INST126-0103 Syllabus

Aric Bills – abills@umd.edu – Version 1.0, 27 August 2019

Table of Contents

INST126: Introduction to Programming for Information Professionals
Fall 2019
TuTh 3:30–4:45 PM
Architecture Building, Room 1101

Instructional Team

Instructor: Aric Bills • abills@umd.edu • (301) 405-3728 • Patapsco Building (https://goo.gl/maps/fCr4UsZ56VA2), suite 1110
TA: Gabriel Cruz
AMP Team: (todo: list AMPs here)

Office Hours (subject to change)

On a walk-in basis in Hornbake 4114:
M 11:05 AM–noon
Th 5:05–6:00 PM

or by appointment:
in person at Patapsco 1110 (https://goo.gl/maps/fCr4UsZ56VA2)
or via WebEx (https://umd.webex.com/meet/abills)

Learning Outcomes

This is an introduction to programming for information science majors. This course provides a path for students with diverse backgrounds to successfully learn programming. You are not expected to have any computer programming experience.

You will learn how programmers analyze problems and design solutions for those problems using **computational thinking**, and practice using computational thinking to solve problems. You will learn how implement computational solutions using the **Python** language (https://python.org/), which is particularly well suited for common problems that information professionals seek to solve, such as data collection, analysis and management, and developing Web applications such as search engines. This is a **hands-on** course - both in and out of class you will be writing, analyzing, testing and debugging code, culminating in a project that tackles a challenging real-world problem.

Throughout the course, we will examine issues like algorithmic bias, ethical/unethical uses of algorithms and disparities in opportunities in tech jobs, which you need to understand to be an **ethical professional** and to be successful in your work. These issues reflect the broader social and cultural context in which we produce software. The social and cultural impacts of information and technology are central concepts in our discipline. Through readings, discussion and writing, we will critically examine how programming is situated in and reflects broader social and organizational structures, the ethical and equity issues this entails, and ways that we might address these issues as information professionals.

After successfully completing this course you will be able to:

1. Explain fundamental programming principles, concepts and methods;
2. Develop small-scale computer programs by applying fundamental programming concepts such as variables, data types, assignments, arrays, conditionals, loops, functions, and input/output operations;
3. Test and assess the quality of small-scale programs;
4. Write clear and effective in-code comments and other documentation;
5. Apply computational thinking techniques to analyze problems and develop computational solutions;
6. Explain how programming is situated in and reflects broader social and organizational structures, and the ethical and equity issues this entails.

Required Resources

- *Python for Everybody: Exploring Data Using Python 3*

  Charles R. Severance
  https://www.py4e.com/html3/
  Download: http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf
  If you prefer, you can buy a hard copy of this book on Amazon for $10: https://www.amazon.com/Python-Everybody-Exploring-Data/dp/1530051126.

- *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 second class.

- *Python:* Python programming language (version 3.6 or greater). You can download the latest version from https://www.python.org/downloads/.
- *Visual Studio Code:* A responsive, customizable, modern code editor. Download the latest version at https://code.visualstudio.com/Download. Please also install the Python (https://marketplace.visualstudio.com/items?itemName=ms-python.python) and Python Indent (https://marketplace.visualstudio.com/items?itemName=KevinRose.vsc-python-indent) extensions.

If you are having trouble installing or using the software listed above, please let me know immediately.

Campus Policies

It is our shared responsibility to know and abide by the University of Maryland's policies that relate to all courses, which include topics like:

- Academic integrity
- Student and instructor conduct
- Accessibility and accommodations
- Attendance and excused absences
- Grades and appeals
- Copyright and intellectual property

Please visit www.ugst.umd.edu/courserelatedpolicies.html for the Office of Undergraduate Studies' full list of campus-wide policies and follow up with me if you have questions.

Academic Integrity and Ethical Use of Other People's Work

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

- you may get help from a fellow student to understand a particular concept
- you may pair program with a student on an assignment that has been designated as a pair assignment
- you may want to use a function or an algorithm from a website or a book
- you may be writing a paper and may wish to share ideas you read in a published scholarly work

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

- You have an obligation to produce your own 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 essay, this means providing a complete, accurate entry in your bibliography as well as an in-text citation. In code, you should provide a comment including the following details:
    - the source of the code (URL if online or bibliographic citation if in print)
    - as much authorship information as 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 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 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, or possess a copy of someone else's solution to an assignment unless the assignment instructions include explicit written instructions to the contrary.**

Course overview and expectations

Each week will typically follow this pattern, with some exceptions (e.g., for exam weeks):

Before the week begins (preparation)

- Do assigned readings, working each example in the text
- Watch any assigned videos;
- Do any pre-class activities – these help you confirm that you understand the basic material and/or help you identify specific aspects that you have questions about. These pre-class activities will sometimes include callbacks to previous material to reinforce understanding;
- Your pre-class activities culminate in completion of a worksheet (on ELMS, unless otherwise specified), which are due at the beginning of each week. The worksheet helps you confirm you are prepared for class and identify any gaps or questions. I review these before class and use them to prepare for class.

During 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, including a substantial amount of **paired programming** (a best practice in software engineering and programming education (https://www.iste.org/explore/articleDetail?articleid=221)).

Class is not a time for solo learning. As members of a learning community, we are mutually responsible to each other as learners. Each of us has to be fully engaged with each other in the activities. We have to be supportive of each other as we try to explain or demonstrate something new, as we inevitably make mistakes. We aren't successful unless everyone is learning.

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. (Before the week begins) Completed all the readings/videos, and attempted all the pre-class activities and either successfully completed them or submitted your questions the night before class, so I have time to prepare and answer them in class.
2. (At each class) Arrived 5 minutes before class starts; are in your seat, with Jupyter Notebook or other Python tools ready. You have downloaded any notes or materials for the day from ELMS. Any paper assignments are ready to hand in. You are ready to take notes.

After class

To provide further practice and opportunities for feedback, and integrate concepts/skills across classes, you will work on homeworks and your team project.

I encourage you all to use the Discussion forum provided by ELMS to ask/answer questions, get clarifications, point out my mistakes, etc. Be sure to check it regularly.

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 sub-conscious 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 in the ELMS discussion forum. 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 or two (longer during weekends, travel, etc.). If I don't respond after two days, send me an email.
6. If you see a question on the ELMS discussion forum that you can answer, or if you have an idea, please respond. Don't wait for me or the AMPs. You will be helping your colleagues.

Assessments

Worksheets (12% of final grade)

These assessments help you assess your preparation for the concepts to be discussed in class. They will typically be quizzes submitted on ELMS before the first lecture of the week, consisting of a mixture of multiple choice, matching, and short answers. They are designed to be straightforward to do well on as long as you have done the reading. The lowest grade is dropped.

These are due at 11:00 pm on the day before the first lecture of this week (usually, except for Week 1 and Labor Day week, on the Monday of each week), so that I can take this into account for planning for the week's lecture (e.g., if there are particular concepts many students are struggling with). These will generally be opened on ELMS at least 1 week before they are due. Please let me know if you need earlier access to a worksheet (e.g., to complete your work early in advance of pre-approved travel; see also policy on missed deadlines below).

In-class learning checks (5% of final grade)

We will use learning checks in the first session of the week to make sure that you are following along with the content. They should be straightforward to do well on if you are paying attention in class and have done the reading. The lowest grade is dropped: consider this a "freebie" for when unexpected life circumstances get in the way (e.g., you get stuck in traffic or have an unexpected emergency to attend to; see policy on Missed Deadlines below for what to do if you know ahead of time that life will get in the way).

In-class exercises (12% of final grade)

Each week, a substantial portion of the last class will be devoted to a hands-on exercise that you will turn in to be graded. These will typically be hands-on programing exercises that apply the concepts and skills discussed in class, the vast majority of which will be completed in pairs (for pair programming). Sometimes these include discussion exercises as well. "Correctness" of your programs will be an important component of your grade, but you should have sufficient support to do well on them in class. Similar to the learning checks, the lowest grade is dropped.

Homeworks (30% of final grade)

Includes coding problems (of course), but will also include analysis questions, brief reflective writing, and other activities. For these assessments, "correctness" of your solutions will be an important part of your grade. There will be 6 of these assignments.

Exams (25% of final grade)

These are diagnostics for you to assess your understanding of the programming basics that are necessary as you proceed through the course and prepare for term project. You will want to address any weaknesses these diagnostics identify to ensure you are well prepared for the project.

There will be two exams. Each will consist of two parts: an in-class (on-paper), closed-book exam, and a take-home exam consisting of coding problems.

Team project (16% of final grade)

The team project will give you an opportunity to apply what you learn in class (especially computational thinking practices and testing/documentation best practices) in a more realistic application than toy problems. You will form a small (~3-5 people) team to **formulate and develop a solution to a real-world problem that interests you** (some examples will be provided to you). The exact problem you choose is open-ended, as long as it conforms to at least one of three application areas that are relevant to information science: data analysis, search, or web services.

There will be 5 components to the team project:

- 1 initial project proposal, where your team submits a document that describes a formulation of your problem in computational terms, along with your team's collaboration plan (proposed roles, time/scheduling expectations). Worth 5% of your final grade for the course.
- 3 progress updates (including current state of the code). Each update is worth 2% of your final grade for the course, for a total of 6%.
- 1 final project submission, worth 5% of your final grade for the course.

Assessment on this project will be process-centric: your grade will heavily focus on assessing the extent to which you are effectively using computational thinking techniques, including a computationally formulating problems, using decomposition and abstraction, and practicing good software engineering techniques like quality testing (e.g., testing for edge cases) and effective documentation. It is possible that some of you may choose problems that turn out to be too difficult to solve (although we will work hard to make sure that your chosen problems are scoped well enough that you have a good chance of succeeding): if this happens, don't worry! As long as you are doing a good job of following *good processes,* you should be able to do well on this assessment.

I also recognize that social loafing can be a problem: to incentivize everyone to pull their weight, I will require you to 1) submit a collaboration plan with your initial proposal, 2) come to at least 2 of the 3 planned in-class project work sessions (your grade on the team project will be reduced by 10% for missing 1, and 20% for missing 2), and 3) complete (confidential) peer reviews of your teammates.

Grading

Grades are not given, but earned. Your grade is determined by your performance on the learning assessments in the course. If earning a particular grade is important to you, please speak with me at the beginning of the semester so that I can offer some helpful suggestions for achieving your goal.

All assessment scores will be posted on the course ELMS page. If you would like to review any of your grades (including the exams), or have questions about how something was scored, please email me to schedule a time for us to meet in my office. I am happy to discuss your grades with you, and if I have made a mistake I will immediately correct it. Any formal grade disputes must be submitted in writing and within one week of receiving the grade.

Your final grade for the course is computed using the learning assessments table below, converted to a letter grade. :

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

Please note that cutoffs are specified to the hundredth of a percent. I do not round grades at the end of the semester.

Course-Specific Policies

Late Work

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.

Exam policy

If you need to miss an exam due to outside circumstances (e.g., a religious holiday, military duties, work/athletic team travel), you must email me before the exam to reschedule your exam time. If you are sick on an exam day, you must provide me with a doctor's note to be excused (see the UMD policies on absences) and should email me before the exam time to let me know you're sick. If you miss an exam due to other circumstances (e.g., oversleeping), you will not be able to make up the exam.

Get some help!

You are expected to take personal responsibility for you own learning. This includes acknowledging when your performance does not match your goals and doing something about it. Everyone can benefit from some expert guidance on time management, note taking, and exam preparation, so I encourage you to consider visiting http://ter.ps/learn and schedule an appointment with an academic coach. Sharpen your communication skills (and improve your grade) by visiting http://ter.ps/writing and schedule an appointment with the campus Writing Center. Finally, if you just need someone to talk to, visit http://www.counseling.umd.edu.

Everything is free because you have already paid for it, and **everyone needs help**... all you have to do is ask for it.

Course Schedule (subject to change)

Week 1: Introduction to computational thinking
Tu 08/27 Introduction to course
Th 08/29 What is computational thinking? What is Python? **worksheet 1 due before class**

Week 2: Basic program components and structure
M 09/02
Tu 09/03 Variables and conditionals; **learning check 1 in class; worksheet 2 due; Survey due**
Th 09/05 Variables and conditionals

Week 3: Basic program components and structure
M 09/09 ***worksheet 3 due***
Tu 09/10 **exercise 1 in class**
Th 09/12 Functions and iteration; **learning check 2 in class**
F 09/13 ***Homework 1 due***

Week 4: Data structures: strings
M 09/16 ***worksheet 4 due***
Tu 09/17 **exercise 2 in class**
Th 09/19 Strings; **learning check 3 in class**

Week 5: Data structures: lists
M 09/23 ***worksheet 5 due***
Tu 09/24 **exercise 3 in class**
Th 09/26 Lists; **learning check 4 in class**
F 09/27 ***Homework 2 due***

Week 6: File I/O
M 9/30 ***worksheet 6 due***
Tu 10/01 **exercise 4 in class**
Th 10/03 **Exam 1 in class**

Week 7: Data structures: dictionaries
M 10/07 ***worksheet 7 due; take-home exam 1 due***
Tu 10/08 Files; **exercise 5 in class**
Th 10/10 Dictionaries; **learning check 5 in class**

Week 8: Applications: search
M 10/14 ***worksheet 8 due***
Tu 10/15 Files; **exercise 6 in class**
Th 10/16 Search engines; **learning check 6 in class**

Week 9: Applications: search
M 10/21 ***worksheet 9 due***
Tu 10/22 Exam 1 review

Th 10/24 **exercise 7 in class**
F 10/25 ***Homework 3 due***

Week 10: Applications: data analysis
M 10/28 ***worksheet 10 due***
Tu 10/29 Files; Project kickoff; **exercise 8 in class; project proposal due**
Th 10/31 Intro to Pandas; **exercise 9 in class**

Week 11: Applications: data analysis
M 11/04 ***project update 1 due***
Tu 11/05 **Exam 2 in class**
Th 11/06 Pandas; **learning check 7 in class**

Week 12: Applications: data analysis
M 11/11 ***worksheet 11 due; take-home exam 2 due***
Tu 11/12 Pandas/Matplotlib **exercise 10 in class; project update 2 due**
Th 11/14 Pandas; **exercise 11 in class**

Week 13: Applications: web services
M 11/18 ***read "Don't Get Distracted"; available on ELMS; worksheet 11 resubmission due***
Tu 11/19 Networked programs (intro) **exercise 12 in class; project update 2 due**
Th 11/21 Ethics in programming/web services; **exercise 13 in class**
F 11/22 ***homework 4 due***

Week 14: Applications: web services
M 11/25 ***worksheet 12 due***
Tu 11/26 Web services; **exercise 14 in class; project update 3 due**
Th 11/28 *Thanksgiving break*

Week 15: Final projects
M 12/02 ***homework 5 due***
Tu 12/03 **Final project presentations**
Tu 12/05 **Final project presentations**

Final
Tu 12/17 **Final projects due by 12:30 PM**

Version 1.0
Last updated 2019-08-27 12:53:18 -0400