# Week 1: Lab pre-read

# Welcome to the first lab session of the course

In this session we will:

* Install Python and some other necessary packages
* Basic usage of python: shell and IDE (IDLE)
* Learning about variables and types of primitives.
* Performing basic operations on different types of primitives.
* Building more complex tools (calculator, introduction to the computer).
* Jupyter Notebook. (maybe as a home assignment?)

Remember, don’t hesitate to ask us any questions if you are stuck at any point or have any confusions, we’re here to help. With that let’s get started.

## 1. Installing Python and other necessary packages

### Windows/Mac System

Start by downloading the executable [python](https://www.python.org/downloads/release/python-397/) package that is appropriate for your system (find the download link under the files section on the page).

Once the file is downloaded, double click on it and the installer should open. Follow along with the instructions. You may also check the box to **add Python 3.9 to PATH** (this essentially makes it possible for you to run Python from your command prompt and access the Python shell). On the final screen you can also **disable the path length limit**. Once completed, you can skip to section 1.1.

### Linux/Ubuntu System

Ubuntu should come with Python pre-installed, to check your version of Python start with opening a terminal (**Ctrl+Alt+T**) and enter the following command:

python --version

If there is an error, or Python is not installed, or if the version is lower than 3.7x then enter the following commands:

sudo apt update

sudo apt -y upgrade

Here sudo stands for super user privileges and apt is Ubuntu’s **A**dvanced **P**ackaging **T**ool and -y agrees to all items to be installed. The sudo command will require you to enter the password that you created at the time of installation. Once this is complete check the version again using the same command as earlier.

Next we will install pip, which is a tool to install the needed packages for programming. Begin with:

sudo apt install -y python3-pip

### 1.1 Installing packages

Packages are groups of modules that allow you to add certain functionality to your code. The general command to install any Python packages is using the pip command (which is a package manager) in the command prompt or terminal:

pip install package\_name

Where package\_name will be the library or package you want to install such as NumPy, matplotlib etc. The installation guidelines for any package can be found on their website. Some of the packages that you can now install are:

1. NumPy
2. Matplotlib
3. JupyterLab or Jupyter Notebook
4. Pandas

You can check the list of installed packages using the following command:

pip list

## 2. Create and run your first program

Congratulations on getting Python ready to go. Let’s take it for a little spin and give you a taste of it. We will now make the simplest and the most famous Python program and run it.

But wait, where will you write your code? Every Python installation comes with an **Integrated Development and Learning Environment** or an **IDLE**. To open this search for *IDLE* in the search bar and open the application or run idle3 from the terminal in Linux. This will open the Python shell (the shell can also be accessed from the command prompt or terminal by running the python command).

You can run single commands from the shell which reads a Python statement, evaluates the result of that statement, and then prints the result on the screen.

**Try out these two tasks:**

* **Now write a command to add two numbers and press enter.**
* **Write a command to print the text “hello” using the print() function as print(**"**hello**"**) and press enter.**

This is simple enough, but what if you want to run multiple commands together? Would you go on typing them one at a time? Hmm, there must be a better way...of course there is. Let’s now make an executable python program. We will now make the simplest and the most famous Python program and run it.

In the IDLE click on **File** and then select **New File** (or **Ctrl + N**). A text editor window will open. In the text editor, add the print statements as follows:

print("Hello, World!")

print("I am IDLE")

Save the file with the name **hello.py**. Finally, to run the program you just created go to **Run** and select **Run Module** or press **F5**. You will see the output in the Python Shell.

The program can also be run from directly from a terminal or command prompt by first navigating to the directory where the file is located using the cd (change directory) command followed by the path, for instance if the file is on the Desktop run the following command:

cd Desktop

Then type:

python hello.py

You should now see the output. Congratulations on building your first program and taking the first step in the world of programming!

## 3. Primitives and Variables

### 3.1 Variables

* Variables are 'containers' to which you assign a value.
* Case sensitive ('a' and 'A' are different variables).
* Must start with a letter or underscore and not a number.
* Can only contain alphanumeric characters (A-Z, a-z, 0-9 and '\_').
* Variable names with more than one word should be joined by an underscore.
* Should not use [keywords](https://www.w3schools.com/python/python_ref_keywords.asp).
* Declaring the same variable again will overwrite the value.

### 3.2 Primitives

There are 4 types of primitives in Python.

* **Integers** - these are whole numbers or integers from negative infinity to infinity e.g. 4, -20.
* **Float** - these are floating point numbers which represent fraction in the decimal format e.g. 4.0, 5.25. (**Note**: 1 is an integer while 1.0 is a float)
* **String** - these are collections of alphabets and/or other characters. A string is declared by wrapping it in single or double quotes e.g. ‘Hello’, “lunch time”, “82”.
* **Boolean** - these are logical data types which only take the value of **True** or **False** and are useful for conditional statements or comparisons.

In the shell, write the following commands assigning a different primitive value to different variables;

a = 3

b = 'Hello!'

c = 56.2

d = True

Now find the primitive type assigned to each variable. (**Hint**: use the **type()** function).

What would happen if you tried to combine different primitive types in an operation? Try to add the 4 variables with themselves and with each other and see what happens.

You will see that apart from the string all other primitive types can be added to each other. This is interesting as the Boolean value behaves as 1 for a value of True and 0 for a value of False. Though these types of operations are not common it is still interesting to see their behaviour.

## 4. Basic operations

* Start by exploring the basic mathematical operations (**+, −, ∗, /, ∗∗, //, %**) on integers in the shell.
* Use them on different primitive types and see what happens.

Make a .py file for these operations that prints the results as well as what the operation does.

### 4.1 Comparison Operators

As the name suggests comparison operators are used to compare two values. The comparison returns a True or False value.

* Greater than >
* Lower than <
* Greater than or equal to >=
* Lower than or equal to <+
* Equal to ==
* Not equal to !=

Try using these operations on different primitive types.

### 4.2 Logical Operators

These are used to combine two or more conditions or comparison expressions.

* and
* or
* not

Try combining comparison operators using the logical operators. The syntax for writing such commands is:

condition\_1 logical\_operator condition\_2

You can read more about operators [here](https://www.tutorialsteacher.com/python/python-operators).

## 5. Build a simple interactive calculator

Write a **calculator.py** program that uses the input() and print() functions to perform multiple operations (addition, subtraction, multiplication, division) on two numbers that the program asks you to input and returns each result with a statement such as “The sum of the two number is:” and so on.

**Hint**: you can use the **input()** function to take an input from the user (you or someone else who will use your program) and assign the input value to a variable. The syntax for writing this is shown below:

a = input("Enter a value:")

The prompt inside the brackets is optional but it always helps to add an instructive text to guide the user as to what the input should be.

Be careful that the input() function converts the input to a string and not an integer, so you will have to use an inbuilt function called int() to change it to an integer. You would do this by the following method:

a = int(a)

Or by wrapping the input() function inside the int() function at the beginning.

If at any point you are stuck with the usage of these functions please feel free to ask us for help. Once you have built your program, run it to see if it is working as expected.

## Congratulations!

You have officially entered the world of programming. Hopefully this has not scared you too much and you could get through it with relatively little trouble. If you do face any trouble feel free to ask for help from either your peers or from us. See you next week!

You can contact us at:

[raghav.awasty@plaksha.edu.in](mailto:raghav.awasty@plaksha.org)

[sachin.lokesh@plaksha.edu.in](mailto:sachin.lokesh@plaksha.org)

## Optional exercises/homework? (for those who would like to explore more)

Using the input() function, write a program that asks for your name and age and prints it in a sentence.

Using the input() function, write a program that converts an integer to a binary number using simple mathematical operations (without using the inbuilt python functions).

Install and explore Jupyter Notebook which is an interactive way of working with Python and allows you to work on large pieces of codes in blocks. You can download and open the **Week\_1.ipynb** file in Jupyter, which is basically the same exercises that you did today with an additional exercise at the end.