Lab 1: Introduction to Linux and Python

Introduction to non-programmers

Operating System: An operating system (OS) is a collection of software that manages computer hardware (CPU, monitor etc) and provides common services (arithmetic, logic, execution etc.) for computer programs. There are three popular group of OS:

1. Windows (Proprietary): Windows XP, Vista, Windows 8 etc.
2. Linux/Unix (Open source/ Free): Ubuntu, Fedora, Redhat, Susie, Gentoo, Android
3. Apple (Proprietary): OS X

Files and File System: All our data/ information is stored in the computer as files. Files usually have an extension that identifies the type of file. For e.g.: *.pdf*, *.doc*, *.xls*, *.java*, *.c*, *.txt*, *.mp3*, *.avi*, *.mov*, etc. These files can be opened, edited, executed using appropriate computer program or software**.**
File system is an organized way to store the files within your computer. We can group the files into folders or directories

Software/ program: There are many software and programs already available to do a variety of tasks. For e.g. *Microsoft Word* and *Libre Office Writer* are software for creating text documents. *Calculator* is a program that does basic calculation, *MATLAB* or *SCILAB* software for doing numerical computations etc.

Log in to computer: Many computers will ask you for *username* and *password* to login to the computer. Contact the lab in-charge for getting your password.

For accessing computers in NSL LAB:
Username or login ID: *nsluser*
Password: *nsluser*

Use the above user id and password to login to your computer.

How to access files/ programs:

- You can simply open the application you want by selecting it using the on screen menu
  (Linux) Application >>
  (Windows) Start >> All Programs >> [Program name]
- For example (Linux):
  Application >> Accessories >> Calculator
  Application >> Internet >> Firefox Web Browser
  Application >> Office >> OpenOffice Word Processor
  Application >> Accessories >> Terminal

Local Machine: Local machine or the individual computer refers to the computer in front of you – the one you are using. You can install any software you want on your computer and run them without any network (or internet) connection

What is a Server? A server is a computer system that responds to requests from many computers across the network. From your computer, you can connect to a server and use the software there.

Passpoli Server: This is a computational server of IEOR. Various useful software packages are already installed in this server machine. Its IP address is 10.105.27.99.

In this lab course, we will connect to the server and use the software there. The advantage is that you can access all your files by connecting to the Server from ANY computer on campus.

To know more about the software available in optimus server, visit: http://10.105.27.47 (Please note: you need to add proxy exception or "No Proxy" for 10.*.*.* in internet explorer/chrome and 10.0.0.0/8 in mozilla firefox to access IITB internal servers with IP address).

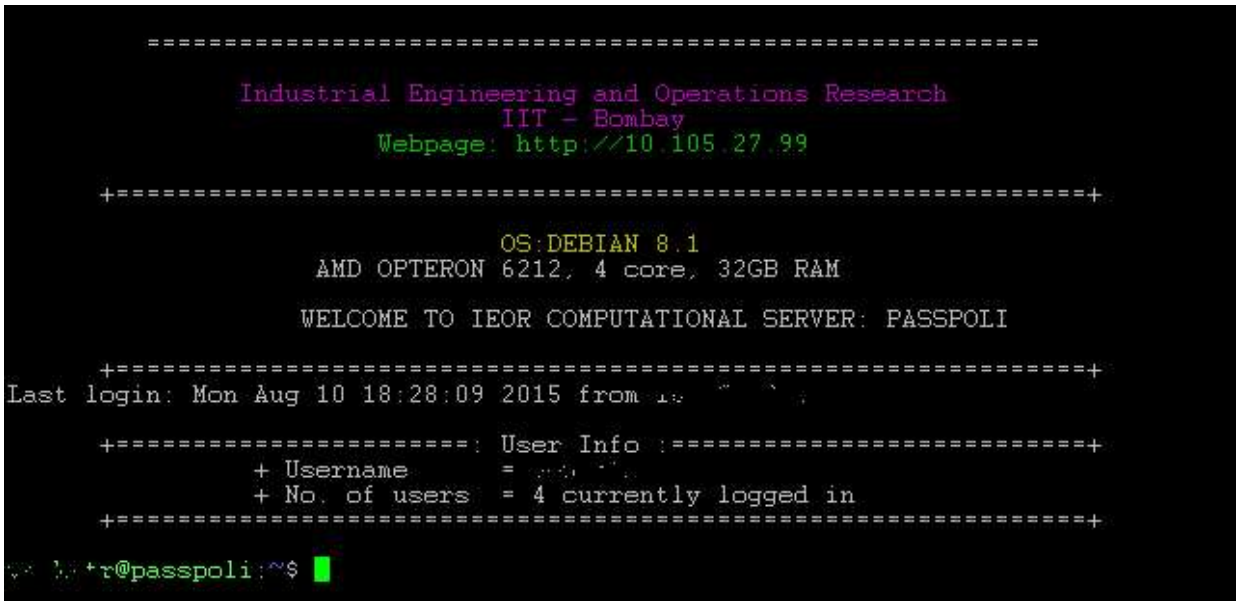
Introduction to Linux Server

A. Connect to Passpoli Server

1. Go to *Applications* --> *Accessories* --> *Terminal*
2. You will see a screen with command prompt $.
3. On that screen, type `$ ssh -X <yourLDAPid>@10.105.27.99`

   For e.g.: `$ ssh -X 10i190002@10.105.27.99`

   [Press *Enter*]
4. You would be prompted for a password: Enter you LDAP password.
5. You should get a screen as shown in figure. If so, CONGRATS. You are now logged on to IEOR Optimus Server. Whatever you now type on this command prompt $ will actually be executed in the server!



B. Basic unix/linux commands

1. Let's type the following at the command prompt at the server

   $ `date` [Press enter](*date* gives the current system date)

   $ `whoami` [Press enter](*whoami* returns your username)

   $ `who` [Press enter](*who* returns the list of all the people logged on to the server)

   $ `ls` [Press enter](*ls* lists the contents of your directory)

   $ `pico Welcome.txt` (Welcome.txt is a new text file that is now opened using a *Text Editor* program called PICO. You can edit the file). Let's try some editing:

   - Using arrow keys move your cursor (the small grey rectangle on the screen) to where it says <YOUR NAME>.
   - Delete <YOUR NAME> and insert your own name there.
   - Now, press 'Control-X'.
   - At the bottom of the screen there will be a message "Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?"
     Type `Y` [Press enter]
   - There will be another message "File name:" [Press Enter]
   - We have now successfully modified the file. We can check it.

$ `more Welcome.txt` (the updated Contents of the file is shown)

2. Now, let's create a separate folder or directory inside server to store the IE505 files

   $ `mkdir ie505` [Press enter](creates a new directory called ie505)

   $ `ls` [Press enter] (You should be able to see 'ie505' in the list)

   $ `cd ie505` (your working directory is changed to ie505)

A few other Linux Commands

- *ls –al* → list the contents of the current directory, with itmestamp etc. Syntax: $ `ls -al`
- *man* →it provides help on linux commands. Syntax: $ `man` *keyword.* For e.g.: $ `man ls`
- *mkdir* →make directory. Syntax: $ `mkdir` *`DirectoryName`*
- *cd* →change directory. Syntax: $ `cd` *`DirectoryName`*
- *cp* →copy a file to another location. Syntax: $ `cp` *`OriginalFile CopyFile`*
- *mv* →move a file to another location. Syntax: $ `mv` *`SourceFile DestinationFile`*
- *rm* →*r*emove/delete a file. Syntax: $ `rm` *`FileName`*
- *ssh* →secure shell is a network protocol for secure data communication, or command execution between two networked computers. Read more $ `man ssh`

Introduction to Python

Python Programming Language

- Created by Guido van Rossum in the late 80s, and early 90s.
- Open source general-purpose high-level language.
- Object Oriented, Procedural, Functional
- Easy to read. Uses English keywords, and has fewer syntactical constructions.
- Programs are compiled at runtime → You don't need to compile the program before executing it.
- Large number of libraries available
- Easy to interface with C/ObjC/Java/Fortran/C++ etc
- Great interactive environment

Python is a great language for beginner's. Support wide range of applications, from simple calculations, to text processing to Internet applications to games. It is well suited for doing 'scientific' programs.

Python official site: http://www.python.org
Download & Installation (Python 2.7.5 recommended):
http://www.python.org/download/
Documentation: http://www.python.org/doc/

FREE books, tutorials, videos & other learning materials:
Overview: http://wiki.python.org/moin/BeginnersGuide/Overview
For non-programmers:
http://wiki.python.org/moin/BeginnersGuide/NonProgrammers
For programmers: http://wiki.python.org/moin/BeginnersGuide/Programmers

It is expected that you will visit the above sites to learn Python further.

Chapter I: Running Python

There are three different ways to run Python

1. Using interactive command prompt interpreter
2. Script from command line
3. Using Integrated Development Environment

1. Running Python using Interactive Interpreter in Command Prompt

In the terminal window, type

```
$ python
```
[Press enter]

This launches an interactive command line interpreter for python, with the prompts '>>>'

We can interact with this interpreter and write programs directly.
Type the following:

```
>>> 6+13-1      [Press enter]
18          (The output of your 'program' is shown immediately)

>>> print('Hello Python')  [Press enter]
Hello Python        (print displays the results of the expression;  print is a reserved word)

>>> for i in range(10):    [Press enter]
        print i  [Press enter twice]
      (numbers from 0 to 9 should be printed)
```

Press 'Control-D' to exit python. Alternately you can type `exit()`

➔You have successfully done your first programming using Python!🡐

2. Script from Command Line
You should now be at the command prompt $.
Let's write our first program and store it in a file.
Type the following at the command prompt at the server

$ `cd ie505` [Press enter] (*cd* changes current directory to ie505)

$ `pico myfirstcode.py` (Let's create our first program and store it in file named *myfirstcode.py*).

- Inside the PICO editor, type

  ```
  print('Hello Python. This is your_name')
  for i in range(5):   [Press enter]
        print i          [Press enter. This statement to be indented
      using TAB]
  ```

- Now, press 'Control-X'.
- At the bottom of the screen there will be a message "Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?"
  Type `Y` [Press enter]
- There will be another message "File name:" [Press Enter]
- We have now successfully created your program file.

Let's execute our code:

```
$ python myfirstcode.py
```

You should get the following output

```
Hello Python. This is your_name
0
1
2
3
4
```

If you DON'T get the above output, contact the TA.

3. Using Integrated Development Environment

You can run Python from GUI environment such as IDLE, Anjuta, PyDev, Spyder, etc. In this course we shall use IDLE (Integrated DeveLopment Environment). It can run in Windows, Linux and Mac.

Type the following at the command prompt at the server

```
$ idle &
```
[Press enter]

This opens IDLE's GUI. We shall use this to do Python programming
IDLE window also has the prompts '>>>', very similar to python.
In IDLE you can do all the things you did with python. For example, try the following:

```
>>> 6+13-1
```
[Press enter]

```
>>> print('Hello IDLE')
```
[Press enter]

```
>>> for i in range(10):
```
[Press enter]

```
        print i
```
[Press enter twice]

Press 'Control-D' to exit IDLE. Alternately you can type `exit()`.

Chapter II: Variables and data types

Now, re-open IDLE from server command prompt.

1. Variables and data types

In the IDLE prompts '>>>', type the following

```
>>> mymessage = 'I am going to learn Python'
>>> n = 42
>>> phi = 1.6180339
```

In the statements above, the first assigns a *string* to a new variable named *mymessage* ; the second gives 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

```
>>> print(mymessage)
>>> print(phi)
```

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

```
>>> type(mymessage)
<class 'str'>
>>> type(n)
<class 'int'>
>>> type(phi)
<class 'float'>
```

There are other built-in data types such as *complex, long, tuple, list, dictionary*, which we shall learn later

As a programmer choose meaningful names for your variables. Names can be long. For e.g., 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.

```
>>> mystr= "IIT'
SyntaxError: EOL while scanning string literal
```

→ Strings to be enclosed using " " or ' '

```
>>> x=2*pi
NameError: name 'pi' is not defined
```

→ we have used *pi* without defining it

```
>>> x=2+
SyntaxError: invalid syntax
```

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

```
>>> 10dulkar = 53
SyntaxError: invalid syntax
```

→ cannot use numbers at start of variable names

```
>>> sachin@ = 53
SyntaxError: invalid syntax
```

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

```
>>> class = 'sachin tendulkar'
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.

Python has 30 keywords:

and	elif	if	print
as	else	import	raise
assert	except	in	return
break	exec	is	try
class	finally	lambda	while
continue	for	not	with
def	from	or	yield
del	global	pass	

You can view the 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

```
>>> 16 / 2 * 3 + 2
```
[Press enter]

result is?

```
>>> 3 * 16 / 2 + 2
```
[Press enter]

Is the result as expected?

```
>>> 2**3
```
[Press enter] (** Computes square)

```
>>> 5*5+2**3-1
```
[Press enter]

Is the result as expected?

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

Is the result as expected?

Rules of Precedence

Highest precedence

( ) (anything in brackets is done first)
** (exponentiation)
-x, +x
*, /, %, //
+, -
relational operators: <, >, <=, >=, !=, ==
logical not
logical and
logical or

Lowest precedence

Integer division vs. floating point division

```
>>> 7/2
```
[Press enter]

```
3
```

If you want to do floating point division, then one of the numbers must be written in decimal:

```
>>> 7.0/2
```
[Press enter]

```
3.5
```

Relational operators

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

String Operation (Mathematical operations cannot be done on strings_

```
>>> college = 'IIT'
>>> city = 'Bombay'
>>> college+city
'IITBombay'
```

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

Miscellaneous

```
>>> x=5
>>> y=x
>>> print(x/2.0, x, y)
>>> x=7
>>> print(x, y)
>>> del x          → The variable x is deleted from memory.
>>> print x        → What happens?
```

Chapter III: Creating .py files using IDLE

Instead of using outside editor, we can use IDLE editor to create and use python (.py) files.
In IDLE, using menu, Click FILE >> New Window

→ This should open a blank window

In this blank window, type the following:

```
message = 'My first .py program using IDLE'
print message
print ('This program outputs the first 20 numbers in
Fibonacci series')
a = 0
b = 1
maxcount = 20
for i in range(1, maxcount+1):
    print i, 'th number is ', a
    temp = a + b
    a = b
    b = temp
```

Click FILE >> Save. Give file name as fibone.py
Click RUN >> Run Module or press F5 button
The output should be displayed in the IDLE command prompt. If any errors are reported, try to correct it.

Class Exercise I

Answer the following questions using PYTHON / IDLE. For each question create separate .py file with your workings and solutions. After your complete the questions, show your work to the TA.

1. Suppose you execute the following statements:

```
>>> length=10
>>> width=6.0
>>> myshape='rectangle'
```

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

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

2. Compute $\frac{1}{2\pi}$, where π=3.1415926

3. Solve $2x^2 - 4x - 3 = 0$ 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*)

Python – Basic Syntax & Decision Making **Step 0.**

Type

`$ cd ie505` [Press enter – to go to *ie505* directory]

`$ idle &` [Press enter – To open IDLE, the Python environment]

Chapter 0: Recap

- Data types (*int, float, string, complex etc*) are implicitly declared by Python
- Variable names should not start with numbers or special characters.
- Variable names should not have 'space' in them.
- Python has many keywords that cannot be used as variable names.
- If we divide 2 integers, the output is also an integer
- Python programs can be saved as .py files.

Chapter I: Basic Syntax (contd from Lab 1)

1. Multi-line statements

Every line is a new statement. However, we can use \ to write statements spanning multiple lines.

```
>>> n = 32 + 456 - \   [Press Enter]
 3**2 - \      [Press Enter]
 1         [Press Enter]
>>> print n
```

The above lines of code should work correctly.

The code below should throw an error (it is incorrect):

```
>>> m = 32 + 456 - \   [Press Enter TWICE]
```

Does the statement below also give an error? Do you understand why?

```
>>> m = 4785 + 45 \    [Press Enter TWICE]
```

2. Quotations in Python

We use single quotes (') or double quotes (") or triple quotes (""") to enclose strings. For example:

```
>>> str1 = 'I know this'
>>> str2 = "This phrase is inside double quotes"
>>> str3 = """I can write a paragraph made up [Press Enter]
 of multiple lines using          [Press Enter]
 triple quotes"""    [Press Enter]
>>> str3
>>> str4 = "I can also write a paragraph \      [Press Enter]
 made up of multiple lines using \         [Press Enter]
 back-slash and single or double quotes"  [Press Enter]
>>> str4
```

3. 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 #
>>> #Use comments in programs to explain the code#
```

4. Data Type conversion

Sometimes we need to convert data from one type to another. For example, convert number to a string or a string to a number.

```
>>> n = 2103
>>> m = str(n)
>>> print m
```

>>> `n*n` [This should work]

>>> `m*m` [This should throw an error since *m* is string]

```
>>> i = int(n)
```

>>> `i*i` [This should work]

>>> `i/2` [This should work, does integer division]

>>> `i/2.0` [This should work, does floating point division]

>>> `float(m)/2` [This should work, converts *m* to floating point (decimal)]

Some other data type conversion functions:	
int(x [,base])	Converts *x* to an integer. base specifies the base if x is a string.
long(x [,base])	Converts *x* to a long integer. base specifies the base if x is a string.
float(x)	Converts *x* to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object *x* to a string representation.
chr(x)	Converts an integer to a character.
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

5. Getting User input

In IDLE, using menu, Click FILE >> New Window → Open a blank window

In this blank window, type the following:

```
print "This is an interactive program"
name = raw_input("What is your name? ")
print 'Hello ' + name
```

Click FILE >> Save. Give file name as *Lab2UserA.py*

Click RUN >> Run Module or press F5 button

In the IDLE command prompt, the following output should be displayed:

```
This is an interactive program
What is your name?
```
<Enter your Name and Press *Enter*>

```
Hello
```
<Name entered>

Now, go back to your *Lab2UserA.py* file. In the file, append the following lines:

```
x = raw_input("Enter any Number : ")
print x
print x-2
print x/3
```

Click FILE >> Save. Click RUN >> Run Module or press F5 button

What happens? Is *x-2* computed correctly? Is *x/3* computed correctly?

Python interpreter will thrown an error since all user inputs are taken as STRINGS BY DEFAULT. We need to convert it into a number if we want to do any numerical computations on it.

Now, go back to your *Lab2UserA.py* file. Modify program so that the full program looks as follows:

```
print "This is an interactive program"
name = raw_input("What is your name? ")
print 'Hello ' + name
x = raw_input("Enter any Number : ")
print x
print int(x)-2
print float(x)/3
```

Click FILE >> Save. Click RUN >> Run Module or press F5 button

What happens when you input an integer? Does it give the correct output now? What happens when you input a decimal number? Does it give the correct output now? → Can you identify the cause of the error? (contact TA if you cannot identify the error)