Learning Objective:

1. What is google colab?
2. Intro to python.
3. Variables in Python
4. List in Python
5. Boolean Values
6. Strings in Python
7. Loops

1. Google Colab

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup.

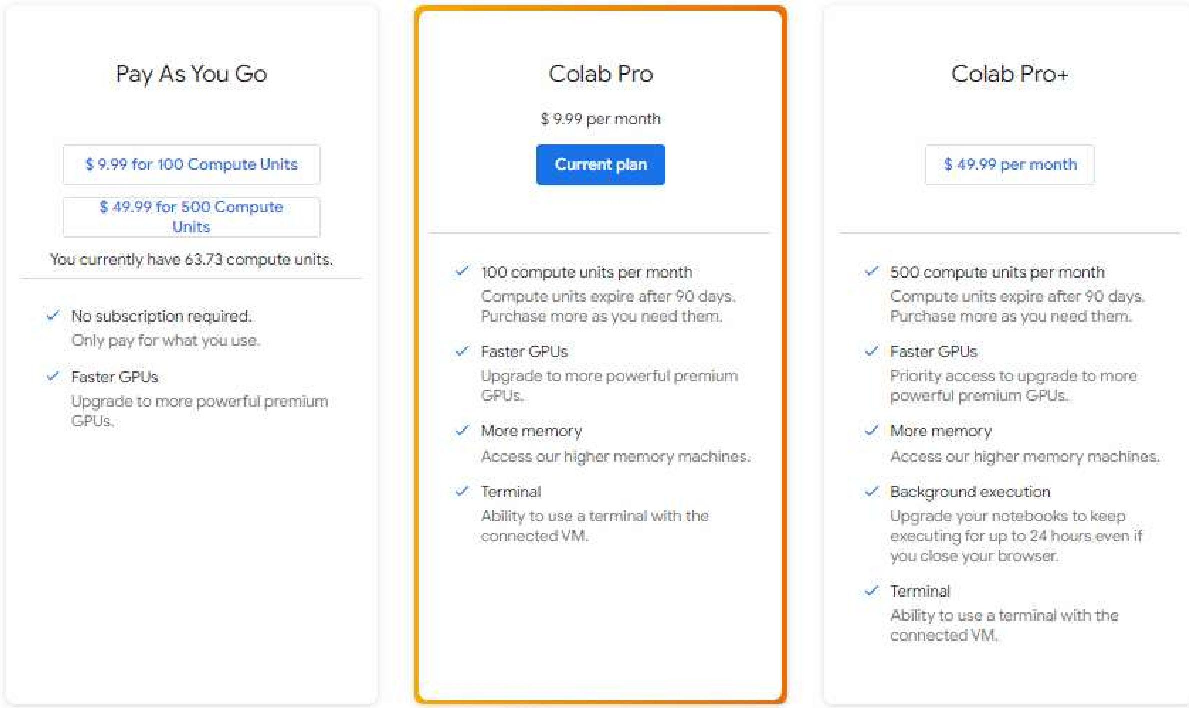
Another attractive feature that Google offers to developers is the use of GPU. Colab supports GPU and it is totally free. The reason for making it free to the public could be to make its software a standard in academics for teaching machine learning and data science. It may also have a long-term perspective of building a customer base for Google Cloud APIs which are sold per-use basis.

What Colab Offers You?

As a programmer, you can perform the following using Google Colab.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

2. Colab subscription plans



---

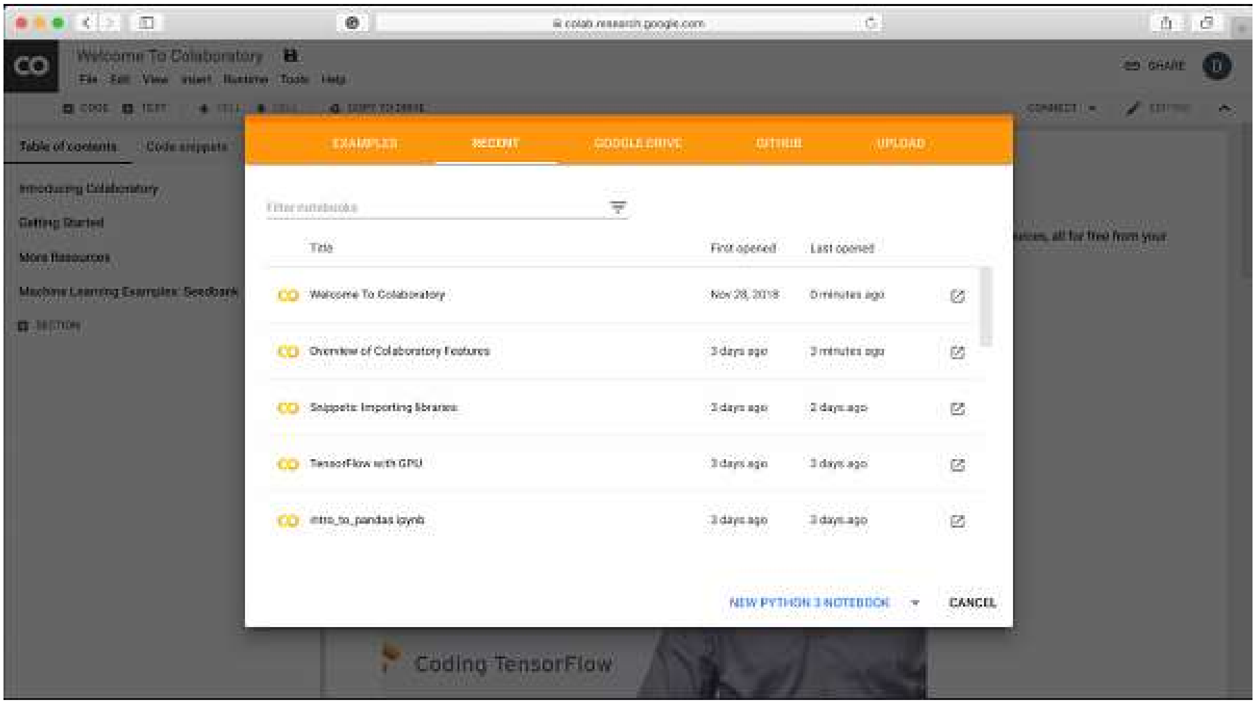
3. Working with Google Colab

If you previously used Jupyter notebook, you would quickly learn to use Google Colab. To be precise, Colab is a free Jupyter notebook environment that runs entirely in the cloud.

Create your first Colab Notebook

Note: As Colab implicitly uses Google Drive for storing your notebooks, ensure that you are logged in to your Google Drive account before proceeding further.

Step 1 : Open the following URL in your browser:
https://colab.research.google.com Your browser would display the following screen (assuming that you are logged into your Google Drive)

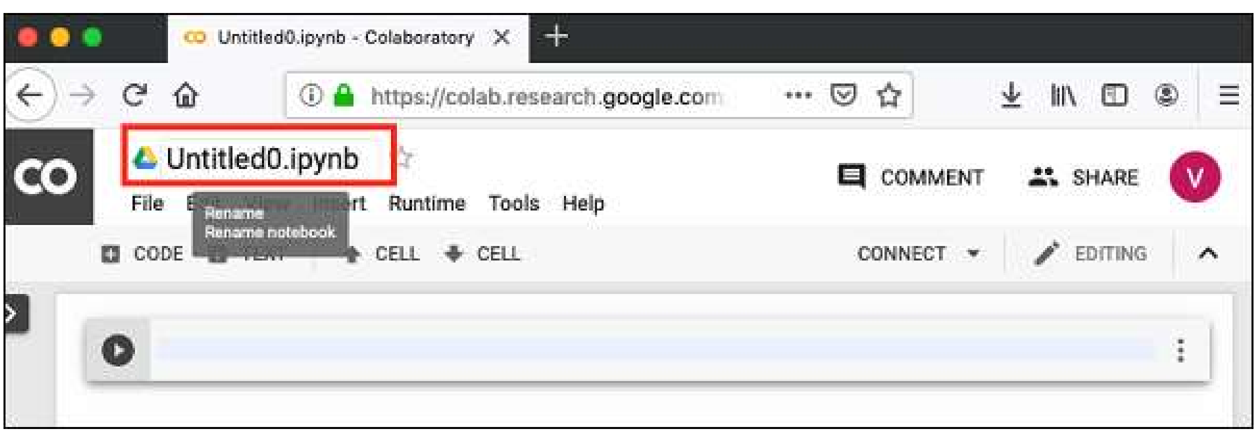


Step 2 – Click on the **New Notebook or NEW PYTHON 3 NOTEBOOK** link at the bottom of the screen. A new notebook would open up as shown in the screen below.

As you might have noticed, the notebook interface is quite similar to the one provided in Jupyter. There is a code window in which you would enter your Python code.

Setting Notebook Name

By default, the notebook uses the naming convention UntitledXX.ipynb. To rename the notebook, click on this name and type in the desired name in the edit box as shown here –



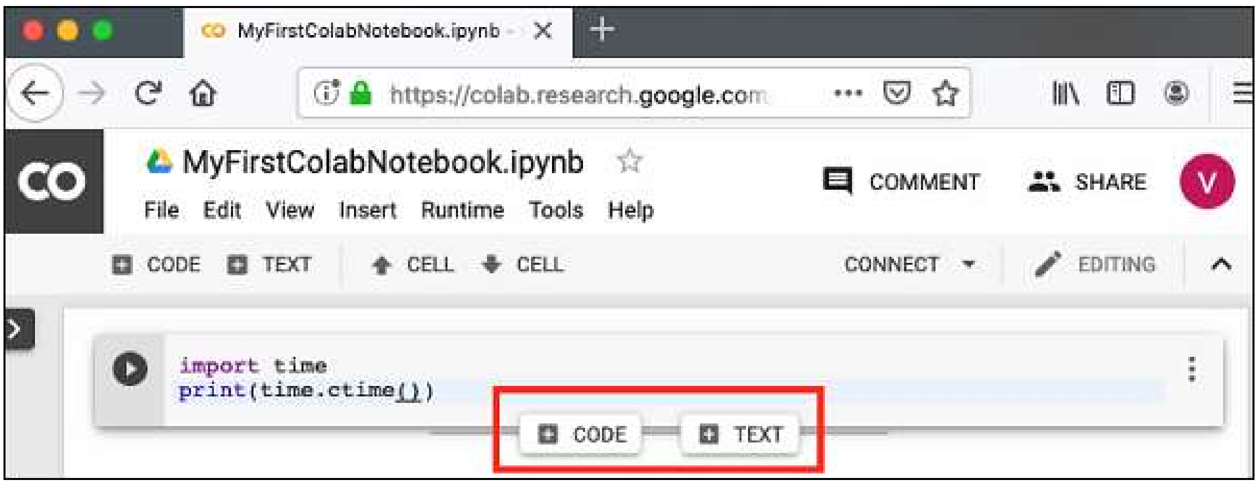
We will call this notebook as **MyFirstColabNotebook/Lab01_IntrotoPython**. So type in this name in the edit box and hit ENTER. The notebook will acquire the name that you have given now.

Adding Code Cells

To add more code to your notebook, select the following **menu** options −

Insert / Code Cell

Alternatively, just hover the mouse at the bottom center of the Code cell. When the **CODE** and **TEXT** buttons appear, click on the CODE to add a new cell. This is shown in the screenshot below −



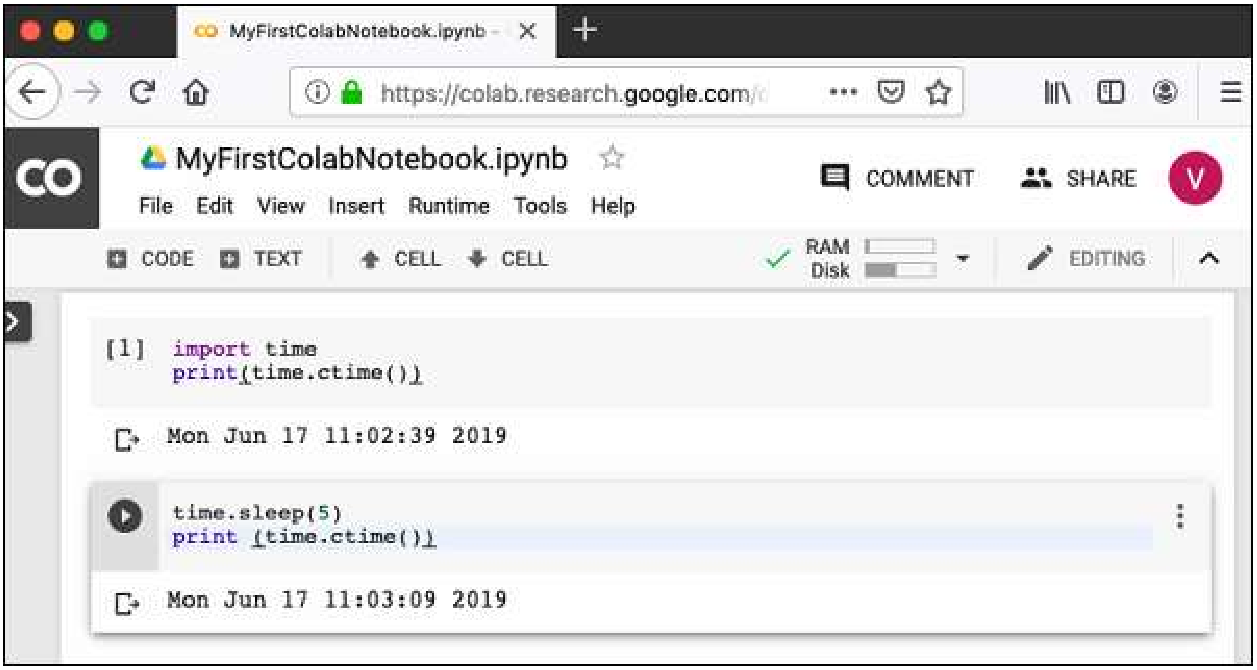
A new code cell will be added underneath the current cell. Add the following two statements in the newly created code window

Run All

To run the entire code in your notebook without an interruption, execute the following menu options:

Runtime / Reset and run all…

It will give you the output as shown below:



Note that the time difference between the two outputs is now exactly 5 seconds.

The above action can also be initiated by executing the following two menu options

Runtime / Restart runtime…

or

Runtime / Restart all runtimes…

Followed by

Runtime / Run all

Study the different menu options under the **Runtime** menu to get yourself acquainted with the various options available to you for executing the notebook.

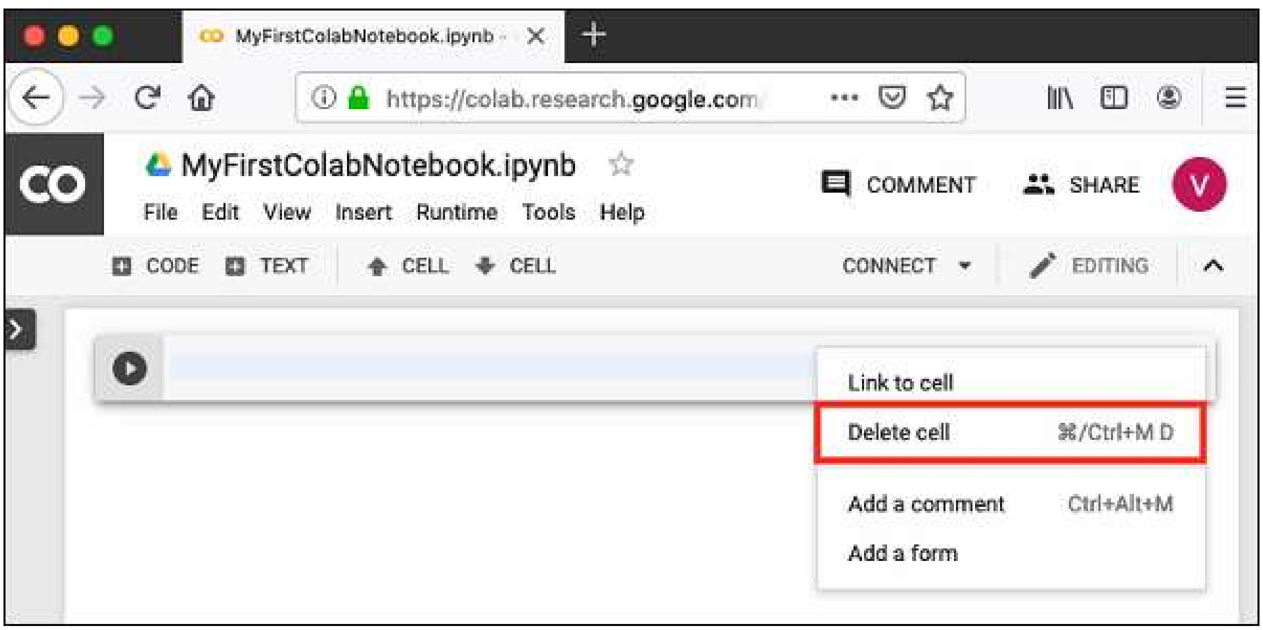
Changing Cell Order

When your notebook contains a large number of code cells, you may come across situations where you would like to change the order of execution of these cells. You can do so by selecting the cell that you want to move and clicking the **UP CELL** or **DOWN CELL** buttons shown in the following screenshot:

You may click the buttons multiple times to move the cell for more than a single position.

Deleting Cell

During the development of your project, you may have introduced a few now-unwanted cells in your notebook. You can remove such cells from your project easily with a single click. Click on the vertical-dotted icon at the top right corner of your code cell.



Click on the **Delete cell** option and the current cell will be deleted.

Now, as you have learned how to run a trivial notebook, let us explore the other capabilities of Colab.

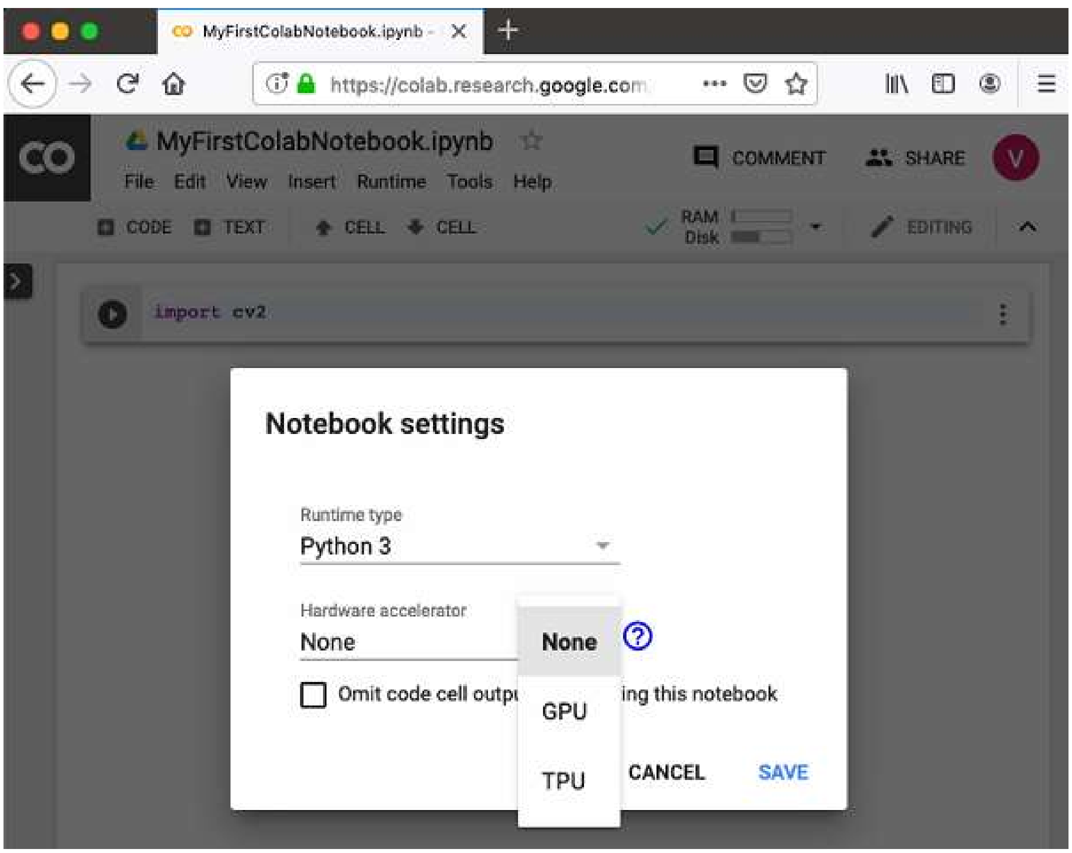
Shortcut Keys of Colab

- Runcell (Ctrl + Enter)
- Runcell and add new cell below (Alt + Enter)
- Runcell and goto the next cell below (Shift + Enter)
- Commentcurrent line (Ctrl + /)
- Addcell above(a)
- Addcell below (b)

Enabling GPU

To enable GPU in your notebook, select the following menu options:

Runtime / Change runtime type
You will see the following screen as the output



Select **GPU** and your notebook would use the free GPU provided in the cloud during processing.

4. Intro to Python

Artificial intelligence is considered to be the trending technology of the future. Already there are a number of applications made on it. Due to this, many companies and researchers are taking interest in it. But the main question that arises here is that in which programming language can these AI applications be developed? There are various programming languages like Lisp, Prolog, C++, Java and Python, which can be used for developing applications of AI. Among them, Python programming language gains a huge popularity and the reasons are as follows:

Simple syntax & less coding

Python involves very less coding and simple syntax among other programming languages which can be used for developing AI applications. Due to this feature, the testing can be easier and we can focus more on programming.

Inbuilt libraries for AI projects

A major advantage for using Python for AI is that it comes with inbuilt libraries. Python has libraries for almost all kinds of AI projects. For example, **NumPy, matplotlib, TnesorFlow, Keras, SimpleAI** are some the important inbuilt libraries of Python.

- **Open source** − Python is an open source programming language. This makes it widely popular in the community.
- **Can be used for broad range of programming** − Python can be used for a broad range of programming tasks like small shell script to enterprise web applications. This is another reason Python is suitable for AI projects.

Python Variables

Python has no command for declaring a variable.
A variable is created the moment you first assign a value to it.

Python Introduction

Datatypes in Python

```
[1] name = "Ali"
    x = 5
    y = 5.5
    print(name)
    print(x)
    print(y)
```

```
Ali
5
5.5
```

```
[2] print("The type of name variable is: ",type(name))
    print("The type of x variable is: ",type(x))
    print("The type of y variable is: ",type(y))
```

```
The type of name variable is:  <class 'str'>
The type of x variable is:  <class 'int'>
The type of y variable is:  <class 'float'>
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the the [Python keywords](#).

Variable casting and its type

Casting

```
[4] x = str(5)    # x will be '3'
    y = int(5)    # y will be 3
    z = float(5)  # z will be 3.0
```

```
[5] print(x)
    print(y)
    print(z)
```

```
5
5
5.0
```

Variable Type

```
[6] x = 5
    print(type(x))
```

```
<class 'int'>
```

6. Boolean Values

Boolean Values

```
[11] print(10 > 9)
     print(10 == 9)
     print(10 < 9)
```

```
True
False
False
```

```
a = 200
b = 33

if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

1. List in Python

- Lists are used to store multiple items in a single variable.
- List items are **ordered**, **changeable**, and **allow duplicate** values.
- List items are **indexed**, the first item has index [0], the second item has index [1] etc.

List in Python

Lists are used to store multiple items in a single variable. List items are **ordered**, **changeable**, and **allow duplicate** values. List items are indexed, the first item has index [0], the second item has index [1] etc.

Allow Duplicates, list() constructor

```
[3] color = ["Black","Blue","Red","Red"]
    color1 = list(("Black","Blue","Red","Red"))
    print(color)
    print(color1)
```

```
['Black', 'Blue', 'Red', 'Red']
['Black', 'Blue', 'Red', 'Red']
```

List length

```
[4] print(len(color))
```

```
4
```

Access List Items

Access List Items

```
[5] print(color[1])
```

```
Blue
```

```
[6] print(color[0:2]) #range of indexes
```

```
['Black', 'Blue']
```

if else condition in python

```
[7] if "Black" in color:
      print("Black is in color list")
    else:
      print("Black is not in color list")
```

```
Black is in color list
```

for loop in python

```
[8] for value in color:
      print(value)
```

```
Black
Blue
Red
Red
```

for loop using range and len

```
[9] for i in range(len(color)):
      print(color[i])
```

```
Black
Blue
Red
Red
```

while loop in pthon

```
[10] i = 0
     while i < len(color):
       print(color[i])
       i = i +1
```

```
Black
Blue
Red
Red
```

Change or Add List Items

Change/Add items in list

```
[11] color[2] = "Orange" #change 3rd index value
     print(color)
```

```
['Black', 'Blue', 'Orange', 'Red']
```

```
[12] color.append("Green")
     print(color)
```

```
['Black', 'Blue', 'Orange', 'Red', 'Green']
```

Remove List Items

Remove list items

```
[13] color.pop()
     print(color)
```

```
['Black', 'Blue', 'Orange', 'Red']
```

```
[14] color.remove("Blue")
     print(color)
```

```
['Black', 'Orange', 'Red']
```

```
[15] del color[0]
     print(color)
```

```
['Orange', 'Red']
```

```
[16] color.clear()
     print(color)
```

```
[]
```

Lab Tasks:

Task 1:

Write a Python program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and 2700 (both included)

Task 2:

Write a Python program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.

Task 3:

Write a Python program to calculate the sum and average of n integer numbers (input from the user). Input 0 to finish.

Task 4:

A student will not be allowed to sit in exam if his/her attendance is less than 75%.

Take following input from user

- Number of classes held
- Number of classes attended.
- And print percentage of class attended
- Is student is allowed to sit in exam or not.

Task 5:

Write a Python program to perform the following tasks using lists:

- Create a list named numbers containing integers from 1 to 5.
- Print the original list.
- Append the number 6 to the list.
- Insert the number 0 at the beginning of the list.
- Print the modified list.
- Remove the number 3 from the list.
- Print the final list.

Lab # 01 Marks distribution

	ER1	ER6	ER8
Task	30 points	30 points	40 points

Lab # 01 Rubric Evaluation Guideline:

#	Qualities & Criteria	0 < Poor <= 10	10 < Satisfactory <= 20	20 < Excellent <=30
ER 1	**Task Completion**	Minimal or no program functionality was achieved.	Some tasks were completed, but the program has errors or incomplete functionalities.	All tasks were completed, and the program runs without errors.
#	**Qualities & Criteria**	**0 < Poor <= 10**	**10 < Satisfactory <= 20**	**20 < Excellent <=30**
ER 6	**Program Output**	Output is inaccurate or poorly presented.	Output is mostly accurate but may lack labels, captions, or formatting.	Output is clear, accurate, and well presented with labels, captions, and proper formatting.
#	**Qualities & Criteria**	**0 < Poor <= 16**	**16 < Satisfactory <= 28**	**28 < Excellent <= 40**
ER 8	**Question & Answer**	Answers some questions but not confidently or based on lab task knowledge.	Answers most questions confidently and based on lab task knowledge.	Answers all questions confidently and demonstrates a deep understanding of the given lab task.