Lecture 1

Steven Walton
PS 232 Computational Methods
Department of Physics
Embry-Riddle Aeronautical University
Prescott, AZ 86301

September 2, 2014

Introduction: Installing and Modules to use

For this class we will be using Python2.7, and not python3. There are a few reasons for this. The first is that there are more third-party module support for python 2, which is one of the key features of python. Second is that most code that is python2 can be compiled by python3. Third, most people still use python2, and one will not have a difficult time switching to python3 (though the other way is a little trickier). Forth, and probably the most important, is that I know python2 best and will not be able to give as quick of help to debugging in python3. You are able to use any programming language in this class, but you do need to remember that you will only receive support for MatLab and Python2.

Installing python2

The first thing we need to do is install python2. To do that visit python.org and download the corresponding package for your system. If you are using Linux then remember that you can do this from your package manager (apt-get, pacman, yum, etc.). Remember to download python2. If you are on Linux or OSX I also suggest installing pip, which is the python package installer. This will help you manage your modules and ensure that they are installed in the right place.
Once you have followed these steps, then you should have python installed.

Interpreter

The next thing we need is something to run python in. Some way to edit code and execute it. I prefer to use vim, but this is not suggested for those that are not comfortable with vim already. When installing python you will already have an interpreter called Spyder. Though you will be able to run everything in here it is not a user friendly space. The first suggestion that we give is PyCharm. This interpreter is extremely user friendly, and has the same capabilities as pip. This will allow you to manage all your modules right from this program. I highly suggest using PyCharm for writing your python code, it will have a similar feel to writing MatLab code in its console. Another commonly used interpreter is ipython. This also has a lot of capabilities, but does not have the same ease of use as PyCharm.
You are welcome to use any interpreter or console that you wish, and are encouraged to try different ones. We are welcome to suggestions if you do find another one.

Modules

We mentioned modules before, and you may be wondering "what the heck are these things?" These are actually one of the most important parts of python, and what gives it a lot of power. With its ease of use, reading and writing code, many people have developed third-party packages to give python capabilities

far beyond its original use. Some of these are NumPy, SciPy, Anaconda (which contains SciPy, NumPy, Pandas, and many other packages), and Pandas. You will become more familiar with these modules in the future. But for now just know that whatever you are trying to do (big data, computer vision, or website interfacing), python probably has a module made to make your life easier.
For now you can install Anaconda. If you'd like to keep your modules to a minimum, then just install NumPy and MatPlotLib for now, we will use them in the next lesson. We will add more later.

Writing Our First Program

As in common programming fashion, the first program that we are going to write is the "Hello World!" program. The point of this program is to print the statement "Hello World!". In many languages this would take a few lines to write. But in python this is one of the easiest things to do. Python's power is in its simplicity and readability. So follow me and we will type

```
print ''Hello World!''   # Outputs ''Hello World!'' to standard i/o
```

That's it! Simple. There are a few things to note though. The first is the quotations marks. We need these here, and they surround the text that we will be printing. The second thing to notice is that we left a comment. A comment is something in code that is left only in the source code. When compiled the compiler completely ignores these sections. Comments are for making code easier to read and to help others debug/review the code. Remember to ALWAYS comment your code. It is always better to over comment your code than to under comment. You will find that even the most simple codes will be difficult to understand without comments.
Now the print command has another trick that we can use. Try typing the following into the console.

```
print ''Hello '' + ''World!'' # This time we are combining strings
```

What does it output? That's right, we get the same output as the first code. This is a great feature of python. Instead of using an fprint statement with multiple flags, like in C/C++ or MatLab, we are easily able to output text/variables in this manner.
Now let's try something a little different. We are going to define a string and output it. For now, think of a string as just a variable of words.

```
text = ''Hello World!''
print text
```

You'll notice that this again outputs the desired text. We can do the same thing as the second code and make two strings and output them. Try this out. A great way to learn code is to experiment with it.

Closing Thoughts

There is a lot of documentation for python out there, mostly for python2. It is important to learn how to become comfortable reading the manuals and finding information on your own. There are a few coding techniques and practices that I will give you to aid in your future endeavours.
Assume that someone else is going to read your code. Even if you are just writing something that only you are going to be using, write it as if someone else will read it at some time. This means commenting it fully and using logical variable names. Readability is extremely important. I can not tell you how many times

I have found old code that I have written and completely forgot what it all means. This is an extremely common mistake in programming, but it can be a deadly one. I can not stress this enough.
Experiment with new commands. This is a great way to learn, and can make programming more enjoyable.
Break your code on purpose. Find out the errors that the compiler is giving you. This will help you learn how to debug future code. Many times the interpreter's debugging messages can be cryptic and difficult to understand. It takes time to learn this, and be sure to read the outputs carefully.
Have fun. This is the most important thing. Do not forget that there can be a lot of fun in the act of coding. It is all about solving problems. The world is full of puzzles, and coding will help you solve many that you could never do without it. Code is an extremely powerful tool.