Introduction to Python programming

Introduction

The goal of this assignment is to get you comfortable with programming in Python, the programming language that will be used in all assignments in this course. Like the previous assignment, this is also ungraded. Also, you do not have to upload any on the autograder for this assignment.

Python is a general purpose, interpreted programming language that is used in a wide variety of fields ranging from GUI programming to data analysis. Python is also one of the most widely preferred programming language by computer security professionals and is supported by numerous popular tools such as debuggers and disassemblers. For more information, please read the FAQs on python.org.

Installation and other setting up

Your first step is to verify if python2 is installed by running the command "python2" in a terminal emulator. We will be testing your solutions on Ubuntu 16.04 and thus recommend you run the same OS. As per Ubuntu 16.04's release notes, it does not come with python2 installed by default. However, python2 is still installable from the repositories via the python2 package. If python2 isn't installed, please install it via the package manager.

On running python2 command, a prompt(>>>) will be displayed waiting for your input. This is one mode of using python - via a command line REPL(read-eval-print loop). This allows you to quickly test some ideas you have. The other approach is to write source code in a file and run the contents using the python interpreter. We recommend that you use the REPL when you need to test some new code that you are learning or want to try out and when you need to quickly view the output of some statements. Otherwise, we recommend writing code into a file.

A better REPL

The default python REPL is fairly limited and not very easy to use. Instead, we recommend using ipython, a better python REPL with features such as auto-completion and saving history among several others. We recommend installing the latest ipython from PyPI than the one available in the Ubuntu repositories. To install latest ipython, install the package "python-pip" from Ubuntu 16.04 repositories using apt-get and then install ipython running the command "pip2 install ipython" as root user(for instance, using sudo command).

Configuring a text editor

If you already have a setup for Python that you are comfortable with, skip this section and move on. Otherwise, read on.

Python code can be written in any editor ranging from the simple nano/ed to complicated IDEs. However, some editors and IDEs can make greatly improve the experience of programming in Python with their features. Here is a small list of editors and IDEs that you can explore. **Don't spend too much time in choosing one. Fix one and move on to learning programming.**

1. **PyCharm**: One of the most popular Python IDEs from JetBrains. They've made several popular IDEs include IntelliJ which powers Android Studio and also created the Kotlin programming language(an official Android programming language). Traditionally not free, JetBrains is offering all their IDEs free for verified students! Visit JetBrains tools for students to get started. You will need your student email ID to avail the student offer.

2. **Atom editor**: A relatively new editor created by GitHub. It's lightweight and quite good. It also comes with several packages for various languages which add features like autocompletion, syntax checking and many themes. Best of all - Atom is an open source editor released under MIT license(free forever!). Visit atom.io to get started.

3. **vim/neovim**: One of the oldest editors out there and free to use. It's also immensely popular editor. However, it is a command line based editor and takes a while to get used to. If you are not familiar with vim or using the Linux terminal or simply prefer GUI based editors, best to avoid this for now. **However, as a security professional, you will frequently work with remote machines which won't offer a GUI. In those cases, vim will be useful.** We recommend that you do explore and become comfortable with using vim later on, if you do not choose to work with vim now. vim is available for installation in the Ubuntu repositories. neovim is a project that aims to rewrite vim without a lot of old features and with new ones. Visit neovim's website for more information. We recommend choosing neovim over vim. Since they are quite similar, if you learn neovim, you will also learn vim.
   Recommended plugins: YouCompleteMe, syntastic and SimpylFold. We use these and they're pretty good and free for all!

There are of course several other editors and IDEs which can be used. However, these are the most popular ones in our opinion and quick to get started with. We recommend choosing and configuring an editor quickly and move on to the remainder of the assignment. The editor/IDE is mostly a tool that aids you: ultimately if you don't learn programming well, the editor won't be of much use.

Be sure to install plugins for syntax highlighting, a language specific plugin, code completion, syntax checking and a theme with a good colour scheme if your chosen editor/IDE doesn't have them already. Solarized is one of the most popular colour schemes supported by nearly all editors. Also, consider installing a good code folding plugin and explore how others are using your chosen editor to learn more and improve your setup. **However, don't spend too long on this: once you have something setup, move on to learning programming.**

Learning Python

Python language basics

One of the most important skills is to be able to learn on your own. Thus, we will start practicing that skill when learning Python programming.

Regardless of where you learn Python from, don't spend too much time exploring available functions for all datatypes etc. One can learn programming by actually writing programs only and you will write a lot of simple Python programs shortly. Thus, gain some knowledge of what is available and find out whatever functions you need when necessary using a search engine.

There are two recommend resources for learning Python. **Pick any one and learn the topics mentioned.**

1. Sections 1 to 6(including 6) and 9.1 to 9.4(including 9.4) of the official Python tutorial.
2. The basic tutorial upto and including modules and classes/objects from advanced section of Tutorials point Python tutorial. You can skip environment setup since we already setup the IDE and REPL above.

If you feel the above two resources are little overwhelming because of the depth they get into, consider using these tutorials. You can use the above two as a reference when needed and learn from the following. These are little easier to follow but not as comprehensive as above two. **Again you need to pick only one of the following. If you already followed the official tutorial or tutorials point documentation, you can skip this and move onto next part**.

1. Units 1 to 9 and 11 of Codecademy's Python tutorial.
2. Lessons from "Hello, World!" to "Modules and packages" and lesson on sets from Learn python.

Useful libraries and functions

Once you are done with whatever you choose from the above choices, go through the following tutorials. These introduce some modules from the Python standard library which will be useful for the assignment:

- **json**: A module for working with JSON data. The official tutorial and the PyMOTW tutorial are pretty good introduction to this module.
- **Working with strings**: These tutorials ([1], [2]) introduce some useful functions for working with string datatype.
- **Accessing command line arguments**: This is possible using the sys module.

These modules may be necessary for some assignments later on. You can explore them along with the programming exercises described in following section, whenever you get time.

- struct module: Refer to the PyMOTW tutorial.
- hashlib module: Refer to the PyMOTW tutorial.
- requests module: Refer to the official documentation.

These are some useful Python modules mentioned here purely for your learning and **optional for this course**. These may not very useful during assignments but these are good to know if you are learning Python.

1. os: Refer to the PyMOTW tutorial.
2. itertools: Refer to the PyMOTW tutorial.
3. collections: Refer to the PyMOTW tutorial.
4. datetime: Refer to the PyMOTW tutorial.
5. random: Refer to the PyMOTW tutorial.
6. re(regular expressions): Refer to the PyMOTW tutorial.
7. os.path: Refer to the PyMOTW tutorial.

Practicing Python programming

Now that you have learnt basic syntax and some modules of Python, let's actually solve some simple programming questions in Python to further strengthen your familiarity and comfort level with the language. You will be solving questions from Exercism, a website that allows you to solve a question, test it locally using their test cases and submit the solution to their servers to get feedback from other users. This is great because you get to not only learn programming but also get reviews for your code from others!

For instructions on how to access questions, submit solution etc, read the nice documentation from exercism. You will need a GitHub account.

Note: GitHub also offers several benefits for students such as private repositories and free access to several services through their GitHub student pack. Consider spending time learning how to use git version control system and exploring the GitHub student pack benefits. You will find knowing git is very useful for maintaining backup of programs you write and much more.

You are strongly urged to attempt the questions sincerely on your own before seeking help. The questions are not very difficult and provide good learning for you. You will need to comfortable programming in Python for the assignments and doing these exercises sincerely will help you achieve that comfort level.

After setting up your Exercism account, you can start solving exercises. Here's the exercises that you should complete. You can pass the exercise name as argument to exercism's fetch command. Most exercises have a template solution file with functions that need to be implemented. For those that don't, read the tests file to determine what classes and/or functions need to be implemented.

Sl No.	Exercise		Sl No.	Exercise
1	hello-world		24	etl
2	leap		25	pig-latin
3	isograms		26	simple-cipher
4	pangram		27	scrabble
5	rna-transcription		28	crypto-square
6	hamming		29	pythagorean-triplet
7	word-count		30	matrix
8	run-length-encoding		31	rail-fence-cipher
9	meetup		32	saddle-points
10	rotational-cipher		33	perfect-numbers
11	difference-of-squares		34	secret-handshake
12	anagram		35	palindrome-products
13	allergies		36	bracket-push
14	series		37	wordy
15	sieve		38	phone-number

16	atbash-cipher		39	triangle
17	sum-of-multiples		40	transpose
18	acronym		41	book-store
19	largest-series-product		42	diamond
20	kindergarten-garden		43	all-your-base
21	grade-school		44	tournament
22	space-age		45	protein-translator
23	luhn		46	robot-simulator

Table 1: List of exercises to complete

We also recommend you complete these exercises if possible.

Sl No.	**Exercise**		**Sl No.**	**Exercise**
1	gigasecond		10	queen-attack
2	say		11	poker
3	roman-numerals		12	rectangles
4	grains		13	binary-search
5	prime-factors		14	list-ops
6	sublist		15	clock
7	nth-prime		16	alphametics
8	minesweeper		17	scale-generator

Table 2: List of optional exercises

After having completed all exercises listed in table 1, you should be sufficienty comfortable in programming with Python. We encourage you to complete all exercises on exercism before end of semester.

Good luck with this assignment! If you have any questions, feel free to contact the instructors for help! Please do these assignments on your own. Otherwise, you will not learn programming, which can be a problem when you attempt the upcoming assignments in the course.