Introduction to Programming

Matthew X. Curinga

Table of Contents

**Computer Science 0145-602-001, Fall 2023**

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

**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

# 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 and Accounts

1. [Create an account on Runestone](https://runestone.academy/runestone/default/user/register), the host of our interactive textbook. This will allow you to read the book ad-free and to save your place.
2. [Join our Slack](https://auedtech.slack.com/) with your mail.adelphi.edu email.
   * install the desktop client so that you can easily share code and screenshots
   * install the mobile client so that you can stay tuned for messages about the class
   * join the #code channel for discussions related to this class
   * DM the instructor at @mxc to get in touch
3. [Chat GPT](https://chat.openai.com/). Since it’s release, I have used ChatGPT as a resource for my own software development projects. I think it will be very beneficial for you, too. Create an account at <https://chat.openai.com/auth/login> to get started. You can use your AU email, but that’s not required.
4. [**Visual Code Studio aka VS Code**](https://code.visualstudio.com/). For this class we will be programming in the [Python](https://python.org) programming language, using an [Integrated Development Environment (IDE)](https://en.wikipedia.org/wiki/%20Integrated_development_environment) called CS Code. This software will allow you to write your Python code in a programmer’s text editor, run you code to see the results, and to run instructor-provided test code to verify your solutions. You will probably use the online programming environment included with the textbook for the simple textbook exercises, but you will want to use VS Code for the more complex programs and to make better screen recordings for your portfolio. Follow the reference materials below for instructions on how to install VS Code and Python for your operating system.
5. **Screenshot software.** To get help, you might need to share a screenshot (more often you will copy-paste code or error messages). Don’t take pictures of your laptop with you phone. Take a screenshot. If you need help setting this up or getting recommendations, ask on #code on slack.
6. **Screen recording and video editing.** Your grades in this class are portfolio based; based narrated screencasts you make of your code and problem solving, where you demonstrate your mastery of key concepts in computer science. Like screenshot software, there are many solutions making screen recordings and editing videos. **Mac users** will be able to use the combination of [Quicktime Player](https://support.apple.com/en-us/HT208721) and [iMovie](https://www.apple.com/imovie/). **Windows users** don’t have quite the same power built in, but Microsoft offers screen recording with the [XBox Toolbar](https://www.theverge.com/2020/4/21/21222533/record-screen-pc-windows-laptop-xbox-game-bar-how-to) and video editing with its [Clipchamp](https://support.microsoft.com/en-us/windows/create-films-with-a-video-editor-94e651f8-a5be-ae03-3c50-e49f013d47f6) application. I recommend [Open Broadcaster Studio (OBS)](https://obsproject.com/) for screen recordings (it works on Mac too). I use [Davinci Resolve](https://www.blackmagicdesign.com/products/davinciresolve/) for editing video – it’s free and cross platform – but it’s full featured and there’s a bit of a learning curve.

# Required Text

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

Miller, B. & Ranum, D. (n.d.) Based on work by Jeffrey Elkner, Allen B. Downey, and Chris Meyers. [*How to Think Like a Computer Scientist: Interactive Edition*](https://runestone.academy/ns/books/published/thinkcspy/index.html)

All reading assignments and exercises are from this book. It is abbreviated TIP (Thinking in Python) in the course syllabus.

# Reference Materials

*Consult this documentation as needed.*

* [[Python Documentation](https://www.python.org/doc/)] official python language docs
  + [[tutorial](https://docs.python.org/3/tutorial/index.html)] basic tutorials
  + [[library reference](https://docs.python.org/3/library/index.html)] reference of the standard libraries
  + [[style guide](https://peps.python.org/pep-0008/)] naming variables, spaces, quotations, comments, etc.
* [VS Code](https://code.visualstudio.com/docs)
  + [Getting Started](https://code.visualstudio.com/docs/getstarted/introvideos)
  + [Source Control](https://code.visualstudio.com/docs/sourcecontrol/overview)
  + [Python in VS Code](https://code.visualstudio.com/docs/python/python-tutorial)

# Class meetings

This is a fully asynchronous online class, which will run on a Wednesday-Wednesday schedule, meaning new topics will begin each Wednesday. 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. As a 3 credit graduate course, you should plan to spend approximately eight hours each week working on materials for this course. This includes assigned readings, videos, programming exercises, group/peer meetings, and tutoring sessions.

### Weekly topics

| Week | Date | Topic | Read | Due |
| --- | --- | --- | --- | --- |
| 1 | 08/30 - 09/05 | The Way of the Program | TIP 1 |  |
| 2 | 09/06 - 09/12 | Data & Variables | TIP 2 |  |
| 3 | 09/13 - 09/19 | Turtle Graphics | TIP 3 & 4 |  |
| 4 | 09/20 - 09/26 | Python Modules | TIP 5 | Portfolio 1 |
| 5 | 09/27 - 10/03 | Functions | TIP 6 |  |
| 6 | 10/04 - 10/10 | Selection | TIP 7 |  |
| 7 | 10/11 - 10/17 | Iteration: for & while | TIP 8 |  |
| 8 | 10/18 - 10/24 | Strings | TIP 9 |  |
| 9 | 10/25 - 10/31 | Lists | TIP 10 | Portfolio 2 |
| 10 | 11/01 - 11/07 | Files | TIP 11 |  |
| 11 | 11/08 - 11/14 | Dictionaries | TIP 12 |  |
| 12 | 11/15 - 11/21 | Exceptions | TIP 13 |  |
| 13 | 11/22 - 11/28 | Recursion | TIP 16 |  |
| 14 | 11/29 - 12/05 | Objects and Classes | TIP 17 |  |
| 15 | 12/06 - 12/12 | Portfolio work | - |  |
| 16 | 12/13 - 12/19 | Final portfolio | - | Portfolio 3 |

### Tutoring

The Adelphi Learning Center offers [individual and group tutoring](https://www.adelphi.edu/learning-writing-centers/tutoring/), which can be either in person or online, scheduled through their website. This is an excellent, free service and you might want to schedule a session to go over some of the labs. In addition, Math and Computer Science has free, drop-in tutoring sessions on weekday afternoons in the Garden City campus. They may also post some Zoom sessions. I will post the schedule and details on the course website after the semester starts.

### Study Group

Everyone is assigned to a 3 or 4 person study group. You should set up a text or Slack channel for your study group so that you have a few people that you can reach out to when you get stuck or need help. It’s highly recommended that you regularly work on weekly exercises with your study group and that you share and get feedback on your portfolios with this team before you submit them for grading. Your study group assignment is available on the course website.

# Assignments and Grading

| Assignment | Pct |
| --- | --- |
| Portfolio 1 | 30% |
| Portfolio 2 | 30% |
| Portfolio 3 | 40% |

## Chapter Exercises

Each week will have a chapter (or 2) assigned in *How to Think Like a Computer Scientist*. You are **required** to work on the exercises at the end, but they are not graded and you do not need to submit your work. You will draw on the code your write for your portfolios.

## Programming Portfolios

Your work and progress in this course will be evaluated based on 3 portfolios that you will submit as a video screencast. In each portfolio you will use work that you’ve done in the course (chapter exercises and challenge problems) to demonstrate your knowledge of key ideas. Your portfolio must show code that you have written, which you use to explain the key concepts for each portfolio. The code samples that you choose must be from chapter exercises in *How to Think Like a Computer Scientist* or challenge problems posted on the course website.

#### Video guidelines

This is not a video production class, so you are not expected to create a polished video with high production quality. However, follow these tips to make a good video:

1. **Make good use of the time.** Write an outline for your video before recording. Open all of the documents (code, browser tabs) you need. Practice what you are going to say.
2. **Record in high definition (aka HD, or 1920x1080 or higher)**. Since reading text on the screen is key to the portfolio, make sure that you are making a high resolution video so that the text isn’t pixelated/blurry.
3. **Record clear audio**. Test out your mic before you record the whole thing and make sure that your audio is coming through clearly. The code on screen pluse your narration *are* the portfolio.
4. **Lightly edit.** It will be hard to make the video all in one take. Consider using video editing software to join clips together, add still screenshots, and edit out dead time. *Don’t* feel obligated to do anything fancy (background music, fast/slow motion, titles, credits, transitions).
5. **Include code you wrote.** Just to be clear, your video should include code that you have written. It is up to you if you want to show several different exercises, a larger program from a challenge problem, or something else that you have written. **Do not show somebody else’s code.** If you include code that you didn’t write (from the book, written by an AI, found online, written by a friend) please indicate that in the comments and in your narration. Passing off someone else’s work as your own is clear academic dishonesty and will lead to a zero for this assignment and possibly further disciplinary action.

#### Late submissions

If you submit your portfolio after late (see due date and time on Moodle), you will lose 2 points. You will lose 1 additional point for each day it is late after the first 24 hours.

#### Submit your portfolio

1. Upload your video to YouTube [[how to](https://support.google.com/youtube/answer/57407?hl=en&co=GENIE.Platform%3DDesktop)].
   * Login with your mail.adelphi.edu account *or* your regular YouTube account.
   * Set the [visibility of your video](https://support.google.com/youtube/answer/157177?hl=en&co=GENIE.Platform%3DDesktop):
     + unlisted (recommended) - anyone with the link can see it
     + private (you must share with mcuringa@adelphi.edu)
     + public (people can search for it and find it on the web and on youtube)
2. Copy the link and submit it on Moodle
3. Optional: put source code files (.py python files) in a .zip archive and submit with your link
4. Optional: post your video on Slack for others to check out

### Portfolio 1

Your first portfolio covers chapters 1-5 in *How to Think Like a Computer Scientist*.

*Key concepts for portfolio 1:*

1. **algorithm:** your own definition of an algorithm and an example of an algorithm that you have written.
2. **debugging:** a demonstration of you debugging your code. Interpret the error message you see, and discuss the *type* of error (syntax, runtime, semantic).
3. **variables:** including understanding data types, assignment, re-assignment
4. **built-in functions:** how to call Python built-in functions using arguments and working with return values
5. **style and organization:** what makes a good variable name? how do comments work? what decisions did you make to write code that is easily understood by humans?
6. **for loops:** what is *repetition*? what are the key aspects of for loops? how to you repeat code n (e.g. 4 times)? how do you **iterate** over a list?
7. **modules:** what are modules or libraries in computer programming? how did you use the math, turtle, random, or other modules in your code example?
8. **learning:** what have you learned? anything interesting, surprising, challenging? anything you are looking forward to learning?

In addition to covering the content above, to receive full points for Portfolio 1, you *must* demonstrate your code running in VS Code (1 point) and that you understand how to use Chat-GPT for feedback, documentation, or general help (1 point).

This video **must be between 7 and 10 minutes** long. If your video is too long or too short, you will lose 2 points on the final score.

### Portfolio 2

Your second portfolio covers chapter 6-10, but will also draw on concepts in chapters 1-5. Specifically, your portfolio must include:

1. **function parameters:** what are parameters? what are arguments? who are parameters different from variables?
2. **return statement:** demonstrate that you can write functions with return statements by highlighting code that you have written that uses return when the exercise prompt did not tell you what value should be returned.
3. **selection:** describe the use of if, else, and elif in your code. Point to examples that use return instead of conditional statements. Demonstrate the use of a *boolean function*.
4. **while and for:** when should we choose to use while loops and when is a for loop more useful?
5. **index notation:** demonstrate code that you used to solve a problem using string index notation and slices.
6. **string methods:** what’s a *method*? demonstrate code that solves a problem using the methods of the Python string class.

## Portfolio 3

Your final portfolio demonstrates the knowledge and skills that you developed during the semester. It covers the content in chapters 11, 12, 13, 16, & 17. Your main goal for this portfolio is to demonstrate that you’ve mastered the key problem solving principles you’ve been working towards, and that you can conceive, design, and code Python programs to solve basic problems.