Introduction to Programming

Matthew X. Curinga

Table of Contents

**Computer Science 0145-602, Fall 2021**

**Keywords:** computer programming, CS1, python, computational thinking, critical computational literacy, jupyter, data science

**Description:** This course introduces students to programming and core concepts of computer science, using a modern, object oriented programming language (currently Python). Students learn concepts of variables, functions, repetition/loops, basic data structures (arrays, lists, dictionaries), and basic object oriented programming.

**Class meetings:** Online, asynchronous (coordinated through Moodle)

**Instructor**

* [Matthew X. Curinga](http://matt.curinga.com), [mcuringa@adelphi.edu](mailto:mcuringa@adelphi.edu)

**Dr. Curinga’s Office Hours by appointment**

* Wednesday, 4:30-5:30PM
* Thursday, 4-5PM

# Learning Goals

* understand the types of problems that can be solved using computational techniques
* understand the basic concepts of computation (CPU, RAM, permanent storage, GUIs, file systems, network connections)
* learn core computer programming concepts (abstraction, variables, conditions, functions, repetition, recursion)
* think algorithmically to design and test computer programs
* master the basic syntax and idioms of the Python programming language
* use technical documentation, APIs, and the internet to learn new technical concepts
* develop step-by-step problem solving and debugging practices

# Required Software

For this class we will be programming in the [Python](https://python.org) programming language, using a development platform called [Jupyter Notebook](https://jupyter.org/).

In order to run this software, it is strongly recommended that you use a computer running Windows, MacOS, or a desktop Linux (e.g. Ubuntu, Debian). If you are working from an iPad, Chromebook, or Android the configuration will be more difficult.

The easiest way to install the latest version of Python 3 and Jupyter is by downloading and running the graphical installer for *Anaconda*. Anaconda is a complete data science platform, but it contains everything we need in a neat package.

[**Click here to find the Anaconda installer for your platform.**](https://www.anaconda.com/products/individual)

# Required Text

*Our textbook is free, open source, and available online.*

Downey, A. B. (2016). [*Think Python: How to Think Like a Computer Scientist, Version 2.4.0*](http://greenteapress.com/thinkpython2/html/index.html). Green Tea Press.

# Required Software & Hardware

For this course we will be programming in the [Python](https://python.org) programming language, using a program/platform called [Jupyter](https://jupyter.org/).

# Class meetings

This is a fully asynchronous online class, which will run on a Wednesday-Wednesday schedule, meaning new topics will begin each Wednesday, and quizzes will be due by end of day on Tuesday. There are no set meeting times, and there will not be Zoom or other video class sessions. You will be able to flexibly schedule your time within the week for each topic.

### Weekly topics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Week | Date | Topic | Read | Due |
| 1 | Sep 01 | The way of the program | TIP 1 | - |
| 2 | Sep 08 | Variables & Statements | TIP 2 | Quiz 1 |
| 3 | Sep 15 | Functions | TIP 3 | Quiz 2 |
| 4 | Sep 22 | Conditionals & Recursion | TIP 5 | Quiz 3 |
| 5 | Sep 29 | Fruitful Functions | TIP 6 | Quiz 4 |
| 6 | Oct 06 | Iteration | TIP 7 | Quiz 5 |
| 7 | Oct 13 | Strings | TIP 8 | Quiz 6 |
| 8 | Oct 20 | Lists | TIP 10 | Quiz 7 |
| 9 | Oct 27 | Dictionaries | TIP 11 | Quiz 8 |
| 10 | Nov 03 | Tuples | TIP 11 | Quiz 9 |
| 11 | Nov 10 | Files | TIP 12 | Quiz 10 |
| 12 | Nov 17 | Word frequency tables | - | Project Thesis & Team |
| - | Nov 24 | *Thanksgiving Break* | - | - |
| 13 | Dec 01 | Visualizing data | - | - |
| 14 | Dec 08 | Working Session | - | - |
| 15 | Dec 15 | Final Projects Due | - | **Final Project** |

TIP: *Thinking in Python*

### Live labs

In addition to the required weekly assignments, there will be several “live” labs on Wednesdays (4:30-5:30). These live **sessions are optional**, but will offer additional help and hands-on demonstration of course concepts. These sessions will be hybrid/flexible (online/in-person) sessions held at the Manhattan Center.

# Assignments and Grading

|  |  |
| --- | --- |
| Assignment | Pct |
| Quizzes | 60% |
| Final Project | 40% |

## Lab Exercises

Most weeks there will be ungraded lab exercises where students can practice the new materials covered. In general, you should spend about one hour working on these exercises. If you understand the exercises, you will be on track with the course. There may be bonus problems that are a little bit more challenging, which are optional. You are encouraged to work on the exercises with other students and friends.

## Quizzes

Quizzes will consist of 1-3 questions, similar to the lab exercises. Each quiz is worth a total of 6 points. Earlier in the semester, the quizzes will have up to 6 questions each, with multiple choice and short answer questions. After we develop more skills writing code, quizzes will consist of a single question and you will copy-paste your solution or upload your source code to Moodle.

Quizzes will be timed, taken through Moodle. You will have 45 minutes once you begin the quiz to submit your answers. Before you attempt the quiz, make sure that you are ready to proceed – you will not be able to pause the quiz once it begins. You can use the textbook, course examples, and any documentation or internet resources you find. **You may not ask other people for help.** While programming is a highly collaborative practice, these quizzes are meant to assess *your* work and understanding.

## Final Project: Content Analysis

During the course of the semester we will be learning about the software design process, and have the opportunity to write a larger program. The final project is a group project, and you should work in a group of 2-4 students.

Content Analysis is a research approach that uses statistical methods to analyze qualitative data. The skills we develop in our first semester of programming will be enough to allow us to conduct our own, novel content analysis research.

We will be focusing on the analysis of textual data, with examples looking at the content found in works of literature and in political speeches.

### Deliverables:

1. **Code Files and Resources**  
   You should turn in all files related to your project in a single archive (e.g., .zip, .tar, .7z). Project files *must* contain your **Python program** in the Jupyter Notebook which runs the main loop of your program and produces the output (tables, graphs, text) which “answers” your research question.

* In addition to the source code and output, you should include any data files, media, or supporting source code your project requires.

1. **Group Video Demo**  
   Your group should provide a narrated screencast or video of 3-4 minutes that demonstrates your program running and explains in greater detail the goals of the program and the results it produces.
2. **Individual Video Walkthrough**  
   Each group member will create their own video (uploaded to their Moodle account only). This video walkthrough of the code should highlight your specific contributions to the project, focusing on key aspects of the code where you were the sole or lead author. In particular, your video should highlight your understanding of the key concept of *abstraction* in computer programming, and how you used it to design your program. See the more specific *coding fundamentals* below for ideas of what to highlight in your video. Videos should be ~5 minutes in length.

### Final Project Grading:

1. **Concept & Design** (*5 points*)  
   Does the project demonstrate a good match between the type of question you explore and the type of answers that computer analysis can provide?
2. **Coding fundamentals** (*20 points*)  
   The program demonstrates a grasp of the programming concepts covered in this class, including:
   * *variables*
     + data is not “hard coded” and can be easily changed by using variables, soliciting user input, and/or reading from files
     + data is separate from functionality
   * *functions*
     + abstraction through function parameters
     + encapsulation through function parameters and return statements; use “pure functions” with zero-side effects when possible
     + composition and re-use of code
   * *design*
     + the program is organized through the use of functions
     + functions’ “scale” is appropriate to the task and discrete: concerns are separated logically, such as one function for gathering results and another for outputting results
     + functions are written in a way that they are used several times in the program
     + code is not repeated
   * *data structures*
     + use python built-in data structures appropriately: list, dict, tuple, set, etc
     + use index/slice notation if needed
     + sort data structures
     + map, filter data as needed
   * *style*
     + is the code style consistent throughout the program?
     + does the code adhere to the style conventions discussed in our readings?
     + are variables and functions named in a clear way?
     + are comments and docstrings included to clarify the program?
   * *testing*
     + does the program include test functions to ensure the program is working as expected?

* Each team member will receive an individual grade for this portion of the project.

1. **Risk Taking** (*5 points*)  
   How “adventurous” is this project? Does the team show that they move beyond the template given to them? Do they come up with a really novel and desirable project? “Riskier” project push beyond the material strictly covered in class and demonstrate the teams’ ability to learn new things and push their horizons.
2. **Results** (*10 points*)  
   How well does the program achieve its goals? Are the results clearly presented in the Notebook file?