INST326: Object-Oriented Programming
Spring 2019
Section 102 • TuTh 12:30-1:45 pm • ~~KEB 1200~~ TWS 1310
Section 104 • TuTh 3:30-4:45 pm • SQH 1117

Instructor

Aric Bills • abills@umd.edu • (301) 405-3728 • Patapsco Building, suite 1110

Office hours (subject to change): TuTh 5:00-6:00 pm, Hornbake 4114; or by appointment

Catalog Description

This course is an introduction to programming, emphasizing understanding and implementation of applications using object-oriented techniques. Topics to be covered include program design and testing as well as implementation of programs. *Prerequisite: (must have completed or be concurrently enrolled in INST201; or INST301); and (INST126; or CMSC106; or CMSC122). Or permission of instructor. Credit only granted for: INST326 or CMSC131.*

Extended Course Description

This course introduces object-oriented design and programming concepts and methods using the Python programming language. Object-oriented programs are built as collections of “objects”, which are software representations of real-world entities and concepts. Objects combine data (attributes) with functionality (methods), and work through communicating with each other as the code is executed. By encapsulating code complexity within objects, OOP allows use and reuse of existing code in a relatively simple and easy manner. Advanced OOP concepts such as inheritance facilitate development of complex code without sacrificing robustness and possibility of code reuse. We apply computational thinking approaches such as abstraction, decomposition, algorithmic design, generalization, evaluation, and debugging.

This course also provides opportunities to develop an understanding of how programming is situated in and reflects broader social structures, constructs and issues, e.g. race, class or gender. Programming is often viewed as a value-neutral technical skill. However, the social and cultural impacts of information and technology are central concepts in our field, and the growing awareness of issues like algorithmic bias, ethical/unethical uses of algorithms and disparities in opportunties in tech jobs require that any informed professional needs to understand the larger context of programming. This is important to be ethical professionals and to be successful in the workplace. Through readings, discussion and writing, we will critically examine issues of racism, sexism and other forms of power and oppression that are pervasive in programming and related technical activities, and discuss what companies and individuals are doing to improve programming practices and professional work environments.

Student Learning Outcomes

After finishing this course, students will be able to:

1. Explain OOP concepts, principles, design patterns and methods;
2. Design, program, and debug Python applications to solve non-trivial problems;
3. Test and assess the quality of object-oriented code;
4. Write clear and effective documentation;
5. Explain how programming is situated in and reflects social issues (e.g., racism, classism, or sexism) and describe actions that individuals or organizations are taking to counteract disparities or inequities.

Teaching Notes

This course builds on a basic understanding of procedural programming, so you have to understand data types, variables, loops, conditionals, etc. and how to use them to write and debug a program. If you are fluent in a language such as JavaScript, Java, C#, Visual Basic, etc. you can readily apply your knowledge to learn Python. If you know a bit of Python already, you might find the first part of the course a bit of review. If you are interested in being challenged, I invite you to talk to me about leading a session (it’s really true that you learn more by teaching), identifying more challenging exercises, or developing a more ambitious project. I want you to learn as much as you can from this course.

Each week will typically follow this pattern, with some exceptions:

Before class (preparation):

- Do assigned readings; watch assigned videos; complete any exercises, homework assignments or quizzes which are due.

In class:

- We will use a mix of lecture, discussion, and lots of hands-on activities to help you apply the materials;

- We will make extensive use of paired and group work in class.

After class (homework):

- There will be weekly assignments to help you practice, reflect, and extend your understanding. All homework assignments are to be completed on your own unless otherwise stated on the assignment handout.

Over the course of the semester, we will also examine selected broader issues of programming and coding – the social and organizational context, issues related to gender, race, disability, etc. This will help you prepare for situations that you are likely to encounter in your professional work. These are noted in the schedule as "Critical perspectives."

Our time together in class is precious. To use it effectively, you must come to class on time and prepared. Being prepared for class means that you have:

1. Completed all the readings/videos;
2. Either successfully completed the exercises/worksheets or submitted your questions the night before class, so I have time to prepare and answer them in class.
3. Arrived 5 minutes before class starts and followed any instructions posted in the classroom. You are ready to take notes and/or participate in a class activity, as instructed.

Here is my suggested general strategy for working on assignments:

1. Start early–don't wait. That will give you time to work through the problems and get help as needed.
2. When you run into a problem, spend 5-10 minutes trying to solve it on your own. Sometimes it's helpful to ask yourself how you would solve the problem on paper or using real-world objects.
3. Then take a break. Sometimes this will allow you to come back and see something you missed. Letting your subconscious work on it for a while (unsupervised, so to speak) will often lead to useful ideas.
4. If you've spent 20-30 minutes and are still stuck, post your question on ELMS. We are here to help each other, so don't beat your head against a brick wall–ask for help! When you post, provide as much information as you can. When helping your fellow students, please do not do their work for them. Help them understand underlying principles and programming techniques and let them arrive at solutions to homework problems on their own.
5. I will be monitoring and will respond as soon as I am able, usually within a day (longer during weekends, travel, etc.).
6. If you see a question on the discussion board that you can answer, or if you have an idea, please respond. Don't wait for me. You will be helping your colleagues.

Textbooks and Readings

Our readings will come from a variety of free online sources; the main ones are:

- ***Python for Everybody: Exploring Data Using Python 3***
  Charles R. Severance
  https://www.py4e.com/book
  Download: http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf
- ***Object-Oriented Programming in Python***
  University of Cape Town
  http://python-textbok.readthedocs.io/en/1.0/
  Download: https://media.readthedocs.org/pdf/python-textbok/1.0/python-textbok.pdf
- ***The Python Tutorial***
  Guido van Rossum and the Python Software Foundation
  https://docs.python.org/3/tutorial/
  Download: https://docs.python.org/3/download.html (as part of the official Python documentation)

Note that all of these sources can be downloaded for offline access. Please take a moment and download copies now, so that you're prepared in the event of internet issues during the semester.

Required Technology

- Laptop - we will do live programming exercises during most classes, so bring your laptop and be prepared to write code. Any reasonably current operating system can be used. If you don't have access to a laptop, contact me before the first class.
- Python - Python programming language (3.4 or newer). Python is freely available from https://www.python.org/downloads/.
- Editor - Please install either Atom (https://atom.io/) or Geany (https://www.geany.org/). Atom is powerful and highly customizable, but resource intensive. Geany is fast and lightweight.

Topics

Topics to be covered include:

- Computational thinking
- Programming patterns
- Variables, expressions, statements

- Conditionals
- Functions
- Iteration
- Strings
- Lists
- Dictionaries
- Tuples
- Classes, objects, methods
- Critical perspectives on programming, which may include:
  - Sociotechnical systems
  - Limitations of computational thinking
  - Coding and gender
  - Search engine bias, algorithmic bias

Grading

Every graded element of the course (e.g., assignments, tests, quizzes, etc.) is assigned to one of the following weighted categories:

Category	**Weight**
Homework (7)	20%
Midterms (2)	20%
Quizzes (5; one gets dropped)	16%
Final project	15%
Exercises	15%
Reflections (3)	9%
Participation	5%
Final exam (optional)	3% (extra credit)

Your grade will be calculated as follows:

- for each category, the sum of your scores is divided by the sum of possible points within the category; this is the proportion of points you have earned for that category
- the proportion of points in each category is multiplied by the category's weight; this is the weighted score for the category
- the sum of the weighted scores is your total score
- your total score is converted into a letter grade according to the table below:

A+: >= 97.00	A: 93.00-96.99	A-: 90.00-92.99
B+: 87.00-89.99	B: 83.00-86.99	B-: 80.00-82.99
C+: 77.00-79.99	C: 73.00-76.99	C-: 70.00-72.99
D+: 67.00-69.99	D: 63.00-66.99	D-: 60.00-62.99
F: 0.00-59.99		

Course Schedule (subject to change)

This course consists of three interconnected parts:

- Fundamentals of programming using Python (~Weeks 1-6)
- Data analysis using Python (~Weeks 7-9)
- Object-oriented programming using Python (~Weeks 10-15)

The table below shows the most current version of the planned course schedule. While this is our planned timeline, please note that this schedule is subject to change at any time. Any revisions to this timeline will be published to ELMS; appropriate announcements will be made in class and posted online accordingly.

Note that all work (homework, reading, exercises, reflections, etc.) is due by 11:59 PM the day before the class period in which it appears below. (The final project is an exception; it is due by 11:59 PM on the day of the scheduled final for your section.) See "Late Work and Resubmissions" below for more information.

Wk	Tuesday	Thursday

Wk	Tuesday	Thursday
1	1/29 Course overview	1/31 Python basics; functions Due: Exercise 1
2	2/5 Functions, docstrings, modules, testing Due: Homework 1	2/7 Iteration Due: Exercise 2 In class: Quiz 1
3	2/12 Strings Due: Exercise 3	2/14 File I/O; command-line arguments Due: Exercise 4
4	2/19 Lists Due: Homework 2 In class: Discuss reflection 1	2/21 Tuples Due: Exercise 5 In class: Quiz 2
5	2/26 Dictionaries Due: Reflection 1	2/28 Sets Due: Exercise 6
6	3/5 Putting it together Due: Homework 3	3/7 Installing Pandas and Matplotlib In class: Midterm 1
7	3/12 Pandas Due: Take-home midterm In class: Discuss reflection 2	3/14 Pandas Due: Exercise 7
Spring break	3/19	3/21
8	3/26 Pandas Due: Reflection 2 In class: discussion of final projects	3/28 Matplotlib Due: Exercise 8
9	4/2 Matplotlib Due: Homework 4 In class: formation of final project groups	4/4 Team programming Due: Exercise 9 In class: sign up for proposal check-in meetings

Wk	Tuesday	Thursday
10	4/9 Project proposal check-in Due: Project proposal *No class today; attend scheduled meeting with instructor; take quiz 4 online*	4/11 Intro to OOP (classes, attributes, methods) Due: Exercise 10
11	4/16 OOP; self-reference Due: Homework 5	4/18 Encapsulation, abstraction, inheritance, polymorphism Due: Exercise 11 In class: Quiz 5
12	4/23 OOP exercise Due: Homework 6 In class: Discuss reflection 3	4/25 OOP exercise Due: Homework 7 (project milestone)
13	4/30 Project check-in; in-class team work session Due: Reflection 3 *You are expected to come to class today*	5/2 Testing In class: Midterm 2
14	5/7 Testing Due: take-home midterm	5/9 Project presentations
15	5/14 Project presentations	
Finals week	Tuesday 5/21 (section 102) Wednesday 5/22 (section 104) Final exam at regularly scheduled time (optional; 1:30 PM [section 102]; 10:30 AM [section 104]); final projects due by 11:59 PM	

University Course Policies

The essential purpose of the university's undergraduate course policies is to enable all of us to fully participate in an equitable, accessible and safe academic environment so that we each can be challenged to learn and contribute most effectively. They address issues such as academic integrity, codes of conduct, discrimination, accessibility, learning accommodations, etc. We are all responsible for following the policies at http://www.ugst.umd.edu/courserelatedpolicies.html. You must read them and send me any questions by the first week of classes.

Academic Integrity and Ethical Use of Other People's Work

In academia and in computer programming, building on the work of other people is often accepted and encouraged. In this class, there will be some situations in which it is appropriate (sometimes even necessary) to build on other people's work. For example:

- you may get help from fellow students to understand a particular concept or technique
- you may collaborate with another student on a project that has been designated as a group assignment
- you may want to use a function or an algorithm published on a website
- you may find it useful to share ideas from a journal article in a reflection you are writing

For the purposes of this class, the following principles govern the ethical use of other people's work:

- You have an obligation to produce original work to satisfy the learning objectives of each assignment. Other people's work should complement, not replace, your own work.
- You should always give credit to individuals whose work you use. In a written document such as a reflection, this means providing a full entry in your bibliography as well as an in-text citation. In code, you should provide a comment indicating the following details:
  - the source of the code
    - URL (if online) or bibliographic citation (if not)
    - whatever authorship information is available
  - the date you accessed it
  - if applicable, the version number and title of the code

You are expected to complete all course work (homework, exercises, quizzes, midterms, reflections, etc.) on your own unless my written instructions on a particular task indicate otherwise. You may not discuss exams or midterms with anyone until the deadline for submitting the exam or midterm has passed for all participants in the discussion (remember, due to personal circumstances, some students may have a different deadline than you). You may discuss exercises and homework with other students; this includes explaining underlying concepts, assisting a fellow student in debugging (without supplying your own code to that student), and discussing algorithms. If you collaborated with one or more fellow students in one of the ways described above, your code must include a comment describing the collaboration and citing all collaborators. **Please note: under no circumstances are you allowed to copy/paste, retype, or work off of someone else's code unless the assignment instructions include explicit written instructions to the contrary.**

UMD students are required to abide by the student honor pledge: *I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.* You will be asked to complete the honor pledge as part of each assignment, quiz, and test in this class.

Suspected cases of cheating, plagiarism, or other academic integrity violations will be referred to the Honor Council.

Late Work and resubmissions

I do not accept late work unless I have approved it by prior arrangement. If you have to miss a deadline, you should inform me as soon as possible, indicating the reason and when you propose to submit your work. If you have a legitimate reason, such as a major medical or family emergency, I may agree to an extension or makeup work, which I will grade by the end of the semester. Documentation of the emergency (e.g., a doctor's letter) may be required.

Homework resubmission

For Homework 2, 3, 4, 5, and 6, after the assignment is graded, you will be given an opportunity to resubmit for up to 50% of the points you missed on the first submission. For example, if you received 2 out of 3 points on your first submission, you would be eligible to resubmit for up to 0.5 additional points. In order to receive points for your resubmission, you must fix the issues for which you lost points on the first submission.

Exercise revision

Exercises cover material that you are assigned to read but that we have not covered in class. You are expected to submit your best attempt on each exercise by the deadline; at a bare minimum this should include the code you wrote (even if it doesn't work) and your questions or an explanation of what you didn't understand. You will be given 48 hours past the deadline to resubmit the exercise before it is graded; this gives you a chance to revisit the exercise after we discuss the material in class. Students who fail to make a "bare minimum" submission by the assignment deadline can still submit within the 48-hour revision window, but they forfeit half the possible points for that exercise.

Syllabus Revision Policy

This syllabus is a guide for the course and is subject to change with advance notice. Changes will be posted in ELMS. The ELMS course site is the definitive location for all course work, and communications, including class schedules, assignments, and deadlines.

Last updated Feb 1, 2019. Version 2019s1.2.