CS 1: Introduction To Computer Programming, Fall 2015

Assignment 1: Getting started

Due: *Thursday, October 15, 02:00:00*

Coverage

This assignment covers the material in lectures 1 to 5.

Things you should have already done by now

In order to be able to submit this assignment (and any of the other assignments in this course), you should have already done several things by now. If you've been to the lectures and/or read the lecture slides and/or read the information on the course website, you should know this, but every year, a number of students apparently don't do those things, so we are repeating this here for clarity, because we know that if you read nothing else, you will read the assignments. **PLEASE DO NOT SKIP THIS SECTION!**

1. You should have enrolled in the course Moodle page. Apparently you did this, or you wouldn't be able to read this document. Well done! :-)

2. You should have a working CMS cluster computer account. Without this, you won't be able to submit your homework, whether or not you intend to use the CMS cluster computers. If you have an account from a previous term, you should have checked it to make sure that it still works (they get deactivated if you don't use them for an extended period) and if it doesn't work, you should have gone to see the system administrators in Annenberg room 112 between 9 and 5 to reset it and/or emailed them at help@cms.caltech.edu. (Going in person is better because they can fix it instantly.) If you have never had a CMS cluster account, you should have applied for one here and changed your password promptly (on a terminal or equivalent (*e.g.* PuTTy if you use Windows)) once you received email notice that the account was set up. **If you haven't changed your password, your account will deactivate in seven days, and you probably won't be able to submit your homework.**

3. You should have filled in the "Lab section/csman signup sheet" on the course website. Without this, we can't add you to the course csman page, which means that you will not be able to submit assignments. **THIS IS INCREDIBLY IMPORTANT!** Every year, a bunch of students forget to do this and then try to email their assignments to the TAs and/or the course instructor. **Emailed assignments do not count, and you will be penalized for lateness for every day between the due date and the date you upload your assignment to csman.** If you try to email your assignment to us, you will only annoy us, so don't do it! We don't care that you really did the assignment on time; if it isn't in csman by the due date, it's late and you will incur late penalties (0.5 marks/day).

4. You should have read the general documents on the course website regarding course policies, especially the pages entitled "Action items for new students", "Administrative information" and "Collaboration policies". The last one is particularly important, because if you violate the collaboration policies, you may get sent to the Caltech Board of Control, which is never pleasant.

If you have questions about general course policies, please refer to the "Course policies and frequently-asked questions (FAQs)" page, which will probably answer most of your questions.

5. If the assignment due date is almost here and you still aren't on the course csman page, you should have emailed the instructor explaining the situation. As long as you have successfully filled out the signup sheet, adding you to the course csman page is very easy. If you haven't filled in the signup sheet, you should do that first and then email the instructor.

If you ignore any of these steps (like, say, if you just skipped reading this section), then you are liable for all late penalties that may occur as a result of your negligence. **We will not waive late penalties for negligence on your part.** However, if there is a problem that is clearly not your fault, we will take that into account.

What to hand in and how to hand it in

What to hand in

You will hand in a single file called `lab1.py`. This will consist of Python code and comments. For this assignment, only part C has anything you need to hand in.

For the exercises where you are asked to evaluate some Python expression or answer a question, write the answer in a comment, along with the exercise number and subsection. For instance, if you are asked to evaluate `2 + 2` for part C, exercise 3, section 6, you would write:

```
# Ex C.3.6:  2 + 2 --> 4
```

You can use multiple lines if you like, but make sure each line is a comment.

For the exercises where you are asked to write some Python code, write the code with a preceding comment indicating the problem number (and subsection). For instance, if you were asked to write a function of one input which returns the input unchanged as the answer to part C, exercise 5, section 2, you could write:

```
# Ex C.5.2:
def identity(x):
    return x
```

The code itself should be uncommented, of course, because we will run it to make sure it works correctly.

How to submit your assignment

The CS 1 lab submission system is called "csman". The csman web site is at http://csman.cms.caltech.edu. The FAQ (Frequently Asked Questions) list has details of how to submit assignments. Once lab sections have been set up and you have been added to the course csman page, you can submit your assignment.

Time hints

Numbers in bold at the start of problems (*e.g.* **[10]**) are a crude estimate of the time (in minutes) the problem should take. If it seems like a problem is taking you disproportionately long to complete (*e.g.* if you are spending hours on a problem rated **[10]**), that is a hint that you might want to talk to a TA. These numbers assume that you have attended the lectures and done the portions of the assignment that precede it.

Parts A and B have no time hints, because there is nothing to hand in for those sections.

Part A: Getting started in the CMS computer lab

Location

The CMS computer lab is in room 104 of the Annenberg building. If you want access to the building (and the room) after hours (*i.e.* after 5 PM), and assuming that you are enrolled in the course, you can use your ID card to get in the building by holding it up to one of the card readers by one of the doors. There is also a card reader outside the computer lab. If you don't have card reader access, email your UID to the departmental secretary (Jeri Chittum, jeri@cms.caltech.edu) to get access (this may be necessary if you registered late). Note that the card readers sometimes malfunction, especially on cold evenings; our experience is that the one on the northwest corner of the building (the side entrance) is more reliable than the one on the north side (the main entrance).

The computer lab has a number of computers (hence the name). Collectively, they are known as the "CMS cluster" (or sometimes the "CS cluster"). Most of them are running the Linux operating system; specifically, a distribution of Linux called "OpenSUSE Linux". You will need to know a little bit about Linux in order to use the system. Hopefully most of you went to the Linux tutorial lecture, but either way you should review the slides from that lecture (posted on the CS 1 web site) before doing any serious work on the computers.

Logging in locally

Once you have a CMS cluster account, you can log in to any of the machines in the CMS cluster. Go to the CMS lab, find an unoccupied machine, and sit down. You will be presented with a login screen. In the top box (labelled with a "man" icon) enter your login name. In the bottom box (labelled with a "key" icon) enter your password. If you have done this correctly, you will see your "desktop". The default desktop is the GNOME desktop, which is pretty sparse, but if you click on the word "Activities" on the top bar to the left, some icons will come up, including icons for a terminal and a web browser. Click on the terminal icon to start a terminal. You will mainly be using the terminal for this course.

At any point, some of the computers in the CMS lab may not work properly. If you try to log in but fail, this may be the reason. Ask a TA to help. If you can, send email describing the problem to the system administrators at help@cms.caltech.edu. Please indicate the name of the machine that isn't working (which is written on a label on the front panel of each computer). Please specify what isn't working *e.g.* say things like *"I couldn't log in to this machine."* or *"There is smoke pouring out of the back of the computer and it's making a horrible grinding noise and people are scared."* but not just *"It's broken"*.

Logging in remotely

You can log in to the computer lab remotely, but don't expect to be able to do your homework this way — our system neither encourages this nor even supports it, and we don't want you to try to do your work this way. Nevertheless, logging in remotely is sometimes useful (*e.g.* for changing your password), so here's how.

To log in remotely, you have to have a terminal program running on your own computer. For Windows computers, we recommend the PuTTy program (which you will have to install); for Mac OS X, either Terminal.app or iTerm.app is fine, and if you use Linux, you should already know how to find the terminal program ;-)

You should log in to `login.cms.caltech.edu`. *DO NOT* type this into a web browser's URL bar! It isn't a web site, and that won't work. On a computer running Linux or Mac OS X, you should do this:

```
% ssh login.cms.caltech.edu
Password: <enter your password>
```

and you can proceed from there. (The `%` is not something you type; it's just the terminal prompt. Your terminal prompt may be different, but we'll use the `%` character when we want to indicate the prompt. When we tell you to type something at the terminal, never type the prompt.)

With PuTTy you will need to enter the hostname `login.cms.caltech.edu` in the configuration window that comes up when you start the program, and then click "Open" at the bottom. A terminal window will come up, and you'll be prompted for your login name and your password. Once this is entered, you will be logged in.

In all cases, type:

```
% exit
```

at the prompt to log out.

Using the terminal

From here on, we'll assume that you are working in the CMS lab. Some of the things we describe may not work if you are logged in remotely, which will hopefully deter you from trying to work that way. Most importantly, WingIDE (the Python development environment) will not work remotely.

Now would be a great time to review the slides from the Linux tutorial lecture if you haven't already. You will be using the terminal mainly to launch programs (*e.g.* WingIDE) and to manage files (create directories, move files between directories, etc.). You don't need to be an expert in Linux for this class; a minimal number of commands will take you a long way.

Changing your password

The very first thing you should do once you start up a terminal (assuming you haven't done this already) is to change your password. The password that was assigned to you when you got your account is only valid for a few days, and you're expected to change it after you log in. This is easy to do from a terminal using the `passwd` command:

```
% passwd
Changing password for user <your login name>.
Enter login(LDAP) password:
```

and you just enter the new password. You will need to enter it twice (to minimize the chances that you typed it wrong the first time). *Make sure you remember your new password!* If you forget it, you will have to go see the sysadmins and get them to straighten things out.

Once you have changed your password, the new password will be good for the entire term, so you won't have to do it again.

Starting WingIDE

To start up WingIDE, type this in the terminal:

```
% wing-101-5.1 &
```

and WingIDE will start up. The `&` is optional but recommended; what it does is allow you to subsequently enter more commands into the terminal. Without the `&`, you would have to wait until WingIDE exits before entering any more commands.

Tip: Just type in `wing` and hit the Tab key; it should type the rest of the program name for you. Then add an `&` at the end, press Return, and WingIDE will start up.

When you start up WingIDE for the first time, a window will come up with a license agreement. Click "Accept" and the main WingIDE window will appear. Make sure that the Python Shell in the lower right-hand corner says "Python 2.7.10" (or perhaps a slightly earlier version). If Python doesn't start up, something has gone wrong and you need to talk to a TA.

Working with WingIDE

The Python Shell

Now that WingIDE is running, we can start experimenting with Python. Let's start by working with the Python Shell in the lower right-hand corner. The first thing you should do is to type the following at the Python prompt (don't print the `>>>` prompt, though!):

```
>>> print "hello, world!"
```

If all goes well, Python should reply:

```
hello, world!
>>>
```

Woo hoo!

Note that Python not only printed the string you wanted it to print, but it also printed another prompt, indicating that it is ready for you to enter more commands. Try a few commands. (The commands are what follows the `>>>` prompt, and Python's expected response is printed on subsequent lines. Don't confuse the Python prompt with the terminal prompt; they aren't related.)

```
>>> 2 + 2
4

>>> 1.0 + 1.0 + 1.0/2.0 + 1.0/6.0 + 1.0/24.0 + 1.0/120.0 + 1.0/720.0 + 1.0/5040.0 + 1.0/40320.0
2.71827876984127

>>> 'foo' + 'bar'
'foobar'

>>> 'foo'.upper()
'FOO'

>>> '     this string has leading and trailing spaces    '.strip()
'this string has leading and trailing spaces'

>>> import math
>>> math.sqrt(2.0)
1.4142135623730951

>>> def func(x, y):
...     return x + y
...

>>> func(2, 3)
5
```

At this point, click on the "Options" button on the lower right side, and select the option "Restart Shell". Then try this:

```
>>> math.sqrt(2.0)
```

You should see something like this:

```
Traceback (most recent call last):
  File "<string>", line 1, in <fragment>
NameError: name 'math' is not defined
```

What this tells us is that when we restart the Python shell, all memory of the previous work we did is lost. Often this is what you want, but sometimes it isn't. This leads us naturally to the next topic.

The Editor

The majority of the WingIDE window is occupied by the top-central window. When you start up WingIDE for the first time, there is a tutorial there called "Introduction for New Users". Do yourself a favor: take a little time and read through this tutorial. It will familiarize you with the way that WingIDE works.

Once you're done with that, click on the "New" button near the top left-hand corner of the window. That will bring up a new tab on the editing window called "untitled-1.py". This is the Python source code editor. It is where you will be doing the majority of your work.

Type the following code into the editing window. (Don't try to cut-and-paste from the browser window; that probably won't work.)

```
def testme():
    print "this is a test"

def foo(x, y, z):
    return x + y + z
```

Notice how the code that you type changes color after you're finished typing it. This is what is called "syntax coloring", and most programming editors do it. It makes code much easier to read. Note also that WingIDE knows how to intelligently indent your code, so you don't have to do that yourself (at least, most of the time you don't).

You can immediately execute this code in the Python shell by hitting the "Run" button. Then, in the Python shell, you can do this:

```
>>> testme()
this is a test
>>> foo(2, 3, 4)
9
>>>
```

When you hit the "Run" button, it's as if you instantly typed the entire contents of the editor into the Python shell. This is often useful when debugging code (which is the process of removing errors from your code).

Saving your work

Much of the time, you will want to save the code you have been typing into a file so that you can continue working on it later. You will also need to save your work to a file in order to submit it for grading. Saving your work is easy: just click the `Save` button near the top (or use the `Save` command in the `File` menu item, or hit `Control-S`). A file browser will come up. If you click on `Save`, a file named `untitled-1.py` will be written to your home directory. Most of the time, though, you will want to rename the file to something else (which you can do in the dialog box at the bottom of the window), and you will probably want to save the file into another directory (which you can do by navigating there using the file browser). Let's rename the file to `foobar.py` and save it. You will note that the name of the file on the tab changes to `foobar.py`. Then, in the Python shell, do this:

```
>>> import foobar
>>> foobar.testme()
this is a test
```

```
>>> foobar.foo(2, 3, 4)
9
>>>
```

Congratulations! You have just written a Python module called `foobar`. The contents of this module are in a file called `foobar.py` which exists in one of your directories, and it can be reloaded later, re-edited, and submitted to the CS 1 grading system.

We *STRONGLY* recommend that you save your work often. Nothing is more frustrating than to work on a program for two hours without saving anything, and have all of that work wiped out because the computer crashed. You should get into the habit of saving your work automatically every ten minutes or so, or else after you've typed (say) ten new lines of code. The `Save` button will save your work to the file with the same name as the name on the tab; that's the one you need to use often. You can do this even more efficiently by typing `Control-S` *i.e.* the "control" key held down while you type the "s" key.

To exit WingIDE, go to the `File` menu and select `Quit`. When you restart WingIDE, the editor will contain the last thing you were working on, which is very convenient.

That's all for this section.

Part B: [*OPTIONAL*] Getting set up on your own computer

Most of you (hopefully *all* of you) own your own computer, and many of you will want to work on the assignments on your own computer in the comfort of your dorm, or use your computer in the CMS lab. Logging in remotely to the CMS cluster computer to do your homework is *strongly* discouraged; the system is not set up to do this and the system administrators may get annoyed at you if you try to do it (also, it's not easy to do anyway). The alternative is to set up the CS 1 software on your own computer and do your work there. This is *not* required, but it is convenient, and we encourage you to do it, because then you'll be able to work on your CS 1 homework wherever your laptop is.

In this assignment we will only show you how to set up Python and the WingIDE development environment. Subsequent assignments will also require extra Python packages; we will discuss how to set those up when the time comes.

We will be using Python version 2.7.10 in this course. Earlier versions will work for most of the course, but there may be occasional problems, so we recommend you install the correct version. Python versions 3.0 and later are *not* acceptable, because there are significant changes to the Python syntax in those versions which are incompatible with the lectures and assignments.

The version of the WingIDE Python development environment we will use is WingIDE 101 version 5.1.7-1. The "101" refers to a free version of WingIDE which can be downloaded from the WingWare web site.

What to do from here depends on which operating system (OS) your computer is running. Most computers run one of three operating systems: Windows, Mac OS X, or Linux. Which specific version of the operating system shouldn't make a difference as long as your computer is not too old. The actual software you need to set up is the same regardless of the operating system, but the details of setting it up are different.

Don't try to set up the CS 1 software on a tablet or *e.g.* a Chromebook. It almost certainly won't work, and you will waste a lot of time.

Windows

Most of you have computers that run some version of the Microsoft Windows operating system. Fortunately, getting the CS 1 software to work on Windows is not too difficult.

First, you will need to install Python 2.7.10. Download the installer for Python version 2.7.10 here. Click on the link to download it; it will most likely be saved in the directory "`My Documents/Downloads`". Use the Windows Explorer (file manager) to inspect that directory; there should be a file called "`Python-2.7.10`" which is a Windows Installer Package. Click on that file and the installation will begin. A dialog box will ask you if you want to run that file; click on the "Run" button. You will be presented with a series of question/answer dialog boxes. Selecting "Next" on each box is fine, except that you should install all the optional packages under "Customize Python 2.7.10" (except for the Test Suite, which you won't need). Click on each extension you want to install and select "Entire feature will be installed on local hard drive". Once you're done, click the "Next" button at the bottom and Python 2.7.10 will be installed.

Now you need to install WingIDE. You can get the Windows Installer Package here. Again, click on the link to download the installer, and again, use the file manager to inspect the "My Documents/Downloads" directory. You should see a file called "wingide-101-5.1.7-1". Click on it and it will begin the installation process. Select the default values for everything (except that you should agree to the license agreement!). Once you finish, WingIDE will start up and you will have to select "Accept" on another license agreement. Do so, and you will see the main WingIDE window. The Python Shell in the lower right-hand corner should say "Python 2.7.10". If not, something has gone wrong and you should ask your TA for help. If everything is working, exit WingIDE. You can start it up again by using the Start button to find the program, but we recommend you make a shortcut to WingIDE by dragging the menu item to the desktop (ask a TA if you don't know how to do this).

Now you're ready to go.

Mac OS X

Installing the CS 1 software on a Mac is almost as easy as installing it on Windows.

First, you need to install Python 2.7.10 (or a version close to it). Most Macs already have some version of Python installed, but it will likely not be the correct one, and even if it is, it may not include some of the libraries we will need. Therefore, we strongly recommend that you install the `ActiveState` version of Python, which is up-to-date and includes pre-installed libraries like `Tkinter` that we will be using a lot. ActiveState's version of Python can be found here. Once you install it, the new version of Python will be found in the `/usr/local/bin` directory. Note that this step will not uninstall the previous version of Python, which is found in the `/usr/bin` directory.

Then you need to install WingIDE. The installation package for WingIDE can be found here. Click on the link to download it to the "`Downloads`" directory. Navigate to that directory using the Finder and search for the file called "`wingide-101-5.1.7-1.dmg`". Click on it, and a new Finder window will come up with some icons, including one that says "`WingIDE.app`". Drag that icon to the Applications folder in order to install WingIDE. Once this is done, you can launch WingIDE by using the Finder to get into the Applications directory and clicking on WingIDE. You will need to agree to the WingIDE license, but after that you'll see the main WingIDE window. Then you have to configure Python. In the menu, select "Edit/Configure Python", and a new window will pop up. Select "Custom" for the Python executable. In the dialog box below this, type in "`/usr/local/bin/python`". Click "OK". Go to the Python Shell and click "Options/Restart Shell". Now it should say "Python 2.7.10" or something very similar (whatever version you installed previously). If this doesn't work, ask a TA for help.

Once all this is done, you're ready to go.

Linux

If you use Linux as your primary OS, congratulations, you rock! We like Linux too ;-) Unfortunately, it's difficult to give general instructions for installing the course software on Linux, as each Linux distribution has its own package manager and does things somewhat differently. Compounding the problem is the fact that nearly all Linux distributions already provide Python, but the version of Python provided will probably not be the one we will be using in this course. To determine this, start up a terminal and type

```
% python
```

at the terminal prompt (`%` here is the terminal prompt character; as usual, don't type that). If it prints:

```
Python 2.7.10 (... some other stuff ...)
(... some other stuff ...)
```

then you're fine (the `2.7.10` is the version of Python running). If not, you may need a different version of Python. Python versions that are slightly older than `2.7.10` are probably OK. If not, and your Linux distribution's package manager has a "`python2.7`" package, you can install that.

Once you have Python installed, it's time to install WingIDE 101. WingIDE 101 is available for download here. Download it and run the following commands:

```
% tar zxf wingide-101-5.1.7-1-x86_64-linux.tar.gz
% cd wingide-101-5.1.7-1-x86_64-linux
% sudo python ./wing-install.py
```

After this, just answer the questions the installer asks. Use the default install locations. Make sure your `PATH` environment variable (set *e.g.* in your `.bashrc` file) includes `/usr/local/bin`; if you don't know how to do this, ask a TA.

If you've done all this correctly, you can then start WingIDE by typing

```
% wing-101-5.1 &
```

and you're ready to go. (The `&` is not required, but if you type it you will be able to enter terminal commands while WingIDE is running.)

The first time you run WingIDE you will have to accept the license of the software; click "Accept" and you won't have to do it again.

Once WingIDE is running, look at the Python shell in the lower right-hand corner. If the Python version is 2.7.10 or something close to it, you are done. Otherwise (and assuming you've installed Python version 2.7.10 somewhere on your system), you'll have to tell WingIDE where the correct version of Python is located. Do this by clicking on Edit/Configure Python, which will bring up a new window. Set "Python Executable" to "Custom". Then enter the location of the correct Python version in the dialog box immediately below. After this is done, click "OK", restart the program, and the Python version in the Python shell should be the correct one. If you have a problem with this, ask your TA for help.

Any other OS

If your computer runs an operating system other than Windows, Mac OS X, or Linux, you are on your own. You can ask a TA for help, but we recommend that you try to obtain a computer that runs one of the three major OSs unless you're willing to invest the time to figure everything out for yourself.

Part B2: [*OPTIONAL*] Setting up a Linux virtual machine

Some students may want to set up their own Linux installation. **This is completely optional!** You do not need to do this, but if you want to, these instructions will help you.

There are a number of ways to set up a Linux computer:

- You could buy a new desktop or laptop computer and install Linux as the only operating system on that computer.
- You could install Linux alongside whichever operating system (Windows or Mac OS X) is already on an existing computer. This is called "dual-booting".
- You could set up a Linux virtual machine (VM) on your existing operating system.

How do to either of the first two approaches is beyond the scope of this course. However, setting up a Linux VM isn't very hard, so if you want to try that, this section will walk you through the process. Once again, **you don't have to do this!** It's strictly for those students who are interested enough/motivated enough to want to do it. You also don't get any extra marks for doing it, but it's a worthwhile learning experience. Setting up a Linux VM is quite an involved process, but it isn't hard; basically, you just need to be able to read carefully and follow directions.

One thing must be made crystal-clear at this point. **These instructions assume that you will be installing a VM on your own computer. Do not, under any circumstances, attempt to install a VM on one of the CMS cluster computers!** This is not supported and will probably make the system administrators very angry, because it will waste a *lot* of disk space.

The other caveat is that setting up a Linux VM from scratch is pretty time-consuming. Don't even bother unless you have at least a couple of hours to spare.

Hardware requirements

In order for this to work, you should have a laptop (or desktop) computer running either Microsoft Windows, Mac OS X, or some version of the Linux operating system. Ideally your computer and its operating system are not too old (say, not more than 2 years old) and the computer has at least 2 gigabytes of RAM (random access memory). You should also have at least 20 gigabytes of free space on your computer's hard disk. Note that Chromebooks (at least, most Chromebooks) are not viable for installing virtual machines. If you don't know whether your laptop's specs are good enough, you need to find out – all programmers should know what operating system they are running and how much RAM and disk space they have on their computers. If your computer's specs aren't good enough to install the VM, you should consider getting a better laptop!

Downloading and installing VirtualBox

You will be setting up a virtual machine, and in order to do this you will need software that manages and runs virtual machines. A virtual machine (at least in this context) is an operating system that runs in software (hence "virtual") inside another operating system. The outer ("host") operating system (which is the operating system of your computer, running directly on the computer's hardware) effectively simulates the hardware that the inner ("guest") operating system (running in software) sees. In fact, the guest operating system is not aware that it is being run as a virtual machine, though it will probably run more slowly than it would if it were running on bare hardware.

The software we will use to run our VM is called **VirtualBox**. You should go to this page and download and install whichever version of VirtualBox runs on your operating system. For instance, if your laptop runs Microsoft Windows, you should download the "VirtualBox X.X for Windows Hosts" installer (where "X.X" is the version number; *e.g.* 5.0). If your laptop runs Mac OS X, download the version for OS X hosts. If your laptop runs Linux, you should install VirtualBox using the package manager of your distribution; install the "virtualbox" packages but not the "virtualbox-guest" packages. Ask the TAs if you're not sure how to do this, but if you're already a Linux user, you probably already know how to do this. (Also, why would you need to install a Linux VM if you already have Linux installed?)

You may notice that some of the download links specify "x86" while others specify "AMD64". These refer to 32-bit and 64-bit operating systems, respectively. Nowadays almost all operating systems are 64-bit, and in fact only the Windows installer supports 32-bit systems (and it also supports 64-bit systems).

Once you've downloaded the installer for your operating system, just click on it to install VirtualBox (unless your host operating system is Linux; see above). Both Windows and OS X will generally give you some grief about installing software from an untrusted source, and you may need to change some settings to allow this. Ask the TAs if you need help doing this. Other than that, the installation should go smoothly.

Downloading Ubuntu Linux MATE

In order to create a VM running Linux, you will need the Linux software. There are literally dozens of Linux versions (also known as Linux distributions or "flavors") that can be installed. We are choosing one called Ubuntu MATE (pronounced "mah-tay" after the caffeinated South American beverage) for a number of reasons. It's stable, highly configurable, light on resources and uses the Ubuntu Linux software repositories, which are among the largest and best-maintained repositories of any Linux distribution. In addtion, Ubuntu MATE is easy for beginning Linux users to pick up, but it's also suitable for more advanced users.

You may be tempted to use another version of Ubuntu Linux, or another version of Linux entirely. **Please do not do so.** There are a number of different "spins" (variants) of Ubuntu Linux, but some of them (including standard "stock" Ubuntu as well as Kubuntu) are not suitable for use in a VM because they are too demanding in terms of 3D graphics acceleration support. Put simply, if you do this, your VM may not be responsive and this will make you angry and frustrated. Other spins, like Xubuntu and Lubuntu, are quite suitable for VMs, but some of the instructions below would need to be modified for those spins, so if you use these don't expect the TAs to be able to help you in case something goes wrong. The same considerations apply for non-Ubuntu flavors of Linux; they probably work fine, but we don't have the same level of expertise with them, so if a problem comes up, you are on your own.

You need to download a "disk image" of the most current version of Ubuntu Linux MATE, which as of this writing is version 15.04. It can be downloaded from this page. We recommend the "HTTP direct download" link. (You can use the "Torrent" download link if you know how to use BitTorrent clients.) Choose the "64-bit PC (AMD64)" version, unless your computer is a 32-bit computer, in which case you'll have to make do with the 32-bit version. (However, if your computer is a 32-bit computer, you really need a more modern computer!) Click on the link and save the file to your disk. **IMPORTANT: This file is quite large (more than 1 gigabyte), so make sure that your computer is plugged in to an AC adapter when downloading!** Also, we **strongly** recommend that if possible you use a wired Ethernet connector to do the download if at all possible, or else use the fastest wireless network you can find. The "Caltech Guest" Wi-Fi is **extremely** slow and not suitable for large downloads, so don't use that. If all else fails, you can use a flash drive to copy the disk image from a friend who has downloaded it successfully. The file (for 64-bit computers) should be called `ubuntu-mate-15.04-desktop-amd64.iso`. **If the file is not downloaded completely, nothing else in this section will work and you will have to start over.** Finally, remember what directory you downloaded the file to, because you will need this information for the next step. Most operating systems put a `Downloads` directory under each user's home directory where downloaded files go by default.

Installing Ubuntu Linux MATE

OK, now we get to the interesting part: creating a new virtual machine which runs Linux!

Preparing the virtual disk

1. First, start up VirtualBox. (You should be able to find it in the program menus, or you can use the terminal command line if you're using Linux (the program name is `virtualbox`.) A smallish window should pop up.

2. Click on the `New` button at the top left to create a new virtual machine. A new window called `Name and operating system` will pop up with information you need to fill in. Under the `Name` entry enter `Ubuntu MATE 15.04`. VirtualBox will automatically select `Linux` in the `Type` entry and `Ubuntu (64 bit)` in the `Version` entry (or `Ubuntu (32 bit)` if you have a 32-bit computer). Click `Next`.

3. The window title changes to `Memory size`. Select `1024` (megabytes, or 1 gigabyte). This is the minimum amount needed for things to work reasonably smoothly. In a pinch, you might get by with `512` (half a gigabyte) but using some programs (*e.g.* a web browser) from inside the VM will be difficult. If you have lots of memory, feel free to specify more (*e.g.* 2048 megabytes/2 gigabytes, or even 4096 megabytes/4 gigabytes; more than this will not help you). Click `Next`.

4. The window title changes to `Hard drive`. Select `Create a virtual hard drive now`. Click `Create`.

5. The window title changes to `Hard drive file type`. Select `VDI (VirtualBox Disk Image)`. Click `Next`.

6. The window title changes to `Storage on physical hard drive`. Select `Dynamically allocated`. Click `Next`.

7. The window title changes to `File location and size`. Leave the upper entry alone; it should say `Ubuntu MATE 15.04`. For the size enter `20 GB` and click `Create`. This should be more than enough disk space for everything you are going to do in this course. If you have a lot of disk space and intend to use this VM for other courses as well, you can make it larger, but you won't be able to change it later. Because it's dynamically allocated, it won't use up the 20 gigabytes right away; 20 gigabytes is simply the maximum size the disk can grow to.

At this point, the window you've been entering information into should go away, and there should be a new VM entry displayed in the main VirtualBox window that says `Ubuntu MATE 15.04` and below that `Powered Off`.

Congratulations! You've just finished the first part of installing the Linux VM! Unfortunately, although you've created storage for your VM (the 20 gigabyte virtual hard disk you just created), there isn't anything in it yet, so we have to install Linux onto it.

Configuring the VM settings

If you've done everything correctly so far, now you will be looking at the main VirtualBox window, but it will now have a new entry on the left for your new VM. At this point we need to adjust some settings.

1. Select the entry for your new VM by clicking on it.

2. Click the `Settings` button at the top left of the window. A new window will pop up with menu entries at the left (Windows, Linux) or on top (Mac). Each one selects a particular set of options.

3. Click on the `System` entry. Under `Motherboard`, you will see an entry called `Boot Order`. Uncheck the `Floppy` option. To be extra safe, you can use the arrow keys to the right of the boot order list to move the `Floppy` option down as low as it can go. (I have no idea why this option even exists.) At this point, the topmost entry in `Boot Order` should be `CD/DVD` and it should be checked.

4. Click on the `Display` entry. Change `Video Memory` to the maximum value (128 megabytes). Also make sure that `Enable 3D Acceleration` is checked but `Enable 2D Video Acceleration` is unchecked (they are mutually exclusive).

5. Click on the `Storage` entry. Under `Storage Tree` you will see an icon that looks like a disk and that says `Empty`. Click on it. This is the virtual CD/DVD drive. We are going to be loading it with the contents of the Linux file we downloaded earlier (the one whose name ended in `.iso`). To the right there is an empty checkbox titled `Live CD/DVD`. Check it. Then at the right edge there is another disk icon. Click on it and select `Choose a virtual CD/DVD disk file`. This will bring up a file browser window. You will need to

navigate to the directory on your computer in which the Linux `.iso` file is located and select it. Remember, it will be called `ubuntu-mate-15.04-desktop-amd64.iso`. Once you've selected it, click `Open`. Then click `OK` in the previous window.

That's the end of the second phase of installation. We're now ready to actually load Linux onto our virtual disk drive and create a working system.

Installing the operating system

At this point, VirtualBox knows where the Linux software you downloaded is located and can start up a working Linux system. Click on the `Start` button on top. This will bring up a new window which will contain the entire Linux desktop. The window is small, but we'll make it bigger later. You will see some garbage and then the words `Ubuntu MATE` as the system boots up. There may be some warning messages as well; ignore them. Finally you'll see a screen containing a pretty graphic and (again) the words `Ubuntu MATE`. Then the installer will start. It will ask you if you want to try Ubuntu MATE or install it.

Now would be a good time to mention one annoying thing about VirtualBox. It has a tendency to "capture" the mouse, which means that when you click on the mouse inside a VM window, the mouse pointer won't exit it (or it will but you'll still see a duplicate pointer inside the window). There is a key you can press that will "release" the mouse and let you use applications outside of the VM. This is called the "Host key". On the Mac the Host key is the left Command key, and on Windows and Linux it's the right Control key (you can change the key binding if you really hate it). It's also listed in the lower-right corner of the window the VM is in. You should practice tapping the Host key in order to get out of the VM (ask a TA if you are having problems with this). Clicking inside the VM window puts you in the VM again.

Select `Install Ubuntu MATE`. This will start the installation process. What this involves is mainly copying a lot of software from the file you downloaded (the `.iso` file) onto the virtual disk. You'll also have to answer a few questions.

1. The installation window title changes to `Preparing to install Ubuntu MATE`. Check the box labeled `Download updates while installing`. If you like, you can also check the box labeled `Install this third-party software`. This will install proprietary software that is necessary for doing certain things (for instance, watching Flash videos). However, it shouldn't be necessary to install this if you don't want to. Then click `Continue`.

2. The window title becomes `Installation type`. Select `Erase disk and install Ubuntu MATE`. **Don't panic!** This will **not** erase **anything** on your computer's hard drive! The "disk" it will be erasing is the virtual disk you created in a previous step (which is actually already empty). As you see, there are other options too, but none of them are important for us, so just go ahead and erase the entire (virtual) disk and install Linux by clicking `Install Now`. A confirmation window will come up. Click `Continue`.

3. The system will then ask you where you are, and preselect "Los Angeles". Click `Continue`.

4. The system will ask you for your preferred keyboard layout, and preselect "English (US)". Click `Continue`.

5. The window title becomes `Who are you?`. Enter your name, a name for your computer (which can be anything you like), a username (which might as well be the same as your CMS cluster username), and a password (which can be the same or different, as you like). Type in your password a second time where indicated. **Make sure that you don't forget your password!** Click `Require my password to log in` and then click `Continue`.

6. At this point, the system will show you a slide show about all the cool things included in Ubuntu MATE while it copies files to your system. Be patient; this step takes a while (a *long* while if you have a very slow internet connection). The progress bar will let you know how far along you are.

7. Once this is done, the virtual machine will tell you that it needs to restart to complete the installation, and ask you if it can. **Do not accept!** If you do, you'll get the same screen you started with (the one asking if you want to install Ubuntu or just try it.). The problem is that the boot order you selected above will cause the system to boot from the CD/DVD drive if possible, and here it is possible because you've loaded the virtual CD/DVD drive with the Linux installation CD (actually the `.iso` file, but the system can't tell the difference). Instead, you have to power down the virtual machine by clicking on the close button of the VM window (which will either be in the upper left or upper right corner, depending on your operating system). VirtualBox will give you some shutdown options, and you should select `Power down the machine`. Click `OK` and the VM will exit, closing the VM window in the process.

8. Now you have to adjust a couple of settings for the VM. Click the `Settings` button and select the `System` menu entry. Under `Boot Order`, uncheck the `CD/DVD` option and move it down below `Hard Disk`; make sure `Hard Disk` is checked. Then click the `Storage` menu entry and click on the disk icon marked `ubuntu-mate-15.04-desktop-amd64.iso`. Uncheck the `Live CD/DVD` checkbox and click on the disk icon above it and to its right. This will bring up a pull-down menu. Scroll down to the bottom and select `Remove disk from virtual drive`. Click `OK`.

Now your Linux system is ready to use without needing the installation `.iso` file. Start it up again and we'll continue setting up.

Updating your system

When you start the VM, you should now see a login window with your name preselected. Enter your password and hit return to log in to the VM.

At this point, you might think you're done, but there are still a few steps to go. First, you need to update your system. Most Linux distributions, including Ubuntu, are continually being updated, and it's important to make sure your system is up-to-date, because updates contain bug fixes and fix security holes. There are actually a few different ways to update your system. When you log in, if you need updates (and in this case you do), a graphical program called `Software Updater` wlll already be running and will tell you that you need to update your system. (The window may be minimized; if so, click on the entry in the task bar at the bottom to bring it up.) Click on `Install now` to update the system. A window will pop up asking you for your password. You should enter this and then hit the `Authenticate` button, and then the system will start updating. **Warning: this will take a while!** If you like using a graphical program to update the system, this is always an option.

The other option is to update the system from the terminal command line. This is actually just as easy and very powerful once you know what you're doing. To do this, first start up a terminal by hitting the key combination `Control+Alt+t` (*i.e.* hold down the `Control` and `Alt` keys while pressing and releasing the `t` key). Then enter the following commands (the `%` is the terminal prompt; don't enter that):

```
% sudo apt update
[enter your password]
% sudo apt upgrade
```

You should only have to enter your password once. After `sudo apt update`, a bunch of stuff will be printed in the terminal, none of which you need to pay attention to. After `sudo apt upgrade`, the system will list all the packages that can be upgraded and offer you a yes/no choice as to whether to proceed or not. Hit the return key (which selects yes) to start updating. Also note that the terminal prompt is more complicated than just `%`; by default it also contains your username, the computer's name and the current directory. We will always use `%` in our examples, but never type the `%` into the terminal! It's possible to change the prompt to whatever you like; see any Linux tutorial for more on this.

Note that you shouldn't try to simultaneously update using the graphical updater and the terminal updater, as whichever one is invoked first will prevent the other one from running.

Once you're done, you need to restart the VM. If you used the graphical software updater program, a window will come up asking if you want to restart now or later; press `Restart now` and the system will restart. Otherwise, click on the button on the top right hand corner of the VM window (on the top panel), which will bring up a shutdown/restart window; select `Restart`. After the system restarts, enter your password to log back in.

Once this is complete, you have an up-to-date Linux system running on your VM. But wait, there's more!

Installing the VirtualBox guest utils

One annoying thing about running a Linux VM is that by default the VM window is much smaller than the computer's monitor. It would be nice if the VM took over the entire monitor display, so it really looked like you were running a different operating system. You can do this by installing the "VirtualBox guest utils". These are a set of programs that give the system a number of new abilities, one of which is to use the full screen. There is more than one way to install the guest utils, but we'll do it from the terminal command line. Bring up a terminal in the VM, and then type the following command:

```
% sudo apt install build-essential dkms m4 virtualbox-guest-dkms
```

You may be asked to enter your password; if so, do so. Then the system will ask if you really want to install all those packages; hit Return to accept. The packages will be installed.

At this point, you need to restart the VM for the changes to take effect. Use the button on the upper-right corner of the VM window to do this. Once it restarts, press Host+f (press the Host key and the `f` key simultaneously) and the window should go to full screen. You should do this when you reach the login screen but before entering your password. The VM may pop up a window telling you that you have just hit the Host+f key (duh!); you should check the checkbox telling it not to show that window again.

Note that the full screen capability may not kick in until you have logged in, but once it does you'll be able to log out and log back in, and it should stay full screen. This is much nicer to work with. Also, when you are in full-screen mode, you can still get out of the VM by hitting the Host key and then doing Alt-tab (Command-tab on Mac OS X) to select a different program. Then just Alt-tab back to the VM window to get back into it.

Some simple customizations and stuff to know

Now you have a fully-functional and up-to-date Linux system installed as a virtual machine. Yay!

You will see a panel on the top of the screen with a few panel "applets" on it to control things like volume, to show the date, to bring up the shutdown/restart window, *etc.*; these are on the right-hand side. On the left-hand side there are menus that allow you to select programs, as well as a launcher that will bring up the Firefox web browser.

Under the `System` menu there is an entry called `Control Center` which you should bring up. This is where most of the configuration of the system happens. Click on that and look around. It's similar to analogous programs on Mac OS X and Windows. One thing to do now is to click on `Appearance` and then `Background` if you want to change the desktop background. There are a number of nice wallpaper images to choose from, or you can choose a simple color (my preference is black) or a color gradient. Another thing to try is `MATE Tweak` which will allow you to change still other settings. I like to hide the `Home` icon on the desktop, which you can do here. There are other options here too; feel free to experiment. Another customization you can do from the Control Center is to swap the Control and Caps Lock keys, which I find very helpful. To do this, click on the `Keyboard` settings, then `Layouts` and `Options` (at the bottom of the window). This brings up a window with a bunch of options. Select `Ctrl key position` and scroll down until you see `Swap Ctrl and Caps Lock` which you should check if you want that behavior. This is all we'll be doing with the Control Center for now.

You will be working mainly in the terminal, so you need to be able to launch terminals quickly. You can always use the `Control-Alt-t` key sequence, but some people prefer having a launcher on the top bar similar to what is there for the Firefox web browser. To create such a launcher, find the terminal menu entry under the menu at the top left: `Applications/System Tools/MATE Terminal`. If you just select this option, a terminal will come up. Instead of selecting it, drag the terminal icon with the mouse onto the top panel. This will create a new launcher icon. Whenever you click on this, a new terminal will be created. Also, every time you log in from now on, that terminal launcher will be there.

Since you'll be working in the terminal so much, it is worth spending a couple of minutes configuring the terminal to be as comfortable as possible. You may want to change the background color, the text color, or the font used for the text to suit your preferences. To do this, click on the `Edit` menu and then select the `Profile Preferences` option. This will bring up a window in which you can select a number of preferences. To change the font, uncheck the option `Use the system fixed width font` and then click on the `Font` pulldown menu immediately below it. You can choose between a number of fonts, each at a number of different sizes. Pick whatever you find most comfortable, then click `OK`. If you've changed the font, you will see the change. If you want to change the background or text colors, click on the `Colors` menu option at the top of the window and uncheck `Use colors from system theme`. Then you can click on `Text color` and `Background color` to set those colors to be whatever you like. I like an off-white text color on a dark green background, but that's just me; use whatever you feel most comfortable with.

Also, note that the terminal program can create multiple terminals or multiple tabs in a single terminal. Tabs are very useful when doing multiple distinct things. Create a new tab by typing `Control+Shift+t`. Create a new terminal window by typing `Control+Shift+n`. (These can also be done from the menu.) Type `Control+d` in a terminal window to close it, or just type `exit`.

There are other terminal configurations you can do, but this is enough for now.

Installing Python and WingIDE

Python will already be installed on the VM. See above for how to install WingIDE.

Whew! That was a long and complicated process, but now you are completely set up on your spiffy new Ubuntu MATE virtual machine. The only additional thing we will do in the rest of the course is install more Python libraries; we'll walk you through that when the time comes. If things didn't work out as described above, see a TA.

Part C: Exercises

In this section we will do some basic programming exercises to test your understanding of the lecture material and to let you experiment with WingIDE and the Python shell. We will also introduce you to a few features of Python that we didn't talk about in the lectures.

1. [**10**] For each of the following expressions, what value will the expression give? Verify your answers by typing the expressions into the Python shell. Write your answers as a Python comment.

    1. `9 - 3`
    2. `8 * 2.5`
    3. `9 / 2`
    4. `9 / -2`
    5. `9 % 2`
    6. `9 % -2`
    7. `-9 % 2`
    8. `9 / -2.0`

9. `4 + 3 * 5`
10. `(4 + 3) * 5`

2. [**10**] Python includes a lot of operators of the form `op=`, where `op` is one of the standard operators. For instance, the following are all operators: `+= -= *= /= %=`. The meaning of these operators is as follows:

   ```
   x op= y
   ```

   is the same as

   ```
   x = x op y
   ```

   For instance, `x += y` is the same as `x = x + y` and `y -= 2` is the same as `y = y - 2`. For each of the following statements, what will be the value of the variable `x` after the statement has been executed? Assume that the statements are entered into the Python shell one after another, so that subsequent statements can depend on the values of previous ones. Write your answers as a Python comment.

   1. `x = 100`
   2. `x = x + 10`
   3. `x += 20`
   4. `x = x - 40`
   5. `x -= 50`
   6. `x *= 3`
   7. `x /= 5`
   8. `x %= 3`

3. [**10**] Write a step-by-step description of what happens when Python evaluates the statement `x += x - x` when `x` has the initial value of `3`. What is `x` after the statement has been evaluated? Write your answer as a Python comment.

4. [**10**] Complex numbers are the sum of a real number and an imaginary number. An imaginary number is just a real number multiplied by the square root of `-1`. The square root of `-1` is often called `i` or `j`. Python allows you to enter complex numbers directly, in the form `Yj` (for imaginary numbers) and `X+Yj` (for complex numbers).

   What are the values of the following expressions?

   1. `1j + 2.4j`
   2. `4j * 4j`
   3. `(1+2j) / (3+4j)`

   What are the results of the following two expressions?

   1. `(1+2j) * (1+2j)`
   2. `1+2j * 1+2j`

   Why do you think they're different? What does this tell you about the way Python handles complex numbers?

   Write all your answers as a Python comment.

5. [**10**] However, not all complex operations act the way you would expect them to. Enter the following into the Python shell:

   ```
   >>> import math
   >>> math.sqrt(-1.0)
   ```

This will give the following error message:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: math domain error
```

That means that the `sqrt` function can't handle square roots of negative numbers. Fortunately, the `cmath` (complex math) module exists which can handle problems like these. Now try this:

```
>>> import cmath
>>> cmath.sqrt(-1.0)
```

Notice that this does give the expected answer of `1j`.

What are the values of the following expressions?

1. `cmath.sin(-1.0+2.0j)`
2. `cmath.log(-1.0+3.4j)`
3. `cmath.exp(-cmath.pi * 1.0j)`

Finally, why do you think it's a better idea to write

```
import math
import cmath
```

than

```
from math import *
from cmath import *
```

in a program?

Write all your answers as a Python comment.

6. [**5**] What are the values of the following expressions? (If there are multiple lines in a question, give the value of the expression on the final line.) If an expression gives an error, write the error message. Write all answers as a Python comment.

   1. `"foo" + 'bar'`

   2. `"foo" 'bar'`

   3.
      ```
      a = 'foo'
      b = "bar"
      a + b
      ```

   4.
      ```
      a = 'foo'
      b = "bar"
      a b
      ```

7. [**5**] Rewrite the following string using single quote characters on either end of the string instead of triple quotes. Don't use any operators or function calls (you don't need them). Write the answer inside a Python comment.

```
'''A
B
C'''
```

8. [**10**] Write a single Python expression which will generate a string of 80 `'-'` characters. The expression itself shouldn't be longer than 10 characters. Write the answer inside a Python comment.

9. [**5**] Write a single-line Python string which, when printed using Python's `print` statement, will print:

   ```
   first line
   second line
   third line
   ```

   (without any leading spaces on each line). Write the string inside a Python comment.

10. [**15**] Given variables `x` and `y`, which refer to the values `3` and `12.5` respectively, write a line of code using `print` to display each of the following messages. When numbers appear in the messages, the variables `x` and `y` should be used in the `print` statement. Use the `%` formatting operator and a format string for each example, but don't use the `%s` catch-all formatting operator. In this case, write your answer as Python code (not inside a comment).

    1. `The rabbit is 3.`
    2. `The rabbit is 3 years old.`
    3. `12.5 is average.`
    4. `12.5 * 3`
    5. `12.5 * 3 is 37.5.`

11. [**10**] Use `raw_input` to prompt the user for a number. Store the number entered as a `float` in a variable named `num`, and then print the contents of `num`. Use the prompt string `"Enter a number: "` as the argument to `raw_input`. Write the answer as Python code (not in a comment).

12. [**10**] Write a function called `quadratic` that takes four input arguments `a`, `b`, `c`, and `x` and computes the value of the quadratic expression

    $ax^2 + bx + c$

    for those values of `a`, `b`, `c`, and `x`. Note that `**` is the power operator in Python, though you don't actually have to use that. (What else could you use if you don't use the `**` operator?)

    Write your answer as Python code (not in a comment).

13. [**30**] DNA is composed of four distinct base pairs: adenine (A), cytosine (C), guanosine (G), and thymine (T). In a DNA molecule, A bases pair with T bases and C bases pair with G bases. A value of interest is the proportion of C/G bases in a DNA molecule, since C/G bases bind more strongly than A/T bases (and thus, a DNA molecule with a large proportion of C/G bases is more stable than one with a large proportion of A/T bases). Given a DNA sequence, this can easily be computed by counting up the number of each base, and taking the ratio of the (G or C) bases as a proportion of the total.

    Use Python's `help()` function to learn about the `count()` method of Python's strings (typing `help(''.count)` at the Python interactive prompt is one way to do this). Then use this to write a function called `GC_content` which takes in a string representing a DNA sequence and returns a single float, which represents the proportion of the bases which are either G or C. You may assume that the input string has only A, C, G, or T bases. For instance:

    ```
    GC_content('ACCAGTGTAG')         --> 0.5
    GC_content('ATATATATA')          --> 0.0
    GC_content('GCGGCCATGCATGGGG')   --> 0.75
    ```

    One pitfall here is that dividing two integers in Python throws away the remainder, so you will want to convert the numerator and denominator of the ratio to floats using the `float()` function (built-in to Python) before doing the division.

Add a docstring to your function so that typing:

```
help(GC_content)
```

in the Python shell will print out a description of what the function does, what its input argument should be, and what its output represents.

Make sure you test your function on the examples given, no matter how sure you are that it works correctly!

Part D: Miniproject

Most assignments will have a miniproject which will involve solving larger-scale programming problems. There is no miniproject in this assignment.
