Python Tips and Tricks

This lab will utilize an Ubuntu 16.04 Virtual Machine. It is recommended that Python 3.x be installed in a VM. This will demonstrate some basic tips and tricks that are good to show students and give you an edge when working with Python.

List Comprehension

Lists are a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as [strings](https://www.digitalocean.com/community/tutorial_series/working-with-strings-in-python-3) are defined as characters between quotes, lists are defined by having values between square brackets. List comprehension consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses. The result will be a new list resulting from evaluating the expression in the context of the for and if clauses which follow it. Below are some examples:

1. Creating a list using the traditional for loop. Type the code below and execute it.



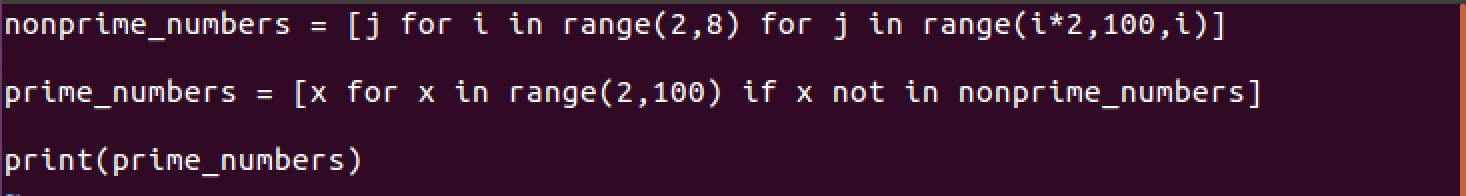
In the program above a list called ***squares*** was created. This list has no contents currently hence the empty []. The next line is a for loop that will iterate through numbers 0 – 9. Range is a function that will start at 0 unless otherwise specified and go to one number just below the value in the (). This is how we get 0 – 9. The command ***squares.append(x\*\*2)*** will add the values of x ^2 to the list. This would produce 0, 1, 4, 9, etc… The last statement will simply print the resulting list. This is a very simple way to create a list of values.

1. Using list comprehension is another way to create a list of items in a very simple fashion. Type the code below and execute it.



In the program above an expression (x \*\*2) is followed by a clause (for x in range(10)). This accomplishes the same goal as the program in #1, but with less code. This is a more efficient way to create the list.

1. With list comprehension, if statements can also be utilized to create a list. Type the following code below and execute it.



In the program above an expression (j) is followed by a clause (for I in range(2,8) for j in range(i\*2,100,i)). In this case a nested for loop is being used to generate a list of numbers that are not prime numbers. The second statement will iterate through number 2 – 99 and add them to the list as long as the number does not exist in the list nonprime\_numbers.

Using list comprehensions is a quick and efficient way to create a list. Practice different types and find the very usefulness of this technique.

Lambda

Lambda is a tool for building functions, or more precisely, for building function objects. we don’t absolutely need lambda; we could get along without it. But there are certain situations where it makes writing code a bit easier, and the written code a bit cleaner. What kind of situations? … Situations in which (a) the function is fairly simple, and (b) it is going to be used only once.

Normally, functions are created for one of two purposes: (a) to reduce code duplication, or (b) to modularize code.

If your application contains duplicate chunks of code in various places, then you can put one copy of that code into a function, give the function a name, and then — using that function name — call it from various places in your code.

If you have a chunk of code that performs one well-defined operation — but is really long and gnarly and interrupts the otherwise readable flow of your program — then you can pull that long gnarly code out and put it into a function all by itself.

But suppose you need to create a function that is going to be used only once — called from only one place in your application. Well, first of all, you don’t need to give the function a name. It can be “anonymous”. And you can just define it right in the place where you want to use it. That’s where lambda is useful.

1. Type the code below and execute it.



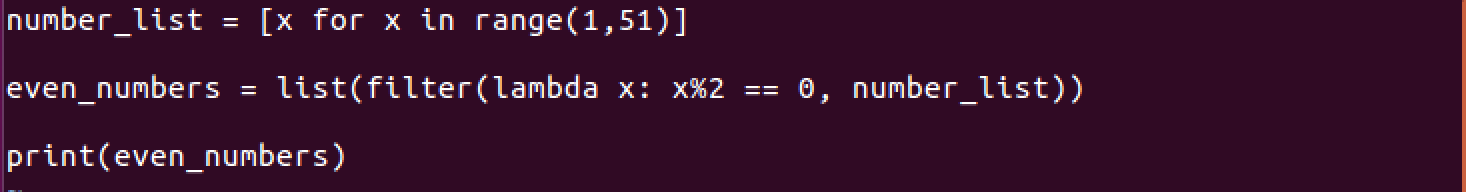
The code above has a function called ***calc*** that will accept two values, x and y, and return the sum of those values. This code can be shortened using an anonymous function called lambda.

1. Type the code below and execute it.



The code above accomplishes the same goals as the previous program but has been done with more efficiency and simplicity. With lambda, you have the code as ***lambda arguments: expression.*** In this case, x,y are the arguments and x + y is the expression.

1. Type the following code and execute it.



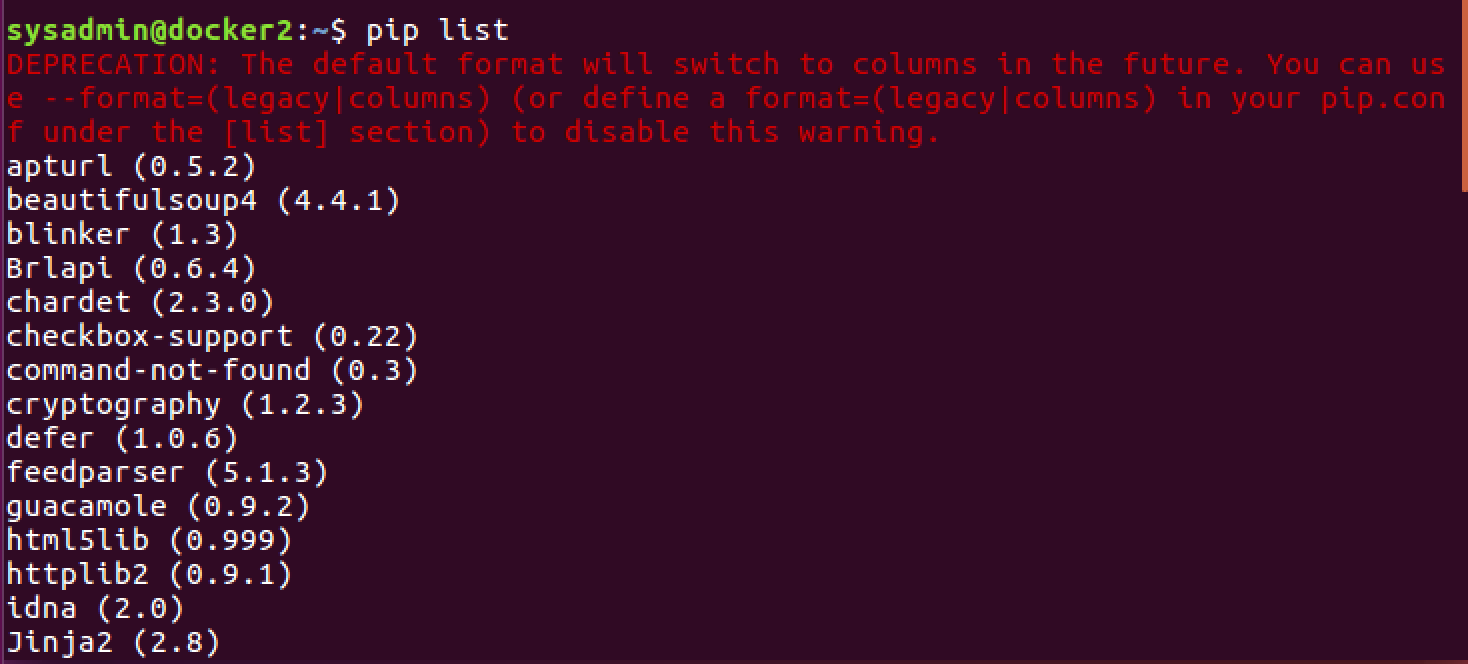
The code above adds a new layer to lambda. In this program a ***number\_list*** is created that stores numbers 1 – 50. This is accomplished using list comprehension. The next line of code will create a list of only even numbers in the ***number\_list***. Filter is a specialized function The filter() function in Python takes in a function and a list as arguments. The function is called with all the items in the list and a new list is returned which contains items for which the function evaluates to True. In this case, the lambda function will take each number in the ***number\_list*** and perform the modulus (modulus is a arithmetic function that returns a value after division) of that number with 2. If this is equal to 0, then the number is added to the list.

Using lambda for anonymous functions is a quick and efficient way to reduce syntax and lines of code in a program and still get the benefit of functions. Practice different types and find the very usefulness of this technique.

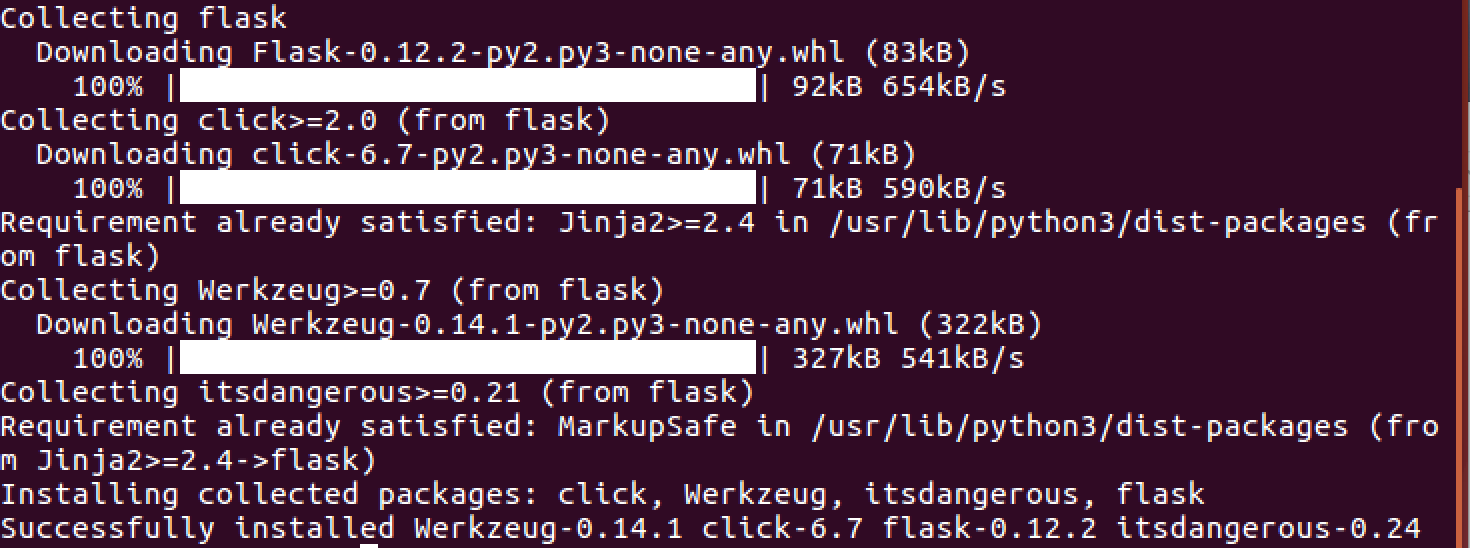
Using PIP (preferred installer program)

There are times that you may need additional functionality that is not provided by the standard installation of Python. A module is a Python object with arbitrarily named attributes that you can bind and reference. Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. There are many different modules that can be added to the Python program. Here is a list of those [modules](https://docs.python.org/3/py-modindex.html).

1. First, open a terminal or command prompt if using Windows. At the prompt, type ***pip list***. This will display all modules that are currently installed.



1. In the previous example, there are many modules installed. To install additional modules, use the ***pip install*** command. Let’s install the module ***flask.*** To do this enter the command ***pip install flask***. If successful, the output below should be similar.



1. To use the modules in a Python program, there are several different ways to accomplish the task. First, use the ***import*** function to do so. This will import the entire module and all methods/functions with it. Second, use the ***from <module> import <method/function>***. This will only import the methods/functions desired. Below is an example of both.

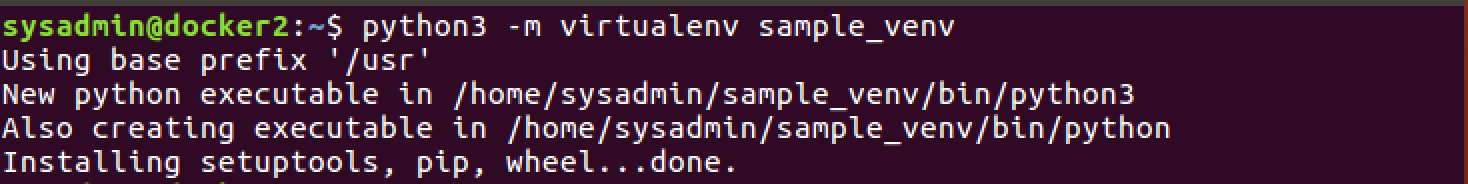


After import ***random***, you can refer to things in ***random*** like ***random.randint***. After ***from x import \****, you can refer to things in ***random*** directly just as ***randint***. Because the second form imports the names directly into the local namespace, it creates the potential for conflicts if you import things from many modules. Therefore, the from x import \* is discouraged. You can also do ***from random import randint***, which imports just the ***randint*** into the local namespace, not everything in ***random***. This is better because if you list the names you import, you know exactly what you are importing and it's easier to avoid name conflicts.

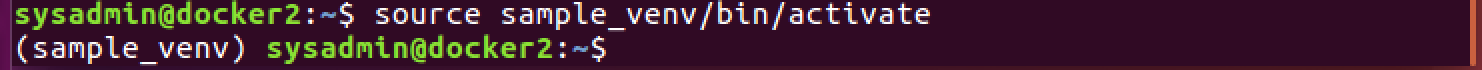
Virtual Environments

virtualenv is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary executables to use the packages that a Python project would need. This is especially useful when different version of Python are going to be used or different versions of modules. The virtual environments DO NOT lock you out of the file system, but lock you into a folder with a specified instance of Python and modules.

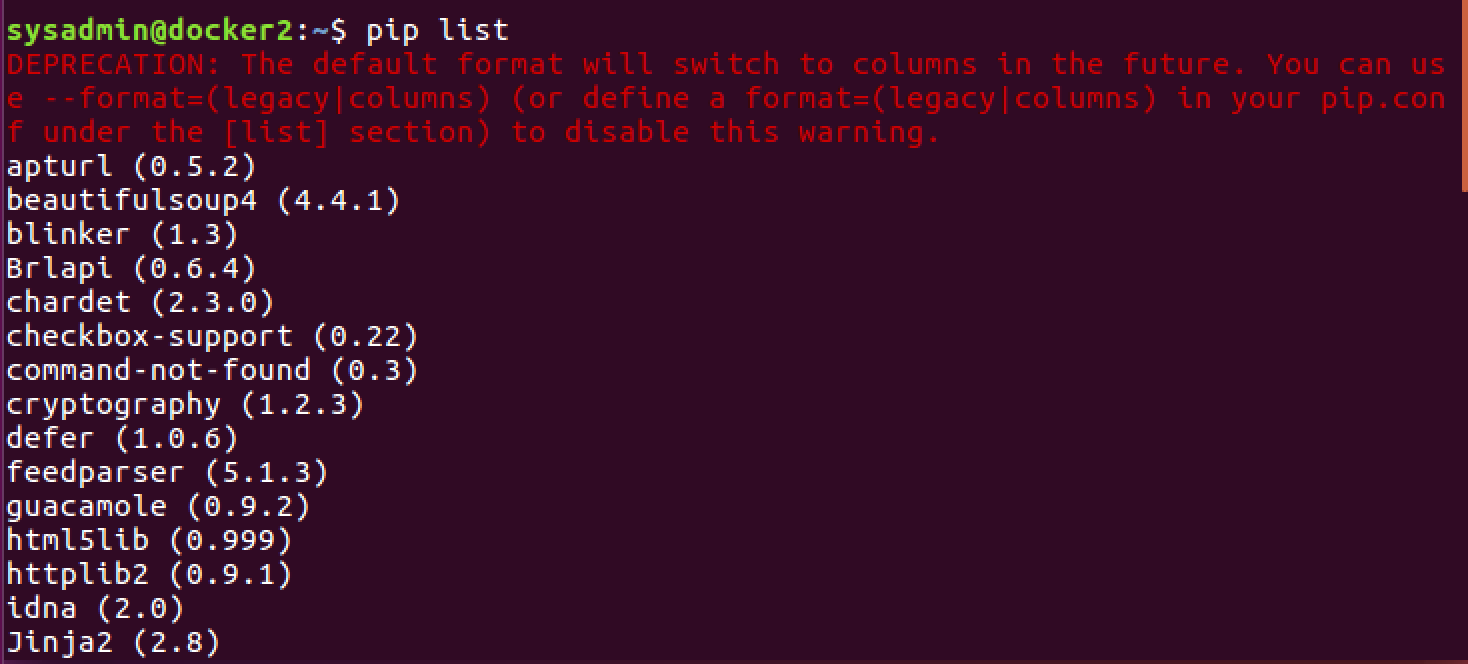
1. The virtual environment module must be installed as it is not a native part of Python. To accomplish this, use the command ***pip install virtualenv.*** After the virtual environment module has been installed, it is time to setup the virtual environment.
2. There are two ways to accomplish the creation of the virtual environment. The virtual environment is nothing more than a folder or directory in the file system. This means a directory can be created for the virtual environment. If a directory is not created the virtual environment tool can be used to create the directory automatically. In either case the virtual environment will be created. To accomplish type ***python3 -m virtualenv sample\_venv***, where sample\_venv is the name of the directory to be used as the virtual environment. Notice there are a couple things automatically copied in the directory.



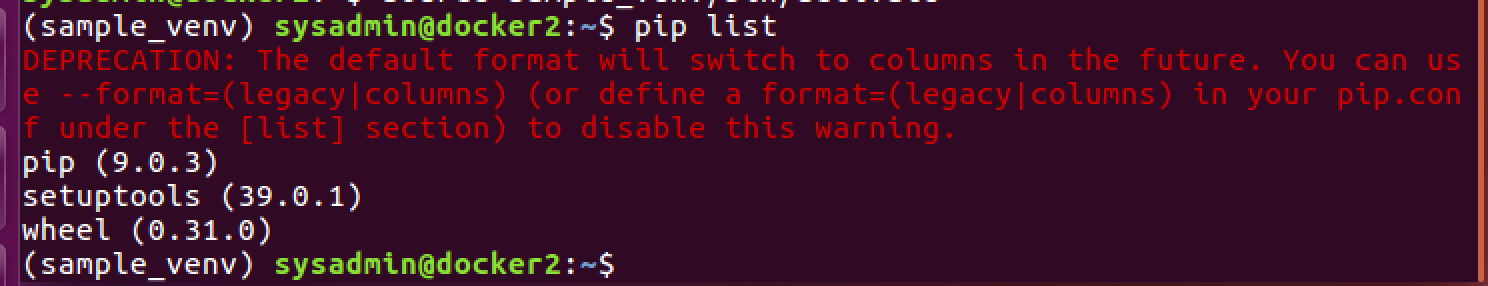
1. After the environment has been created it is time to activate it for use. At the prompt type, ***source sample\_venv/bin/activate.*** This will allow the user to enter the virtual environment. This is further shown by the new prompt.



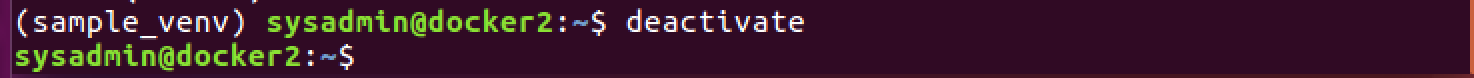
1. In the section about PIP, notice the modules that were installed on the Linux system.



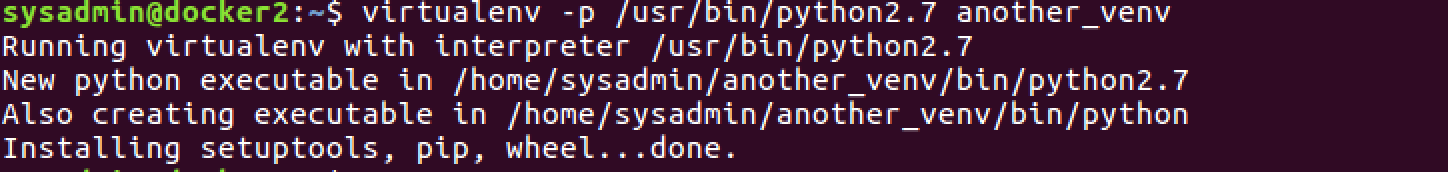
At the new prompt type ***pip list*** and notice there are significantly less modules. This is due to the environment being separated from the rest of the Python environment.



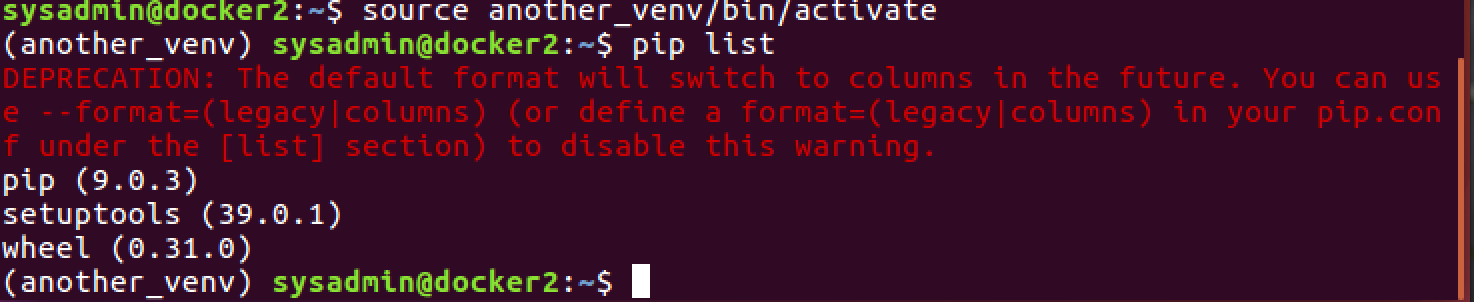
To exit the virtual environment, enter the command ***deactivate.***



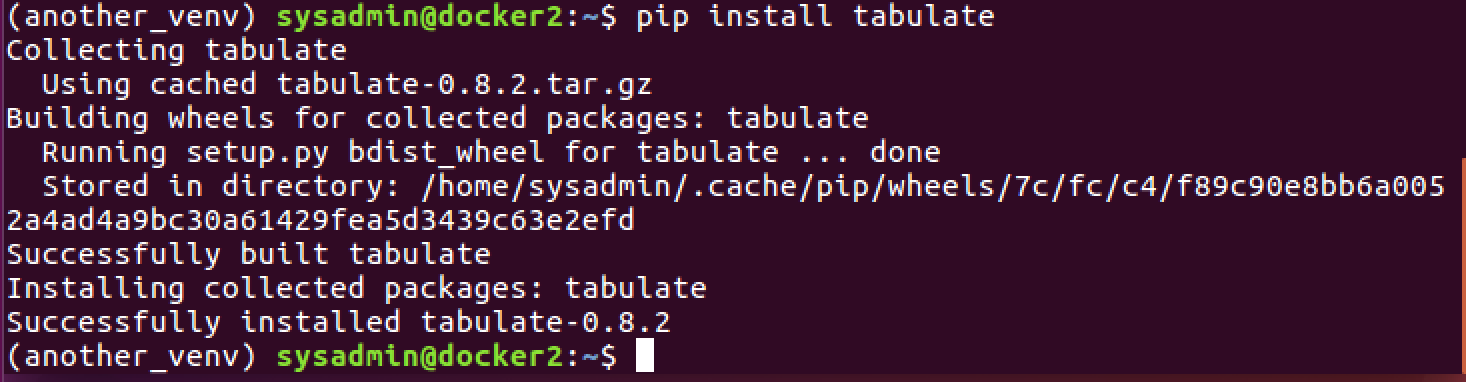
1. The virtual environment gives the user the ability to have different modules, different versions of those modules, as well as different versions of Python. Below is an example of adding a different version of Python (Python 2.7) to a virtual environment as well as installing a module in that same environment.



After the version of Python is installed into the environment, activate, and enter the environment. After entering the environment, list the modules installed.



Now install a new module into the virtual environment.



These are some tips and tricks that may be useful to some of the more advanced topics in Python.