Lab 1: Introduction to Python

Python is a great language for beginners to learn programming, and also for advanced users! Support wide range of applications, from simple calculations, to text processing to Internet applications to games. It is well suited for doing *scientific* programs.

These notes are not comprehensive and you are encourage to learn Python using material available online.

Documentation: http://www.python.org/doc/

Overview: http://wiki.python.org/moin/BeginnersGuide/Overview

Useful site: https://www.learnpython.org/

For non-programmers:

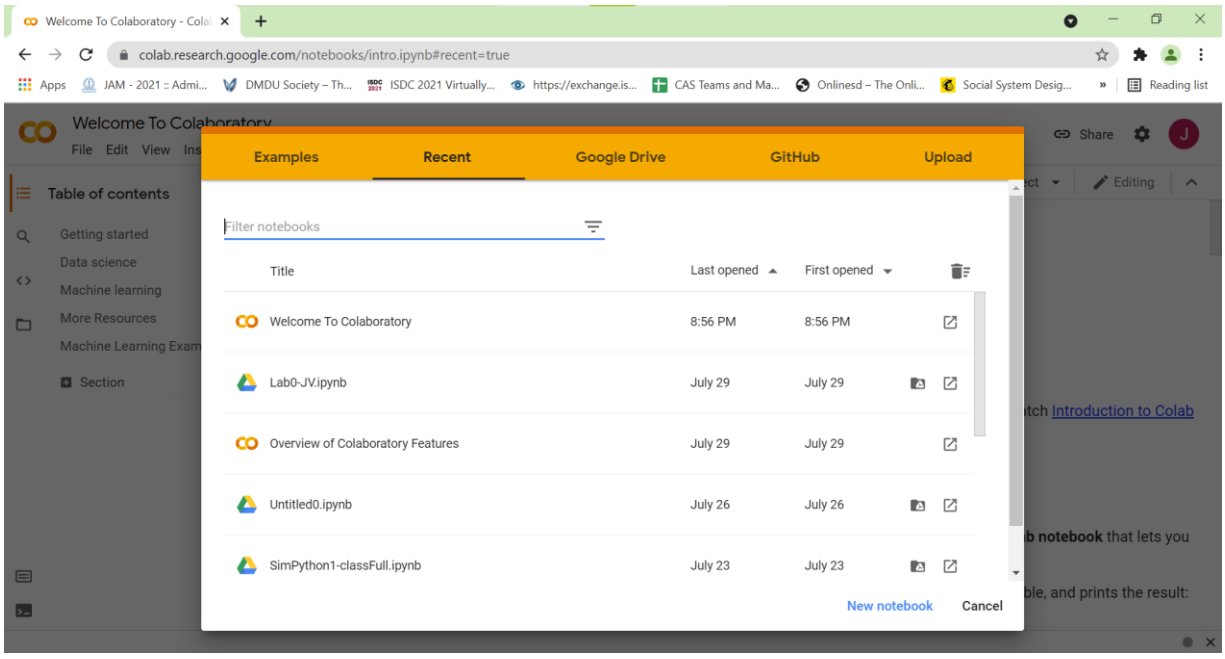
http://wiki.python.org/moin/BeginnersGuide/NonProgrammers

For programmers:

http://wiki.python.org/moin/BeginnersGuide/Programmers

Login to Google Colab https://colab.research.google.comusing IITB LDAP

You should get the window as shown in figure below.
Click New notebook at bottom of the window. It will open a blank python notebook.



Python notebook: "It is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media."

The following tutorial is designed for self-learning using Python 3.

Read the notes and instructions given. Whatever you see written against '>>>' in `Courier New font` type ONLY that in the Python Notebook Cell. The phrases inside square brackets [ ] are instructions you need to follow. Do NOT type them out.

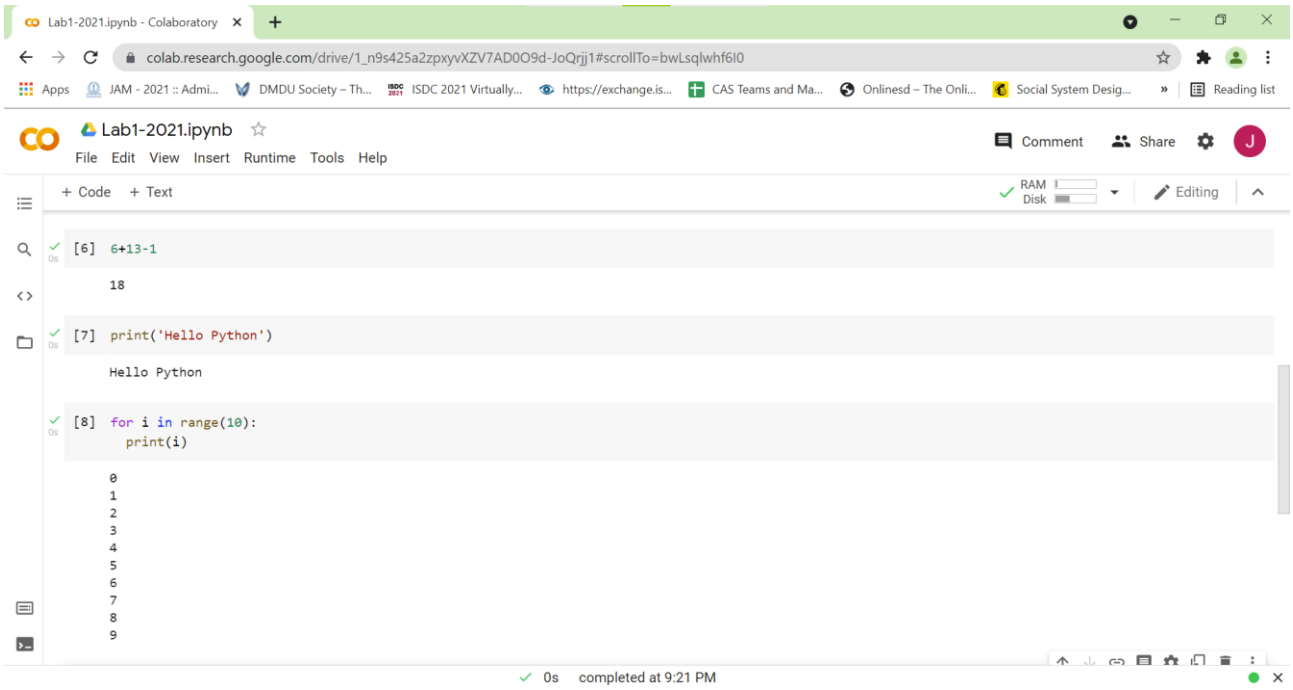
You execute the commands given, observe the output of your Python program and understand what the operation does.

Using the Python Google Colab GUI

In Python cell GUI (Graphical user Interface), type the following statements, one in each cell, and press SHIFT ENTER. (**DON'T type '>>>'**)

```
>>> 6+13-1      [Press shift enter]
>>> print('Hello IDLE')      [Press shift enter]
>>> for i in range(10):      [Press enter]
        print (i)                    [Press shift enter]
```

Your screen should look as shown.



Note: To run the code, you can press the Play button at the left of each cell; or press Control Enter; or press Shift Enter.

Remember, unlike C program, in Python there is no semi-colon ';' at the end of any statement. Indentations or TAB spaces are used instead of "{ }". Use only TAB to create indentations.

Chapter I: Variables and data types

1. Variables and data types

In a single Python Cell, type the following (**DON'T type '>>>'**)

```
>>> mymessage = 'I am going to learn Python'    [ Press Enter]
>>> n = 42                                      [ Press Enter]
>>> phi = 1.6180339                             [ Press SHIFT Enter]
```

In the statements above, the first assigns a *string* to a new variable named *mymessage* ; the second assigns the *integer* 42 to *n* ; the third assigns the approximate value of φ (a *decimal value or float*) to *phi*.

Notion of pointers: The variable names points to the data.

mymessage → 'I am going to learn Python'

n → 42

phi → 1.6180339

To display the values of the variables, use *print* function. Type following in new cell:

```
>>> print(mymessage)
>>> print(phi)
>>> print('the value of n is ', n)
```

Run the above statements, and view result.

The type of variable is the type of value it refers to:

```
>>> type(mymessage)                             [ Press Shift Enter]
<class 'str'>

>>> type(phi)                                   [ Press Shift Enter]
<class 'float'>
```

There are other built-in data types such as *complex, long, tuple, list, dictionary,...*

As a programmer choose meaningful names for your variables. Names can be long. For e.g., you can use `boiling_point_of_water = 100` instead of `t = 100`.

2. Syntax Errors, Variable names, and Keywords

Syntax errors are mistakes made while typing the code (similar to spelling mistakes). Such errors make it impossible for the interpreter (compiler) to understand what you have coded. This makes the code un-executable. However, to HELP you correct the error, the Python interpreter will display a message & highlight the error. You can read the message, and appropriately correct the code.

Let's try to make errors and correct them.

```
>>> x=2*pi                    [ Press Shift Enter]
NameError: name 'pi' is not defined
```

→ we have used *pi* without defining it

```
>>> Num=10          [ Press Enter]
>>> print(num)      [ Press Shift Enter]
NameError: name 'num' is not defined
```

→ Variable names are case sensitive. `Num` and `num` are different

```
>>> mystr= "IIT'              [ Press Shift Enter]
SyntaxError: EOL while scanning string literal
```

→ Strings to be enclosed using " " or ' '

```
>>> y=2+                      [ Press Shift Enter]
SyntaxError: invalid syntax    → the expression is incomplete
```

Also, if you give illegal variable name, you will get a syntax error:

```
>>> 10dulkar = 53             [ Press Shift Enter]
SyntaxError: invalid syntax
```

→ cannot use numbers at start of variable name

```
>>> dhoni@ = 100              [ Press Shift Enter]
SyntaxError: invalid syntax
```

→ @, $,&, etc cannot be part of variable name

```
>>> class = 'Kohli'           [ Press Shift Enter]
SyntaxError: invalid syntax
```

→ This is because, *class* is one of **Python's keywords.** They are predefined for use by Python and cannot be used as variable names.

View the all keywords by typing:

```
>>> help('keywords')
```

3. Operators

The mathematical operators +, -, *, / , **, and // perform addition, subtraction, multiplication, division, exponentiation (power), and floor division.
There are relation operators such: <, >, <=, >=, !=, and ==, representing strictly less than, strictly greater than, less than or equal to, greater than or equal to, not equal to and if equals.

Try the following and see if the results are you expect? Try the following

```
>>> 16 / 2 * 3 + 2          [Press SHIFT Enter]

>>> 3 * 16 / 2 + 2          [Press SHIFT Enter]

>>> 2**3                    [Press SHIFT enter]    (** computes square)

>>> 5*5+2**3-1              [Press SHIFT Enter]

>>> 5*5+2**(3-1)            [Press SHIFT Enter]
```

Rules of Precedence

When more than one operator appears in an expression, the order of evaluation depends on the rules of precedence. Python follows the same precedence rules for its mathematical operators that mathematics does.The precedence order is as follows:

( ) (anything in brackets is done first) ← ***Highest precedence***
** (exponentiation: Computes the square)
-x, +x
*, /, %, //
+, -
relational operators: <, >, <=, >=, !=, ==
logical not
logical and
logical or ← ***Lowest precedence***

NOTE:
1. Parentheses have the highest precedence and can be used to force an expression to evaluate in the order you want. Hence, `2*3+4` = 10 while `2*(3+4)` = 14.
2. Exponentiation has the next highest precedence, so `3*1**3` is 3 and not 27.
3. Multiplication and both division operators have the same precedence, which is higher than addition and subtraction, which also have the same precedence. So 2*3-1 yields 5 rather than 4, and 5-2*2 is 1, not 6.
4. Operators with the same precedence are evaluated from left-to-right. In algebra we say they are left-associative. So in the expression 5-3+2, the subtraction happens first, yielding 2. We then add 2 to get the result 4.

Let's see what kind of results we get in Python for various algebraic expressions

```
>>> 7/2
```
[Press SHIFT enter] → Gives 3.5. Note `7.0/2` also gives 3.5.

```
>>> 11%3
```
[Press SHIFT enter] → Gives 2, the remainder upon integer division

```
>>> 11//3
```
[Press SHIFT enter] → Gives 3, the quotient upon integer division.

Relational Operators: used to check if given statement is True or False

```
>>> 3**2 < 2**3
```
→ Gives *False*. The intermediate steps not shown.

```
>>> 2**2 >= 10-4/2*3-1
```
→ Gives *True*

Tasks to try
Try the following, and understand the results.

```
>>> -11%3
>>> -11//3
>>> 10-4/2*3-1
>>> 10-4/2*3**2-1
>>> 10-4/2*-3+-1

>>> x=3
>>> y = 3.0
>>> x == y
>>> x != y
```

4. String Operation (Mathematical operations cannot be done on strings).

```
>>> college = 'IIT'
>>> city = 'Bombay'
>>> college+city
```
[Press Shift Enter]

```
'IITBombay'
```
[Output; shown in RED]

→ The + operator on string, concatenates or joins strings together.

5. Print statements

```
>>> x=3
>>> print(3)
>>> print(x)
```
→ Both the above print statements will print 3

```
>>> print("value of x is ", x)
```
→ Whatever is written in parenthesis inside a print statement are simply reproduced as is. The *x* appearing outside the parenthesis after the comma indicate the variable *x*, and hence its value is printed).

```
>>> print(x , "is value of x here AND this is", 2*x)
```
→ Anything written outside the parenthesis are treated as variables and the value is printed. Expressions are computed first).

```
>>> print(x , 2*x, 3*x, 4*x)
3 6 9 12
```
→ Displays answer separated by spaces.

```
>>> print("when x=5, x = ", x )
when x=5, x = 3
```
→ This is the output. Whatever is written in parenthesis inside a print statement are simply reproduced as is. The *x* appearing after the comma indicate the variable *x*).

```
>>> y=x
```
→ Assigns value of *x* to *y*.

```
>>> print(x, y)
```
→ Should give 3 3 as output.

```
>>> x=4
>>> print(x, y)
```
→ Should give 4 3 as output.

```
>>> del x
```
→ The del variable x is deleted from memory.

```
>>> print(x)
```
→ What happens?

4. Comments in Python

We use hash (#) to indicate comments. Anything following # will be ignored by Python interpreter. For example:

```
>>> 32 * 12 #this is a sample comment
>>> 32 #* 12 #-->multiplication is not done due to #
```

Chapter II: Create a full program

Instead of using multiple cells, let's write entire code in a single cell.

In this **blank CELL**, type the following. After each line press ENTER only.

```
message = 'My first program in a single cell'
print(message)

a = 12
b = 7
print('a%b is ', a%b)
print('a//b is ', a//b)
print('a/b is ', a/b)
print('a**b is ', a**b)
```

Run the above code by pressing either SHIFT ENTER or CONTROL ENTER or the play button

If any errors are reported, try and correct it.

Tasks for submission

Answer the following questions in a SEPARATE NOTEBOOK (named as your rollnumber-Labnumber).

Instructions

The complete code for each question to be entered in a single cell only.
Each question to be answered in a separate cell.
After your complete the questions, download the files as .ipynb files and submit the same in Moodle. The ipynb file to be fully executed with all outputs visible.

Questions

1. Suppose you have the following variables:

```
length=15
width=7.0
myshape='rectangle'
```

Now, for the each of the following expression, determine the output, and the type of the value of the following expressions:

length / 3
width / 2
length / 3.0
length / width
(width+length) / length > length / width
myshape+myshape
myshape*3

2. Compute $1/2\pi$, where, where π=3.1415926. You need to define a variable to store value of π and then do the computation

3. Solve $6x^2 - 17x + 12 = 0$ using the quadratic formula.

4. If you run a 10 kilometer race in 43 minutes 30 seconds, what is your average time per km? What is your average speed in km per hour? (Hint: *speed = distance / time*)