notebook document.

A cell is a container for text to be displayed in the notebook or code to be executed by the notebook's kernel.

**3.6 Cells**

We'll return to kernels a little later, but first let's come to grips with cells. Cells form the body of a notebook. In the screenshot of a new notebook in the section above, that box with the green outline is an empty cell. There are two main cell types that we will cover:

A code cell contains code to be executed in the kernel and displays its output below.

**Student Name:**

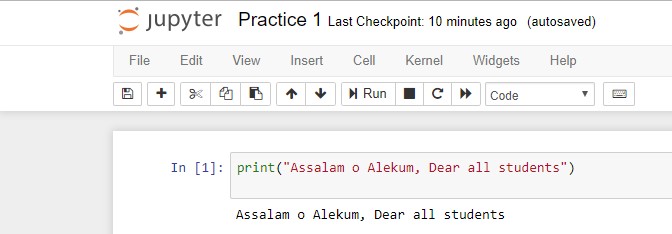
**Roll No:**

**Section:**

A Markdown cell contains text formatted using Markdown and displays its output in-place when

it is run.

The first cell in a new notebook is always a code cell. Let's test it out with a classic hello world example. Type print('Hello World!') into the cell and click the run button Notebook Run Button in the toolbar above or press Ctrl + Enter. The result should look like this:



print("Assalam o Alekum, Dear all students") Assalam o Alekum, Dear all students

When you ran the cell, its output will have been displayed below and the label to its left will have changed from In [ ] to In [1]. The output of a code cell also forms part of the document, which is why you can see it in this article. You can always tell the difference between code and Markdown cells because code cells have that label on the left and Markdown cells do not. The "In" part of the label is simply short for "Input," while the label number indicates when the cell was executed on the kernel — in this case the cell was executed first. Run the cell again and the label will change to In [2] because now the cell was the second to be run on the kernel. It will become clearer why this is so useful later on when we take a closer look at kernels.

**3.7 Keyboard Shortcuts**

One final thing you may have observed when running your cells is that their border turned blue, whereas it was green while you were editing. There is always one "active" cell highlighted with a border whose colour denotes its current mode, where green means "edit mode" and blue is "command mode."

So far we have seen how to run a cell with Ctrl + Enter, but there are plenty more. Keyboard shortcuts are a very popular aspect of the Jupyter environment because they facilitate a speedy cell-based workflow. Many of these are actions you can carry out on the active cell when it's in command mode.

**Student Name:**

**Roll No:**

**Section:**

Below, you'll find a list of some of Jupyter's keyboard shortcuts. You're not expected to pick them up immediately, but the list should give you a good idea of what's possible.

Toggle between edit and command mode with Esc and Enter, respectively. Once in command mode:

Scroll up and down your cells with your Up and Down keys. Press A or B to insert a new cell above or below the active cell. M will transform the active cell to a Markdown cell.

Y will set the active cell to a code cell.

D + D (D twice) will delete the active cell. Z will undo cell deletion.

Hold Shift and press Up or Down to select multiple cells at once. With multple cells selected, Shift + M will merge your selection. Ctrl + Shift + -, in edit mode, will split the active cell at the cursor.

You can also click and Shift + Click in the margin to the left of your cells to select them.

**Student Name:**

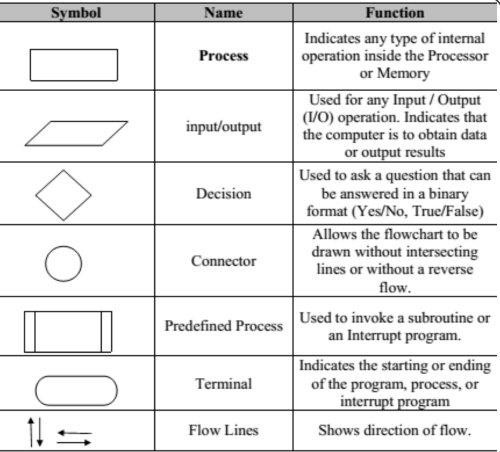
**Roll No:**

**Section:**

**4. Introduction to Flow Charts**

A flowchart is a graphical representation of an algorithm. These flowcharts play a vital role in the programming of a problem and are quite helpful in understanding the logic of complicated and lengthy problems. Once the flowchart is drawn, it becomes easy to write the program in any high level language. Often we see how flowcharts are helpful in explaining the program to others. Hence, it is correct to say that a flowchart is a must for the better documentation of a complex program.

Flowcharts are usually drawn using some standard symbols; however,



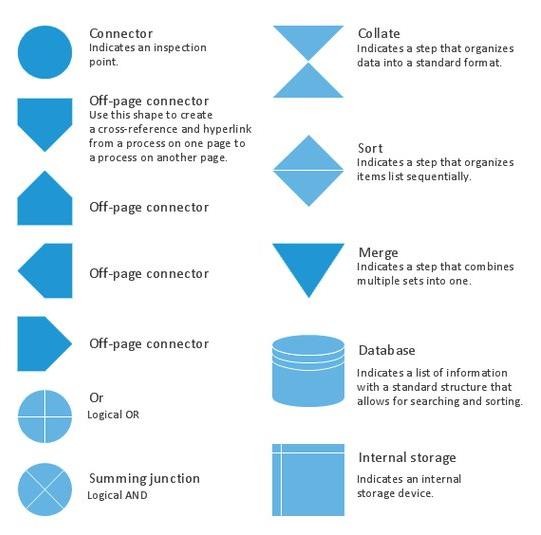
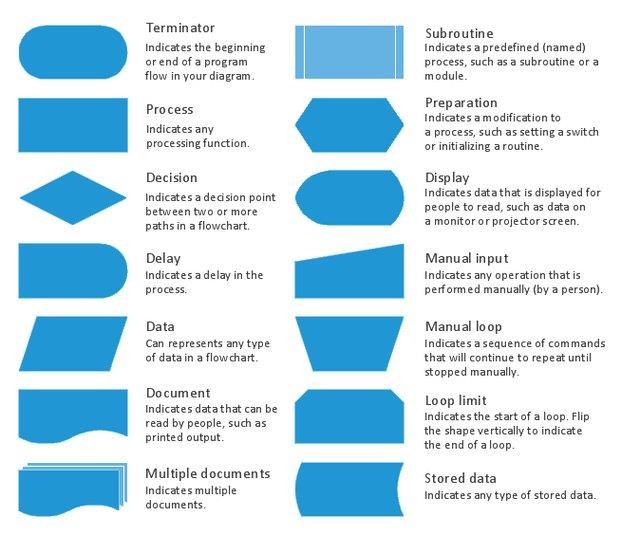
*Figure 7 - Flow chart Symbols*

**Student Name:**

**Roll No:**

**Section:**

**4.1. List of Flow Chart Symbols**



**Student Name:**

**Roll No:**

**Section:**

The following are some guidelines in flowcharting:

a. In drawing a proper flowchart, all necessary requirements should be listed out in logical order. b. The flowchart should be clear, neat and easy to follow. There should not be any room for

ambiguity in understanding the flowchart.

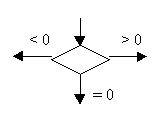
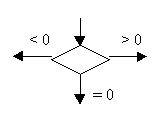
c. The usual direction of the flow of a procedure or system is from left to right or top to bottom.

d. Only one flow line should come out from a process symbol.

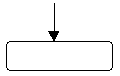
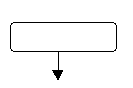
or



e. Only one flow line should enter a decision symbol, but two or three flow lines, one for each possible answer, should leave the decision symbol.



f. Only one flow line is used in conjunction with terminal symbol.



h. If the flowchart becomes complex, it is better to use connector symbols to reduce the number of flow lines. Avoid the intersection of flow lines if you want to make it more effective and better way of communication.

i. Ensure that the flowchart has a logical *start* and *finish.*

j. It is useful to test the validity of the flowchart by passing through it with a simple test data.

**Student Name:**

**Roll No:**

**Section:**

**Problem 1:** Write an algorithm and draw the flowchart for finding the average of two numbers

**Algorithm:**

Input: two numbers x and y Output: the average of x and y Steps:

1. input x

2. input y

3. sum = x + y

4. average = sum /2

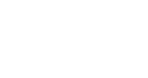
5. output average

START

Input x

Input y

Sum = x + y



Average = sum/2

Output

Average

END

**Student Name:**

**Roll No:**

**Section:**

**Problem 2:** Write an algorithm for finding the area of a rectangle

Hints:

 define the inputs and the outputs

 define the steps

 draw the flowchart

**Problem 3:** Write a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.

name = input("What s your name: ") age = int(input("How old are you: ")) year = str((2019 - age)+100)

print(name + " will be 100 years old in the year " + year)

**Problem 4:** Write a Python program to print the following string in a specific format

Twinkle, twinkle, little star,

How I wonder what you are!

Up above the world so high, Like a diamond in the sky.

Twinkle, twinkle, little star,

How I wonder what you are

print("Twinkle, twinkle, little star, \n\tHow I wonder what you are! \n\t\tUp above the world so high, \n\t\tLike a diamond in the sky. \nTwinkle, twinkle, little star, \n\tHow I wonder what you are!")

**Problem 5:** Write a Python program to display the current date and time. import datetime

now = datetime.datetime.now()

print ("Current date and time : ")

print (now.strftime("%Y-%m-%d %H:%M:%S"))

**Student Name:**

**Roll No:**

**Section:**

**Problem 6:** Write a Python program which accepts the radius of a circle from the user and compute the area.

from math import pi

r = float(input ("Input the radius of the circle : "))

print ("The area of the circle with radius " + str(r) + " is: " + str(pi \* r\*\*2))

**Problem 7:** Write a Python program which accepts the user's first and last name and print them in reverse order with a space between them.

fname = input("Input your First Name : ")

lname = input("Input your Last Name : ")

print ("Hello " + lname + " " + fname)

Problem 8: Write a Python program to print the calendar of a given month and year.

Note: Use 'calendar' module.

import calendar

y = int(input("Input the year : "))

m = int(input("Input the month : "))

print(calendar.month(y, m))

**Student Name:**

**Roll No:**

**Section:**

**Programming Exercise**

1. Write a program for converting Degree Centigrade to Fahrenheit.

2. Write a program for converting Degree Fahrenheit to Centigrade.

3. Write a program to calculate the area of rectangle.

4. Write a program to calculate the volume of a sphere.

5. Write a program that can write your name is upper case, lower case, and title case.