1. Basic Syntax

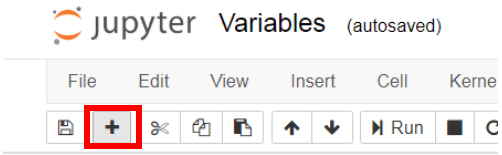
In Task 0 you learnt how to install Jupyter Notebook. So, now we will go through some very basic codes snippets to get you familiar with the platform.

Follow the steps in Task 0 to execute/run the code.

Example 1:

```
In [2]: 5+2
Out[2]: 7
```

Add new cell by clicking on **+** tab available in menu bar at top



Example 2:

Hope you've understood how to write and run a code in Jupyter Notebook.

```
In [3]: a = 5
        b = 9
        c = a+b
        c
Out[3]: 14
```

So, moving forward, in this section you will be learning about the basic syntax of python 3.x. The prescribed rule of writing a code is called **syntax**.

1. Lines/statement and Indentation

1.1 Lines/statement

In C/C++/C#, a semicolon (;) denotes the end of a statement. But in Python, a piece of text terminated by pressing "Enter" key is considered as one statement.

Example:

```
In [17]: print("Hello")
         print("Welcome")
```

Outcome:

```
Hello
Welcome
```

Here **print("Hello")** and **print("Welcome")** are the 2 lines/statements.
Note: *print( )is a function in python used to diplay the output. Detailed examples of the same are given in point 5 of this section.*

1.1.1 Continuation of Statement

However, you can show the text spread over more than one line to be a single statement by using the backslash (\) as a continuation character. Look at the following example.

Example:

```
In [19]: print("Hello")
         print("Welcome to online summer training \
         program by R-Labs")
```

Outcome:

```
Hello
Welcome to online summer training program by R-Labs
```

1.1.2 Multiple Statements in Single Line

If you want to print multiple statements on single line, use semicolon

Example:

```
In [23]: print("Hello"); print("Welcome")
```

Outcome:

```
Hello
Welcome
```

1.2 Indentation

- The meaning of indentation is "to start a line/ block of text few spaces away from the margin than from the main part of the text."
- In C, C++, statements inside curly brackets '{' and '}' are treated as a block.
  For example:

```
void main()
{
   Statement 1;
   Statement 2;
}
```

From the above example, Statement 1& Statement 2, are considered as the part of the part of the 'main' function. We could identify it as a part of main function because they were enclosed in curly brackets.

- Similarly in python, **uniform indentation** is used to denote a block of statements.
- When a block is to be started, type the colon symbol (**:**) and press Enter.

- After pressing 'Enter', the cursor goes to the next line and automatically leaves an additional whitespaces (called **indent**).

  Note: *By default the indent consists of* ***'4 spaces (of spacebar)'*** *which is equivalent to* ***'1 Tab'****. So if you want to give indentation you need to give* ***'4 spaces'*** *or* ***'1 Tab'.***

Example:

// Code in C language	# Code in python language

int a = 10;
if(a = = 10)
{
printf(" Value of a is 10 ");
}
else
{
printf(" Value of a is not 10 ");
}
	a = 10
	if a = = 10:
	print(" Value of a is 10. ")
	else:
	print(" Value of a is not 10. ")

Explanation:

Explanation of C code	***Explanation of python code***
1. After 'if' condition is given a '{' is placed, indicating the start of 'if' statement.	1. After 'if' condition, a semicolon (:) if given, indicating the start of 'if' statement..
2. Statement is written inside the 'if' condition. printf(" Value of a is 10 ")	2. Statement is written inside the 'if' condition. if a = = 10: print(" Value of a is 10 ") Here, you can see the print statement doesn't start from the level of 'if' condition. It has few spaces called indent.

3. To close the 'if' condition a '}' is placed.
.
	3. To close 'if' condition, the indent on next line is removed.
	if a = = 10:
	print(" Value of a is 10. ")
	else:
	Here, the 'else' condition starts from the level of 'if' condition.

4. Similarly an else statement is also written.	4. Similarly, the statement is also written in 'else' condition.

You can also add a block inside a block. In such cases, indentation of inner block will increase.

Example:

```
a =10;  b = 6
if a = = 10:
    print("a is equal 10")
    if b = = 6:
        print("b is equal to 6")
```

2 Comments

2.1 Single line-comments

- A comment is a piece of text to let someone know what is being done in a block of code.
- It doesn't affect the outcome of a code.
- In a Python, the symbol '#' indicates the start of a comment line. It is effective till the end of the line in the editor.
- If # is the first character of the line, then the entire line is a comment.

Example:

```
In [1]: # this is a comment starting
        # from first charater of the line
        print ("Hello World")
```

```
Hello World
```

Here you can see that the output is only "Hello World". The comments are not reflected in the output.

- We can also comment after a statement of the code

Example:

```
In [2]: print ("Hello World") #this is also a comment
```

```
Hello World
```

2.2 Multi-line Comments

- In Python, there is no provision to write multi-line comments, or a block comment. Each line should have the # symbol at the start to be marked as a comment.
- A triple quoted multi-line string is also treated as a comment (if it is not a docstring of a function or a class).

Example:

```
In [3]: '''
        this is a for
        commenting multiple lines
        '''
        print ("Hello World")
```

```
Hello World
```

3 Getting User's Input

- In some codes you might need to take inputs from the user.
- This is done with the help of input( ) function in python.
- The input taken with the input( ) function is in form of a string (you will learn about strings in future sections).

Syntax:

input(prompt)

Here, the prompt is the message that you want to display to the user. This message is to be written **between inverted quotes ("message")**.

Example 1:

```
In [*]: college = input("Enter your college name: ")
```

Here, "Enter your college name: " is the message/prompt, which will be visible to the user once the code is run.

Once the code is run, you'll get a block below the cell, with your message/prompt. You need to type your answer in that block.

```
In [*]: college = input("Enter your college name: ")
            Enter your college name: SVPCET
```

Then press Enter. You will get the following output.

```
In [3]: college = input("Enter your college name: ")
            Enter your college name: SVPCET
```

Here, the name of the college which you have entered is stored in the variable 'college'.
So if you print the variable 'college', you'll get the following output.

```
In [4]: print(college)
            SVPCET
```

Example 2:

Writing a code to input 2 numbers from user and perform addition on them.

```
In [ ]: a = int(input("Enter value for a: "))
        b = int(input("Enter value for b: "))
```

Note: *In the topic 'Getting user's input', we learnt that the input( )function always takes inputs in form of strings. Thus we need to convert it into integer before adding. The 'int' written before input( ) does this conversion for us. This is called 'typecasting'. You will learn about this in future sections.*

Run the code.

You'll be asked to enter the value for 'a'.
Enter the value as shown below & press 'Enter'.

```
In [*]: a = int(input("Enter value for a: "))
        b = int(input("Enter value for b: "))

        Enter value for a: 10
```

Next you'll be asked to enter the value for 'b'.
Enter the value as shown below & press 'Enter'.

```
In [*]: a = int(input("Enter value for a: "))
        b = int(input("Enter value for b: "))

        Enter value for a: 10

        Enter value for b: 3
```

```
In [7]: a = int(input("Enter value for a: "))
        b = int(input("Enter value for b: "))

        Enter value for a: 10
        Enter value for b: 3
```

You will get the following output.

Now we will add the values of 'a' & 'b' and print the result.

```
In [8]: c = a+b
        print(c)

        13
```

4 Keywords

- All computer programming languages comprise of a set of predefined words which are called keywords.
- These are predefined words, hence cannot be used for any other purpose, than for which it is defined.
- Some of the keywords which you might come across in this training are mentioned below.

and	**break**	**class**	**or**
continue	**def**	**del**	**pass**
elif	**else**	**if**	**return**
for	**while**	**import**	**from**
not	**False**	**True**	**in**

***Note**: All these are case sensitive, hence while using these keywords make sure that you use the correct casing.*

- If you want to know what exactly is the use these keywords, then you can use the following command on Jupyter Notebook.

Syntax:

help("name_of_the_keyword"**)**

Example:

```
In [5]: help("if")
```

Output:

```
The "if" statement
******************

The "if" statement is used for conditional execution:

   if_stmt ::= "if" expression ":" suite
               ( "elif" expression ":" suite )*
               ["else" ":" suite]
```

5 Displaying output

Any statement can be displayed in the output with the help of print( ) function in python. Different ways to use print( ) function is as shown below.

Example 1:

Printing a string.

```
In [4]: print("R-Labs")
```

Output 1:

```
R-Labs
```

Example 2:

Printing a value stored in a variable.

```
In [7]: college = "SVPCET"
        print(college)
```

Output 2:

```
SVPCET
```

Here "college" is the variable.

Example 3:

Printing multiple strings and variable values using single print.

```
In [8]: message = "print"
        print(message,'multiple values.')
```

Output 3:

```
print multiple values.
```

Example 4:

Printing multiple values separated by separators.

From Example 3, it is clear that the default separator is 'space'.
Thus to replace the space with a separator we use the parameter 'sep'

```
In [10]: num1 = 1
         num2 = 2
         letters = "abc"
         print(num1, num2, letters, sep=",")
```

Output 4:

```
1,2,abc
```

In this example, we have used comma (,) as the separator. You can use any symbol, alphabet, numbers as a separator

Example 5:
Using multiple print statements.

```
In [12]: num1 = 1
         num2 = 2
         letters = "abc"
         print(num1, num2)
         print(letters)
```

Output 5:

```
1 2
abc
```

Here you can see that all the elements of same print statement are displayed on the same line.

Example 6:
Display values in separate print functions on single line.

In the previous example you saw that, the elements of separate print functions were displayed of separate lines.
This is because the output of print function ends with a newline (equivalent to Enter).
Thus to do this we use the parameter 'end'.

```
In [13]: college = "SVPCET"
         print("College Name:", end=" ")
         print(college)
```

Output 6:

```
College Name: SVPCET
```

Video link:
https://drive.google.com/file/d/1APfEkslOgXLzGPtwmIo312oVaokeKcNR/view?usp=sharing

References:

1. https://www.w3schools.com/python/python_ref_keywords.asp
2. https://www.tutorialsteacher.com/python/python-basic-syntax
3. https://www.geeksforgeeks.org/statement-indentation-and-comment-in-python/