INST 326, Object-Oriented Programming for Information Science
Course Syllabus

Fall 2018, Section 0104
Tu - Th 9:30-10:45
1204 John S. Toll Physics Building

Instructor

Douglas K. Lay
Email: douglay@umd.edu
Phone: 410-302-3562 (cell)
Office Hours: 4121M Hornbake, 11-12 Tuesday 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.

Prerequisites: 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 opportunities 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:

- Explain OOP concepts, principles, design patterns and methods;
- Design, program and debug Python applications to solve non-trivial problems;
- Test and assess the quality of object-oriented code;
- Write clear and effective documentation;
- 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 and 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 worksheets, 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 homework 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 hand-out.

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:

- Completed all the readings/videos;
- 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;
- Arrived 5 minutes before class starts; are in your seat, with your IDE or text editor running. You are ready to take notes.

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.
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 still are 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. Often it helps to post a screenshot with the problem.
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 & 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.php
Download: http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf

The Python Tutorial
Guido van Rossum and the Python Software Foundation
https://docs.python.org/3/tutorial/index.html
Download: https://docs.python.org/3/download.html

Object-Oriented Programming in Python
University of Cape Town
https://python-textbok.readthedocs.io/en/1.0/
Download: https://media.readthedocs.org/pdf/python-textbok/1.0/python-textbok.pdf

Note that all of the above sources may be downloaded for offline access. It is recommended that you download them as soon as possible so that you are prepared in the event of Internet connectivity problems during the semester.

Additional readings will be assigned during the semester.

All readings will be added to the ELMS class schedule and announced in class at least a week before the reading is due.

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 (Windows, OSX or Linux) can be used. If you don't have access to a laptop, contact me as soon as possible.
- Python - Python programming language (version 3) platform that supports object-oriented programming. The programming platform is freely available from.*
- Editor - An advanced text editor (such as Aton, Sublime Text, Notepad++, Crimson Editor) and/or an integrated development environment (such as NetBeans, Eclipse, PyCharm, Geany).*

** Please note that we will install all necessary environments together in class during the first week.*

Topics

Topics to be covered include:

- Computational thinking
- Programming patterns
- Variables, expressions, statements
- Conditionals
- Functions
- Iteration
- Strings
- Lists and Tuples
- Dictionaries
- Classes and objects; methods and attributes
- Encapsulation
- Composition
- Inheritance
- Critical perspectives on programming, including:
  - Sociotechnical systems
  - Limitations of computational thinking
  - Coding and gender
  - Search engine bias, algorithmic bias

Grading

Your final grade for the course is computed as the sum of your scores on the individual elements below (100 possible points total), converted to a letter grade:

A+ 97-100	B+ 87-89.99	C+ 77-79.99	D+ 67-69.99	F 0-59.99
A 93-96.99	B 83-86.99	C 73-76.99	D 63-66.99	
A- 90-92.99	B- 80-82.99	C- 70-72.99	D- 60-62.99	

Final grades will be calculated based on the following components:

Homework(9)	27 points
Quizzes (4)*	20 points
Midterms (2)	20 points
Final Project (1)	16 points
Reflections (3)	12 points
Participation	5 points
TOTAL	**100 points**
Final Exam (optional)	up to 3 points extra credit

** There will be five (5) quizzes. Your lowest quiz score will be dropped from the final grade calculation.*

Course Schedule (subject to Change)

The following table shows the most current version of the planned course schedule. This course is intended to be delivered via three interconnected modules:

Module 1 - Fundamentals of Python Programming (Weeks 1-6)
Module 2 - Data Analysis using Python (Weeks 7-9)
Module 3 - Object-Oriented Programming using Python (Weeks 10-15)

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.

Week	Tuesday	Thursday
1	*8/28* Introductions; Course Overview; Best Practices	*8/30* Data types; Variables; Operations; Expressions **Reading:** P4E 1-2; Style Guide (ELMS); Docstrings page (ELMS)
2	*9/4* Conditional execution; functions; modules **Practice Quiz** (in class) **Reading:** P4E 3-4; Tut 4.7.1-4.7.2; 6-6.1.2 **Homework 1 Due**	*9/6* Iteration **Quiz 1** (in class) **Reading**: P4E 5
3	*9/11* Strings **Reading:** P4E 6	*9/13* File I/O; command-line arguments **Reading:** P4E 7; Tut 7.2, 7.2.1 **Homework 2 Due**
4	*9/18* Putting It Together	*9/20* Critical Perspectives 1 discussion **Quiz 2** (in class) **Reading:** Critical Perspectives 1 (TBD)
5	*9/25* Lists and Tuples **Reading**: P4E 8, 10.1-10.3, Tut 5.3 **Homework 3 Due**	*9/27* Lists and Tuples continued; Sets **Reading:** TBD **Reflection 1 Due**
6	*10/2* Dictionaries **Quiz 3** (in class) **Reading:** P4E 9, 10.4-10.9	*10/4* Putting It Together **Homework 4 Due**
7	*10/9* Introduction to Data Analysis **Reading**: TBD **Homework 5 Due**	*10/11* Critical Perspectives 2 discussion **Midterm 1** (in class and take-home) **Reading**: Critical Perspectives 2 (TBD)
8	*10/16* Data Analysis with Pandas **Reading**: TBD **Midterm 1 Take-Home Due**	*10/18* Data Visualization with Matplotlib **Reading**: TBD **Reflection 2 Due**

Week	Tuesday	Thursday
9	*10/23* Remote APIs and JSON **Reading**: TBD **Homework 6 Due**	*10/25* Putting It Together
10	*10/30* Critical Perspectives 3 Discussion **Quiz 4** (in-class) **Reading**: Critical Perspectives 3 (TBD) **Homework 7 Due**	*11/1* Introduction to OOP; Final Project Discussion **Reading**: OOP 9-9.5
11	*11/6* Encapsulation, abstraction and composition **Reading**: OOP 10-10.2 **Reflection 3 Due**	*11/8* Inheritance **Reading**: OOP 10.3-10.5 **Project proposals due**
12	*11/13* Team programming best practices **Quiz 5 (in-class)** **Reading**: TBD **Homework 8 Due**	*11/15* **Midterm 2**
13	*11/20* Project work and check-in (optional attendance) **Homework 9 Due**	*11/22* Thanksgiving Holiday. No class.
14	*11/27* Object-Oriented programming exerclse	*11/29* Object-Oriented programming exercise
15	*12/4* Project Presentations	*12/6* Project Presentations
Finals Week		*12/13* **Final projects due** **Final Exam (optional) 8:00 AM**

Legend:

P4E: Python for Everybody (textbook)
Tut: Python Tutorial
OOP: Object-Oriented Programming (textbook)

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 end of the first week of class.

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 other students 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 an article or other source 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 bibliography entry 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;
- Whatever authorship information is available;
- The date you accessed it;
- If applicable, the version number and title of the code.

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

I do not accept late work unless I have approved it by prior arrangement. This is because we usually review solutions in class the day after the assignment is due. 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 at the end of the semester. Documentation of the emergency (e.g. a doctor's letter) may be required.

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 communication, including class schedules, assignments and deadlines.