**You do not have to read through these at the start of the course. Come back to these resources as the course progresses!**

**Add your own at the end of the list, by clicking "Show related resources", and rate ones you found useful!**

If you're having trouble with a particular concept or simply want to have access to more information, try one of the following links.

**Documentation**

* [Official Python 3 Documentation](https://docs.python.org/3/library/index.html) - "official"/technical explanation of what a particular function/operator does, examples of correct syntax, what the various libraries are, etc.

**Textbooks/Tutorials**

* [Dive Into Python](https://www.cmi.ac.in/~madhavan/courses/prog2-2012/docs/diveintopython3/index.html) - another survey of Python syntax, datatypes, etc.
* [Think Python](http://greenteapress.com/wp/think-python-2e/) by Allen Downey - a good general overview of the Python language. Includes exercises.
* [The Official Python Tutorial](https://docs.python.org/3/tutorial/) - self-explanatory
* [Learn Python the Hard Way](http://learnpythonthehardway.org/book/) - (note: for Python 2) another free online text
* [Reserved Keywords in Python](https://docs.python.org/3.0/reference/lexical_analysis.html#id8) - don't use these as variable names
* [PEP 8](https://www.python.org/dev/peps/pep-0008/) - Style Guide for Python Code - learn what is good and bad style in Python
* [CheckIO](https://checkio.org/) - learn Python by exploring a game world
* [Invent with Python](https://inventwithpython.com/) - develop your Python skills by making games or hacking ciphers
* [Codecademy](https://www.codecademy.com/learn/python) - (note: for Python 2) learn Python by building web apps and manipulating data; interactive tutorial sequence
* [Python Tutor](http://www.pythontutor.com/) - interactive tutorial sequence of exercises
* [Blog with tutorials](http://mitxcsjourney.blogspot.com/) - created by one of our community TAs

**Debugging**

* [Python Tutor](http://www.pythontutor.com/) - an excellent way to actually visualize how the interpreter actually reads and executes your code
* [DiffChecker](https://www.diffchecker.com/) - compares two sets of text and shows you which lines are different
* [Debugging in Python](https://pythonconquerstheuniverse.wordpress.com/2009/09/10/debugging-in-python/) - steps you can take to try to debug your program

**Software**

* [Python Tools for Visual Studio](https://microsoft.github.io/PTVS/) - Visual Studio plug-in enabling Python programming

**Other Q&A**

* [Stack Overflow](http://stackoverflow.com/questions/tagged/python) - a large Q&A forum for programming concepts (not just Python). Try searching here before you post on the edX forum, and you may find that someone has already answered your question.

**More practice problems**

* [Python Challenge](http://www.pythonchallenge.com/) - a series of puzzles you can try to test your Python abilities
* [Project Euler](https://projecteuler.net/) - additional programming challenges you can try once your Python knowledge becomes stronger; problems are sorted by increasing difficulty
* [Coding Bat](http://codingbat.com/python) - problems you can solve within an online interpreter
* [Codewars](https://www.codewars.com/?language=python) - improve your skills by training on real code challenges