INST326: Object Oriented Programming Course Syllabus

Spring 2019
Section 105
Benjmain (EDU) 2119
MWF 10:00-10:45

Instructor

Name: Ed Summers
Email: edsu@umd.edu
Phone: 240-478-7086 (cell)
Office: 0301 Hornbake Library (MITH)
Office Hours: TBD and 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 covers (1) the core features of the Python programming language, (2) using programs to collect, process, and analyze data, and (3) object-oriented programming. 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:

1. Design, program, and debug Python programs.

2. Collect, process, and analyze data.
3. Test and assess code quality.
4. Understand how code expresses values & ethics.
5. Write clear and effective documentation.
6. Gain competency with tools for managing code.

How We'll Work

This course assumes a basic understanding of procedural programming, and begins with a comprehensive review of Python fundamentals—including data types, variables, loops, and conditionals—that is designed to deepen your mastery of these concepts. The first part of the course will thus be an opportunity to consolidate and extend what was covered in INST126. Alternatively, if you have worked with a language such as JavaScript, Java, C#, or Visual Basic, you should be able to apply that knowledge to learning Python. The later parts of the course will cover certain topics in program design and programming best practices (documentation, testing, etc.) that are a necessary part of producing complex, reliable, and maintainable applications.

If you have a strong foundation in Python programming already, and 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.

The course is divided into weekly modules, and will typically follow this pattern, with some exceptions:

Before class (preparation):

- Do assigned readings and/or watch assigned videos.
- Complete any online worksheets, exercises, or quizzes that 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 use of paired and group work in class.
- Occasional quizzes may be administered online in class.
- Each midterm exam will include both an in-class and take-home section.

After class (programming homework):

- There will be six homework assignments to help you apply, reflect and extend your understanding by working on a practical task.
- 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 Reflections."

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 respond as soon as I am able, usually within a day.

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

This course will make extensive use of the following book/website, which was also used in INST126:

- Charles R. Severance, [Python for Everybody: Exploring Data Using Python 3] ISBN-13: 978-1530051120 http://py4e.com
- The Python Tutorial, v3.7.2, Python Software Foundation
- Object-Oriented Programming in Python, University of Cape Town https://www.cs.uct.ac.za/mit_notes/python/

Other readings (generally available online for free, or through Library subscriptions) will be assigned as needed.

Required Technology

- Laptop - We will do programming in class, so bring your laptop. Any current OS can be used. If you do not have access to a laptop, please contact me immediately.
- Python - Python3 is freely available from https://www.python.org/downloads.
- Git - We'll be using it to distribute code related to the course materials. Code will also be on Canvas in case you'd rather not use it. https://git-scm.com/downloads
- [Code] - Visual Studio Code is an open source editor created by Microsoft with nice Python and Git support. If you are already comfortable using an advanced text editor like Atom, Vim or Emacs feel free to use it. But I will be using VS Code in class.
- Slack - A widely used team collaboration platform. The iSchool has its own Slack workspace, and there is an inst326 channel we will be using for discussion. In addition to using Slack in your browser there are apps for your desktop and snmartphone. https://umd-cis.slack.com

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

Group Chat

We will be using the iSchool's Slack as a discussion forum for the class. You should receive an invitation to your UMD email address on how to join the Slack. Once you are in you need to join the #inst326 channel, and any other channels that look interesting to you. The table below contains information about what's appropriate to discuss in the Slack channel.

Acceptable in Group Chats	Unacceptable in Group Chats
To use in cases when electronic devices are allowed in the course.	To use during exams or at other times when electronic devices are not allowed.
To discuss the concepts and ideas in the course, on homework assignments, and in other course assignments.	To give out an answer to a homework question, clicker question, or other assignment.
To discuss the mathematical equations in the course, how they work, and to do examples.	To give out numerical answers and exact math work to math problems on a homework assignment or other course assignment.
To report to the instructor any conduct or remarks on the group chat that go against the university's Code of Academic Integrity or Non-Discrimination Policy.	To use to complete an in-class assignment when you are not in class, such as clicker questions or pop quizzes.

Acceptable in Group Chats	Unacceptable in Group Chats
To ask about the mechanics of the course, such as when a due date is or where the class is being held.	To facilitate (help) others cheat, such as by passing on answers to assignments or quizzes, or telling others when in-class activities are occurring (i.e., clicker questions or pop quizzes).
To coordinate with other course members on a groupwork assignment.	To actively exclude another student in the course.

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	**C+**	77-79	**D+**	67-69		
A	93-96	**B**	83-86	**C**	73-76	**D**	63-67	**F**	0-59
A-	90-92	**B-**	80-82	**C-**	70-73	**D-**	60-63		

Final grades will be computed based on the following components:

Exercises/Quizzes (10)	20 points
Homework (5)	25 points
Midterms (2)	20 points
Final Project	15 points
Reflections (3)	10 points
Participation	10 points
TOTAL	100 points

Course schedule

The following table shows the most current version of the planned schedule. The course content can be roughly divided into three interrelated units:

Unit 1: Procedural Programming Review using Python (~weeks 1-4)
Unit 2: Object-Oriented Programming using Python (~weeks 5-9)
Unit 3: Data Analysis using Python (~weeks 10-14)

Week 1
01/28 - Intro and Overview
01/30 - Module 1: Fundamentals
02/01 - Module 1: Fundamentals

Week 2
02/04 - Module 2: Functions & Iteration
02/06 - Module 2: Functions & Iteration
02/08 - Module 2: Functions & Iteration

Week 3
02/11 - Module 3: Data Types **HW1**
02/13 - Module 3: Data Types
02/15 - Module 3: Data Types

Week 4
02/18 - Module 4: Serialization & File I/O
02/20 - Module 4: Serialization & File I/O
02/22 - Module 4: Serialization & File I/O

Week 5
02/25 - Module 5: OOP Fundamentals **HW2**
02/27 - Module 5: OOP Fundamentals
03/01 - Module 5: OOP Fundamentals

Week 6
03/04 - **Critical Reflection #1**
03/06 - Catch up & Review
03/08 - **Midterm #1**

Week 7
03/11 - Module 6: Inheritance & OOP Patterns
03/13 - Module 6: Inheritance & OOP Patterns
03/15 - Module 6: Inheritance & OOP Patterns **HW3**

Week 8
03/18 - Spring Break
03/20 - Spring Break
03/22 - Spring Break

Week 9
03/25 - Module 7: Exceptions & Logging
03/27 - Module 7: Exceptions & Logging
03/29 - Module 7: Exceptions & Logging

Week 10
04/01 - Module 8: Databases and SQL
04/03 - Module 8: Databases and SQL
04/05 - Module 8: Databases and SQL

Week 11
04/08 - **Critical Reflection #2**
04/10 - Catch up & Review
04/12 - **Midterm #2**

Week 12
04/15 - Module 9: Testing **HW4**
04/17 - Module 9: Testing
04/19 - Module 9: Testing

Week 13
04/22 - Module 10: Data on the Web
04/24 - Module 10: Data on the Web
04/26 - Module 10: Data on the Web

Week 14
04/29 - Module 11: **HW5**
05/01 - Data Analysis
05/03 - Data Analysis

Week 15
05/06 - **Critical Reflection #3**
05/08 - Final Presentations
05/10 - Final Presentations

Week 16
05/13 - Final Presentations