Agenda of Class 3

- Introduction to Python
- Installation of Anaconda Distribution

Introduction about Python:

- Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

- Python can be used for
    - analyzing data,
    - games,
    - web development (server-side),
    - software development,
    - mathematics,
    - system scripting.
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax:

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. **Eg: Sample python code**

```
a=5
print("Now a is an integer. Its value is ", a)


b='Our India'
print("Now b is a string. Its value is ", b)
```

- Python relies on indentation, using whitespace, to define scope;
  - such as the scope of loops,
  - functions and
  - classes.
  - Other programming languages often use curly-brackets for this purpose.

Python Indentation

- Indentation refers to the spaces at the beginning of a code line.
- Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
- Python uses indentation to indicate a block of code.

- Indentation is a very important concept of Python because without proper indenting the Python code, you will end up seeing IndentationError and the code will not get compiled.
- Python indentation is a way of telling a Python interpreter that the group of statements belongs to a particular block of code.
- A block is a combination of all these statements.

Eg: Block of statements

Python Code

```
a=5
b=7

if a > b:
 print("a is greater than b.")
 print("I am in a block.")
 print("a block is bigger than b block.")

else:
 print("b is greater than a.")
 print("I am in b block.")
 print("b block is bigger than a block.")
```

C Code

```
int main()
{
   int a=5;
int b=7;

if (a > b)
{
printf("a is greater than b.\n");
printf("I am in a block.\n");
    printf("a block is bigger than b block.");
     }

else
{
printf("b is greater than a.\n");
    printf("I am in b block.\n");
 printf("b block is bigger than a block.");
     }
return 0;
}
```

- Block can be regarded as the grouping of statements for a specific purpose.
- Most of the programming languages like C, C++, Java use braces { } to define a block of code.
- Python uses indentation to highlight the blocks of code. Whitespace is used for indentation in Python.
- All statements with the same distance to the right belong to the same block of code.
- If a block has to be more deeply nested, it is simply indented further to the right.
- **Note:** Python uses 4 spaces as indentation by default.
- However, the number of spaces is up to you, but a minimum of 1 space has to be used.

Python Comments

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

Creating a Comment

Comments starts with a #, and Python will ignore them:

Example

```
#This is a comment
print("Hello, World!")
```

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

Example

```
print("Hello, World!") #This is a comment
```

- Comments does not have to be text to explain the code, it can also be used to prevent Python from executing code:

Example

```
#print("Hello, World!")
print("Cheers, Mate!")
```

Multi Line Comments

- They are useful when the comment text does not fit into one line; therefore needs to span across lines.
- Multi-line comments or paragraphs serve as documentation for others reading your code.
- Python does not really have a syntax for multiline comments.
- To add a multiline comment you could insert a `#` for each line:

Example

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

We can also use a multiline string

- Python multi-line comment is a piece of text enclosed in a delimiter `("""")` on each end of the comment.
- There should be no white space between delimiter `("""")`.
- Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

Example

```
"""
This is a comment
written in
more than just one line
```

```
"""
print("Hello, World!")
```

- As long as the string is not assigned to a variable,
    - Python will read the code,
    - but then ignore it,
    - and you have made a multiline comment.

Python Variables

- Variables are containers for storing data values.
- Unlike other programming languages, Python has no command for declaring a variable.
- In Python, variables are created when you assign a value to it.
- Variables do not need to be declared with any particular *type (integer, character, string, etc.)*, and can even change type after they have been set.
- String variables can be declared either by using single or double quotes:

Example:

```
a=5  # Here, a is of type int
print("Now a is an integer. Its value is ", a)
c='x'
a="Our VIT"  # Now, a is of type string
print("Now a is a string. Its value is ", a)
print("\n") # New line
b='Our India' # Now, b is of type string
print("Now b is a string. Its value is ", b)
 d='India Science'
```

Rules for Python variables:

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

#Legal variable names:

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

#Illegal variable names:

```
2myvar = "John"
my-var = "John"
my var = "John"
```

Assign Value to Multiple Variables

Python allows you to assign values to multiple variables in one line:

Example

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

We can assign the *same* value to multiple variables in one line:

Example

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Output Variables

The Python print statement is often used to output variables.

To combine both text and a variable, Python uses the + character:

Example

```
x = "awesome"
print("Python is " + x)
```

Output: **Python is awesome**

You can also use the + character to add a variable to another variable:

Example

```
x = "Python is "
y = "awesome"
z =  x + y
print(z)
```

For numbers, the + character works as a mathematical operator:

Example

```
x = 5
y = 10
print(x + y)
```

If you try to combine a string and a number, Python will give you an error:

Example

```
x = 5
y = "John"
print(x + y)
```

Error: Unsupported operand types

Taking input in Python

- Developers often have a need to interact with users, either to get data or to provide some sort of result.
- Most programs today use a dialog box as a way of asking the user to provide some type of input.
- Python provides us with two inbuilt functions to read the input from the keyboard.

Example:

```
val = input("Enter your value: ")
print(val)
```

How the input function works in Python :

- When input() function executes, program flow will be stopped until the user has given an input.
- The text or message display on the output screen to ask a user to enter input value is optional i.e. the prompt, will be printed on the screen is optional.
- Whatever you enter as input, input function convert it into a string.
    - if you enter an integer value still input() function convert it into a string.
    - You need to explicitly convert it into an integer in your code using typecasting.

Typecasting

1. **Typecasting the input to Integer:** There might be conditions when you might require integer input from user/Console, the following code takes two input(integer/float) from console and typecasts them to integer then prints the sum.

```
# input
num1 = int(input())
num2 = int(input())

# printing the sum in integer
print(num1 + num2)
```

2. **Typecasting the input to Float:** To convert the input to float the following code will work out.

```
# input
num1 = float(input())
num2 = float(input())
```

```
# printing the sum in float
print(num1 + num2)
```

3. **Typecasting the input to String:** All kind of input can be converted to string type whether they are float or integer. We make use of keyword str for typecasting.

```
# input
string = str(input())

# output
print(string)
```

Example code – without type casting:

```
a = input("Enter the value for a: ")

print(a)

b = input("Enter the value for b: ")

print(b)


if a > b:

 print("a is greater than b.")

 print("I am in a block.")

 print("a block is bigger than b block.")


else:

 print("b is greater than a.")

 print("I am in b block.")

 print("b block is bigger than a block.")
```

Example code – with type casting:

```
a = int(input("Enter the value for a: "))
print(a)
b = int(input("Enter the value for b: "))
print(b)

if a > b:
 print("a is greater than b.")
 print("I am in a block.")
 print("a block is bigger than b block.")

else:
 print("b is greater than a.")
 print("I am in b block.")
 print("b block is bigger than a block.")
```

Activity-1:

- **Installing Anaconda Distribution**
- Anaconda is a free and open-source distribution of the Python used for scientific computing, that aims to simplify package management and deployment.
- The distribution includes data-science packages suitable for Windows, Linux, and macOS.
- **URLs:**
  - https://www.anaconda.com/
  - https://www.anaconda.com/products/individual

Activity-2:

Using Anaconda distribution, execute the following programs.

a) Write a python program to read 5 numbers from users and print the addition result.

b) Write a python code to read radius of a circle (say ‘r’) from user and calculate the circumference of the circle. Print the result.