**Initial comments**

**Deadlines**

This course is instructor-led, which means that we have deadlines you'll need to meet, with something due approximately every week. All deadlines are in the [Calendar](https://courses.edx.org/courses/course-v1:MITx+6.00.1x+2T2020a/e9b01dbc537a47aea8f77bed14675329/). Most notably, the midterm and final exam take place over 4 specific days, in an 8-hour time limit of your choosing within those 4 days. All deadlines are in the UTC time zone.

**Our Course Philosophy**

It is our intent that over the entire course, students should be able to get the work done in at most 15 hours a week. If, on average, you are spending lots more than that, let us know.

We also teach computational methods. As such, we may not cover every detail of the Python language. Otherwise, this would become a course on “Introduction to Python”, which it is not. If you are unfamiliar with Python, we have posted many Python resources available online. The forum participants, staff, and community TAs are also willing to help you if you have a question about certain Python constructs.

**Set up your Coding Environment**

Interesting notes:

1. To clear variables (Spyder)

Tools > Preferences > Run > “Clear all variables before execution” (or similar)

2. Modules of interest

-math

-math.log

-math.cos

-math.sin

-math.sqrt

-matplotlib

-numpy

**Resources**

**Textbook/Tutorials**

[Dive Into Python](http://diveintopython3.problemsolving.io/) - 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](https://learnpythonthehardway.org/python3/) - 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

**Common Grader Issues and Fixes**

**Grader Downtime**

It is possible for the automatic graders to go down, especially when there are many people trying to submit code for grading.

**The grader marked your code as INCORRECT**

If you submit code to the grader (by clicking Check) and find you get an Incorrect from the grader, use the “See Full Output” link to the right of the "Incorrect". This will show you the test cases that ran. It will print what your code outputted and what the correct answer code outputted.

They must match exactly, so:

* check spaces
* check capitalization
* check general formatting
* re-read the problem specification
* finger exercises may be submitted as many times as you want, so if you get an answer wrong, try again!
* problem sets have a limited number of attempts so make sure your code is correct (in your own development environment, like Anaconda or IDLE) before you submit it.

**Spinning queue icon**

It is possible that when you submit your code to the grader (by hitting Check), you will get the spinning processing icon. Usually, this should only last a few seconds and you will get a reply back from the grader within those few seconds.

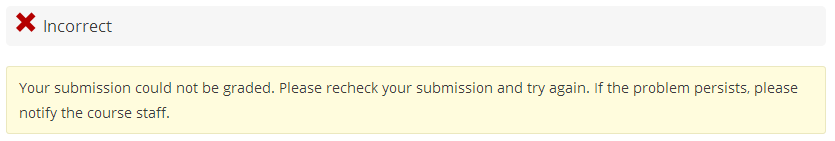
**Infinite loops in your programs**

If you accidentally created an infinite loop and your program sits for a few seconds without printing anything out or terminating, restart the console.

You can do this in Spyder by hitting Ctrl + c (Ctrl key and the c button on your keyboard) in the console. Or go to the button beside the red square at the top of your console and select Restart Kernel. If this doesn't work, restart Anaconda/Spyder.

**Program timed out (Slow Code or Infinite Loops)**

If you see this error, you have an infinite loop in your program (or more rarely, slow code). The grader uses test cases not shown in the problem, so check your code with more test cases. Most likely, there is a path through your code that leads to an infinite loop.

**Submission cannot be graded**

After pasting code from your own working environment and hitting Check, you may see this message (or a similar one inside a yellow box). Those students who use non-ascii characters are most likely to see this. After pasting, some special characters (like accented letters) were introduced. To the grader, they are a sequence of characters (\u200b for example). Go through the code in the textbox and check that all your characters are ASCII (a-z and 0-9 but none with accents). These special characters may appear in bright red font so should be easy to spot.