Programming Environment

Your programming environment is the computer you do your work on, and all the software that's installed on your system which helps you write and run programs. Some systems are better for programming than others, but the best system to learn on is probably the one you are using right now. This section will help you get your system set up so you can start writing programs quickly.

Overview

The goal is to help you get Python up and running on your computer, so that you can write and run your own programs. To do this, we want to:

- Find out if Python is already installed on your computer.
- Install Python, if it is not already installed.
- Install a text editor that will make it easy to run your first programs.
- Help you enter and run Hello World, your first Python program.
- Congratulate you if it works, and give you some options for troubleshooting if it doesn't work.

Linux

If you already know that you'd like to take programming seriously, you might want to consider learning to use Linux. The people who build Linux expect you to program at some point, so they've built the system to make it as easy as possible to get started.

Most Linux systems already have Python installed, so we'll just verify that it's installed, and then install Geany.

Setting up a Linux system

Mac

If you go to a Python conference or meetup, you're likely to see more Macbooks than any other computer. So Macs are certainly a good platform on which to learn Python. Many of the people you see using Macbooks are actually running some distribution of Linux on their Mac, and running Python within a virtual Linux machine. You don't have to worry about that yet, just know that if you continue to grow as a programmer your Mac will probably continue to serve you quite well. Python 2.7 is probably already installed on your Mac, so we'll just verify that it is installed, and then install Sublime Text. If you want to start with Python 3, we'll walk you through setting up Python 3 on your system as well.

Setting up an OS X system

Windows

Python doesn't come pre-installed on most Windows computers, but you can download an installer that will set Python up for you. Once you have Python installed and running, it's pretty straightforward to install Geany, and then configure it to run Python programs.

Setting up a Windows system

Some simple programming projects for beginners

1) Hello World

Ah, the all familiar "hello world," exercise that you do every time you start learning a new programming language. The goal here is to output a small message to introduce yourself to the language.

In Python, this is incredibly simple. All you need to do is open the interpreter and type the following:

```
print("Hello World")
print("My name is") #add your name after the word "is" obviously
```

If all goes well, you should see something like this:

```
> python3 #to call upon Python on MAC OS X use this command, for Windows use "python"
```

```
Python 3.5.1 (default, Jan 14 2016, 06:54:11)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on
darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> print("Hello World")
>>> print("My name is Bob")
Hello World
My name is Bob
```

Clearly, the command print is used to display content on the screen. Remember this command because you'll be using it often.

The text you see after the # symbol is called a comment. Comments do not appear at runtime, and are instead meant for the developers who will be working with the code. The comment we left above provides instructions for adding your name to the message. More often than not, comments will provide labels or quick descriptions for a snippet of code, so you can easily identify what a particular section is for.

2) Performing Calculations

Next, let's take a simple calculation and feed it through the interpreter to see what happens. Enter the following:

```
7 + 2
```

After typing in the equation above and hitting enter - to submit - you should see something like the following:

```
>>> 7 + 2
9
```

Notice how the interpreter automatically answers the equation and spits out the result?

3) Creating Your First String

A string is a sequence of characters that can be processed by a computer. The string is usually stored for manipulation later.

Strings must always begin and end with the same character, this is a requirement. In addition you can use either single or double quotations to signify a string, there is no difference

between the two. The quotation marks only serve to inform Python that what's inside of them is a string.

Let's save your name as a string to call upon later. To do that, type the following into the interpreter:

```
>>> "Bob"
'Bob'
```

Congrats! You just created your first string, and this is signified by the information sent back to you. We can see that the name was saved as a string.

Now, we want to test out this string and see what kinds of things we can do with it. First, let's use multiple strings in tandem. Enter the following into the interpreter:

```
>>> "Hello there " + "my name is " + "Bob"
'Hello there my name is Bob'
```

Notice how Python adds the strings together before outputting the content?

Another neat trick you can do is multiply strings or manipulate them through equations.

```
>>> "Bob" * 4
'BobBobBobBob'
```

This may seem silly right now, as you would probably never need to multiply your name like this in the real world. However, this type of manipulation can really come in handy when you're working on large projects in Python that have a lot of strings.

To see your name in upper case - instead of using caps - try working with the following command:

```
>>> "Bob".upper()
'BOB'
```

Pretty cool, right?

4) Return the Length of a Phrase or Word

Normally, if you want to know the number of letters in a word or phrase you would just count them yourself, but that's no

fun! There's actually a dedicated command to do just this!

To determine the number of letters in a word or string, type the following into the interpreter:

```
>>> len("BobIsTheGreatestEver")
20
```

You can also calculate the length (size) of a list using the same command.

```
>>> players = ['bryan', 'john', 'chris']
>>> len(players)
3
```

5) Storing Variables

Each entry in the list of "players" we created above is called a variable. Variables are nothing more than names or titles for a particular set of data so that you can store them and call upon them later. For example, the variable in the tutorial above was "players" because that's what we used to store the names of the players.

Let's create a new variable of our own:

```
>>> movie = "Terminator"
```

Our variable is "movie" and in that variable we stored the data "Terminator," as you can see.

One thing you'll notice about variables is that the interpreter doesn't return anything once the information is stored. You may be wondering how we know the variable was actually stored?

You can test this by simply entering "movie" in the interpreter and hitting enter. It should return the data stored in that particular variable, like so:

```
>>> movie
'Terminator'
```

Good job! You created your first variable! Feels great, doesn't it?

But, let's say we get sick of seeing "Terminator" as the data stored in that variable. We can change this easily.

```
>>> movie = "Cinderella"
>>> movie
'Cinderella'
```

Sweet! No more crazy robots or androids! Just a compassionate, naive girl named Cinderella who will finish all our chores for us!

You can store just about anything inside a variable, including numbers, equations, and more.

6) Comparisons

One remarkably useful - yet underrated - thing you can do with a programming language is compare sets of data. Let's try that now, using numbers.

```
>>> 7 > 2
True
```

```
>>> 9 < 1
False
>>> 6 > 2 * 4
False
>>> 3 == 3
True
>>> 5 != 2
True
```

Notice how we used two equal signs (==) to check if sets of data are equal? You must always use two equal signs if that's what you are trying to do. This is because a single equal sign, or "=" is used to assign a value to a variable.

In addition, if you want to check whether or not two values are unequal you can use an exclamation mark followed by an equal sign like so: "!="

6) Define (main) functions

```
def squareit(n):
    return n * n
```

```
def cubeit(n):
    return n*n*n

def main():
    anum = int(input("Please enter a number"))
    print(squareit(anum))
    print(cubeit(anum))

if __name__ == "__main__":
    main()
```

Some useful Python libraries

1. NumPy. How can we leave this very important library? It provides some advance math functionalities to python.

2. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

3. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

4. BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.

5. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

6. matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

7. nltk. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

Some useful links:

Official Python site. Find a complete list of all documentation, installation, tutorials, news etc.

https://docs.python.org/3.6/tutorial/index.html

https://www.python.org/about/gettingstarted/

https://wiki.python.org/moin/BeginnersGuide

A Gentle Introduction to Programming Using Python

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2008/

Resources for learning python (libraries, examples, tutorials)

https://opensource.com/education/16/4/teaching-python-and-more-with-oer

Useful Books on Python

1. Learning Python, 5th Edition
2. Python Programming for the Absolute Beginner, 3rd Edition
3. The Quick Python Book, Second Edition
4. Python Essential Reference
5. Python Programming: An Introduction to Computer Science

Update (3/6/2018)

Installing PyDev for Eclipse

PyDev is a plug-in for eclipse which let you to program in Python.
The link below has a step by step guide for installing eclipse, and PyDev for Python:

https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/eclipsepython%20neon.html

After setting it up, you can use eclipse IDE for your Python programs and run them from there.

More Examples:

https://www.programiz.com/python-programming/examples

https://wiki.python.org/moin/SimplePrograms