בס"ד

**Python:**

Aims---Control of python language, basic syntax, familiarity with Python's features, libraries, ability to search for and use packages, ability to debug python programs, ability to write modules, ability to design, write and run applications in Python ---

Introduction

Python is a programming language. Unlike C#, it is not compiled but interpreted (As you recall, a C# program is run in several stages – the source code, written in .cs files, is linked and compiled into a .exe file, which can then be run on any computer. Python code is written in .py files, and every time it is run a program called an **interpreter** converts the text to code, in run time. This means that in order to run Python programs, you must have an interpreter on your machine).

Python’s syntax is a little different from C#, as you will see in the documentation of the language. One of the most key differences in its syntax is the usage of indents to mark different levels of execution, instead of {} in C#, and the usage of dynamic data types all the time (Python does not have set data types like in C#, each variable can hold any kind of data). Like C#, Python is also object oriented, and allows for creation and usage of objects.

One of the most powerful features in Python is the ability to run parts of code in the interpreter. This means that in order to try and test parts of your code, you do not have to run your entire project. It is enough to type snippets of code into the interpreter program, and it will compute the results immediately (like the JavaScript console available in the developer tools in Chrome, if you are familiar with it).

Python is more intuitive than C#, and is easier to program in. It should not take much time to be comfortable with it; afterwards of course there is much to deepen in its many features. The main reason I think it is better to learn Python is the abundance of open source frameworks and technologies available for it, that are usually more elegant and simpler to use than their C# equivalents.

\*A note on Python 2 and Python 3 (and software versioning in general):

Like all programs, the Python programming language has versions. You should be familiar with this, for example Microsoft Word 2013 is not like Microsoft Word 2003, or Microsoft Word 2018. The internal, professional way to mark program versions is MAJOR\_NUMBER.MINOR\_NUMBER.PATCH\_NUMBER, for example version 3.1.9. There are other ways to number programs, but this is the common one.

The Python version you have will impact the packages you can install and the features the language offers you. Not all features are available in all versions of the program (usually the opposite is true – the main reason for releasing a new build is to incorporate new features and to fix bugs).

Usually, a basic rule in software updates is **backwards compatibility.** This means that if I had a program that worked in Python version 2.3.2, updating to Python version 2.3.4 should not stop my program from running. This is very important (Imagine that after updating to Word 2018 all your previous documents would not open). However, the writers of Python decided, when upgrading to Python 3, that there are changes that are very important to incorporate into the language despite breaking compatibility with previous versions. In effect, this means that Python 2 and Python 3 are not exactly the same language – they are very similar, but source code written for Python 2 will not work with Python 3.

(One of the obvious differences is in the print statement (Python’s Console.WriteLine). In Python 2 the syntax is -- print “Hello world” --, while in Python 3 the syntax is – print(“Hello world”) -- )

This is designed for study of Python 3. However, switching from one to the other is not difficult.

Installation: (30 minutes)

**Install Python Interpreter:** [**https://www.pythontn.org/downloads/**](https://www.python.org/downloads/)

\*Download version Python 3.7.0 – 2018-06-27 (the main version it offers) \*

This is the basic program that will enable you to run Python source code on your system. After installing it, you should be able to press Winkey+r and type python in the little box, and a command line window should appear. This is the interpreter you can type code into. For example, try typing

print(“Hello World!”)

If you have a source file, saved with the ending “.py”, you will be able to run it by typing in a command line window (Winkey+r -> cmd) python filename.py.

**Install IPython:** [**https://ipython.org/install.html**](https://ipython.org/install.html)

\*The installation is meant to be run from command line

Open IPython by typing winkey+r-> ipython. IPython is an upgraded Python interpreter, it is much easier to use. Graphically it uses colors and more spaces than the regular interpreter, however its main power is its ability to let you see the methods available for each class and module. For example, try typing

import os

and then type os. and press tab. You should see a list of all the methods and objects that the os module exports to you.

\*Useful knowledge: ctrl+u in IPython clears the current line, typing exit closes your current session\*

**Install PyCharm :** [**https://www.jetbrains.com/pycharm/download/#section=windows**](https://www.jetbrains.com/pycharm/download/#section=windows)

Pycharm is an IDE for Python, like Visual Studio for C#. It enables you to debug your projects, and in general offers a comfortable working environment. Download the Community Edition ☺

**Basic PyCharm tutorial:** [**https://www.jetbrains.com/help/pycharm/basic-tutorials.html**](https://www.jetbrains.com/help/pycharm/basic-tutorials.html)

This walks you through the first stages of creating and running a Python program with PyCharm. Do only the first two parts of the tutorial: creating a project and debugging your project. PyCharm tip: there are many keyboard shortcuts, that are worth learning as they will speed up your work. Ctrl+shift+a opens a search bar where you can type the action you want and shows you the shortcut for it.

Basic Learning: (2 hours)

The overall concepts of Python and C# are not so different. Seeing as you are familiar with another programming language, the learning curve is to get used to a different syntax, and learning Python’s different features. Go over the tutorial, it covers the basics of the language, and test out interesting parts for yourself using IPython:

This part contains reading materials, while the next sections contain exercises. It is your choice, whether you prefer to do the exercises during your learning or separately.

\* I recommend to skip “Part 2: Using the Python Interpreter”, because it is very technical and confusing. In parts 10 and 11, “A Standard Tour of the Standard Library”, you will find a listing of useful modules. Look over these but you do not need to be familiar with each one of them.

In general, read in high level, you do not need to know every detail described in the tutorial. Try out interesting things, and skip the things that seem less relevant.

[**https://docs.python.org/3/tutorial/**](https://docs.python.org/3/tutorial/)

The library is the documentation for all of Python’s built in libraries and methods. You do not need to go over it entirely, the main thing is to know how to look for what you need – for example, if you need a library to open Excel files in Python, how would you find out if maybe there is a module in Python that does that already? (These are the standard documentations, of course there is a lot of information in general, looking up your questions is a great way to find solutions, you do not necessarily have to look only in these)

[**https://docs.python.org/3/library/**](https://docs.python.org/3/library/)

Exercises:

Basic Usage of the Language: (6 hours)

This contains many different exercises, separated by levels (I don’t know why, but the levels do not appear in order on the page), starting with the basic syntax, creating variables, loops, methods, and ending with getting familiar with important libraries. You do not need to do all the exercises, do as many as you feel are necessary from each part.

I recommend doing no more than 4-5 exercises from each part, unless of course you feel that you need more practice with the concepts introduced. The main purpose of each section is to build familiarity with a certain concept. Keep in mind the time frame for this section – it is about 15 minutes for the shorter sections, and an hour each for the longer sections.

The topics covered are:

**Level 1 – Variables** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SEhl6FBYSgZJLsO9/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SEhl6FBYSgZJLsO9/problems)

**Level 2 – Basic Operators** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1S8SjqcW0zdohNZBP/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1S8SjqcW0zdohNZBP/problems)

**Level 3 – Functions** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SAvEcdraqYALKgnr/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SAvEcdraqYALKgnr/problems)

**Level 4 – If Statements** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SRDZca6jL\_65hQmL/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SRDZca6jL_65hQmL/problems)

**Level 5 – Loops** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SW-67bEcHGOSP6TA/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SW-67bEcHGOSP6TA/problems)

**Level 6 – Strings** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SGN68siAqQ3TmxFF/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SGN68siAqQ3TmxFF/problems)

**Level 7 – Lists** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SLLAnZm-8vFpw3mp/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SLLAnZm-8vFpw3mp/problems)

**Level 8 – Dictionaries** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1S3y1Jb8ghlk1aBYb/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1S3y1Jb8ghlk1aBYb/problems)

**Level 9 – Tuples** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1S636OP4Y7aPSSV9N/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1S636OP4Y7aPSSV9N/problems)

**Level 10 – Library Functions** [**http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SZTsXi1syOPj2OR1/problems**](http://www.singpath.com/#/paths/-JltRrSkwdZ0IWUoxuLG/levels/-Jo1SZTsXi1syOPj2OR1/problems)

Advanced Features of the Language: (2 hours)

Do the **Advanced Tutorials** section. This is a tutorial along with exercises for each part. The exercises are mostly to change existing pieces of code. **Copy the exercises to your own computer**, I recommend not doing the exercises online. Don’t forget, for every concept you have a lot of documentation online, provided you look for it.

[**http://learnpython.org/**](http://learnpython.org/)

OOP: (2 hours)

OOP is a design concept as you know. The knowledge of how to use it should be learned in design. These reference materials are meant to help you understand the ways to implement OOP concepts in Python.

Basic Exercises: <https://realpython.com/python3-object-oriented-programming/>

More reading material: <https://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-programming/>

Advanced Exercises:

Do the SafeFile exercise (it should be in a directory by the same name). This is a summary exercise for Python in general, it contains many concepts you should be familiar with by now.

Bonus: do the UnicodeTranslator exercise. This is an exercise that uses external libraries, and includes research of implementing a feature.