

Contents

| | |
|---|----|
| Acknowledgements | 3 |
| About the Book | 6 |
| Chapter 1: Algorithms and Flowcharts | 10 |
| Recap..... | 10 |
| Quiz Time! | 10 |
| Introduction | 14 |
| What is an Algorithm? | 15 |
| Activity | 15 |
| Challenge Time..... | 16 |
| What is a flowchart?..... | 18 |
| How to Use Flowcharts to Represent Algorithms | 19 |
| Challenge time! | 20 |
| Test Your Knowledge | 20 |
| Chapter 2: Introduction to Python..... | 21 |
| What is a program?..... | 21 |
| What is Python?..... | 21 |
| Why Python for AI? | 22 |
| Applications of Python..... | 22 |
| Getting started with Python..... | 23 |
| Downloading and Setting up Python for use..... | 23 |
| Python IDLE installation | 23 |
| Run in the Integrated Development Environment (IDE) | 29 |
| Interactive Mode | 29 |
| Script Mode..... | 30 |
| Python Statement and Comments | 32 |
| Python Statement | 32 |
| Python Comments | 32 |
| Python Keywords and Identifiers..... | 33 |
| Variables and Datatypes | 35 |
| Python Operators I | 41 |
| Arithmetic Operators | 41 |
| Python Input and Output..... | 41 |
| Python Output Using print() function | 41 |
| User input | 42 |
| Type Conversion | 42 |

| | |
|---|----|
| Implicit Type Conversion | 42 |
| Explicit Type Conversion..... | 44 |
| Python Operators II | 47 |
| Comparison operators..... | 47 |
| Logical operators | 47 |
| Assignment operators | 48 |
| Let's Practice..... | 48 |
| Test Your Knowledge | 49 |
| Chapter 3 - Introduction to tools for AI..... | 50 |
| Recap | 50 |
| Introduction to Anaconda..... | 50 |
| How to install Anaconda?..... | 50 |
| Jupyter Notebook | 56 |
| Introduction | 56 |
| What is a Notebook?..... | 56 |
| Installing Jupyter Notebook..... | 57 |
| Working with Jupyter Notebook..... | 57 |
| Notebook Interface - Explained! | 58 |
| Test Your Knowledge | 64 |
| Chapter 4 - More About Lists and Tuples | 65 |
| Introduction to Lists | 65 |
| How to create a list ? | 65 |
| How to access elements of a list ?..... | 65 |
| List Index | 66 |
| Negative Indexing | 66 |
| Adding Element to a List..... | 67 |
| Using append() method..... | 67 |
| Using insert() Method..... | 68 |
| Using extend() method..... | 68 |
| Removing Elements from a List..... | 68 |
| Using remove() method..... | 69 |
| Using pop() method | 69 |
| Slicing of a List | 70 |
| List Methods | 72 |
| Let's Practice..... | 72 |
| Introduction to Tuples | 72 |

| | |
|---|----|
| How to Create a tuple ? | 73 |
| Accessing of Tuples | 73 |
| Deleting a Tuple | 73 |
| Test Your Knowledge | 74 |
| Chapter 5 - Flow of Control and Conditions..... | 75 |
| If Statement..... | 75 |
| If Statement | 77 |
| Python if...else Statement | 78 |
| Python if...elif...else Statement..... | 79 |
| Syntax of if...elif...else | 79 |
| Python Nested if statements | 81 |
| Let's Practice..... | 81 |
| The For Loop..... | 82 |
| Syntax of for Loop..... | 82 |
| Flowchart of for Loop | 82 |
| Example: Python for Loop..... | 82 |
| The while Statement..... | 83 |
| Syntax of while Loop in Python | 83 |
| Flowchart of while Loop..... | 84 |
| Example: Python while Loop | 84 |
| Let's Practice..... | 85 |
| Test Your Knowledge | 85 |
| Chapter 6 : Introduction to Packages | 86 |
| Recap..... | 86 |
| CHALLENGE TIME!..... | 86 |
| Introduction | 87 |
| What is a package? | 90 |
| Package Installation..... | 91 |
| Working with a package | 91 |
| What is NumPy? | 92 |
| Exploring NumPy! | 93 |
| Let's Practice!..... | 94 |
| TASK 1 | 94 |
| TASK 2 | 94 |
| Test Your Knowledge | 94 |
| Additional Resources | 95 |

Chapter 1: Algorithms and Flowcharts

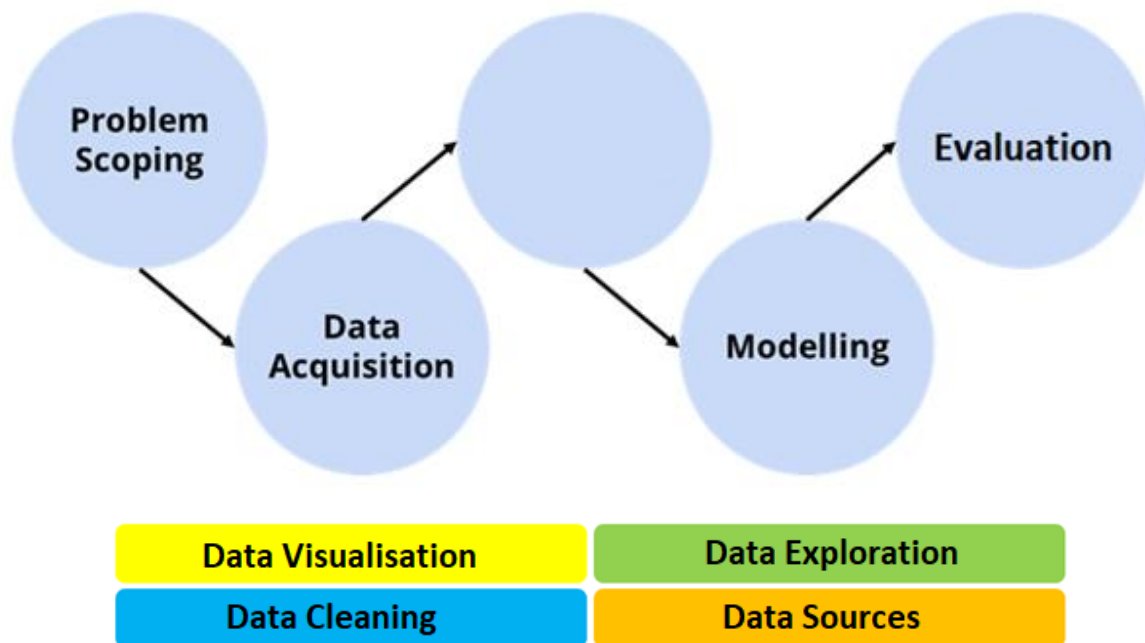
Recap

Till now we have gone through an experiential learning around Artificial Intelligence in which we got to know about so many new concepts like what is Artificial Intelligence, Machine Learning, Deep learning, Rule-based approach, learning based approach, neural networks, etc. Now, it is time to move ahead in this journey towards AI Readiness and work around the concepts we are introduced to through hands-on learning sessions.

But before we start getting our hands dirty, let us recall what has been done so far. Here is a quiz through which you can challenge yourself and see how well do you remember things.

Quiz Time!

1. Which stage is missing in AI Project Cycle?



2. Which one of the following is not an SDG?



3. What will be the output for this program?

```
options = ['data', 'AI', 'CV', 'NLP']  
choice = 1  
print(options[1])
```

NLP

CV

AI

Data

4. Large Neural Networks are time consuming and computationally expensive.

TRUE

FALSE

5. Which one of the following is not an AI application?

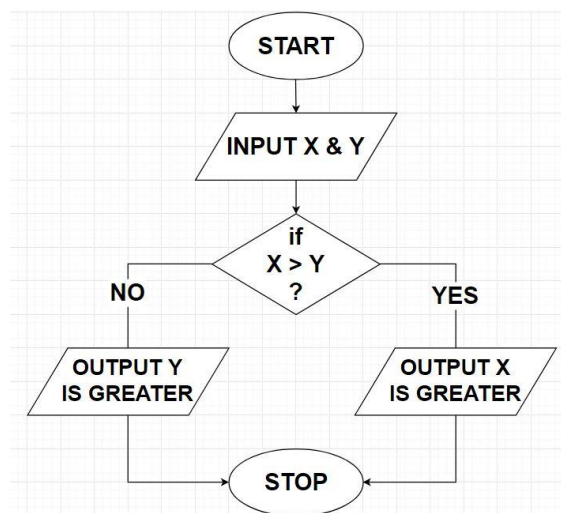
Facial Recognition

Writing Suggestion in Gmail

Kahoot

Google Maps

6. What is this flowchart for?



Find Largest Number

Print X & Y

Take X & Y as Input

None of the Above

7. Google Deep Mind AI is based on reinforcement learning.

TRUE

FALSE

8. Wikipedia can be an appropriate Data Source for which domain?

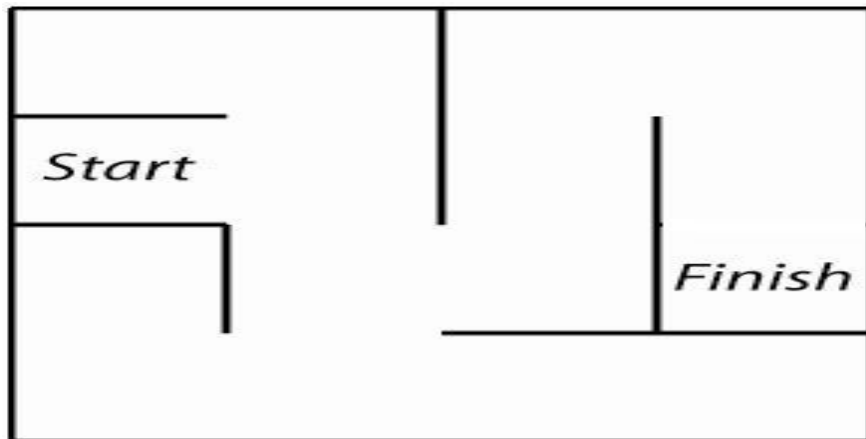
Data Science

Computer Vision

Natural Language Processing

None of the Above

9. What is the correct step to reach finish from first?



Straight, Right, Left, Left, Right, Right

Right, Left, Straight, Left, Right

Straight, Right, Left, Right

Straight, Right, Left, Right, Left, Right

10. Pixel It is an example of machine learning AI approach.

TRUE

FALSE

11. AI can be implemented in which of the following field?

Medical Science

Agriculture

Education

All of the above

12. Problem statement template does not talk about goal of the project?

TRUE

FALSE

13. Snapchat is an example of Natural Language Processing domain?

TRUE

FALSE

14. Which of the following is commonly used for image processing in python?

Pandas

Matplotlib

OpenCV

CV

15. Box plots help in finding out_____?

Erroneous Data

Outliers

Mean of the data

None of the above

16. The AI domain which can be used to predict the Air Quality Index is?

Data Science

Computer Vision

Natural Language Processing

None of the Above

17. Where is decision tree used?

Classification

Regression

Clustering

Dimensionality Reduction

18. Which process is depicted by these steps?

Calculate average marks

Calculate sum

Calculate percentage

Analyse marks of student

19. Which of the following is used to plot graphs in python?

Numpy

Pandas

Scikit

Matplotlib

20. How was the overall experience of the workshop?

It was great!


It was too complicated!

It was easy & understandable

It was boring

How well did you score in the quiz? _____

Analyze your thoughts here:




What you **K**now?

What you **W**ant to Know?

What have you **L**earned?

How have your Learned?




What you **K**now?

What you **W**ant to Know?

What have you **L**earned?

How have your Learned?




What you **K**now?

What you **W**ant to Know?

What have you **L**earned?

How have your Learned?



What you **K**now?

What you **W**ant to Know?

What have you **L**earned?

How have your Learned?

Introduction

Every machine, whether AI enabled or not, follows a particular path of action. It goes through exact specific steps which have been programmed into it to accomplish the given task. For example, a washing machine can wash clothes for us and not do anything else because it is designed explicitly for this task. It follows the steps programmed into it and completes the work.

Can you think of possible steps to wash clothes in a washing machine?

This image shows a single sheet of white paper with horizontal blue ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Ever wondered how do people create such programs? Write your thoughts below.



To develop a program, the very first step that comes into the picture is to specify the task which the machine needs to do. Once the task or the main objective of the program is finalized, the task is broken into smaller tasks which altogether contribute towards achieving the main goal. To make sure that the flow of the process is proper, algorithms and flowcharts are used which help us into developing a stepwise framework to achieve the main goal.

What is an Algorithm?

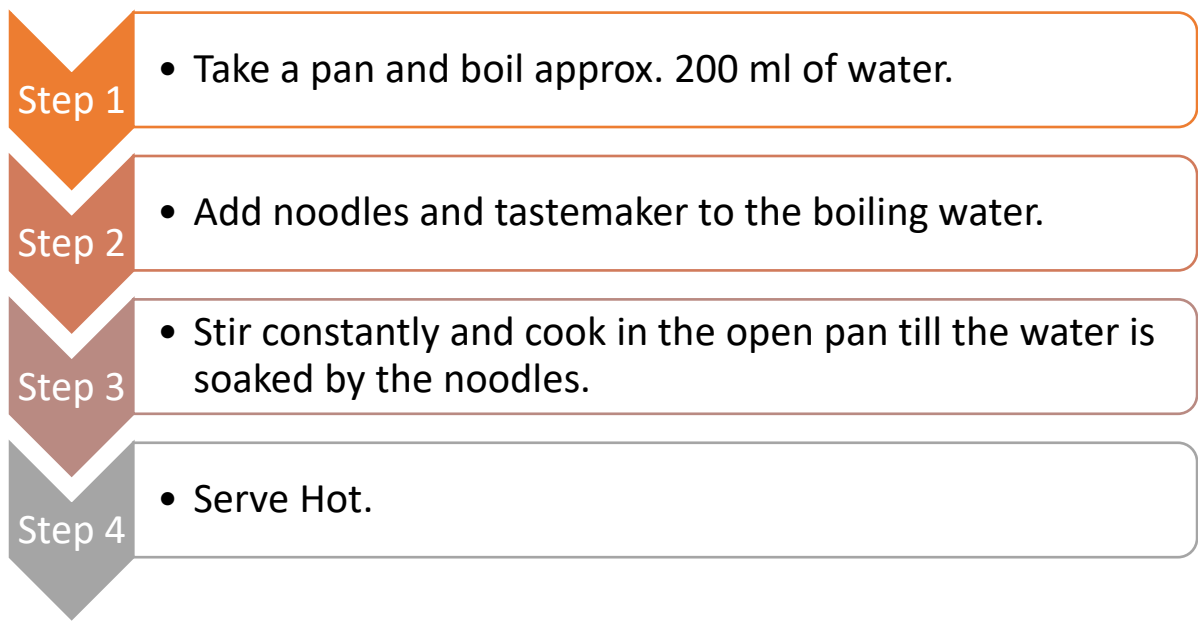
To write a logical step-by-step method to solve the identified problem is called algorithm, in other words, an algorithm is a procedure for solving problems. In order to solve a mathematical or computer problem, this is the first step of the procedure. An algorithm includes calculations, reasoning and data processing. Algorithms can be presented by natural languages, pseudocode and flowcharts, etc.

Activity

Goal: To give a glimpse of how to write a step by step algorithm for any problem/process in a bidirectional manner.

To understand algorithms better, let us take an example of the process of making instant noodles.

Basic steps to make any instant noodles are:



Can you think of other ways to make instant noodles? Write them down.

[illegible]

Challenge Time

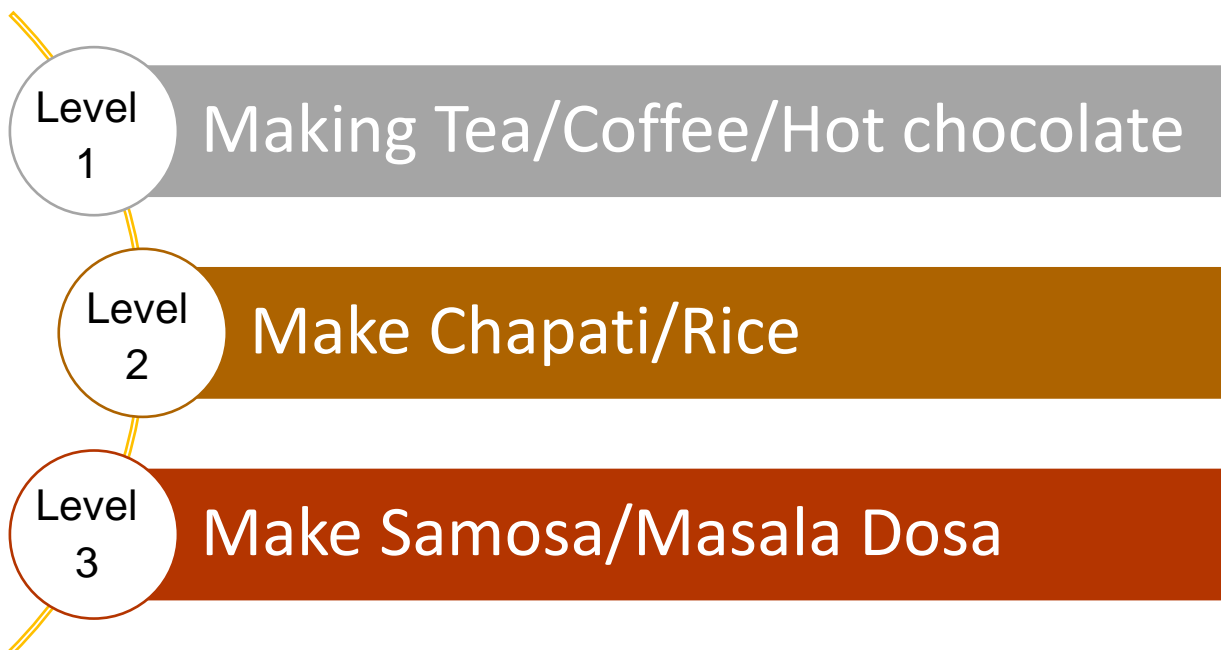
Before we jump into challenging ourselves, let us go through some ground rules for this activity:

GROUND RULES

This is an individual activity.
Do not discuss your answers
with your peers.

Think of various ways in
which a challenge could be
accomplished. Solutions can
vary from person to person.

Write algorithms for the challenges given.



Now, exchange your attempts with your friend and see what procedure have they followed for the same tasks.

Did you notice any difference?

| Yes! Write the differences below. | No! How are they so similar? |
|-----------------------------------|------------------------------|
| | |

Re-look at your own algorithms now. Do you wish to make any changes to it? Is there a need to add/subtract step(s) in it? Take your time and make the necessary changes if required.

Conclusion: Through this activity, we understood:

1. How to divide a task into several sub-tasks and the advantage of doing so.
2. There can be multiple approaches in terms of the number of sub-tasks or the methods used in sub-tasks to solve a problem and the one which is the simplest and the most efficient should be chosen above all.

What is a flowchart?

A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes and arrows in order to demonstrate a process or a program. While computers work with numbers at ease, humans need visual representations to understand the information well and communicate it effectively. Thus, flowcharts are used to break a process into smaller parts and elaborate it using visual representations.

Several standard graphics are applied in a flowchart:



Terminal Box
(Start / End)

Input / Output

Process /
Instruction

Decision

Connector /
Arrow

The graphics above represent different parts of a flowchart. The process in a flowchart can be expressed through boxes and arrows with different sizes and colors. In a flowchart, we can easily highlight a certain element and the relationships between each part. Let us take a look as to how can we use flowcharts to represent algorithms.

How to Use Flowcharts to Represent Algorithms

Now that we have the definitions of algorithm and flowchart, how do we use a flowchart to represent an algorithm? Let us take a look at given examples and see how they work.

Example 1: Print 1 to 20:

Algorithm:

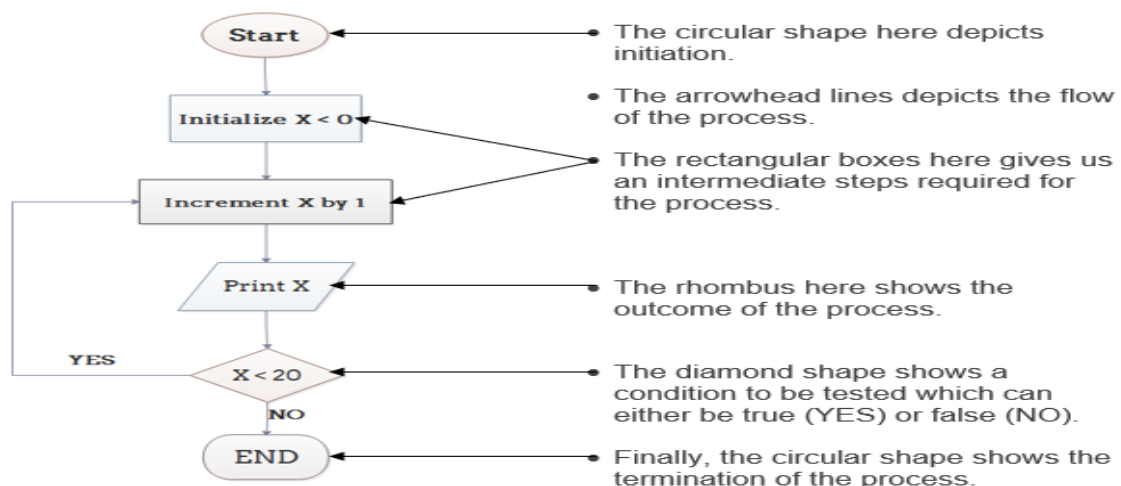
Step 1: Initialize X as 0,

Step 2: Increment X by 1,

Step 3: Print X,

Step 4: If X is less than 20 then go back to step 2.

Flowchart:



Example 2: Convert Temperature from Fahrenheit (°F) to Celsius (°C)

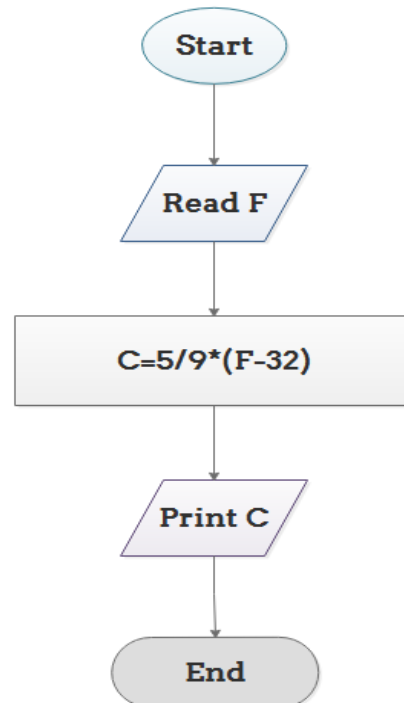
ALGORITHM

Step 1: Read temperature in Fahrenheit

Step 2: Calculate temperature with formula $C = 5/9 * (F - 32)$

Step 3: Print C

FLOWCHART



Challenge time!

Let us practice what we have learnt so far. Write algorithms and draw flowcharts for the tasks given below.

TASK 1: How to check whether the input number is prime or not?

TASK 2: How does a traffic signal work?

TASK 3: How to make an ATM transaction?

TASK 4: How to check if a light bulb is working or not?

TASK 5: How to win a Rock, Paper, Scissors Game – To know more about this game, go to <https://www.wikihow.com/Play-Rock,-Paper,-Scissors>

Test Your Knowledge

Q1) What is an algorithm?

Q2) What is a flowchart?

Q3) List down all the standard graphics for making a flowchart.

Chapter 2: Introduction to Python

In the previous section, you learnt about the different methodologies for programming. A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output.

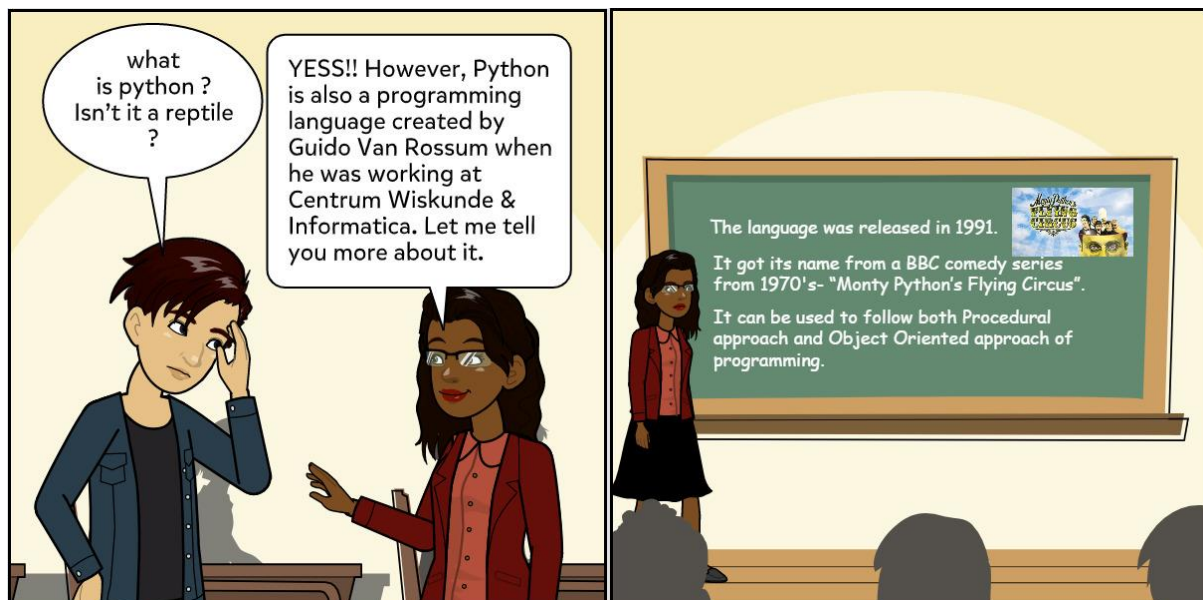
In simple Words, a programming language is a vocabulary and set of grammatical rules for instructing a computer to perform specific tasks. Though there are many different programming languages such as BASIC, Pascal, C, C++, Java, Haskell, Ruby, Python, etc. we will study Python in this course.

What is a program?

A computer program is a collection of instructions that perform a specific task when executed by a computer. It is usually written by a computer program in a programming language.

Before getting into understanding more about Python, we need to first understand what is Python and why we need to use Python?

What is Python?



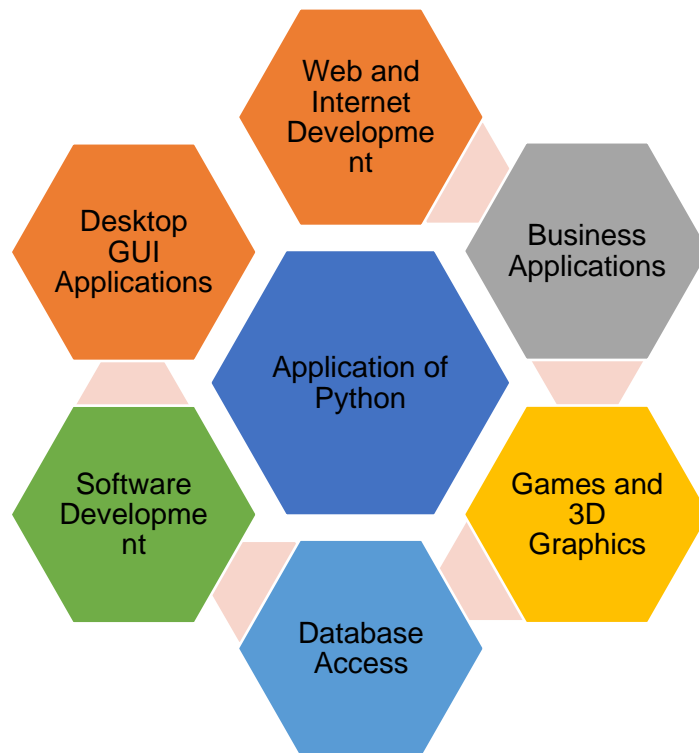
Why Python for AI?

Artificial intelligence is the trending technology of the future. You can see so many applications around you. If you as an individual can also develop an AI application, you will require to know a programming language. There are various programming languages like Lisp, Prolog, C++, Java and Python, which can be used for developing applications of AI. Out of these, Python gains a maximum popularity because of the following reasons:



Applications of Python

Python is used for a large number of applications. Some of them are mentioned below:



Getting started with Python

Python is a cross-platform programming language, meaning, it runs on multiple platforms like Windows, MacOS, Linux and has even been ported to the Java and .NET virtual machines.

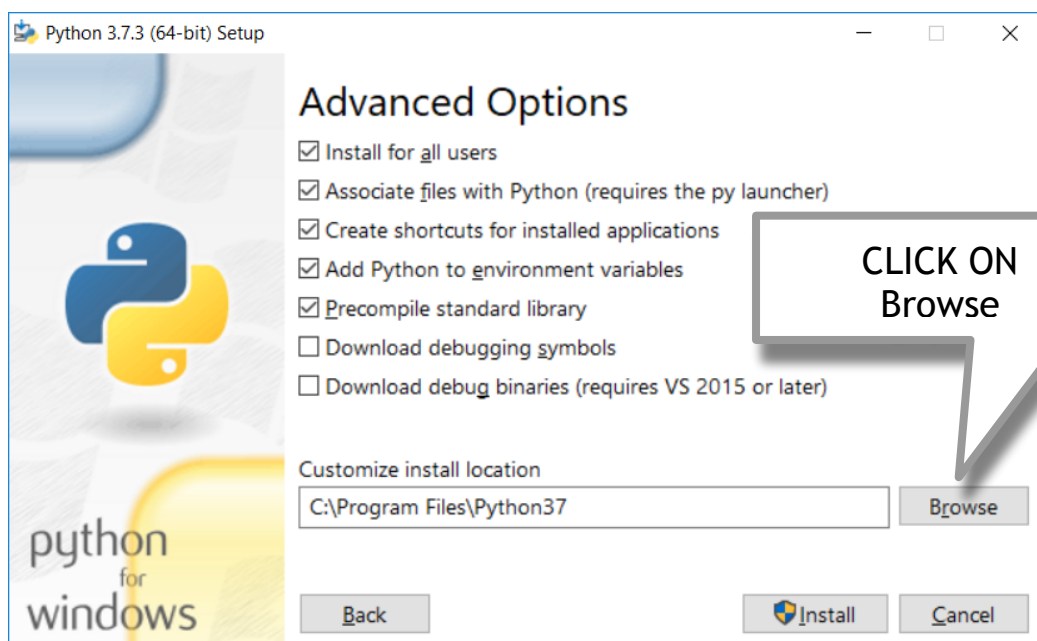
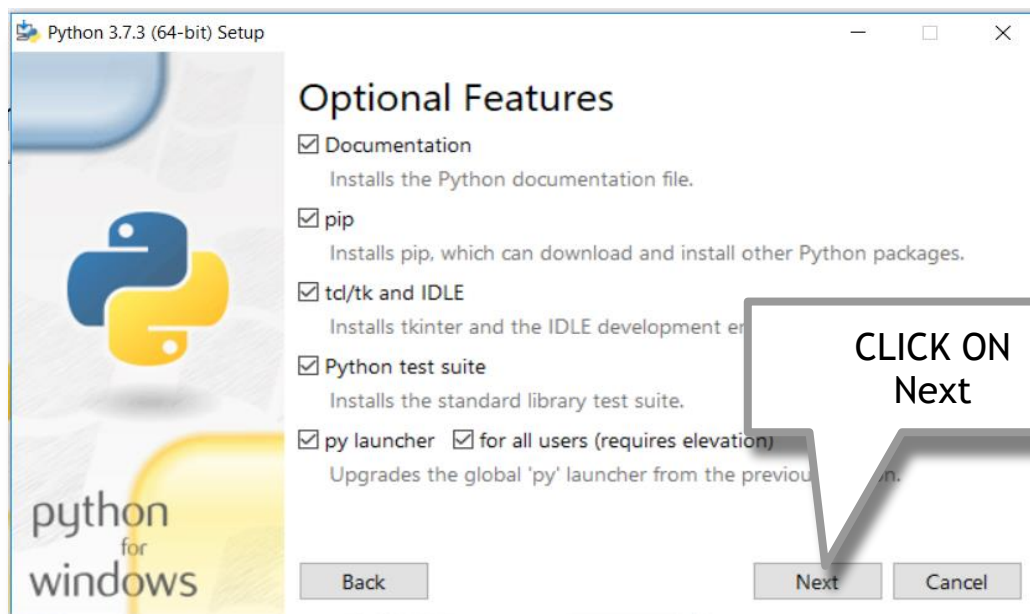
To write and run Python program, we need to have Python interpreter installed in our computer.

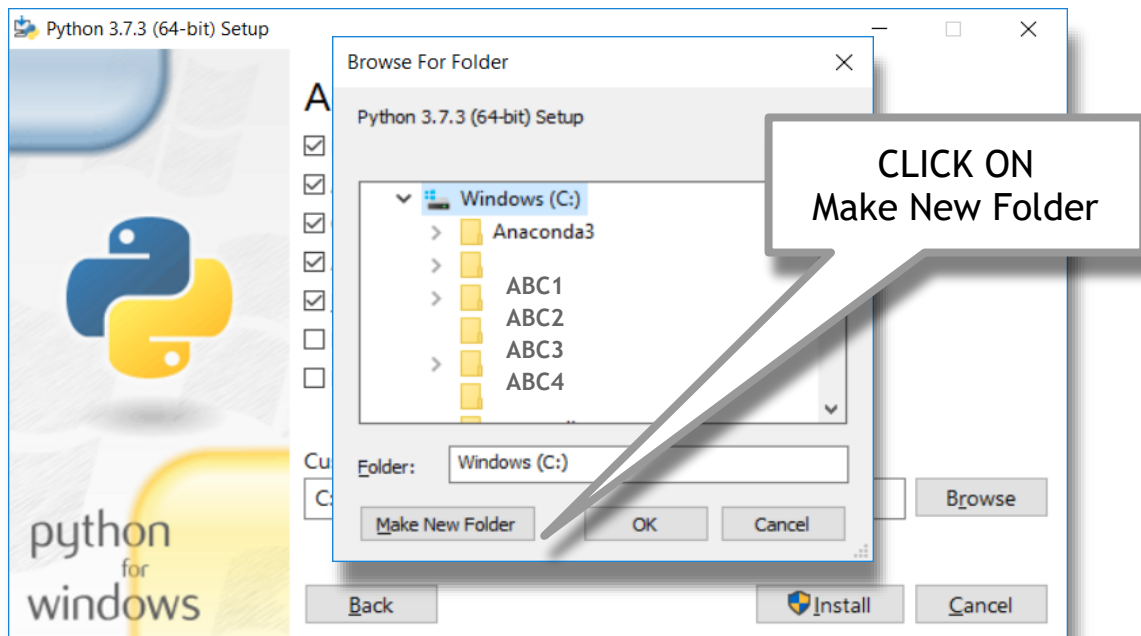
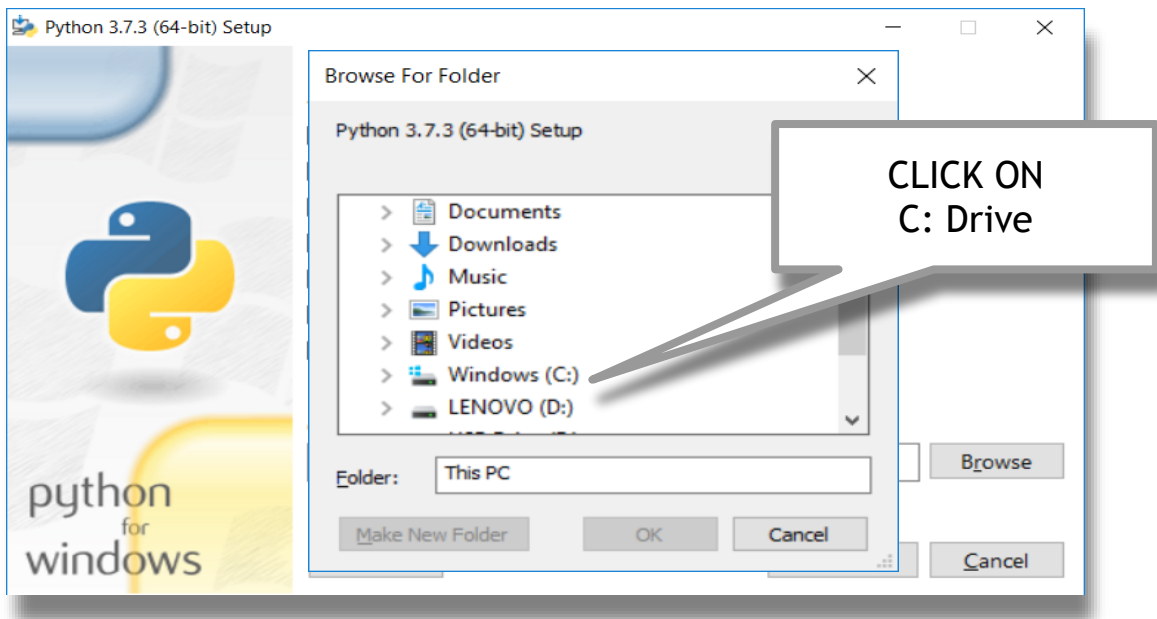
Downloading and Setting up Python for use

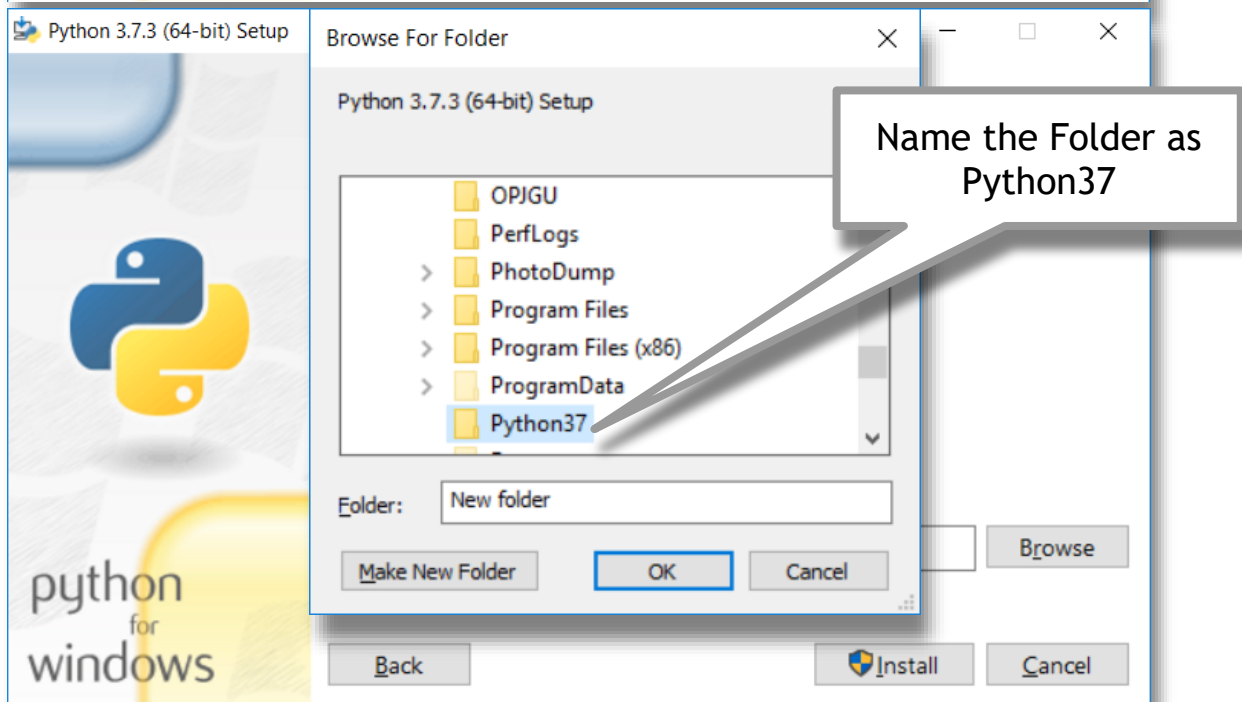
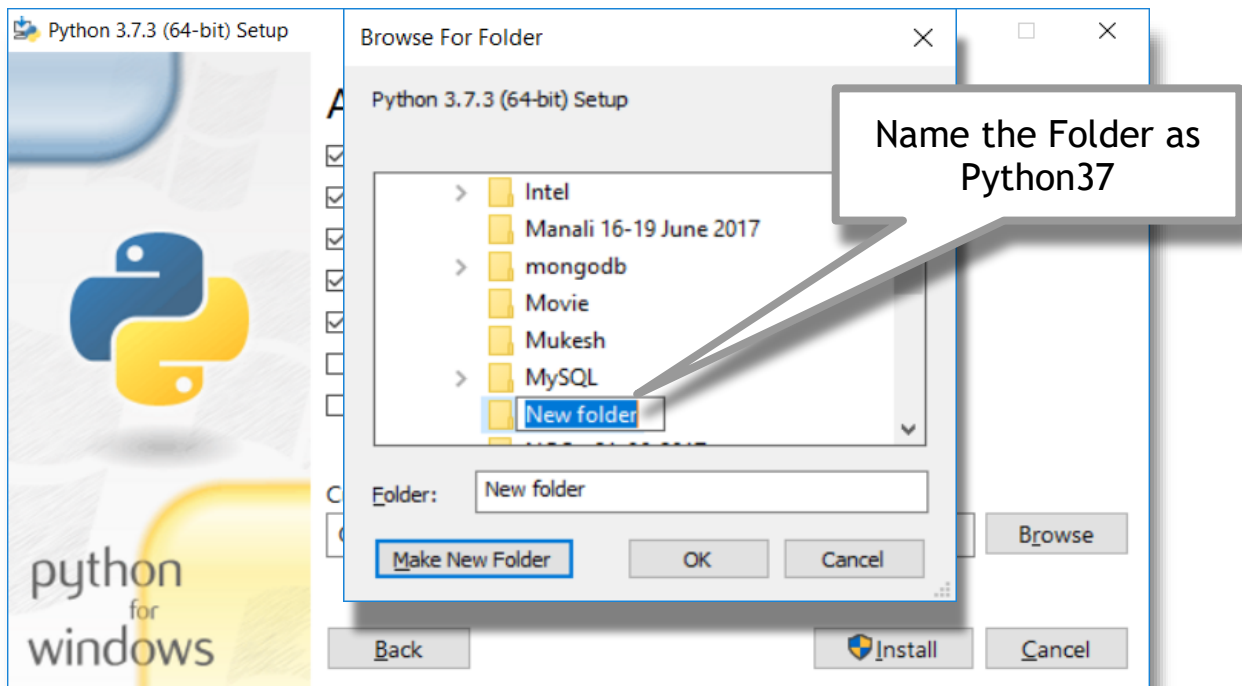
- Download Python from **python.org** using link **python.org/downloads**
- Select appropriate download link as per Operating System [Windows 32 Bit/64 Bit, Apple iOS]
- FOR EXAMPLE, for Windows 64 Bit OS
- Select the following link

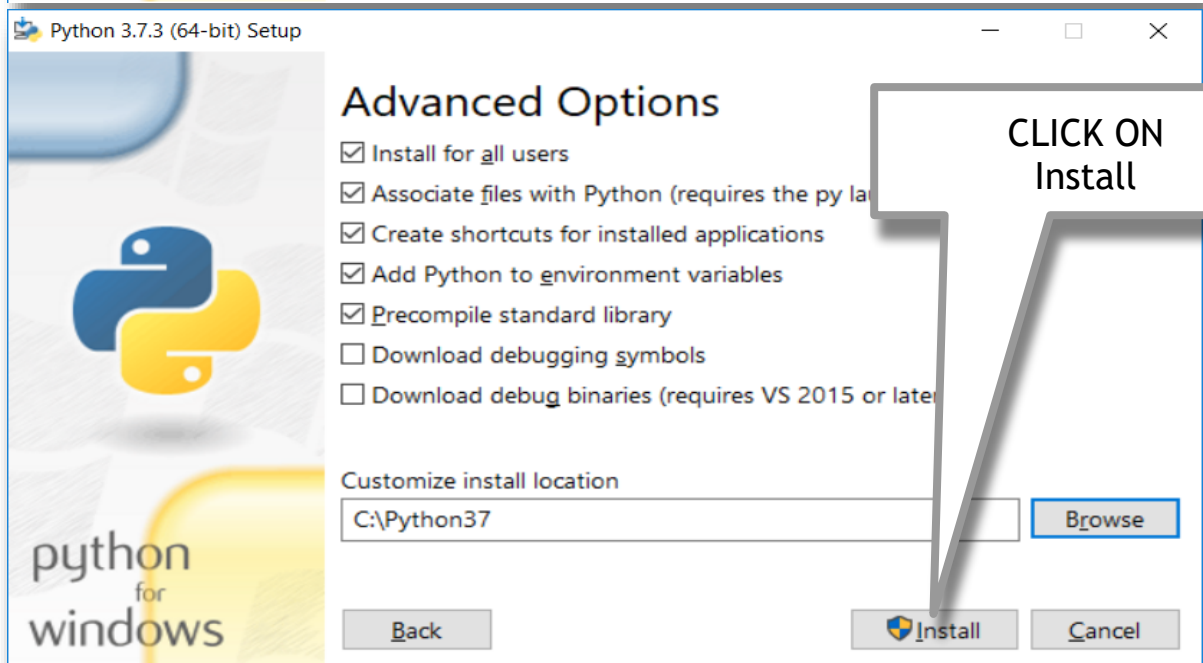
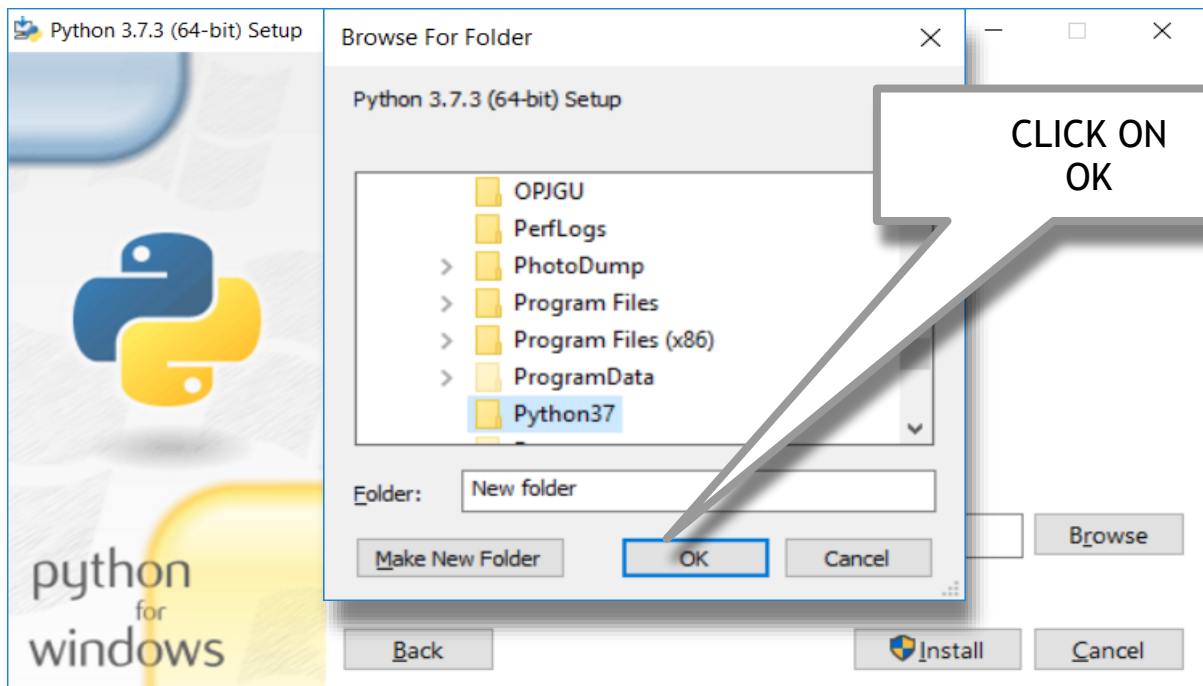
▪ [Download Windows x86-64 executable installer](https://www.python.org/downloads/windows/)

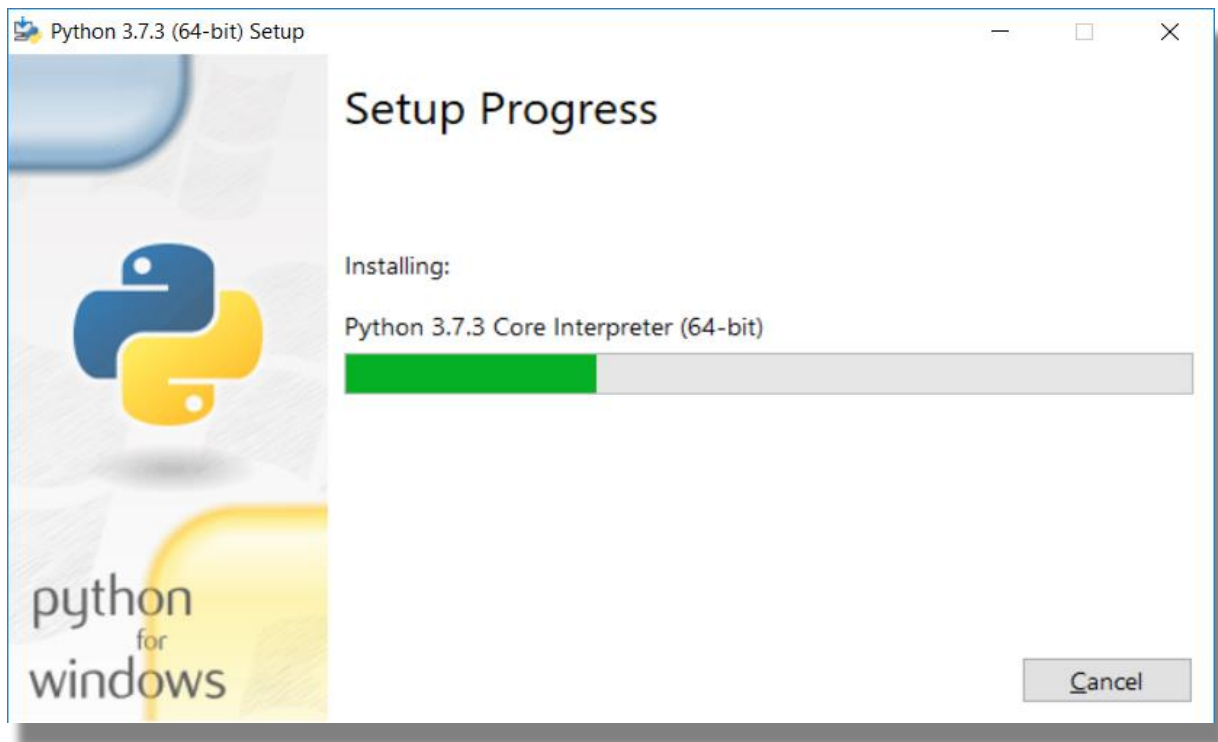
Python IDLE installation











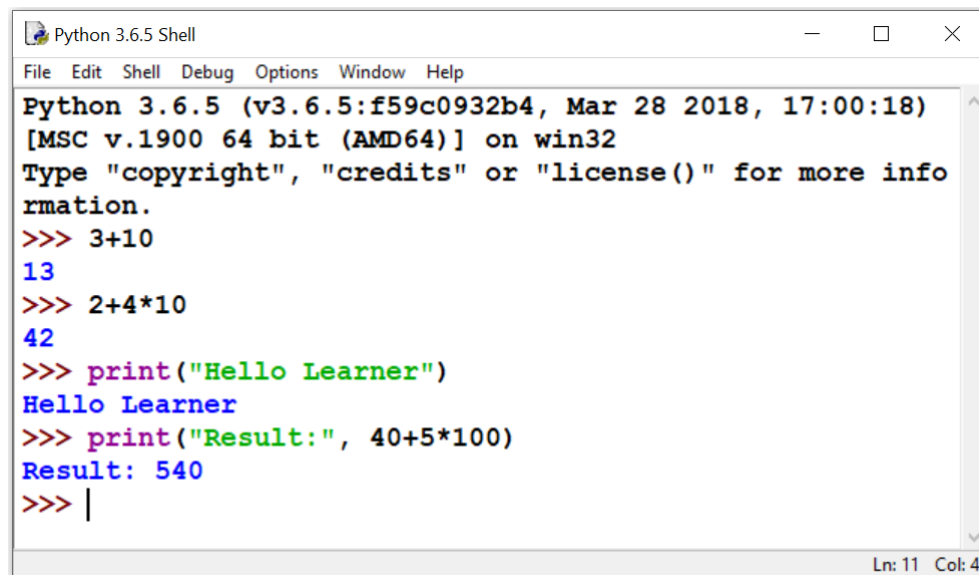
Run in the Integrated Development Environment (IDE)

When we install Python, an IDE named **IDLE** is also installed. We can use it to run Python on our computer.

IDLE (GUI integrated) is the standard, most popular Python development environment. IDLE is an acronym of Integrated Development Environment. It lets one edit, run, browse and debug Python Programs from a single interface. This environment makes it easy to write programs.

Python shell can be used in two ways, viz., interactive mode and script mode. Where Interactive Mode, as the name suggests, allows us to interact with OS; script mode lets us create and edit Python source file.

Interactive Mode

A screenshot of the Python 3.6.5 Shell window. The window has a title bar "Python 3.6.5 Shell" and a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following text:

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18)
[MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more info
rmation.
>>> 3+10
13
>>> 2+4*10
42
>>> print("Hello Learner")
Hello Learner
>>> print("Result:", 40+5*100)
Result: 540
>>> |
```

The status bar at the bottom right shows "Ln: 11 Col: 4".

You can see the above example, Python IDLE Shell account has `>>>` as Python prompt, where simple mathematical expressions and single line Python commands can be written and can be executed simply by pressing enter.

The first expression `3+10` written on the first Python prompt shows 13 as output in the next line.

The second expression `2+4*10` written on the second Python prompt shows 42 as output in the next line.

The third statement `print("Hello Learner")` written on the third Python prompt shows Hello Learner as output in the next line.

The third statement `print("Result:", 40+5*100)` written on the fourth Python prompt shows Result: as output in the next line.

Try Yourself

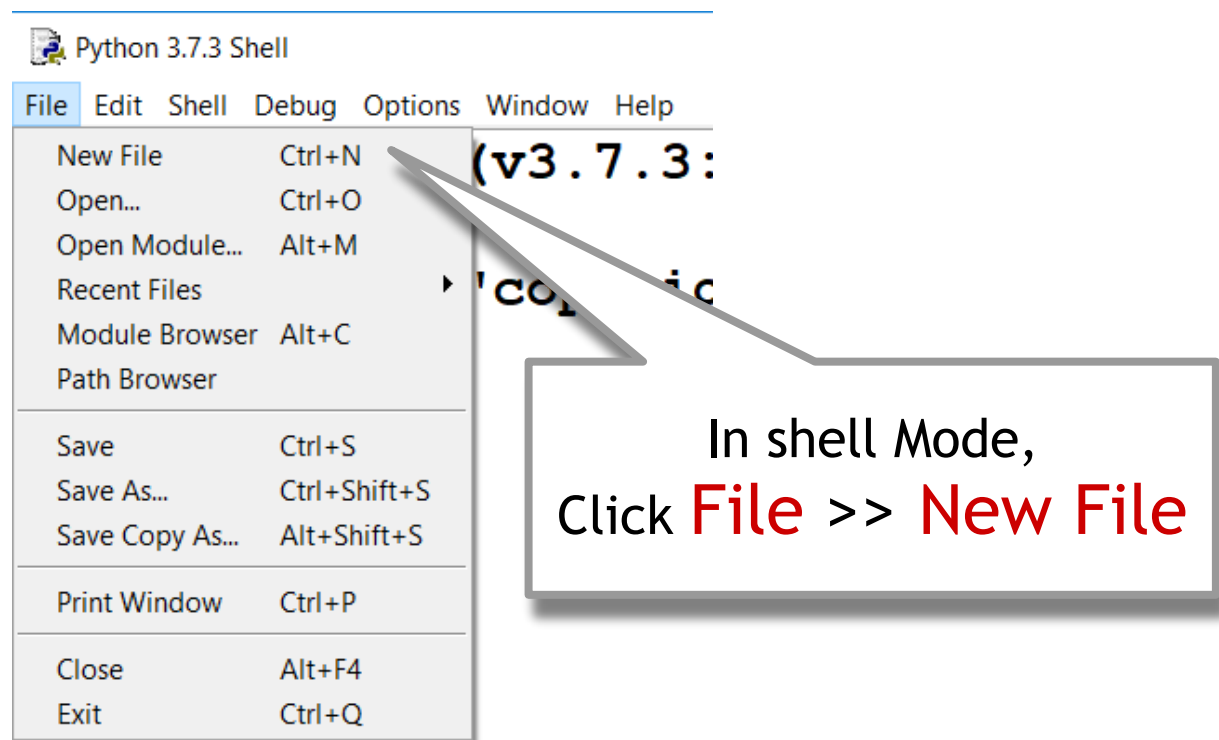
1. Find the result of $(75+85+65)/3$ i.e. the average of three marks
2. Find the result of $22/7 * 5 * 5$ i.e. the area of circle having radius as 5
3. Find the result of "RAVI"+"Kant"
4. Find the result of "####" * 3

Script Mode

In script mode, we type Python program in a file and then use the interpreter to execute the content from the file. Working in interactive mode is convenient for beginners and for testing small pieces of code, as we can test them immediately. But for coding more than few lines, we should always save our code so that we may modify and reuse the code.

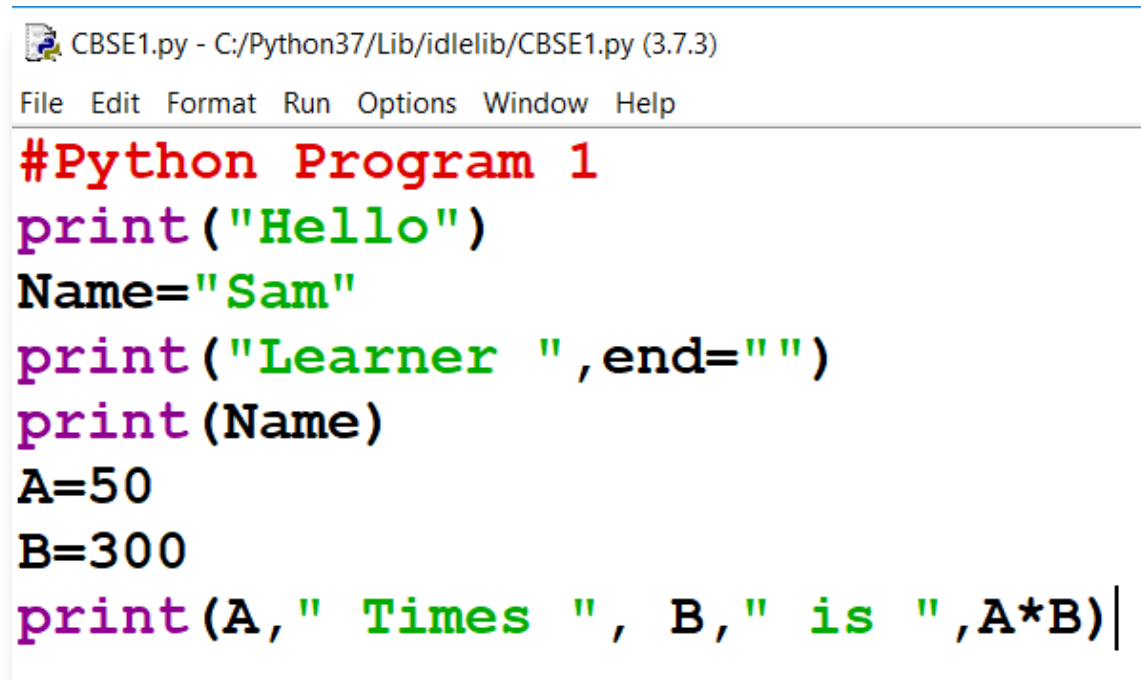
Note: Result produced by Interpreter in both the modes, viz., Interactive and script mode is exactly the same.

Python Script/Program: Python statements written in a particular sequence to solve a problem is known as Python Script/Program.



To write a Python script/program, we need to open a new file - **File >> New File**, type a sequence of Python statements for solving a problem, save it with a meaningful name - **File >> Save**, and finally **Run** the program to view the output of the program.

Now, type your first Python Program

A screenshot of a Python IDE window titled 'CBSE1.py - C:/Python37/Lib/idlelib/CBSE1.py (3.7.3)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

```
#Python Program 1
print("Hello")
Name="Sam"
print("Learner ",end="")
print(Name)
A=50
B=300
print(A," Times ", B," is ",A*B)|
```

Code Explanation:

Line 1 in the above code starting with # is a comment line, which means the line is non-executable and it is only for the programmer's reference.

Line 2 will simply display

Hello

Line 3 will assign a string value "Sam" to a variable **Name**

Line 4 will display **Learner** and will allow the next output to get displayed in the same line

Line 5 will display **Sam** in the same line as **Learner**

Line 6 will assign an integer **50** to a variable **A**

Line 7 will assign an integer **300** to a variable **B**

Line 8 will display **50** times **300** is **15000**

So, the complete output of the Python Code will be

```

Hello
Learner Sam
50 times 300 is 15000
```

Python Statement and Comments

In this section we will learn about Python statements, why indentation is important and how to use comments in programming.

Python Statement

Instructions written in the source code for execution are called statements. There are different types of statements in the Python programming language like Assignment statement, Conditional statement, Looping statements etc. These help the user to get the required output. For example, `n = 50` is an assignment statement.

Multi-line statement

In Python, end of a statement is marked by a newline character.

However, Statements in Python can be extended to one or more lines using parentheses (), braces {}, square brackets [], semi-colon (;), continuation character slash (\). When we need to do long calculations and cannot fit these statements into one line, we can make use of these characters.

Examples:

| Type of Multi-line Statement | Usage |
|----------------------------------|---|
| Using Continuation Character (\) | <pre>s = 1 + 2 + 3 + \ 4 + 5 + 6 + \ 7 + 8 + 9</pre> |
| Using Parentheses () | <pre>n = (1 * 2 * 3 + 4 - 5)</pre> |
| Using Square Brackets [] | <pre>footballer = ['MESSI', 'NEYMAR', 'SUAREZ']</pre> |
| Using braces {} | <pre>x = {1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9}</pre> |
| Using Semicolons (;) | <pre>flag = 2; ropes = 3; pole = 4</pre> |

Python Comments

A **comment** is text that doesn't affect the outcome of a code, it is just a piece of text to let someone know what you have done in a program or what is being done in a block of code.

In Python, we use the hash (#) symbol to start writing a comment.

Single line Comments

#Defining a variable to store number.

```
a = 10
```

```
fb = "messi" #Store Messi as value in fb
```

Multi Line Comments

```
"""
```

```
This is an example code.
```

```
This is to learn Python for AI
```

```
"""
```

```
'''
```

```
This is an example code.
```

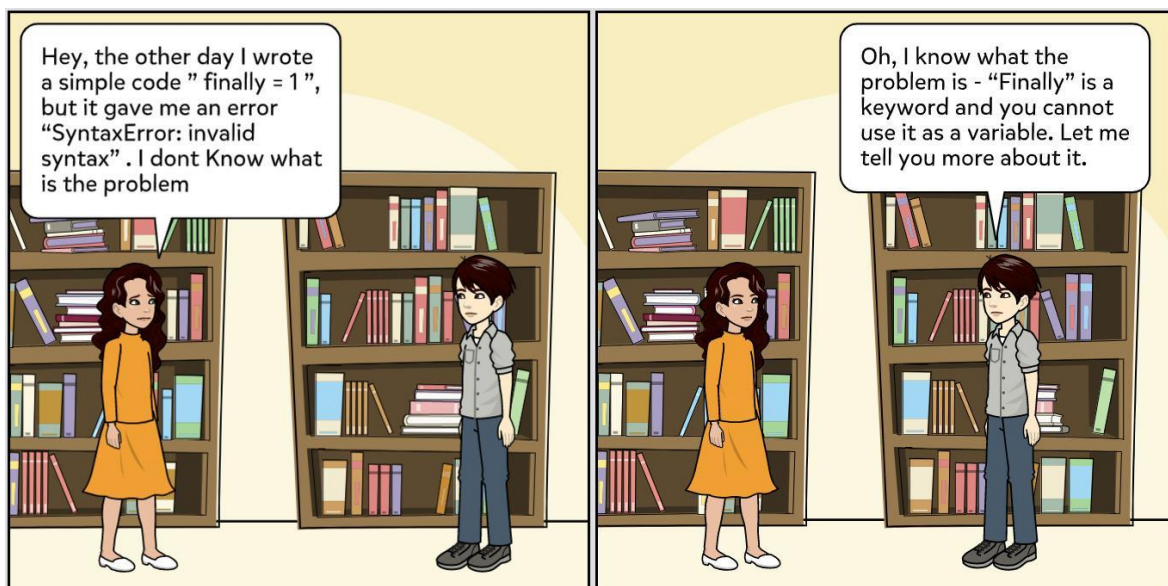
```
This is to learn Python for AI
```

```
'''
```

Python Keywords and Identifiers

In this section, we will learn about keywords (reserved words in Python) and identifiers (names given to variables, functions, etc.).

Keywords



Keywords are the reserved words in Python used by Python interpreter to recognize the structure of the program.

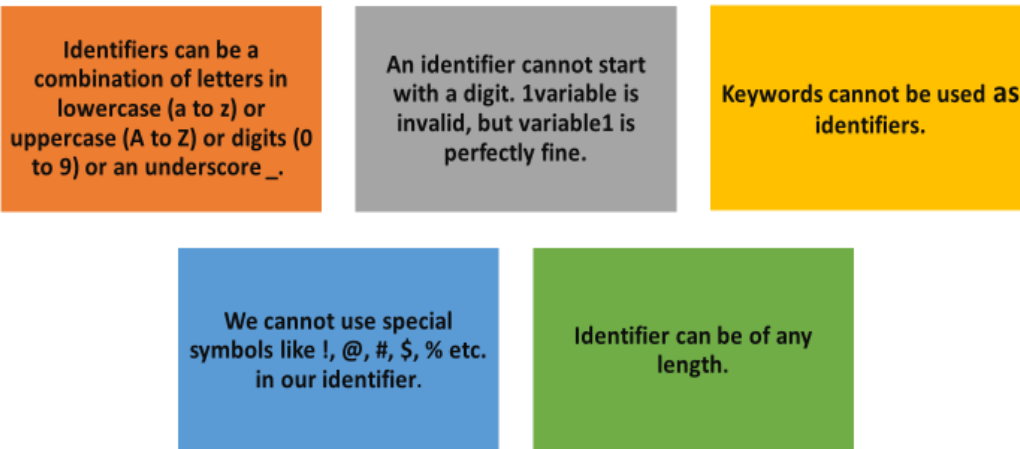
The list of all the keywords is given below.

Keywords in Python

| | | | | |
|--------|----------|---------|----------|--------|
| False | class | finally | is | return |
| None | continue | for | lambda | try |
| True | def | from | nonlocal | while |
| and | del | global | not | with |
| as | elif | if | or | yield |
| assert | else | import | pass | |
| break | except | in | raise | |



An identifier is a name given to entities like class, functions, variables, etc.
It helps to differentiate one entity from another.



Note:

Python is a case-sensitive language. This means, `Variable` and `variable` are not the same. Always name identifiers that make sense.

While, `c = 10` is valid. Writing `count = 10` would make more sense and it would be easier to figure out what it does even when you look at your code after a long gap.

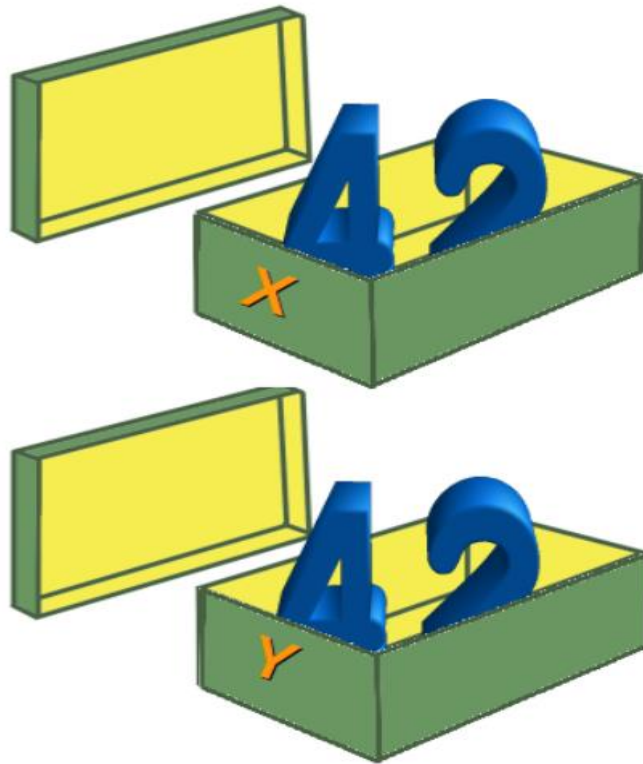
Multiple words can be separated using an underscore, for example `this_is_a_long_variable`

Variables and Datatypes

Variables

A variable is a named location used to store data in the memory. It is helpful to think of variables as a container that holds data which can be changed later throughout programming. For example,

```
x = 42
y = 42
```



These declarations make sure that the program reserves memory for two variables with the names `x` and `y`. The variable names stand for the memory location. It's like the two shoeboxes, which you can see in the picture. These shoeboxes are labelled with `x` and `y` and the corresponding values are stored in the shoeboxes. Like the two shoeboxes, the memory is empty as well at the beginning.

Note:

Assignment operator is used in Python to assign values to variables. For example, `a = 5` is a simple assignment operator that assigns the value 5 on the right to the variable `a` on the left.

Examples on Variables:

| Task | Sample Code | Output |
|---|--|----------------------------|
| Assigning a value to a variable | <pre>Website = "xyz.com" print(Website)</pre> | <pre>xyz.com</pre> |
| Changing value of a variable | <pre>Website = "xyz.com" print(Website) Website = "abc.com" print(Website)</pre> | <pre>xyz.com abc.com</pre> |
| Assigning different values to different variables | <pre>a,b,c=5, 3.2, "Hello" print(a) print(b)</pre> | <pre>5 3.2 Hello</pre> |

| | | |
|--|---|----------------------|
| | <code>print(c)</code> | |
| Assigning same value to different variable | <code>x=y=z= "Same"</code> <code>print(x)</code> <code>print(y)</code> <code>print(z)</code> | Same Same Same |

Constants:

A constant is a type of variable whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later.

Non technically, you can think of constant as a shoe box with a fixed size of shoe kept inside which cannot be changed after that.

Assigning Value to a constant in Python

In Python, constants are usually declared and assigned on a module. Here, the module means a new file containing variables, functions etc. which is imported to the main file. Inside the module, constants are written in all capital letters and underscores separating the words.

Example : Declaring and assigning value to a constant

- Create a info.py

```
NAME = "Ajay"
AGE = 24
```

- Create a main.py

```
import info

print(info.NAME)
print(info.AGE)
```

- When you run the program the output will be,

```
Ajay
24
```

In the above program, we create a constant.py module file. Then, we assign the constant value to PI and GRAVITY. After that, we create a main.py file and import the constant module. Finally, we print the constant value.

Note: In reality, we don't use constants in Python. The global or constants module is used throughout the Python programs.

Rules and Naming convention for variables and constants

Create a name that makes sense. Suppose, vowel makes more sense than v.

Use camelCase notation to declare a variable. It starts with lowercase letter. For example: myName

Use capital letters where possible to declare a constant. For example: PI

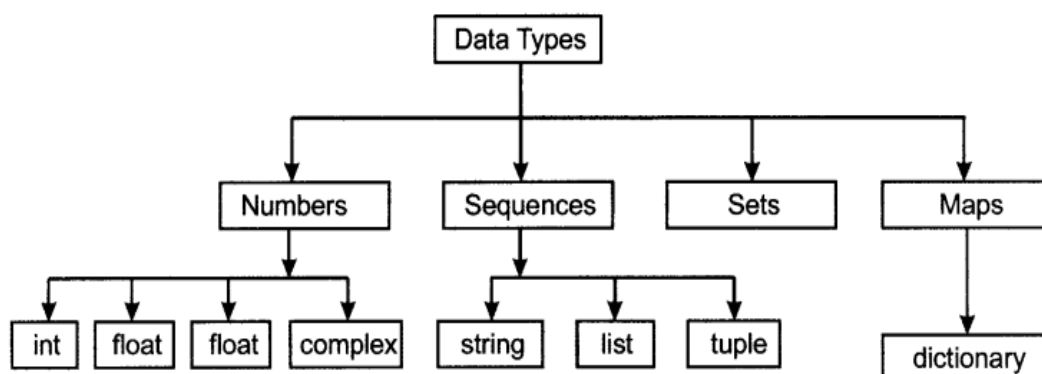
Never use special symbols like !, @, #, \$, %, etc.

Constant and variable names should have combination of letters in lowercase or uppercase or digits or an underscore (_).

Datatypes

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

There are various data types in Python. Some of the important types are mentioned below in the image



Python Data types

1) Python Numbers

Number data type stores Numerical Values. These are of three different types:

- a) Integer & Long
- b) Float / floating point

Integer & Long Integer

Range of an integer in Python can be from -2147483648 to 2147483647, and long integer has unlimited range subject to available memory.

Integers are the whole numbers consisting of + or – sign with decimal digits like 100000, -99, 0, 17. While writing a large integer value, don't use commas to separate digits. Also, integers should not have leading zeros.

Floating Point:

Numbers with fractions or decimal point are called floating point numbers.

A floating-point number will consist of sign (+,-) sequence of decimal digits and a dot such as 0.0, -21.9, 0.98333328, 15.2963. These numbers can also be used to represent a number in engineering/ scientific notation.

-2.0 x 10⁵ will be represented as -2.0e5
2.0X10⁻⁵ will be 2.0E-5

2) None

This is special data type with single value. It is used to signify the absence of value/false in a situation. It is represented by **None**.

3) Sequence

A sequence is an ordered collection of items, indexed by positive integers. It is a combination of mutable and non-mutable data types. Three types of sequence data type available in Python are:

- a) Strings
- b) Lists
- c) Tuples

String

String is an ordered sequence of letters/characters. They are enclosed in single quotes (' ') or double (" "). The quotes are not part of string. They only tell the computer where the string constant begins and ends. They can have any character or sign, including space in them.

Lists

List is also a sequence of values of any type. Values in the list are called elements / items. These are indexed/ordered. List is enclosed in square brackets.

Example:

```
dob = [19,"January",1990]
```

Tuples:

Tuples are a sequence of values of any type, and are indexed by integers. They are immutable. Tuples are enclosed in ().

Example:

```
t = (5,'program',2.5)
```

4) Sets

Set is an unordered collection of values, of any type, with no duplicate entry.

Example:

```
>>> a = {1,2,2,3,3,3}
>>> a
{1,2,3}
```

5) Mapping

This data type is unordered. Dictionaries fall under Mappings.

Dictionaries

[Dictionary](#) is an unordered collection of key-value pairs. It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.

In Python, dictionaries are defined within braces {} with each item being a pair in the form `key: value`. Key and value can be of any type.

Example

```
>>> d = {1:'Ajay','key':2}
>>> type(d)
<class 'dict'>
```

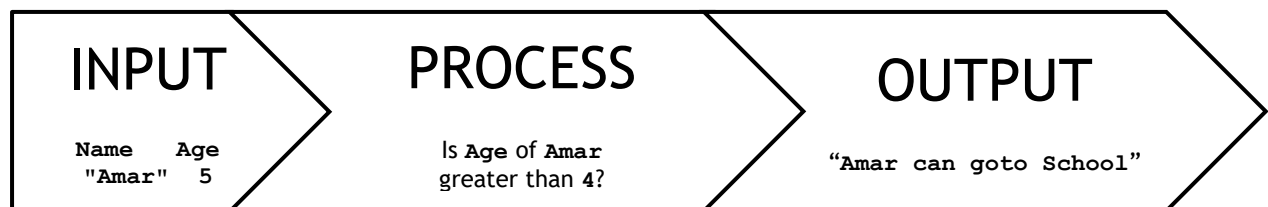
Python Operators I

Operators are special symbols which represent computation. They are applied on operand(s), which can be values or variables. Some operators can behave differently on different data types. Operators when applied on operands form an expression. Operators are categorized as Arithmetic, Relational, Logical and Assignment. Value and variables when used with operator are known as operands.

Arithmetic Operators

| Operator | Meaning | Expression | Result |
|----------|------------------|------------|--------|
| + | Addition | 10 + 20 | 30 |
| - | Subtraction | 30 - 10 | 20 |
| * | Multiplication | 30 * 100 | 300 |
| / | Division | 30 / 10 | 20.0 |
| | | 1 / 2 | 0.5 |
| // | Integer Division | 25 // 10 | 2 |
| | | 1 // 2 | 0 |
| % | Remainder | 25 % 10 | 5 |
| ** | Raised to power | 3 ** 2 | 9 |

Python Input and Output



Python Output Using print() function

We use the `print()` function to output data to the standard output device (screen). We can also output data to a file. An example is given below.

```
a = "Hello World!"  
print(a)
```

The output of the above code will be:

```
Hello World!
```

| Example Code | Sample Output |
|--------------|---------------|
|--------------|---------------|

| | |
|--|--------------------|
| <code>a = 20 b = 10 print(a + b)</code> | 30 |
| <code>print(15 + 35)</code> | 50 |
| <code>print("My name is Kabir")</code> | My name is Kabir |
| <code>a = "tarun" print("My name is :",a)</code> | My name is : tarun |
| <code>x = 1.3 print("x = /n", x)</code> | x = 1.3 |
| <code>m = 6 print(" I have %d apples",m)</code> | I have 6 apples |

User input

In all the examples till now, we have been using the calculations on known values (constants). Now let us learn to take user's input in the program. In python, `input()` function is used for the same purpose.

| Syntax | Meaning |
|--|----------------------------|
| <code><String Variable>=input(<String>)</code> | For string input |
| <code><integer Variable>=int(input(<String>))</code> | For integer input |
| <code><float Variable>=float(input(<String>))</code> | For float (Real no.) input |

Type Conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion. Python has two types of type conversion.

1. Implicit Type Conversion
2. Explicit Type Conversion

Implicit Type Conversion

In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

Example:

```
# Code to calculate the Simple Interest

principle_amount = 2000
roi = 4.5
time = 10

simple_interest = (principle_amount * roi * time)/100

print("datatype of principle amount : ", type(principle_amount))
print("datatype of rate of interest : ", type(roi))

print("value of simple interest : ", simple_interest)
print("datatype of simple interest : ", type(simple_interest))
```

When we run the above program, the output will be

```
datatype of principle amount : <class 'int'>
datatype of rate of interest : <class 'float'>

value of simple interest : 900
datatype of simple interest : <class 'float'>
```

In the above program,

- We calculate the simple interest by using the variable `principle_amount` and `roi` with time divide by 100
- We will look at the data type of all the objects respectively.
- In the output we can see the datatype of `principle_amount` is an integer, datatype of `roi` is a float.
- Also, we can see the `simple_interest` has float data type because Python always converts smaller data type to larger data type to avoid the loss of data.

| Example Code | Sample Output | Explanation |
|--|--|---|
| <pre>a = 10 b = "Hello" print(a+b)</pre> | <pre>File "cbse2.py", line 3, in <module> print(a+b) TypeError: unsupported operand type(s) for +: 'int' and 'str'</pre> | The output shows an error which says that we cannot add integer and string variable types using implicit conversion |
| <pre>c = 'Ram' N = 3 print(c*N)</pre> | RamRamRam | The output shows that the string is printed 3 times when we use a multiply operator with a string. |
| <pre>x = True y = 10 print(x + 10)</pre> | 11 | The output shows that the boolean value x will be converted to integer and as it is true will be considered as 1 and then give the output. |
| <pre>m = False n = 23 print(n - m)</pre> | 23 | The output shows that the boolean value m will be converted to integer and as it is false will be considered as 0 and then give the output. |

Try It Yourself:

- 1) Take a Boolean value and add a string to it
- 2) Take a string and float number and try adding both
- 3) Take a Boolean and a float number and try adding both.

Explicit Type Conversion

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like `int()`, `float()`, `str()`, etc to perform explicit type conversion.

This type of conversion is also called typecasting because the user casts (changes) the data type of the objects.

Syntax:

```
(required_datatype)(expression)
```

Typecasting can be done by assigning the required data type function to the expression.
Example: Adding of string and an integer using explicit conversion

```
Birth_day = 10
Birth_month = "July"

print("data type of Birth_day before type casting :", type(Birth_day))
print("data type of Birth_month : ", type(Birth_month))

Birth_day = str(Birth_day)
print("data type of Birth_day after type casting :", type(Birth_day))

Birth_date = Birth_day + Birth_month

print("birth date of the student : ", Birth_date)
print("data type of Birth_date : ", type(Birth_date))
```

When we run the above program, the output will be

```
data type of Birth_day before type casting : <class 'int'>
data type of Birth_month : <class 'str'>

data type of Birth_day after type casting : <class 'str'>

birth date of the student : ' 10 July '
data type of Birth_date : <class 'str'>
```

In above program,

- We add Birth_day and Birth_month variable.
- We converted Birth_day from integer(lower) to string(higher) type using str() function to perform the addition.
- We got the Birth_date value and data type to be string.

| Example Code | Sample Output | Explanation |
|---|---------------|--|
| <pre>a = 20 b = "Apples" print(str(a)+ b)</pre> | 20 Apples | Writing str(a) will convert integer a into a string and then will add to the string b. |
| <pre>x = 20.3 y = 10 print(int(x) + y)</pre> | 30 | Writing int(x) will convert a float number to integer by just considering the integer part of the number and then perform the operation. |
| <pre>m = False n = 5 print(Bool(n)+ m)</pre> | 1 | Writing Bool() will convert the integer value to Boolean. If it is zero then it is converted to False, else to True for all other cases. |

Try it yourself:

- 1) Take a Boolean value "False" and a float number "15.6" and perform the AND operation on both
- 2) Take a string "Zero" and a Boolean value "True" and try adding both by using the Bool() function.
- 3) Take a string "Morning " and the float value "90.4" and try and add both of them by using the float() function.

Perform the above mentioned Try it Yourself in the lab and write down the observations to be discussed later in the class.

Note:

There is no **difference** in **single** or **double quoted** string. Both representations can be used interchangeably. However, if either **single** or **double quote** is a part of the string itself, then the string must be placed in **double** or **single quotes** respectively.

Python Operators II

Comparison operators

| Operator | Meaning | Expression | Result |
|----------|--------------------------|------------|--------|
| > | Greater Than | 20 > 10 | True |
| | | 15 > 25 | False |
| < | Less Than | 20 < 45 | True |
| | | 20 < 10 | False |
| == | Equal To | 5 == 5 | True |
| | | 5 == 6 | False |
| != | Not Equal to | 67 != 45 | True |
| | | 35 != 35 | False |
| >= | Greater than or Equal to | 45 >= 45 | True |
| | | 23 >= 34 | False |
| <= | Less than or equal to | 13 <= 24 | True |
| | | 13 <= 12 | False |

Comparison operators are used to compare values. It either returns True or False according to the condition.

Logical operators

| Operator | Meaning | Expression | Result |
|----------|--------------|----------------|--------|
| And | And operator | True and True | True |
| | | True and False | False |
| Or | Or operator | True or False | True |
| | | False or False | False |
| Not | Not Operator | not False | True |
| | | not True | False |

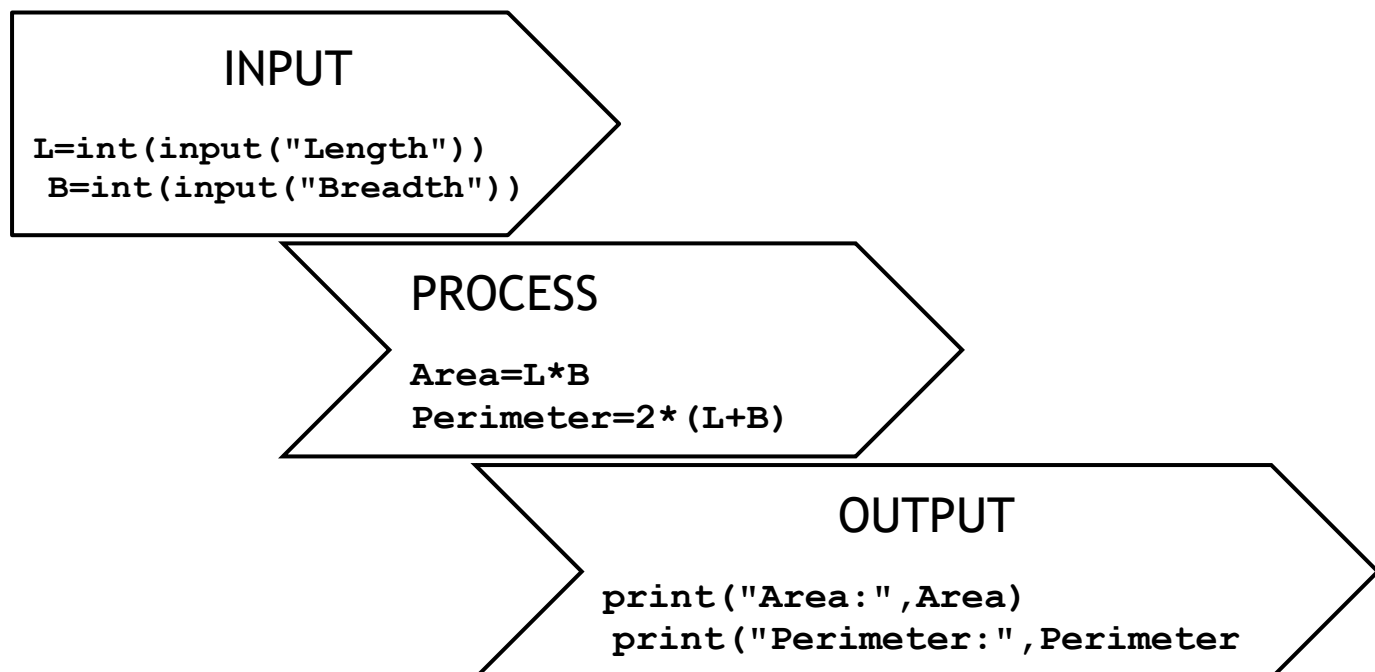
Logical operators are the and, or, not operators.

Assignment operators

Assignment operators are used in Python to assign values to variables.

| Operator | Expression | Equivalent to |
|----------|------------|---------------|
| = | x=5 | x = 5 |
| += | x +=5 | x = x + 5 |
| -= | x -= 5 | x = x - 5 |
| *= | x *= 5 | x = x * 5 |
| /= | x /= 5 | x = x / 5 |

Let's Practice



| Python Code | Sample Output |
|---|--|
| <pre># To calculate Area and # Perimeter of a rectangle L=int(input("Length")) B=int(input("Breadth")) Area=L*B Perimeter=2*(L+B) print("Area:",Area) print("Perimeter:",Perimeter)</pre> | <pre>Length:50 Breadth:20 Area:100 Perimeter:140</pre> |

| Try writing your code | Sample Output |
|--|--|
| <pre># To calculate Area of a triangle # with Base and Height _____ #Input Base _____ #Input Height _____ #Calculate Area _____ #Display Area</pre> | <pre>Base:20 Height:10 Area:100</pre> |
| <pre># To calculating average marks # of 3 subjects _____ #Input English Marks _____ #Input Maths Marks _____ #Input Science Marks _____ #Calculate Total Marks _____ #Calculate Average Marks _____ #Display Total Marks _____ #Display Average Marks</pre> | <pre>English:80 Maths :75 Science:85 Total Marks : 240 Average Marks: 80.0</pre> |
| <pre># To calculate discounted amount # with discount % _____ #Input Amount _____ #Input Discount% _____ #Calculate Discount _____ #Calculate Discounted Amount _____ #Display Discount _____ #Display Discounted Amount</pre> | <pre>Amount: 5000 Discount%:10 Discount:500 Discounted Amt:4500</pre> |
| <pre># To calculate Surface Area and Volume # of a Cuboid _____ #Input Length _____ #Input Breadth _____ #Input Height _____ #Calculate Surface Area _____ #Calculate Volume _____ #Display Surface Area _____ #Display Volume</pre> | <pre>Length:20 Breadth:10 Height:15 Surface Area:1300 Volume:3000</pre> |

Test Your Knowledge

- Q1) What are the key features of python ? Mention various applications along with it
- Q2) What are the different modes for coding in python?
- Q3) What are comments in python ? List down the various types of comments.
- Q4) What are the different properties of an identifier ?
- Q5) What are the rules for naming of variables and constants
- Q6) Explain python input and output with the help of an example.
- Q7) What is type conversion ? Explain the types of type conversion with the help of an example.
- Q8) What is a variable? Give example.

Chapter 3 - Introduction to tools for AI

Recap

Till now we have learnt about flowcharts & algorithms, Python programming language, and how to run programs in IDLE, what are packages and modules, and how it helps us in better and faster coding. Now, we will learn and explore new tools which will help us with better understanding, faster debugging of codes, and a lot more.

Introduction to Anaconda

We have explored about 3 major domains of AI: Data, NLP and CV. It often happens that while writing a code, these domains have different *packages* which need to be installed. Though we can install them all in IDLE, it is difficult to manage them all.

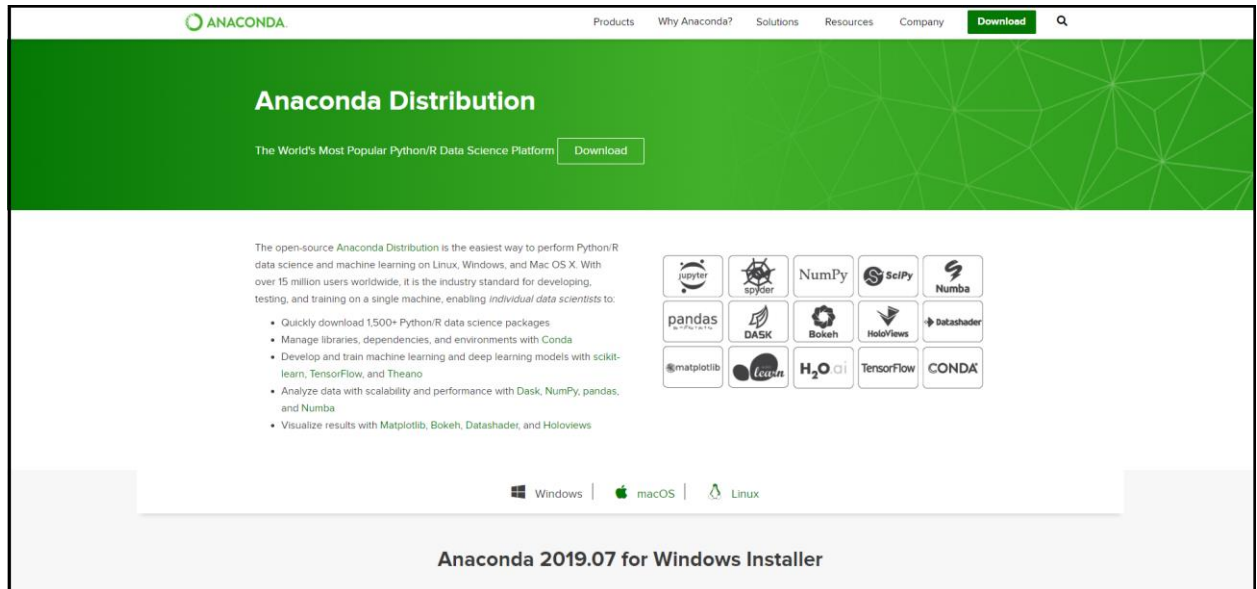
Anaconda is a free and open-source distribution of the Python language for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. It provides the facility to create different *virtual environments*, each having its own packages and settings, as per user's need.

Anaconda Navigator is a desktop graphical user interface(GUI) included in Anaconda that allows you to launch applications and easily manage conda packages, environments and channels without the need to use command line commands.

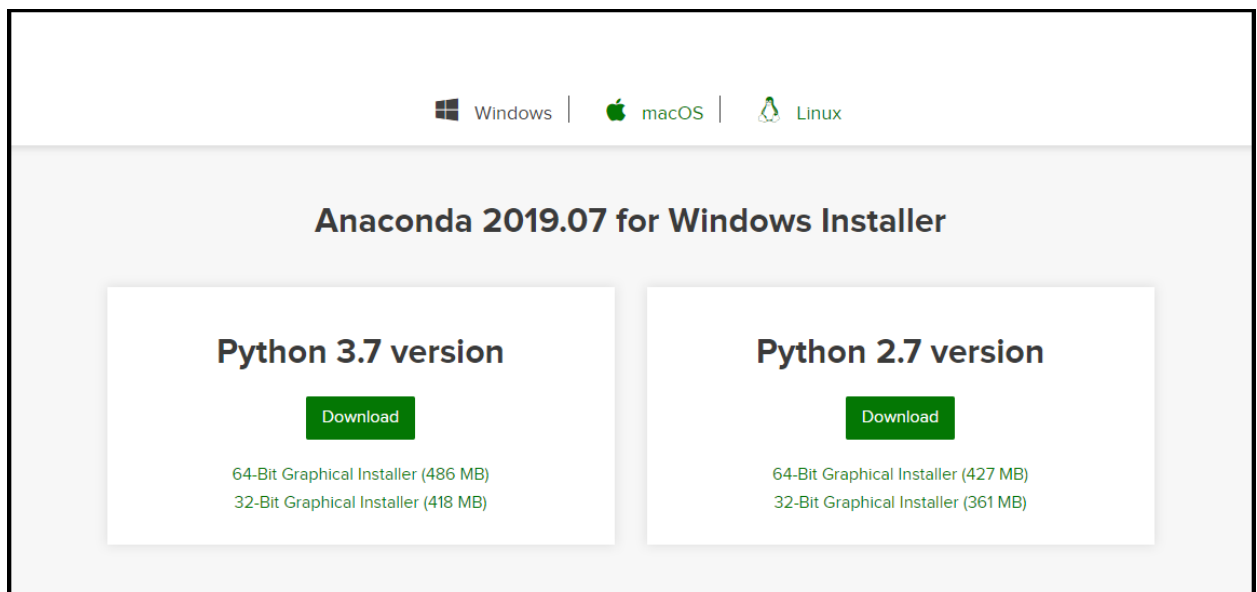
How to install Anaconda?

Anaconda distribution is open source and is available for Windows, Linux and Mac OS. Here are the steps of how to download and install Anaconda for windows.

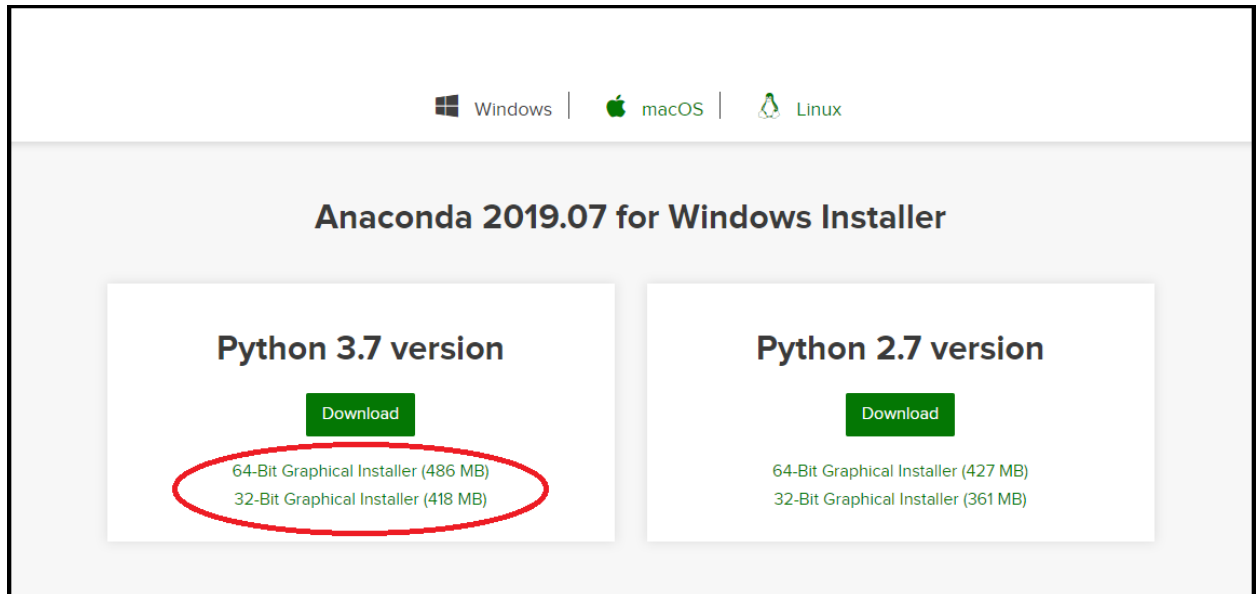
- 1) Log on to <https://www.anaconda.com/distribution/>



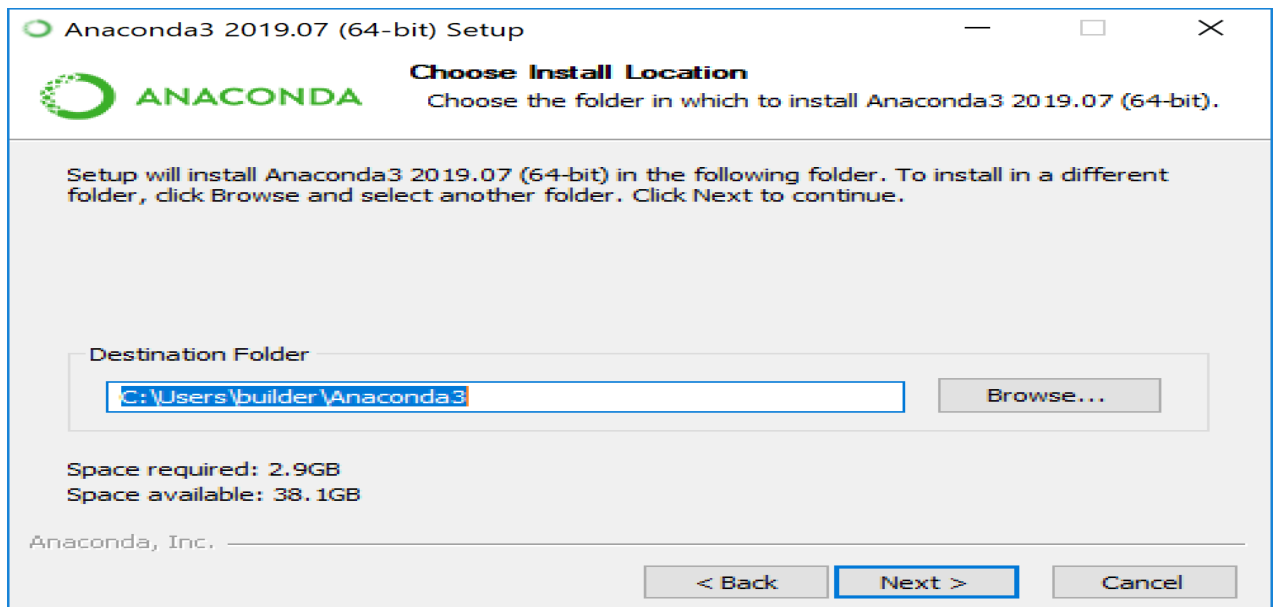
2) Scroll down to the bar with operating system options and click on windows.



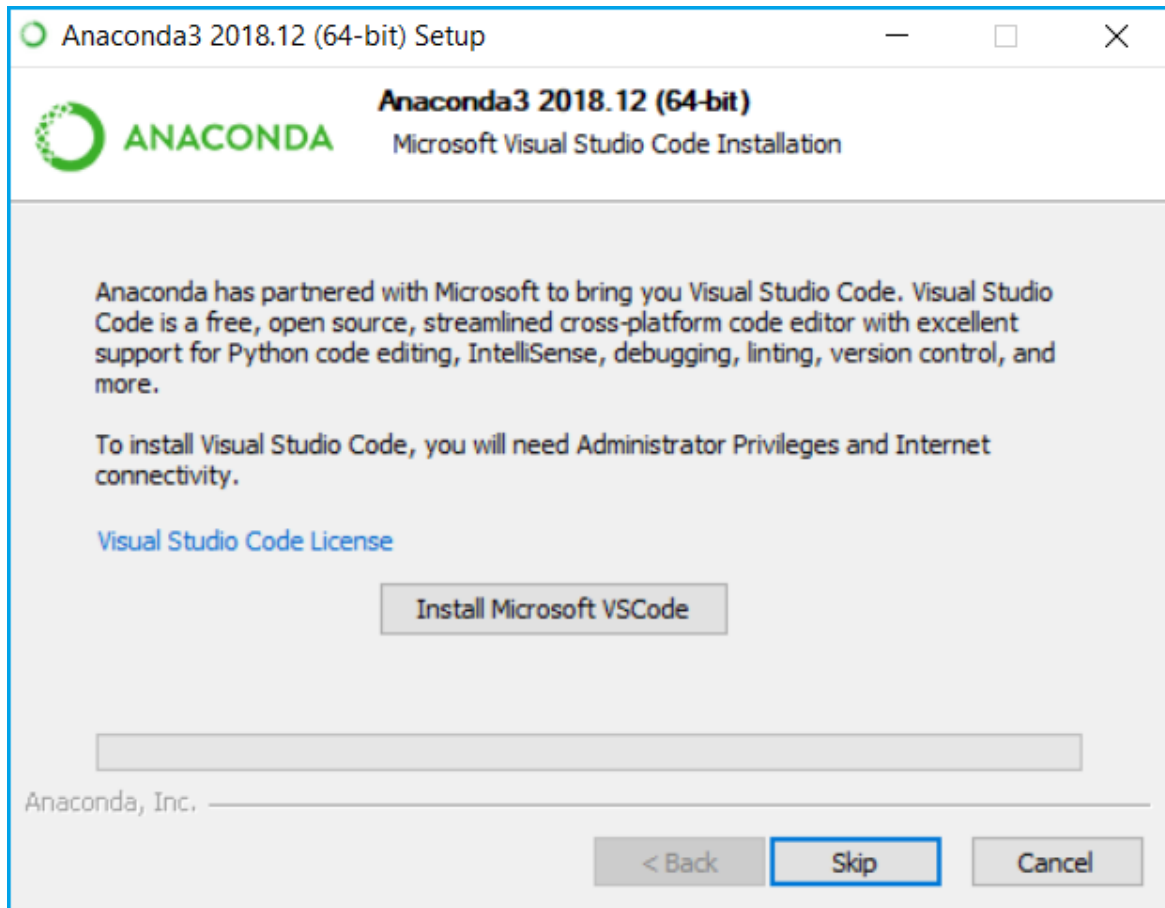
- 3) Under Python 3.7 version, select the right option according to the configuration of your pc(32-bit/64-bit). The download will begin.

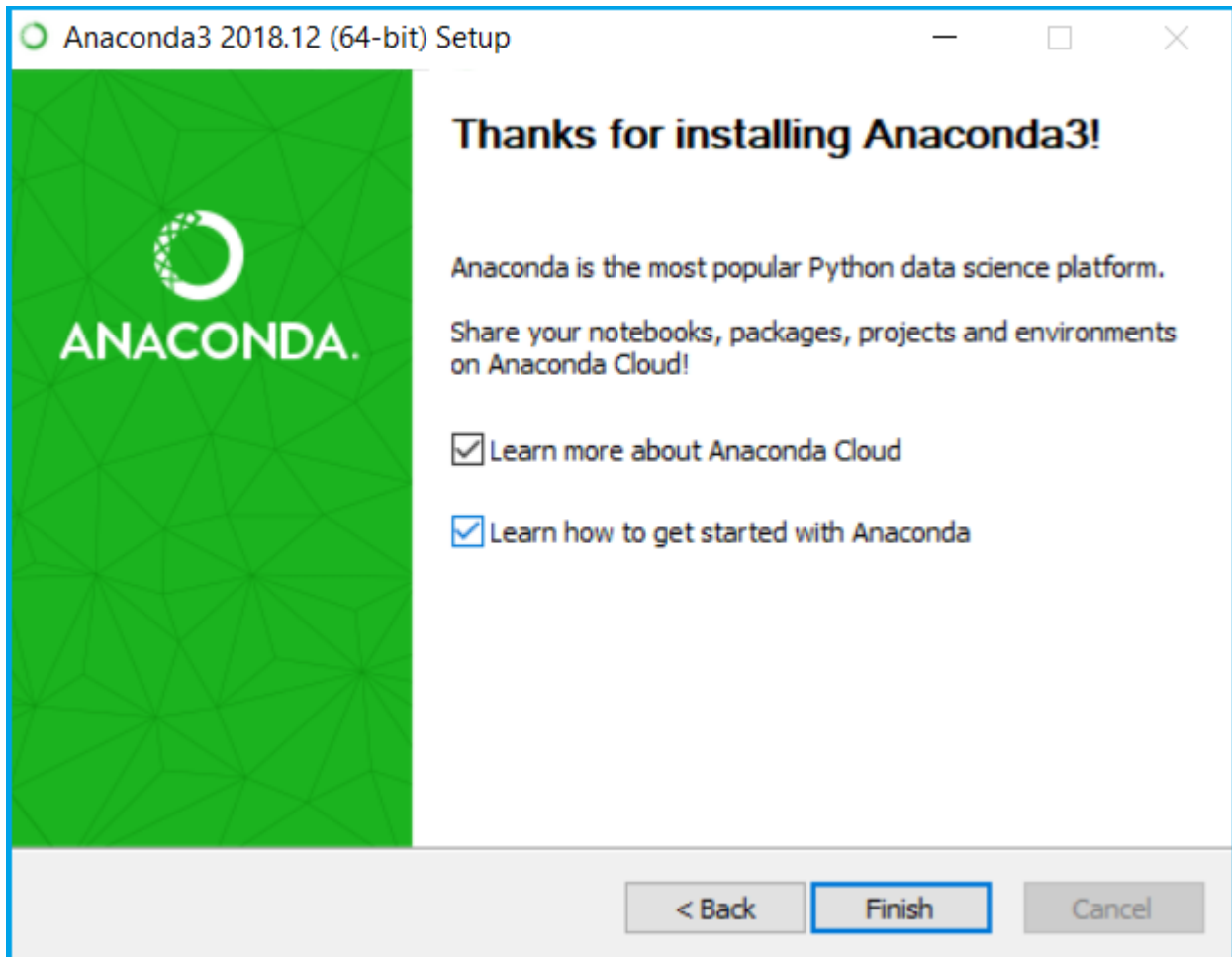


- 4) Double click the installer to launch.
5) Click on "Next".
6) Read the license agreement and click on "I Agree".
7) Select an install for "Just Me" unless you're installing for all users (which requires Windows Administrator privileges) and click "Next".
8) Select destination folder, and click "Next".



- 9) Do not change anything in PATH Options, click "Next".
- 10) Wait for the installation to complete.
- 11) Click on "Skip" to continue.





12) Click on “Finish”. Your Anaconda setup is complete!

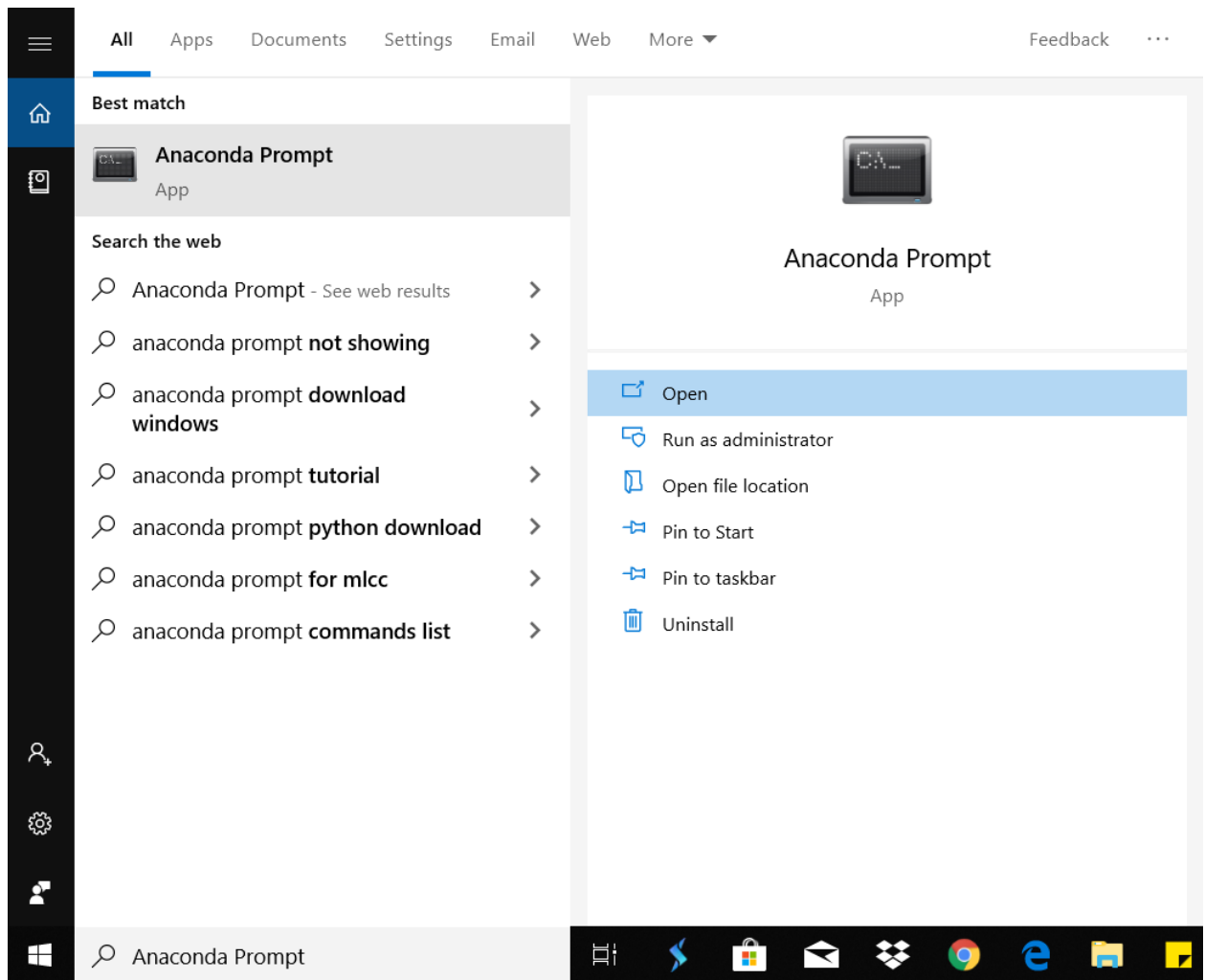
You can also install anaconda for MacOS and Linux from <https://www.anaconda.com/distribution/>

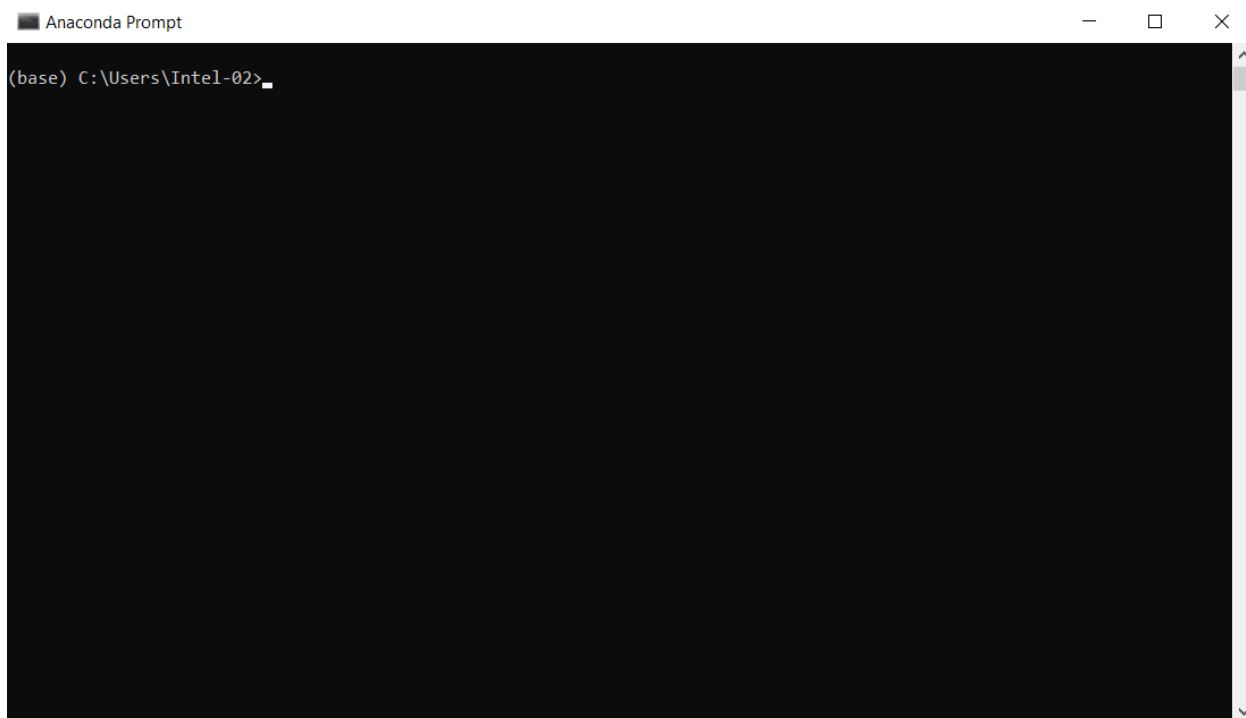
What did we install?

Anaconda Prompt- Anaconda’s Command Line Interface.

Anaconda Prompt is a Python CLI where we can create different virtual environments and install packages into them as per our need.

Anaconda prompt can be opened by writing “Anaconda Prompt” in windows search bar.





Note the **(base)** written at the beginning of the line, it shows that the active environment is base, as it is the default environment.

Jupyter Notebook

<https://www.dataquest.io/blog/jupyter-notebook-tutorial/>
<https://realpython.com/jupyter-notebook-introduction/>
<https://jupyter.readthedocs.io/en/latest/glossary.html#term-kernel>
<https://www.dataquest.io/blog/jupyter-notebook-tutorial/>

Introduction

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting AI related projects. 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, Python remains the most commonly used language for it. In other words, we can say that the Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

What is a Notebook?

Before we dive deep into Jupyter Notebooks, let us first understand what a notebook is. A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. This 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.

Installing Jupyter Notebook

The easiest way to install and start using Jupyter Notebook is through Anaconda. Anaconda is the most widely used Python distribution for data science and comes pre-loaded with all the most popular libraries and tools. With Anaconda, comes the Anaconda Navigator through which we can scroll around all the applications which come along with it.

Working with Jupyter Notebook

To work with Jupyter Notebook, it is necessary to have a kernel on which it operates. A kernel provides programming language support in Jupyter. IPython is the default kernel for Jupyter Notebook. Therefore, whenever we need to work with Jupyter Notebook in a virtual environment, we first need to install a kernel inside the environment in which the jupyter notebook would run.

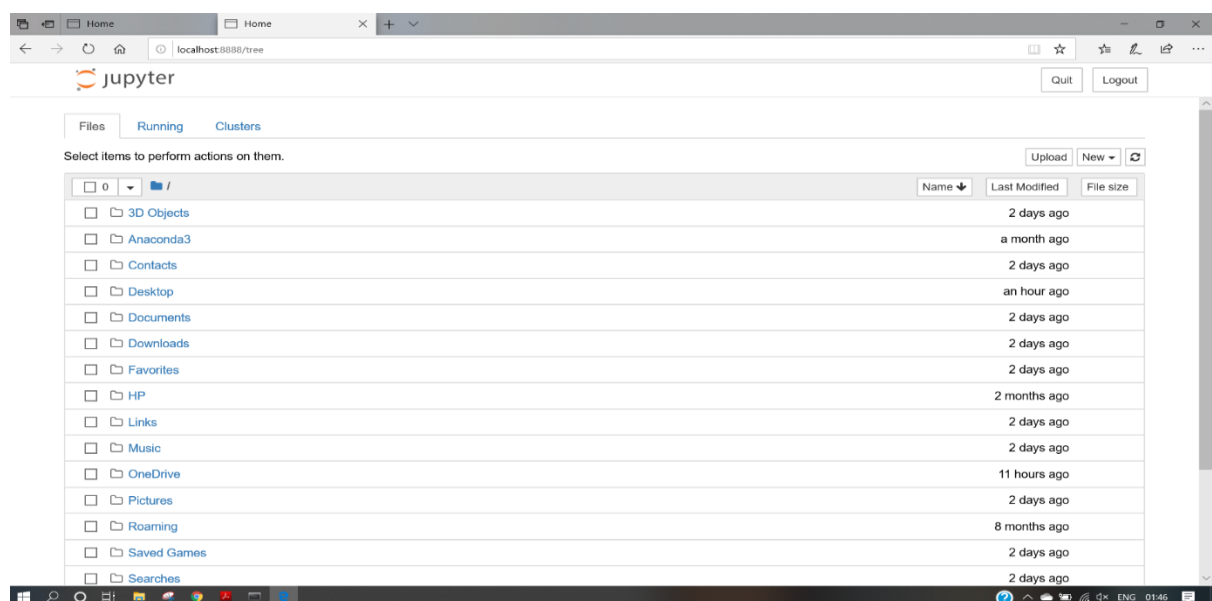
To install the kernel, Open Anaconda Prompt and execute the following command:

```
conda install jupyter nb_conda ipykernel
```

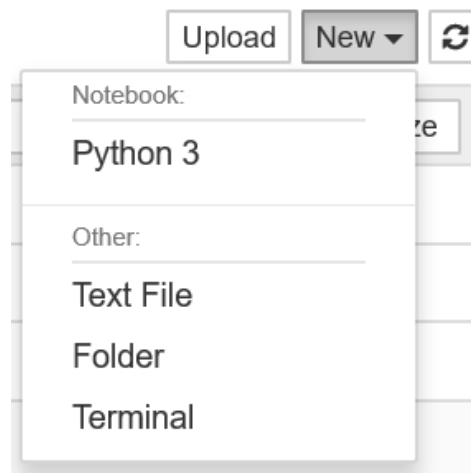
Here, Jupyter is an extension to the Jupyter Notebook which gets installed. Ipykernel is a powerful and interactive Python shell and a jupyter kernel to work with python code in Jupyter Notebooks and nb_conda refers to notebook conda which is an extension to jupyter kernel to set the kernel for a notebook's execution. Once the installation is done, write the following command to open the Jupyter Notebook.

```
jupyter notebook
```

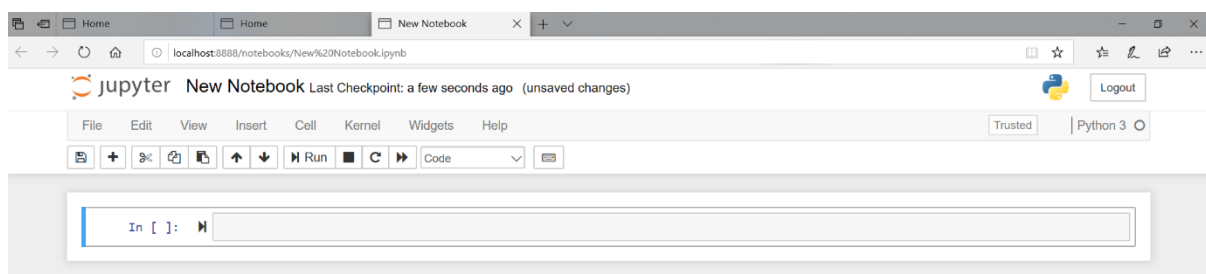
The Jupyter Notebook opens in the default browser with <http://localhost:8888/tree> URL.



In this page, click on New and select Python3 which would open a new Jupyter notebook with Python3 as the default language.



Clicking on Python3 opens a new Jupyter notebook:

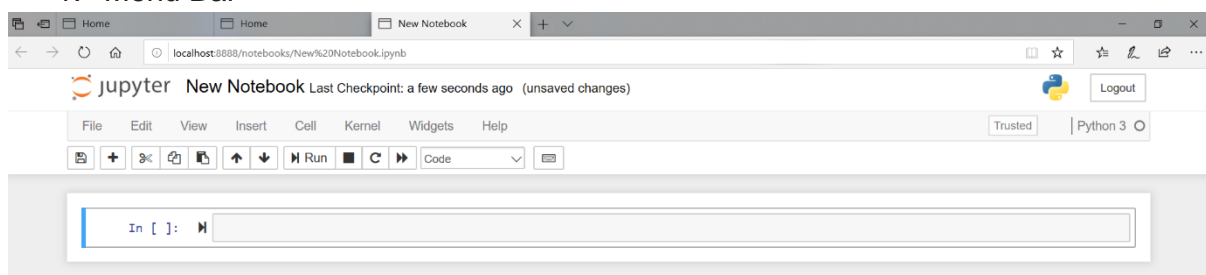


Notebook Interface - Explained!

As we have learnt, Jupyter Notebook is a Graphical User Interface (GUI) which means that the Notebook interface contains a lot of easily-accessible tools for making the work easier as all of them are clicks away.

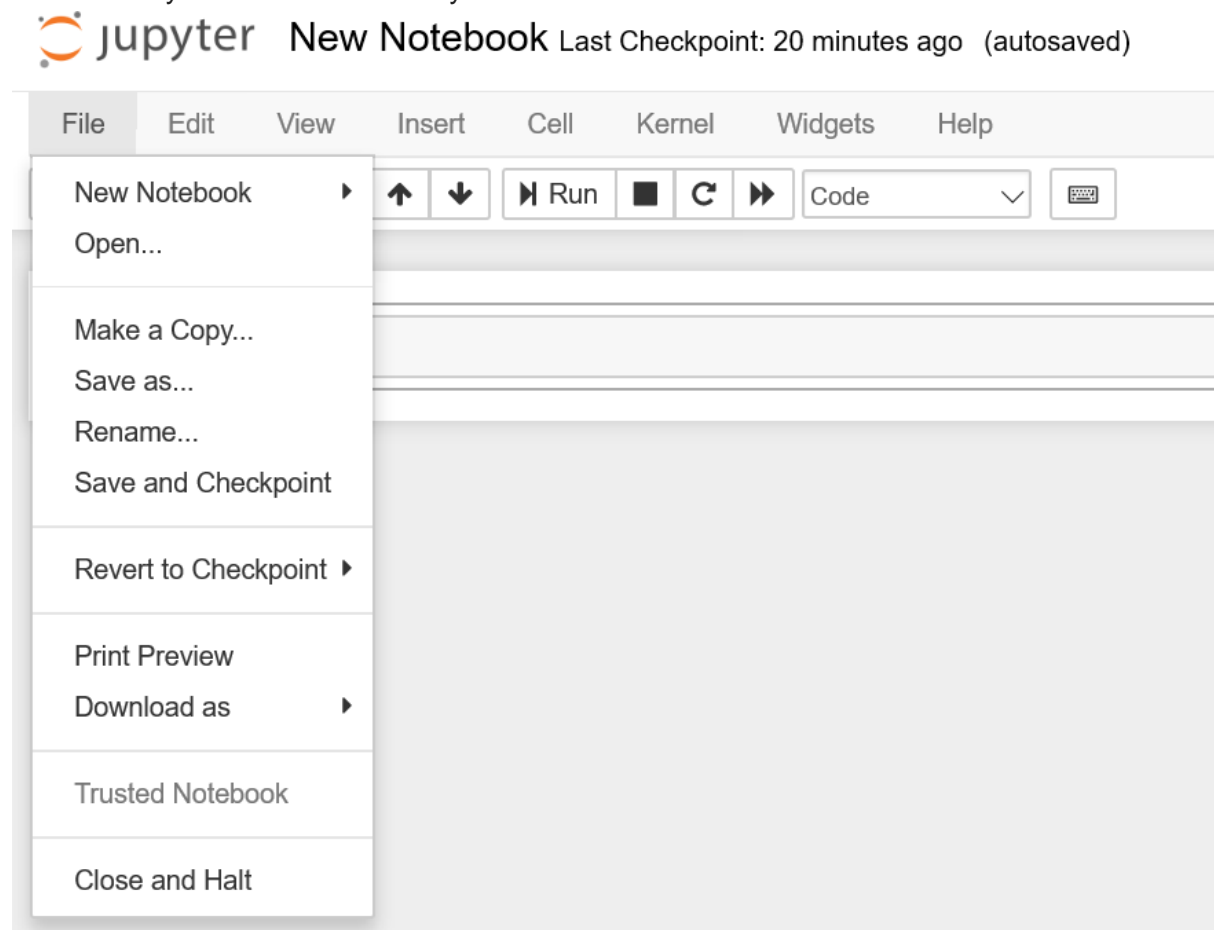
Let us take a tour around the Notebook and understand its features.

1. Menu Bar

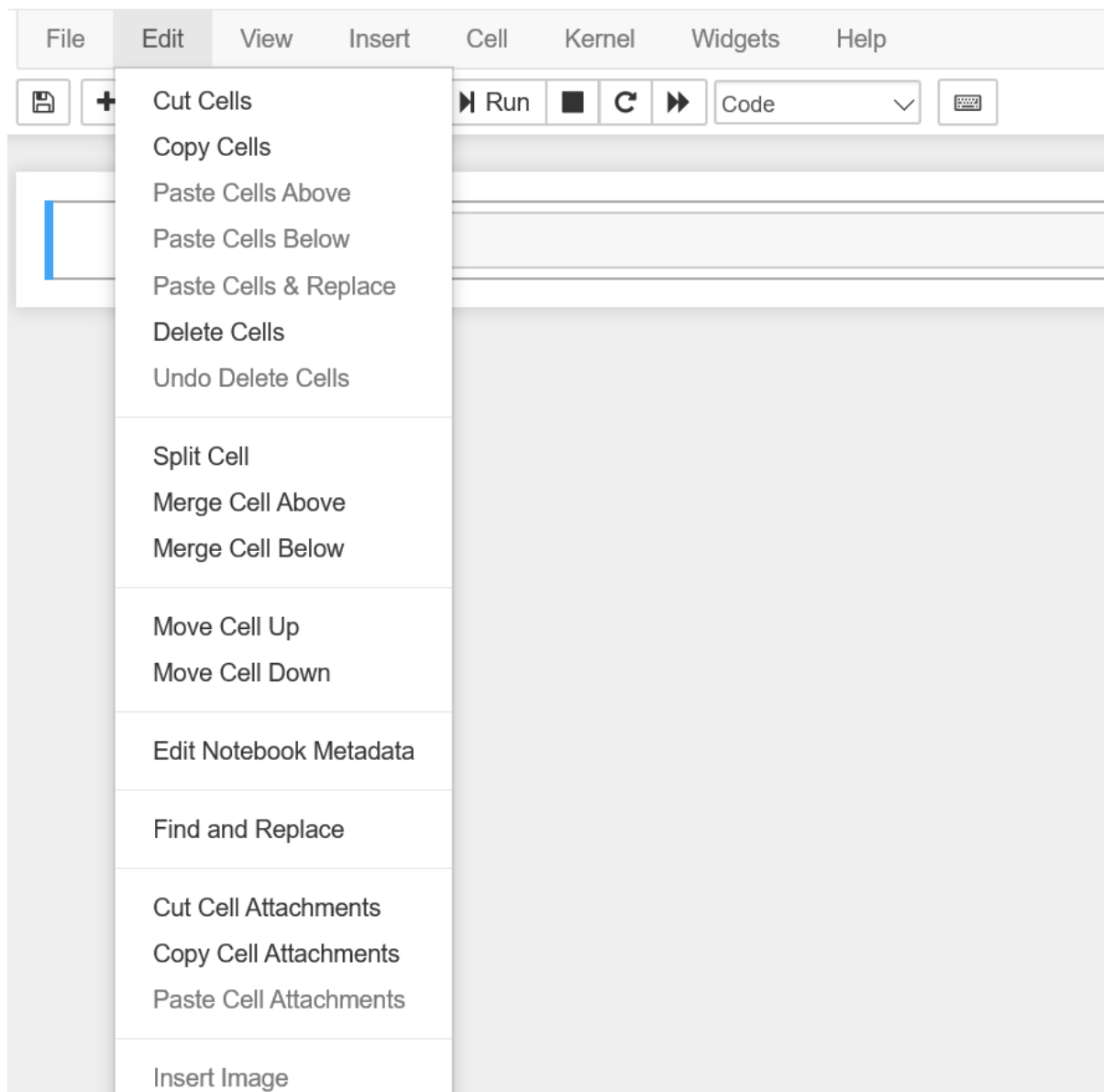


Jupyter Notebook has its own Menu bar which has the following options:

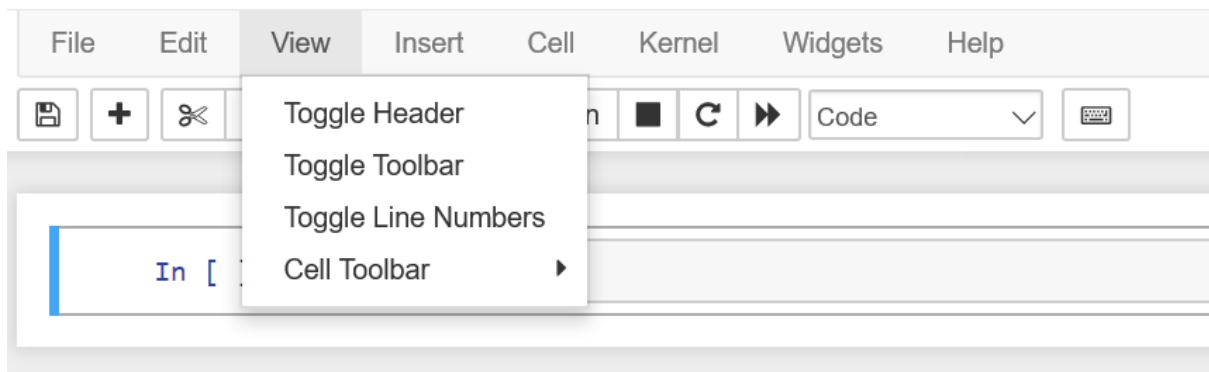
1. File: In the file menu, you can create a new Notebook or open a pre-existing one. This is also where you would go to rename a Notebook. I think the most interesting menu item is the *Save and Checkpoint* option. This allows you to create checkpoints that you can roll back to if you need to.



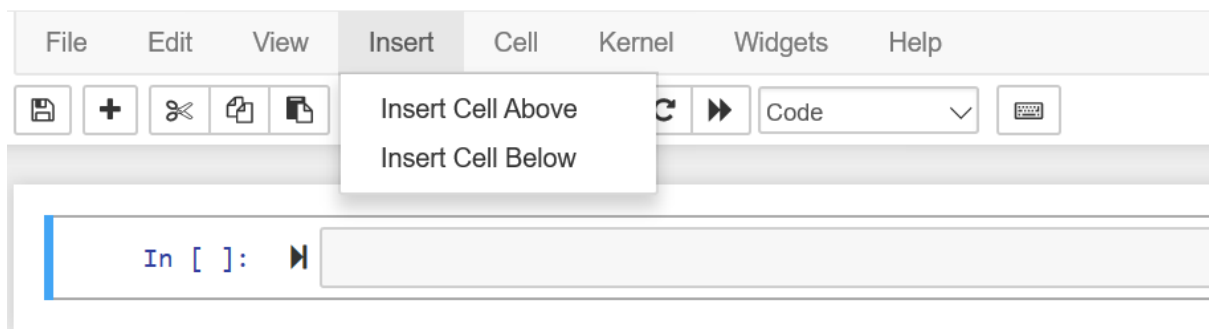
2. Edit Menu: Here you can cut, copy, and paste cells. This is also where you would go if you wanted to delete, split, or merge a cell. You can reorder cells here too.



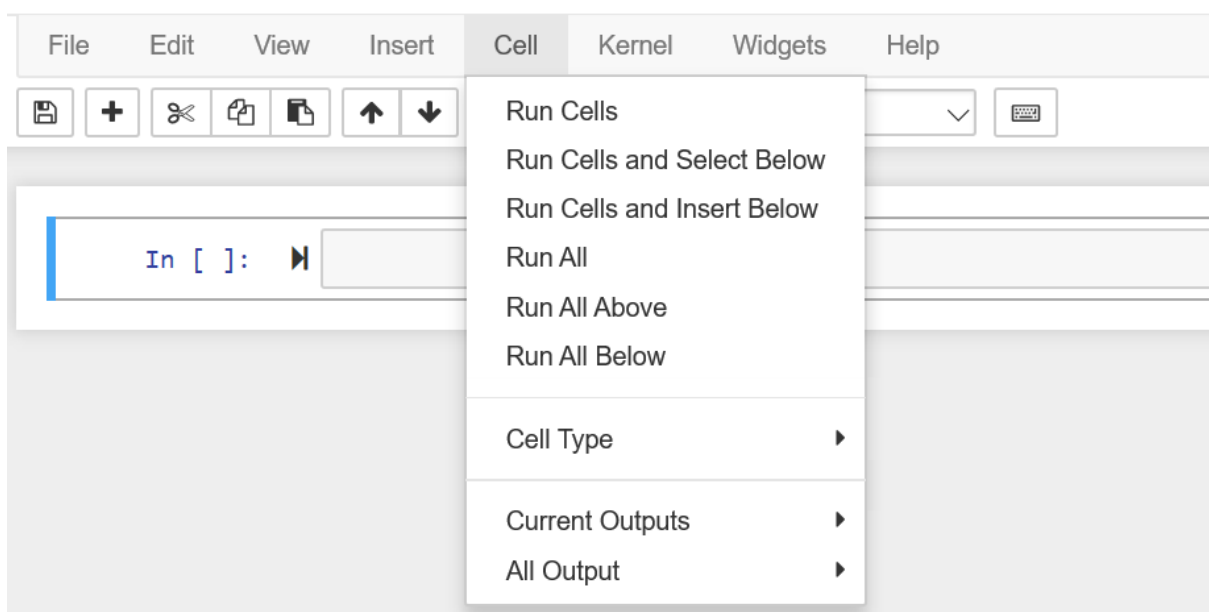
3. View menu: The *View* menu is useful for toggling the visibility of the header and toolbar. You can also toggle *Line Numbers* within cells on or off. This is also where you would go if you want to mess about with the cell's toolbar.



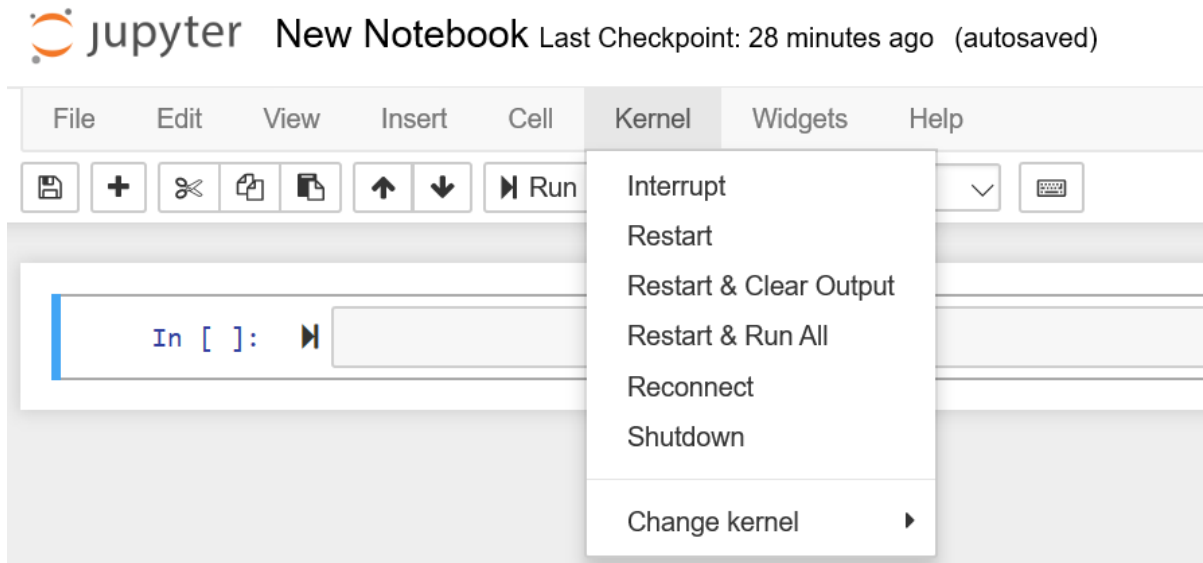
4. Insert menu: The *Insert* menu is just for inserting cells above or below the currently selected cell.



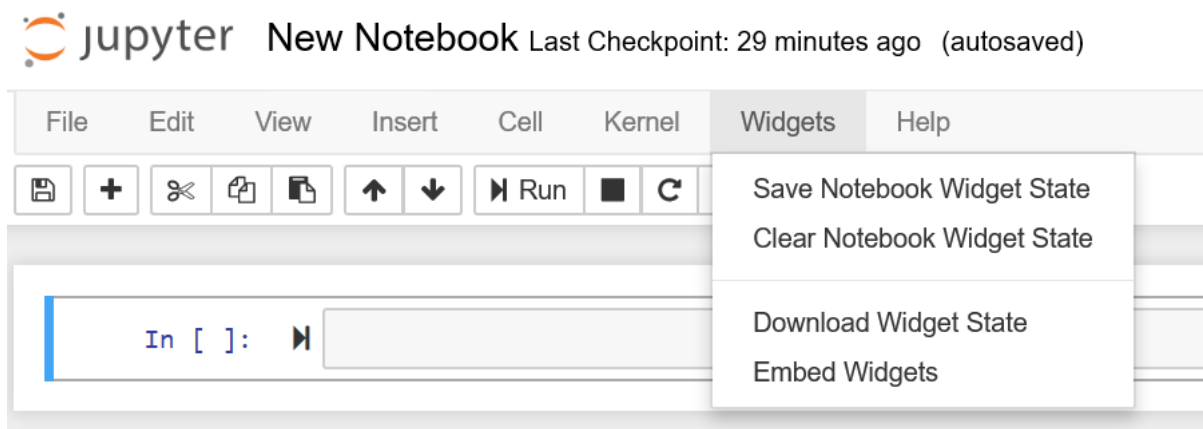
5. Cell menu: The *Cell* menu allows you to run one cell, a group of cells, or all the cells. You can also go here to change a cell's type, although the toolbar is more intuitive for that. The other handy feature in this menu is the ability to clear a cell's output.



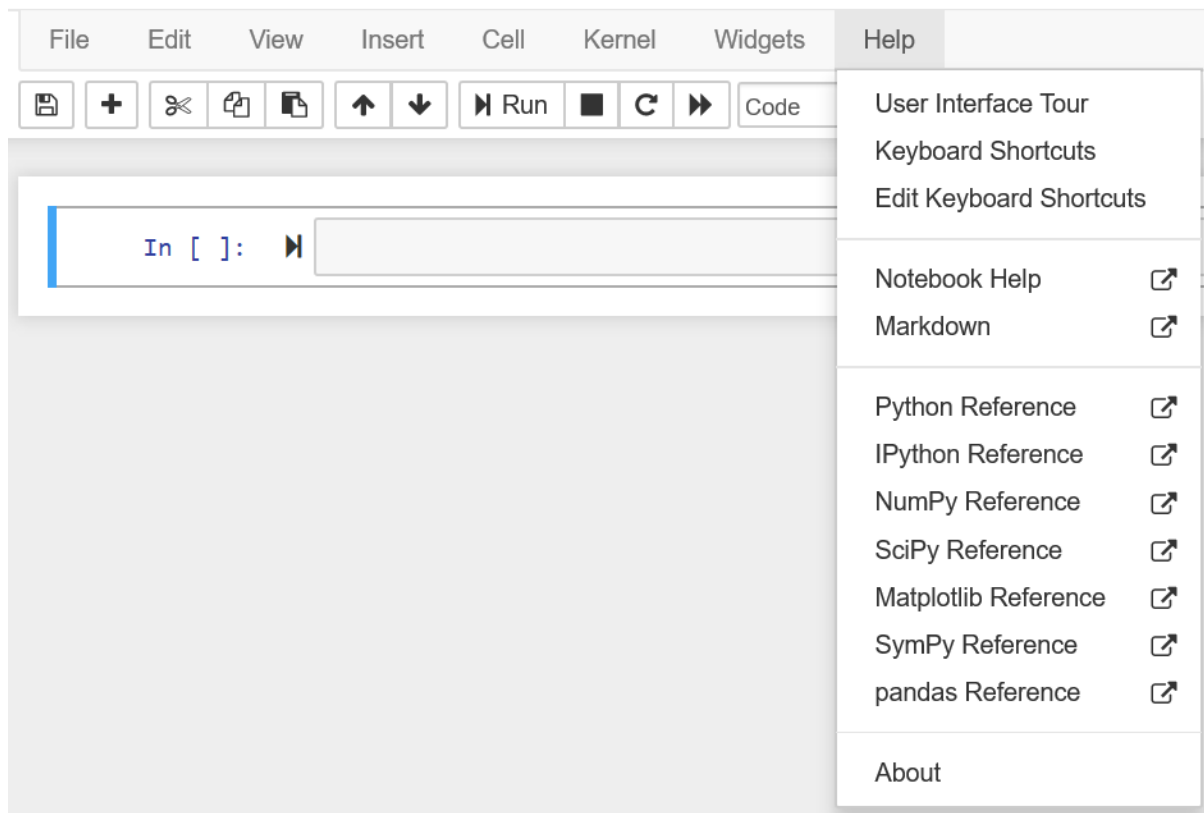
6. Kernel Menu: The *Kernel* cell is for working with the kernel that is running in the background. Here you can restart the kernel, reconnect to it, shut it down, or even change which kernel your Notebook is using.










7. Widgets Menu: The *Widgets* menu is for saving and clearing widget state. Widgets are basically JavaScript widgets that you can add to your cells to make dynamic content using Python (or another Kernel).





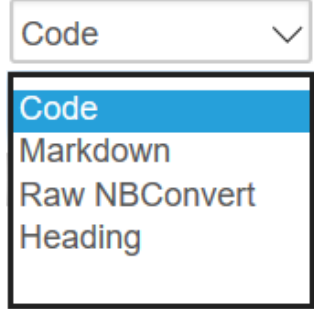


8. Help Menu: Finally, you have the *Help* menu, which is where you go to learn about the Notebook's keyboard shortcuts, a user interface tour, and lots of reference material.



Other than the Menu Bar, a tool bar is also given for our ease in the Notebook interface. Let us get to know about each of the tools.

| | | |
|---|-------|--|
|  | Save | Used to save the progress of Jupyter Notebook. |
|  | Add | Add a cell next to the selected cell in the notebook. |
|  | Cut | Cut/Remove a cell from its location. |
|  | Copy | Copy the contents of a cell. |
|  | Paste | Paste the cut/copied cell below the selected location. |
|  | Shift | Shift selected cells up/down respectively. |
|  | Run | Execute the selected cell. |

| | | |
|---|-------------------|--|
|  | Stop | Break execution of the selected cell. |
|  | Restart | Restart the kernel. |
|  | Restart & Run all | Restart the kernel and re-run the whole notebook. |
|  | Command Palette | Open the command palette containing all the features of Jupyter Notebook. |
|  | | <p>Cell type selection.</p> <p>Code: Executable cell containing python syntax.</p> <p>Markdown: Textual Information</p> <p>Raw NBConvert: Raw text to be kept unmodified in execution.</p> <p>Heading: Add textual headings using #.</p> <p># - Heading level 1</p> <p>## - Heading level 2 and so on.</p> |

Test Your Knowledge

- Q1) What is anaconda and anaconda prompt ?
- Q2) What is a notebook ?
- Q3) How can one install jupyter notebook in anaconda prompt?
- Q4) Explain the different tools in a toolbar
- Q5) Why do we need to install a kernel before running jupyter notebook?

Chapter 4 - More About Lists and Tuples

Introduction to Lists

As studied in the previous section, List is a sequence of values of any type. Values in the list are called elements / items. List is enclosed in square brackets.

Example:

```
a = [1,2.3,"Hello"]
```

List is one of the most frequently used and very versatile data type used in Python. A number of operations can be performed on the lists, which we will study as we go forward.

How to create a list ?

In Python programming, a list is created by placing all the items (elements) inside a square bracket [], separated by commas.

It can have any number of items and they may be of different types (integer, float, string etc.).

Example:

```
#empty list
empty_list = []

#list of integers
age = [15,12,18]

#list with mixed data types
student_height_weight = ["Ansh", 5.7, 60]
```

Note: A list can also have another list as an item. This is called nested lists.

```
# nested list
student_marks = ["Aditya", "10-A", [ "english",75]]
```

How to access elements of a list ?

A list is made up of various elements which need to be individually accessed on the basis of the application it is used for. There are two ways to access an individual element of a list:

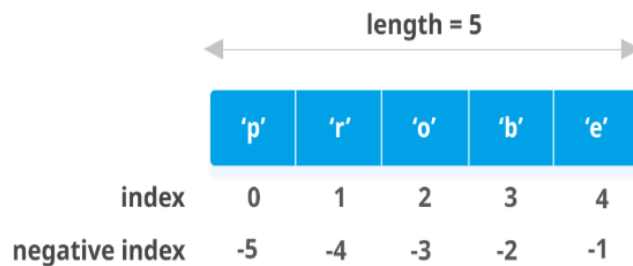
- 1) List Index
- 2) Negative Indexing

List Index

A list index is the position at which any element is present in the list. Index in the list starts from 0, so if a list has 5 elements the index will start from 0 and go on till 4. In order to access an element in a list we need to use index operator [].

Negative Indexing

Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.



In order to access elements using negative indexing, we can use the negative index as mentioned in the above figure.

| Task | Sample Code | Output |
|----------------------------------|---|---|
| Accessing Using List Index | <pre>language = ['p','y','t','h','o','n'] print(my_list[0]) print(my_list[4])</pre> | <pre>p e</pre> |
| | <pre>language = ['p','y','t','h','o','n'] print(my_list[4.0])</pre> | <pre>Y Error! Only Integer can be used for indexing</pre> |
| Accessing Value in a nested list | <pre>n_list = ["Happy",[2,0,1,5]] print(n_list[0][1]) print(n_list[1][3])</pre> | <pre>A 5</pre> |
| Accessing using Negative Index | <pre>day = ['f','r','i','d','a','y'] print(a[-1]) print(a[-6])</pre> | <pre>y f</pre> |

Adding Element to a List

We can add an element to any list using two methods :

- 1) Using append() method
- 2) Using insert() method
- 3) Using extend() method

Using append() method

| Task | Sample Code | Output |
|-----------------------------|---|---|
| Using append() method | <pre>List = [] print("Initial blank List: ") print(List) # Addition of Elements # in the List List.append(1) List.append(2) List.append(4) print("\nList after Addition : ") print(List)</pre> | <pre>Initial blank List: [] List after Addition: [1, 2, 4]</pre> |
| | <pre># Addition of List to a List List2 = ['Good', 'Morning'] List.append(List2) print("\nList after Addition of a List: ") print(List)</pre> | <pre>List after Addition of a List: [1, 2, 4, ['Good', 'Morning']]</pre> |
| Using insert() method | <pre># Creating a List List = [1,2,3,4] print("Initial List: ") print(List) # Addition of Element at # specific Position # (using Insert Method) List.insert(3, 12) List.insert(0, 'Kabir') print("\nList after Insert Operation: ") print(List)</pre> | <pre>Initial List: [1, 2, 3, 4] List after Insert Operation: ['Kabir', 1, 2, 3, 12, 4]</pre> |
| Using | <pre># Creating a List</pre> | <pre>Initial List:</pre> |

| | | |
|--------------------|---|---|
| extend() method | <pre> List = [1,2,3,4] print("Initial List: ") print(List) # Addition of multiple elements # to the List at the end # (using Extend Method) List.extend([8, 'Artificial', 'Intelligence']) print("\nList after Extend Operation: ") print(List) </pre> | <pre> [1, 2, 3, 4] List after Extend Operation: [1, 2, 3, 4, 8, 'Artificial', 'Intelligence'] </pre> |
|--------------------|---|---|

Elements can be added to the List by using built-in append() function. Only one element at a time can be added to the list by using append() method, for addition of multiple elements with the append() method, loops are used. Tuples can also be added to the List with the use of append method because tuples are immutable. Unlike Sets, Lists can also be added to the existing list with the use of append() method.

Using insert() Method

append() method only works for addition of elements at the end of the List, for addition of element at the desired position, insert() method is used. Unlike append() which takes only one argument, insert() method requires two arguments(position, value).

Using extend() method

Other than append() and insert() methods, there's one more method for Addition of elements, extend(), this method is used to add multiple elements at the same time at the end of the list.

Removing Elements from a List

Elements from a list can removed using two methods :

- 1) Using remove() method
- 2) Using pop() method

Using remove() method

| Task | Sample Code | Output |
|-----------------------|--|--|
| Using remove() method | <pre># Creating a List List = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,12] print("Intial List: ") print(List) # Removing elements from List # using Remove() method List.remove(5) List.remove(6) print("\nList after Removal: ") print(List)</pre> | <p>Intial List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]</p> <p>List after Removal: [1, 2, 3, 4, 7, 8, 9, 10, 11, 12]</p> |
| Using pop() method | <pre># Removing element from the # Set using the pop() method List.pop() print("\nList after popping an element: ") print(List) # Removing element at a # specific location from the # Set using the pop() method List.pop(2) print("\nContent after pop ") print(List)</pre> | <p>List after popping an element: [1, 2, 3, 4]</p> <p>List after popping a specific element: [1, 2, 4]</p> |

Elements can be

removed from the List by using built-in [remove\(\)](#) function but an Error arises if element doesn't exist in the set. [Remove\(\)](#) method only removes one element at a time, to remove range of elements, iterator is used. The remove() method removes the specified item.

Note – Remove method in List will only remove the first occurrence of the searched element.

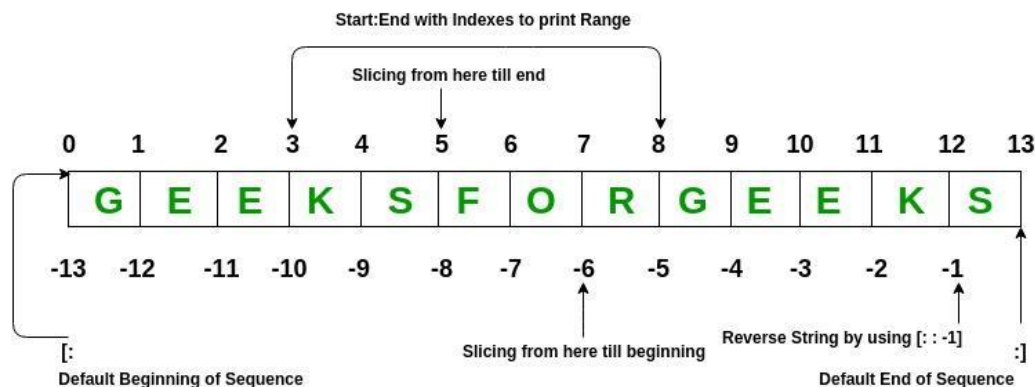
Using pop() method

Pop() function can also be used to remove and return an element from the set, but by default it removes only the last element of the set, to remove an element from a specific position of the List, index of the element is passed as an argument to the pop() method.

Slicing of a List

In Python List, there are multiple ways to print the whole List with all the elements, but to print a specific range of elements from the list, we use [Slice operation](#). Slice operation is performed on Lists with the use of colon(:). To print elements from beginning to a range use [:Index], to print elements from end use[:-Index], to print elements from specific Index till the end use [Index:], to print elements within a range, use [Start Index: End Index] and to print whole List with the use of slicing operation, use [:]. Further, to print whole List in reverse order, use[::-1].

Note – To print elements of List from rear end, use Negative Indexes.



| Task | Sample Code | Output |
|---------|--|--|
| Slicing | <pre># Creating a List List= ['G','O','O','D','M','O', 'R','N','I','N','G'] print("Initial List: ") print(List) # using Slice operation Sliced_List = List[3:8] print("\nSlicing elements in a range 3-8: ") print(Sliced_List)</pre> | <p>Initial List:</p> <pre>['G','O','O','D','M', 'O', 'R','N','I','N','G']</pre> <p>Slicing elements in a range 3-8:</p> <pre>['D', 'M', 'O', 'R', 'N']</pre> |
| | <pre># Print elements from a # pre-defined point to end Sliced_List = List[5:] print("Elements sliced from 5th element till the end: ") print(Sliced_List)</pre> | <p>Elements sliced from 5th element till the end:</p> <pre>['M', 'O', 'R', 'N', 'I', 'N', 'G']</pre> |
| | <pre># Printing elements from # beginning till end Sliced_List = List[:] print("\nPrinting all elements</pre> | <p>Printing all elements using</p> |

| | | |
|---|---|--|
| | <pre>using slice operation: ") print(Sliced_List)</pre> | <pre>slice operation: ['G','O','O','D','M', 'O', 'R','N','I','N','G']</pre> |
| Slicing using negative index of list | <pre># Creating a List List= ['G','O','O','D','M','O', 'R','N','I','N','G'] print("Initial List: ") print(List) # Print elements from beginning # to a pre defined point using # Slice Sliced_List = List[:-6] print("\nElements sliced till 6th element from last: ") print(Sliced_List)</pre> | <pre>Initial List: ['G','O','O','D','M', 'O', 'R','N','I','N','G'] Elements sliced till 6th element from last: ['G','O','O','D','M', 'O']</pre> |
| | <pre># Print elements of a range # using negative index List # slicing Sliced_List = List[-6:-1] print("\nElements sliced from index -6 to -1") print(Sliced_List)</pre> | <pre>Elements sliced from index -6 to -1 ['R', 'N', 'I', 'N', 'G']</pre> |
| | <pre># Printing elements in reverse # using Slice operation Sliced_List = List[::-1] print("\nPrinting List in reverse: ") print(Sliced_List)</pre> | <pre>Printing List in reverse: ['G','N','I','N','R', 'O', 'M','D','O','O','G']</pre> |

List Methods

Some of the other functions which can be used with lists are mentioned below:

| Python List Methods |
|---|
| append() - Add an element to the end of the list |
| extend() - Add all elements of a list to the another list |
| insert() - Insert an item at the defined index |
| remove() - Removes an item from the list |
| pop() - Removes and returns an element at the given index |
| clear() - Removes all items from the list |
| index() - Returns the index of the first matched item |
| count() - Returns the count of number of items passed as an argument |
| sort() - Sort items in a list in ascending order |
| reverse() - Reverse the order of items in the list |
| copy() - Returns a shallow copy of the list |

Now that you have learnt the basic concepts of Lists in python, it is time for us to get our hands on to practicing about list using a Jupyter Notebook.

To open jupyter notebook, go to start menu and open anaconda prompt and **jupyter notebook** in it.

Let's Practice

Go through the List Jupyter Notebook to get an experiential learning experience for Lists. To download the Jupyter Notebook, go to the following link : http://bit.ly/lists_jupyter

Note:

To open Jupyter notebook, go to start menu → open anaconda prompt → write “jupyter notebook”

Introduction to Tuples

Tuple is a collection of Python objects which is ordered and unchangeable. The sequence of values stored in a tuple can be of any type, and they are indexed by integers. Values of a tuple are syntactically separated by ‘commas’. Although it is not necessary, it is more

common to define a tuple by closing the sequence of values in parentheses. This helps in understanding the Python tuples more easily.

Example

```
fruits = ("apple", "banana", "cherry")
```

How to Create a tuple ?

In Python, tuples are created by placing sequence of values separated by 'comma' with or without the use of parentheses for grouping of data sequence. It can contain any number of elements and of any datatype (like strings, integers, list, etc.).

Example

```
#Creating an empty Tuple
Tuple1 = ()

#Creating a Tuple with the use of string
Tuple1 = ('Artificial', 'Intelligence')

#Creating a Tuple with Mixed Datatype
Tuple1 = (5, 'Artificial', 7, 'Intelligence')
```

Accessing of Tuples

We can use the index operator `[]` to access an item in a tuple where the index starts from 0. So, a tuple having 6 elements will have indices from 0 to 5. Trying to access an element outside of tuple (for example, 6, 7,...) will raise an `IndexError`. The index must be an integer; so, we cannot use float or other types. This will result in `TypeError`.

Example

```
fruits = ("apple", "banana", "cherry")
print(fruits[1])
```

The above code will give the output as `"banana"`

Deleting a Tuple

Tuples are immutable and hence they do not allow deletion of a part of it. Entire tuple gets deleted by the use of `del()` method.

Example

```
num = (0, 1, 2, 3, 4)
del num
```

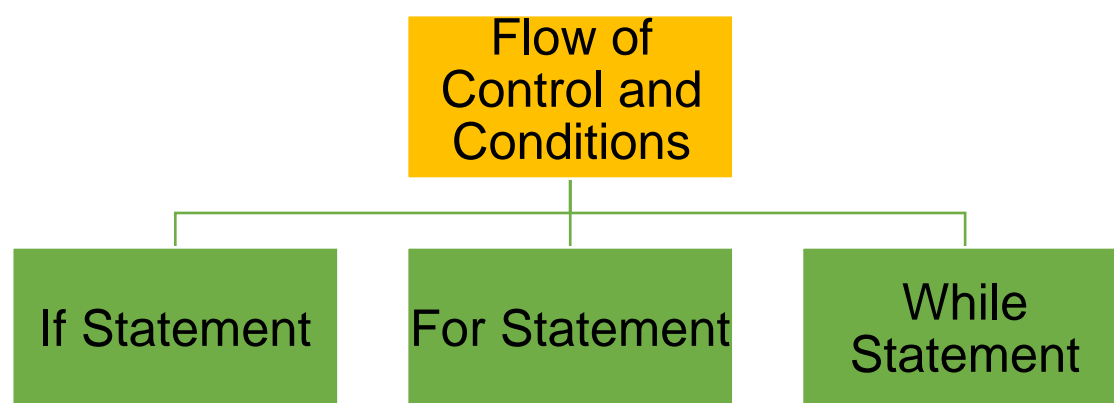
Test Your Knowledge

- Q1) What is the difference between a list and a tuple ?
- Q2) Explain the different ways of slicing a list with a help of an example.
- Q3) What is negative indexing ? Explain with an example.
- Q4) Mention the methods to remove elements from a list.
- Q5) What is the method to delete an element from a tuple?

Chapter 5 - Flow of Control and Conditions

In the programs we have seen till now, there has always been a series of statements faithfully executed by Python in exact top-down order. What if you wanted to change the flow of how it works? For example, you want the program to take some decisions and do different things depending on different situations, such as printing 'Good Morning' or 'Good Evening' depending on the time of the day?

As you might have guessed, this is achieved using control flow statements. There are three control flow statements in Python - if, for and while.



If Statement

On the occasion of World Health Day, one of the schools in the city decided to take an initiative to help students maintain their health and be fit. Let's observe an interesting conversation happening between the students when they come to know about the initiative.



There come situations in real life when we need to make some decisions and based on these decisions, we need to decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code.

Decision making statements in programming languages decide the direction of flow of program execution. Decision making statements available in Python are:

- if statement
- if..else statements
- if-elif ladder

If Statement

The if statement is used to check a condition: *if* the condition is true, we run a block of statements (called the *if-block*).

Syntax:

```
if test expression:  
    statement(s)
```

Here, the program evaluates the test expression and will execute statement(s) only if the text expression is True.

If the text expression is False, the statement(s) is not executed.

Note:

- 1) In Python, the body of the if statement is indicated by the indentation. Body starts with an indentation and the first unindented line marks the end.
- 2) Python interprets non-zero values as True. None and 0 are interpreted as False.

Python if Statement Flowchart

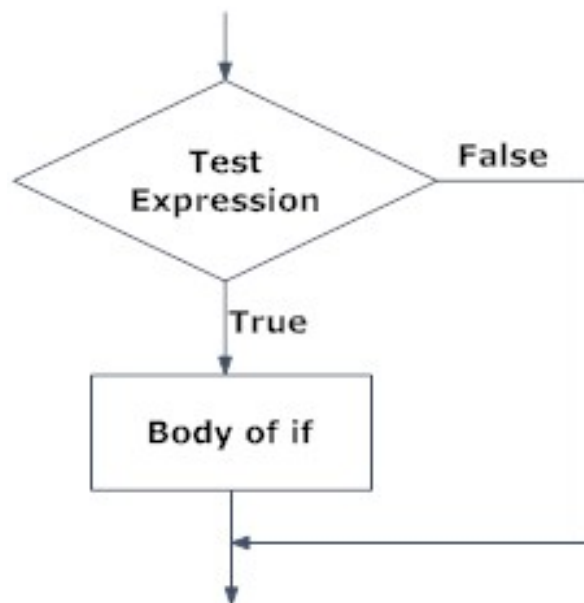


Fig: Operation of if statement

Example :

```
#Check if the number is positive, we print an appropriate message

num = 3
if num > 0:
    print(num, "is a positive number.")
print("this is always printed")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("this is always printed")
```

When you run the program, the output will be:

```
3 is a positive number
This is always printed
This is also always printed.
```

In the above example, `num > 0` is the test expression. The body of `if` is executed only if this evaluates to `True`.

When variable `num` is equal to 3, test expression is true and body inside body of `if` is executed.

If variable `num` is equal to -1, test expression is false and body inside body of `if` is skipped.

The `print()` statement falls outside of the `if` block (unindented). Hence, it is executed regardless of the test expression.

Python if...else Statement

Syntax of if...else

```
if test expression:
    Body of if
else:
    Body of else
```

The `if..else` statement evaluates test expression and will execute body of `if` only when test condition is `True`.

If the condition is `False`, body of `else` is executed. Indentation is used to separate the blocks.

Python if..else Flowchart

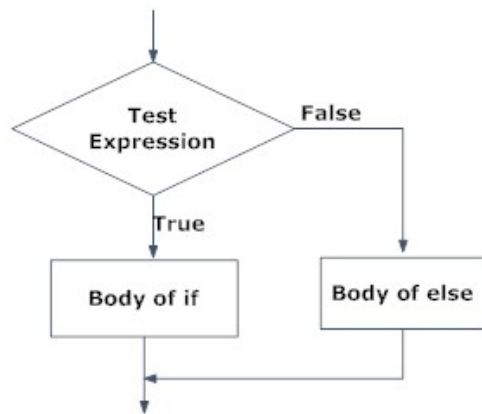


Fig: Operation of if...else statement

Example of if...else

```
#A program to check if a person can vote
age = input("Enter Your Age")
if age >= 18:
    print("You are eligible to vote")
else:
    print("You are not eligible to vote")
```

In the above example, when if the age entered by the person is greater than or equal to 18, he/she can vote. Otherwise, the person is not eligible to vote

Python if...elif...else Statement

Syntax of if...elif...else

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

The elif is short for else if. It allows us to check for multiple expressions.

If the condition for if is False, it checks the condition of the next elif block and so on.

If all the conditions are False, body of else is executed.

Only one block among the several if...elif...else blocks is executed according to the condition.

The if block can have only one else block. But it can have multiple elif blocks.

Flowchart of if...elif...else

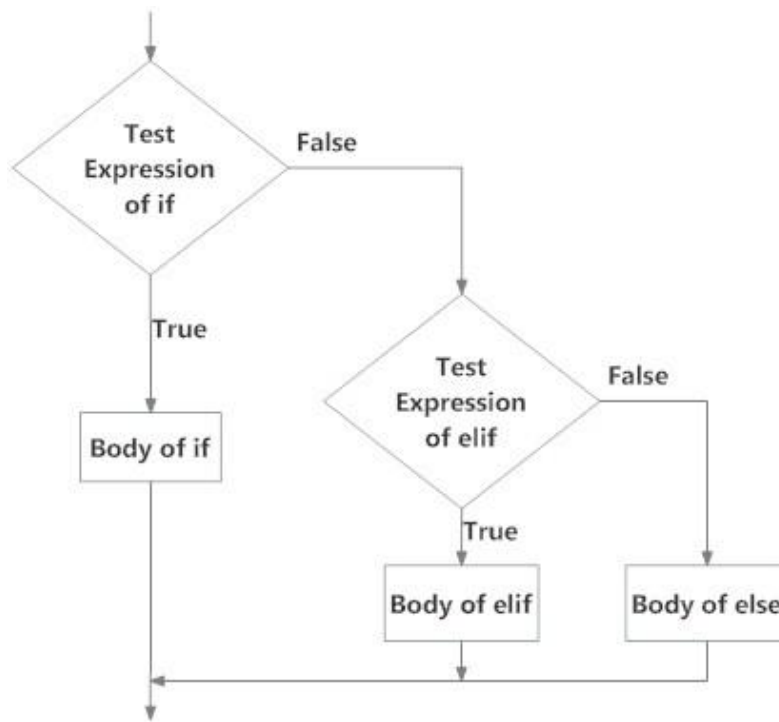


Fig: Operation of if...elif...else statement

Example of if...elif...else

```
#To check the grade of a student

Marks = 60

if marks > 75:
    print("You get an A grade")
elif marks > 60:
    print("You get a B grade")
else:
    print("You get a C grade")
```

Python Nested if statements

We can have an if...elif...else statement inside another if...elif...else statement. This is called nesting in computer programming.

Any number of these statements can be nested inside one another. Indentation is the only way to figure out the level of nesting. This can get confusing, so must be avoided if it can be.

Python Nested if Example

```
# In this program, we input a number
# check if the number is positive or
# negative or zero and display
# an appropriate message
# This time we use nested if

num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

When you run the above program

Output 1

```
Enter a number: 5
Positive number
```

Output 2

```
Enter a number: -1
Negative number
```

Output 3

```
Enter a number: 0
Zero
```

Let's Practice

Let's Go through the If-Else Jupyter Notebook to get an experiential learning experience for If-Else. To download the Jupyter Notebook, go to the following link : http://bit.ly/ifelse_jupyter

Note:

To open Jupyter notebook, go to start menu → open anaconda prompt → write "jupyter notebook"

The For Loop

The for..in statement is another looping statement which *iterates* over a sequence of objects i.e. go through each item in a sequence. We will see more about sequences in detail in later chapters. What you need to know right now is that a sequence is just an ordered collection of items.

Syntax of for Loop

```
for val in sequence:  
    Body of for
```

Here, val is the variable that takes the value of the item inside the sequence on each iteration.

Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.

Flowchart of for Loop

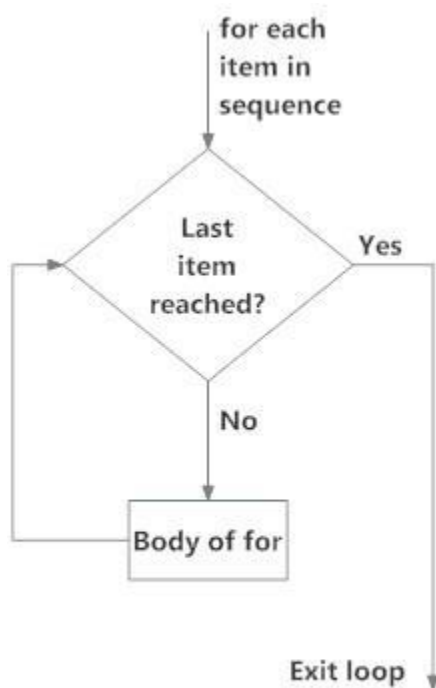


Fig: operation of for loop

Example: Python for Loop

```
# Program to find the sum of all numbers stored in a list  
  
# List of numbers  
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
```

```
# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

# Output: The sum is 48
print("The sum is", sum)
```

when you run the program, the output will be:

```
The sum is 48
```

The while Statement

The while statement allows you to repeatedly execute a block of statements as long as a condition is true. A while statement is an example of what is called a *looping* statement. A while statement can have an optional else clause.

Syntax of while Loop in Python

```
while test_expression:
    Body of while
```

In while loop, test expression is checked first. The body of the loop is entered only if the test_expression evaluates to `True`. After one iteration, the test expression is checked again. This process continues until the test_expression evaluates to `False`. In Python, the body of the while loop is determined through indentation. Body starts with indentation and the first unindented line marks the end. Python interprets any non-zero value as `True`. `None` and `0` are interpreted as `False`.

Flowchart of while Loop

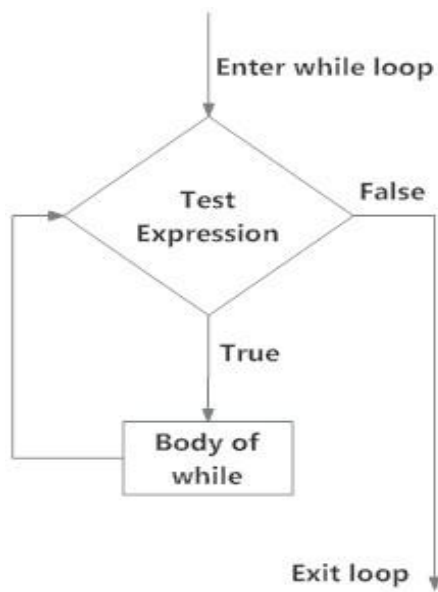


Fig: operation of while loop

Example: Python while Loop

```
# Program to add natural
# numbers upto
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))

n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

When you run the program, the output will be:

```
Enter n: 10
The sum is 55
```

In the above program, the test expression will be True as long as our counter variable *i* is less than or equal to *n* (10 in our program).

We need to increase the value of the counter variable in the body of the loop. This is very important (and mostly forgotten). Failing to do so will result in an infinite loop (never ending loop).

Finally, the result is displayed.

Let's Practice

Let's Go through the flow control Jupyter Notebook to get an experiential learning experience for 'for' and 'while' loop . To download the Jupyter Notebook, go to the following link : http://bit.ly/loops_jupyter

Note:

To open Jupyter notebook, go to start menu → open anaconda prompt → write "jupyter notebook"

Test Your Knowledge

Q1) Explain different types of 'If' statements with the help of a flowchart.

Q2) What is the difference between 'for' loop and 'while' loop. Explain with a help of a flowchart?

Q3) What are python nested if statements? Explain with example.

Q4) Write a program to find numbers which are divisible by 7 and multiple of 5 between 1200 and 2200.

Q5) Write a program to find the whether a number is prime or not using 'while' loop.

Chapter 6 : Introduction to Packages

Recap

Till now, you have worked around various python syntaxes. Conditional statements, Control flow statements, variables, datatypes, etc. are some of the concepts you now are quite familiar with. Let us test your skills and see how well have you understood it all.

CHALLENGE TIME!

Here are 5 questions. Go through them individually and try answers them all.

- 1) Which of the following is NOT a legal variable name ?

| | |
|--------------|---------------|
| mark_student | student_mark |
| StudentMark | _Mark_Student |

- 2) What is the correct syntax to output the type of variable in python ?

| | |
|-----------------|------------------|
| print(typeof x) | print(typeof(x)) |
| print(type(x)) | print(type x) |

- 3) What is the command to open Jupyter Notebook in anaconda prompt?

| | |
|------------------------|---------------------------|
| conda jupyter notebook | open jupyter notebook |
| jupyter notebook | activate jupyter notebook |

- 4) Which of the following "if" statement will not be executed ?

| | |
|------------------------|-----------------------|
| if (a>b): print(a) | if (a>b): print(a) |
| if (a>b) : print(a) | if (a>b): print(a) |

- 5) What will be the output of the following code?

```
x=2
while x<=10:
    print(x)
    x = x + 1
```


Print number from 1 to 10

Print numbers from 1 to 9

Print numbers from 2 to 10

Print numbers from 2 to 9

Or did you find a need to take a look back at the concepts?

Analyze your score and mention it here:

This image shows a single page of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page, leaving small margins at the top and bottom. There are no vertical margin lines, and the page is completely blank except for the lines themselves.

Step 1: Collect the exam scores for Mathematics, Science, Social Science, Hindi and English for all the students.

87

| Student's Roll No. | Marks in Science | Marks in Mathematics | Marks in Social Science | Marks in Hindi | Marks in English |
|--------------------|------------------|----------------------|-------------------------|----------------|------------------|
| Student 1 | 89 | 75 | 73 | 84 | 78 |
| Student 2 | 54 | 65 | 57 | 75 | 48 |
| ... | ... | ... | ... | ... | ... |
| Student 40 | 98 | 100 | 97 | 94 | 97 |

Step 3: New columns get added to the database for total marks and the percentage.

What is the formula to calculate percentage?

| Student's Roll No. | Marks in Science | Marks in Mathematics | Marks in Social Science | Marks in Hindi | Marks in English | Total Marks | Percentage |
|--------------------|------------------|----------------------|-------------------------|----------------|------------------|-------------|------------|
| Student 1 | 89 | 75 | 73 | 84 | 78 | 399 | 79.8% |
| Student 2 | 54 | 65 | 57 | 75 | 48 | 299 | 59.8% |
| ... | ... | ... | ... | ... | ... | ... | ... |
| Student 40 | 98 | 100 | 97 | 94 | 97 | 486 | 97.2% |

Step 4: Finally, the database has been successfully created. Now, we need to analyze class performance as a whole. In this case, statistics come to our rescue. Various parameters that come into picture are mentioned below. Can you write how to find them all? List them down!

1. Average Score of the class.

2. Average percentage of the class performance.

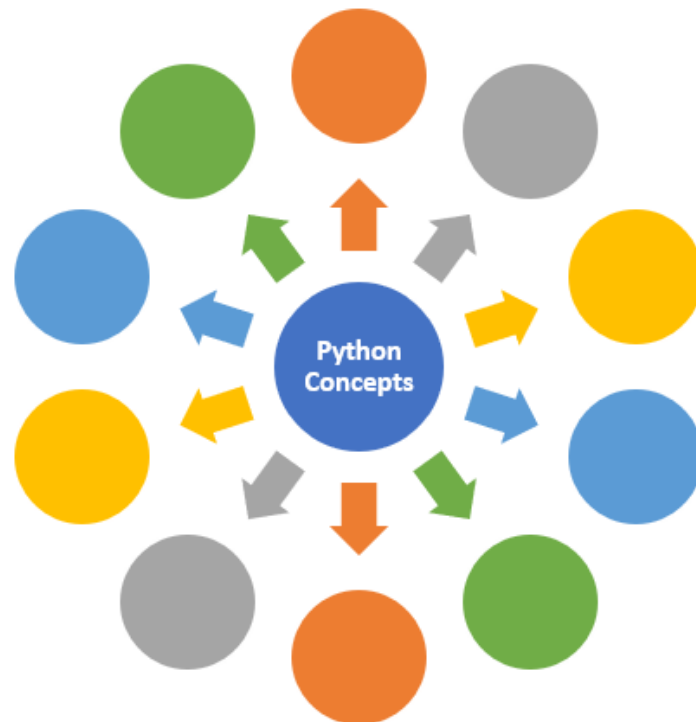
3. Number of students passed / failed.

4. Success percentage of the class

5. Top 10 students of the class.

After going through this process, we can say that preparing the exam result manually is a tedious and time-consuming process. Therefore, it should be automated by creating a python script!

As you have already learnt a lot of Python concepts, list down the ones which can be used to create a python script of result creation.



There are a lot of functions which can be used in this process. But as it is all about numbers, we need to look for something that explicitly works around numbers so that the work becomes easier. With python, we get the advantage of using open-sourced packages available on the internet. But what are packages? Let's find out.

What is a package?



Here, we can draw analogy for packages with the book shelf. The history bookshelf is nothing but a package which contains multiple books of similar type and provides the user with a variety to choose from to make it easier for them to find out the exact information needed.

In other words, a package is nothing but a space where you can find codes or functions or modules of similar type. There are various packages readily available to use for free (perks of python being an open-sourced language) for various purposes.

Some of the readily available packages are:

NumPy

- A package created to work around numerical arrays in python.
- Handy when it comes to working with large numerical databases and calculations around it.

OpenCV

- An image processing package which can explicitly work around images and can be used for image manipulation and processing like cropping, resizing, editing, etc.

Matplotlib

- A package which helps in plotting the analytical (numerical) data in graphical form.
- It helps the user in visualizing the data thereby helping them in understanding it better.

NLTK

- NLTK stands for Natural Language Tool Kit and it helps in tasks related to textual data.
- It is one of the most commonly used package for Natural Language Processing.

Pandas

- A package which helps in handling 2-dimensional data tables in python.
- It is useful when we need to work with data from excel sheets and other databases.

Package Installation

Using these packages is easy. All we need to do is install the package and import it wherever required.

Any package can be installed by directly writing the following command in the Anaconda

```
conda install numpy
```

The name of package can be replaced at the end of the statement.

Multiple packages can also be installed in just one command:

```
conda install numpy pandas matplotlib
```

This would install NumPy, Pandas and Matplotlib altogether.

Once you begin the installation, after a bit of processing, the prompt would ask if you wish to proceed for the installation or not:

```
Proceed ([y]/n)?
```

Press **Y** to continue with the installation. Within a few minutes, the packages will be installed and would be ready to use.

Working with a package

To use a package, we need to import it in the script wherever it is required. There are various versions of importing a package in Python:

```
import numpy
```

Meaning: Import numpy in the file to use its functionalities in the file to which it has been imported.

```
import numpy as np
```

Meaning: Import numpy and refer to it as np wherever it is used.

```
from numpy import array
```

Meaning: import only one functionality (array) from the whole numpy package. While this gives faster processing, it limits the package's usability.

```
from numpy import array as arr
```

Meaning: Import only one functionality (array) from the whole numpy package and refer to it as arr wherever it is used.

A lot of other combinations can also be explored while importing packages like importing multiple functionalities of a package in a single statement, etc.

What is NumPy?

NumPy, which stands for Numerical Python, is the fundamental package for mathematical and logical operations on arrays in Python. It is a commonly used package when it comes to working around numbers. NumPy gives a wide range of arithmetic operations around numbers giving us an easier approach in working with them. NumPy also works with arrays, which is nothing but homogenous collection of Data.

An array is nothing but a set of multiple values which are of same datatype. They can be numbers, characters, Booleans, etc. but only one datatype can be accessed through an array. In NumPy, the arrays used are known as ND-arrays (N-Dimensional Arrays) as NumPy comes with a feature of creating n-dimensional arrays in python.

An array can easily be compared to a list. Let us take a look how different they are:

| NumPy Arrays | Lists |
|--|---|
| <ol style="list-style-type: none">1. Homogenous collection of Data.2. Can contain only one type of data, hence not flexible with datatypes.3. Cannot be directly initialized. Can be operated with Numpy package only.4. Direct numerical operations can be done. For example, dividing the whole array by 3 divides every element by 3.5. Widely used for arithmetic operations.6. Arrays take less memory space.7. Example: To create a Numpy array 'A': <pre>import numpy A=numpy.array([1,2,3,4,5,6,7,8,9,0])</pre> | <ol style="list-style-type: none">1. Heterogenous collection of Data.2. Can contain multiple types of data, hence flexible with datatypes.3. Can be directly initialized as it is the part of python syntax.4. Direct numerical operations are not possible. For example, dividing the whole list by 3 cannot divide every element by 3.5. Widely used for data management.6. Lists acquire more memory space.7. Example: To create a list: <pre>A = [1,2,3,4,5,6,7,8,9,0]</pre> |

Since NumPy package is not included in the basic Python installation, we need to install it separately. Once it is installed, it can be readily used in any Python code whenever imported.

Exploring NumPy!

NumPy package provides us with various features and functions which helps us in arithmetic and logical operations. Let us look at some of them:

1. NumPy Arrays: As discussed before, arrays are homogenous collection of datatypes. With NumPy, we can create n-dimensional arrays (where n can be any integer) and operate on them using other mathematical functions.

Here are some ways by which you can create arrays using NumPy package assuming the NumPy packages is imported already.

| Function | Code |
|--|---------------------------------------|
| Creating a Numpy Array | <code>numpy.array([1,2,3,4,5])</code> |
| Creating a 2-Dimensional zero array (4X3 – 4 rows and 3 columns) | <code>numpy.zeros((4,3))</code> |
| Creating an array with 5 random values | <code>numpy.random.random(5)</code> |
| Creating a 2-Dimensional constant value array (3X4 – 3 rows and 4 columns) having all 6s | <code>numpy.full((3,4),6)</code> |
| Creating a sequential array from 0 to 30 with gaps of 5 | <code>numpy.arange(0,30,5)</code> |

One of the features of array is that we can perform arithmetic functions on the elements of the array directly by performing them on the whole array.

Let us assume the array is “ARR” and it has been initialized as:

```
ARR = numpy.array([1,2,3,4,5])
```

Now, let us look at various operations that could be implemented on this array:

| Function | Code |
|---|------------------------|
| Adding 5 to each element | <code>ARR + 5</code> |
| Divide each element by 5 | <code>ARR / 5</code> |
| Squaring each element | <code>ARR ** 2</code> |
| Accessing 2 nd element of the array (element count starts from 0) | <code>ARR[1]</code> |
| Multiplying 2 arrays {consider BRR = <code>numpy.array([6,7,8,9,0])</code> } | <code>ARR * BRR</code> |

As you can see, direct arithmetical operations can be implemented on individual array elements just by manipulating the whole array variable.

Let us look at the functions which talk about the properties of an array:

| Function | Code |
|--|------------------------|
| Type of an array | <code>type(ARR)</code> |
| Check the dimensions of an array | <code>ARR.ndim</code> |
| Shape of an array | <code>ARR.shape</code> |
| Size of an array | <code>ARR.size</code> |
| Datatype of elements stored in the array | <code>ARR.dtype</code> |

Some other mathematical functions available with NumPy are:

| Function | Code |
|--|--------------------------------|
| Finding out maximum element of an array | <code>ARR.max()</code> |
| Finding out row-wise maximum elements | <code>ARR.max(axis = 1)</code> |
| Finding out column-wise minimum elements | <code>ARR.min(axis = 0)</code> |
| Sum of all array elements | <code>ARR.sum()</code> |

Let's Practice!

TASK 1

To understand these functions better, let us try and execute all the functions we read above on a Jupyter Notebook. To download the Jupyter Notebook, go to the following link: http://bit.ly/numpy_jupyter and download NumPy Basic notebook.

TASK 2

Go through the NumPy Jupyter Notebook to get an experiential learning experience for NumPy. To download the Jupyter Notebook, go to the following link : http://bit.ly/numpy_jupyter and download NumPy Advance notebook

Note:

To open Jupyter notebook, go to start menu → open anaconda prompt → write "jupyter notebook"

Test Your Knowledge

- Q1) What is a package ? Give Example with its use.
- Q2) What is the command to install a package.
- Q3) How can we use a package in a code? Explain with an example.
- Q4) What is a NumPy array ? Give example
- Q5) What is a difference between a NumPy array and python list.

Additional Resources

- NCERT Class 11 Python Text Book
- Introduction to Python :
 - https://www.w3schools.com/python/python_intro.asp
 - <https://www.programiz.com/python-programming/first-program>
 - <https://www.geeksforgeeks.org/python-programming-language/>
- Introduction to Flowchart and Algorithm: <https://www.edrawsoft.com/explain-algorithm-flowchart.php>
- Python Lists : https://www.w3schools.com/python/python_lists.asp
- Python While Loop : https://www.w3schools.com/python/python_while_loops.asp
- Python For Loop: https://www.w3schools.com/python/python_for_loops.asp
- Python Condition Statements: https://www.w3schools.com/python/python_conditions.asp
- Python Packages: https://www.w3schools.com/python/python_modules.asp