Lab 1: An Introduction to Linux and Python

CSE/IT 107

NMT Computer Science

1 Introduction

The purpose of this lab is to introduce you to the Linux environment (including some common, useful commands) and to the Python programming environment. In this lab, you will learn how to

1. Log into Linux at the Tech Computer Center (TCC).
2. Learn to use IDLE as an interpreter.
3. Edit, compile, and run a few short Python programs.
4. Create a tarball and submit a lab to Moodle.

Throughout this semester's labs, we will be using specially formatted text to aid in your understanding. For example,

```
text in a console font within a gray box
```

is text that either appears in the terminal or is to be typed into the terminal verbatim. Even the case is important, so be sure to keep it the same! If a $ is used that indicates the terminal prompt. You do not type the $.

```
Text that appears in console font within a framed box is
sample Python code or program output.
```

Text in a `simple console font`, without a frame or background, indicates the name of a file, a function, or some action you need to perform, but not necessarily type into the terminal. Don't worry: as you become familiar with both the terminal and Python, it will become obvious which is being described!

2 Instructions

The following instructions will guide you through this lab. Because this is an introductory lab, the instructions are rather detailed and specific. However, it is important that you understand the concepts behind the actions you take—you will be repeating them throughout the semester. If you have any questions, ask the instructor, teaching assistant, or lab assistant.

2.1 Log in

To log into a TCC machine, you must use your TCC username and password. If you do not have a TCC account, you must visit the TCC Office to have one activated.

1. First, sit down at a computer. If you're looking at some flavor of Linux (such as Fedora), skip forward to Step 6. If you're looking at Windows, press `Ctrl+Alt+Delete`.
2. Click on the `Shutdown...` button.
3. Select `Restart` from the menu and click `OK`. Wait for Windows to shutdown and for the computer to restart.
4. When the computer reboots, you will be presented with a menu to select the operating system you prefer. Select `Fedora` or the flavor of Linux installed on the computer you are using.
5. Wait while the Linux operating system boots.
6. When the login screen comes up, type your username and your password followed by `Enter`.

It should be noted that most of these instructions will work on Windows or Mac OS with Python installed, though we recommend using Linux simply to gain familiarity, as later Computer Science classes will require it.

2.2 Open IDLE

Throughout this semester, we will primarily be working with IDLE. IDLE is an integrated development environment included with every Python installation. In this lab we will cover how to use it as a calculator, as well as how to use it to write simple programs.

3 Python as a calculator

Forgot ever using your calculator! Python rocks as a calculator.

When you first open IDLE, you should see something like this

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Note the first line thing printed out: `Python 3.4.1`. This is the version of Python IDLE is running. Make sure you have Python 3 or higher. If you do not, check for another program called IDLE (or IDLE 3) or, if on your own computer, install Python 3.

At the Python prompt `>>>` you can type in Python statements. For instance, to add 2 + 3

```
>>> 2 + 3
5
>>>
```

Notice how after a statement is executed, you are given a prompt to enter another statement.

The `+` is the addition operator. Other basic math operators in python include:

`-` Subtraction: subtracts right hand operand from left hand operand

```
>>> 2 - 3
-1
```

`*` Multiplication: Multiplies values on either side of the operator

```
>>> 2 * 3
6
```

`/` Division, `//` Floor division - Divides left hand operand by right hand operand

Python 3

```
>>> 2 / 3
0.6666666666666666
>>> 2 // 3
0
>>> 2 / 3.0
0.6666666666666666
>>> 2 // 3.0
0.0
>>> 2. / 3.
0.6666666666666666
>>> 2 / float(3)
0.6666666666666666
```

This is an example of a major difference between Python 2 and Python 3. If you are using Python 2 some of these calculations will give different results. Make sure you are using Python 3.

Using `/` will always include the decimal portion of the answer. If you wish to drop the decimal portion, you must use `//`. Dropping the decimal portion of the answer is called integer division (since your answer will always be an integer) or floor division (since it will always round down).

`%` Modulus: Divides left hand operand by right hand operand and returns the remainder.

```
>>> 5 % 3
```

```
2
>>> 3 % 5
3
```

`**` Exponent: Performs exponential (power) calculation on operator. `a ** b` means a raised to b.

```
>>> 10 ** 5
100000
>>> 5 ** 10
9765625
>>> 9 ** .5
3.0
>>> 5.5 ** 6.8
108259.56770464421
```

Besides fractional powers, python can handle very large numbers. Try raising 10 to the 100th power (a googol)

```
>>> 10 ** 100
10000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000L
```

The `L` at the end of the number indicates it is a long. Long integers are integers that are too big to represent with a normal integer. In general you are able to treat integers and longs as equivalent data types.

Of course these commands can be combined to form more complex expressions. They follow typical precedence rules, with `**` having higher precedence than `* / % //`, which have higher precedence than `+ -`. If operators have equal precedence they execute left to right. You can always use parenthesis to change the precedence.

```
>>> 6 * 5 % 4 - 6 ** 7 + 80 / 3 #((6 * 5) % 4) - (6 ** 7) + (80 / 3)
-279908
>>> 6 * 5 % 4
2
>>> 6 * (5 % 4)
6
>>> (6 * 5) % 4
2
```

The `#` indicates the start of a comment. Comments are ignored by the interpreter and can be used to provide reminders to yourself as to what a section of code does.

The real power of python as a calculator comes when you add variables. Using `=` allows you to assign a value to a variable. A variable can have any name consisting of letters, numbers, and underscores. Once a variable has been assigned, it can be used in future computations.

```
>>> x = 7
>>> y = 3 * x ** 2 + 7 * x + 6
>>> y
202
```

In general you should try to give your variables descriptive names so that it is clear what a section of code does.

```
>>> food_price = 4.50
>>> drink_price = 1.00
>>> subtotal = food_price + drink_price
>>> tax = subtotal * 0.07
>>> total = subtotal + tax
>>> total
5.885
```

4 Running Python programs

So far, you have been running Python interactively. This is a nice feature as it lets you test ideas quickly and easily. However, there are times you want to save your ideas to a text file and to run the code as a program. Your programs will be almost identical to what you enter when using the Python interpreter, though you will have to use the `print` function in order to display output.

Open IDLE's text editor from `File > New File` or with `Ctrl + N` and create a one line Python program:

```
print('Hello, world!')
```

`print` is a function. It will print the value of whatever is within its parentheses. In order to print plaintext (not variables), you must additionally surround it with quotes. This is called a string. It is important to note that

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

is equivalent to the first example.

`\n` prints a newline. This means that the text following the `\n` will start on a new line. Test this with the following example.

```
print('Hello, world!\nHow are you?')
```

Save your file as `hello.py`. The `py` extension indicates it is a Python file.

To run the program, press `F5`.

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32
Type "copyright", "credits" or "license()" for more information.
>>>  ========================= RESTART ==========================
>>>
Hello, world!
```

```
How are you?
>>>
```

You can also print out the values of variable that you may have assigned earlier in the program.

```
x = 5
y = 4
print(y)
y = y + x
print(y)
```

```
>>>
4
5
```

But what if you wish to print a variable's value on the same line as another string? Unfortunately, the simplest method of doing this does not work:

```
>>> x = 5
>>> print('x = ' + x)
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    print('x = ' + x)
TypeError: Can't convert 'int' object to str implicitly
```

However, we can get around this by converting `x` into a string using `str()`.

```
>>> x = 5
>>> print('x = ' + str(x))
x = 5
```

At this point it is not necessary to completely understand the differences between a string and an actual number, just remember that `x = '5'` and `x = 5` denote different things.

5 Exercises

5.1 Conversions

Download `conversions.py`. It is a simple Python program for converting temperatures from Celsius to Kelvin. We would like to modify it to also convert the temperatures to Fahrenheit. Currently, the output of the program is something like this:

```
Please input a temperature in Celsius: 10
Kelvin temperature: 283.15
```

Add code (you should not need to modify the existing lines, though you are free to) so that its output is as follows:

```
Please input a temperature in Celsius: 10
Kelvin temperature: 283.15
Fahrenheit temperature: 50.0
```

6 Submitting

You may submit your code as either a tarball (instructions below) or as a .zip file. Either one should contain all files used in the exercises for this lab. The submitted file should end with either `.zip` or `.tar.gz` depending on which you used.

Tar is used much the same way that Zip is used in Windows: it combines many files and/or directories into a single file. Gzip is used in Linux to compress a single file, so the combination of Tar and Gzip do what Zip does. However, Tar deals with Gzip for you, so you will only need to learn and understand one command for zipping and extracting.

In the terminal (ensure you are in your `lab1` directory), type the following command, replacing `firstname` and `lastname` with your first and last names:

```
tar czvf cse107_firstname_lastname_lab1.tar.gz *.py
```

This creates the file `cse107_firstname_lastname_lab1.tar.gz` in the directory. The resulting archive, which includes every python file in your `lab1` directory, is called a tarball.

To check the contents of your tarball, run the following command:

```
tar tf cse107_firstname_lastname_lab1.tar.gz *.py
```

You should see a list of your Python source code files.

Upload your tarball or .zip file to Canvas.