EECE 230X – Introduction to Computation and Programming
Open Enrollment Section
Summer 2022-23
Course Information

1. Catalog Description

This is an introductory programming course with an emphasis on abstractions and elementary algorithmic ideas. It uses the Python programming language. Topics include data types, selection, repetition, lists, tuples, strings, functions, files, plotting, exception handling, program efficiency, recursion, divide and conquer algorithms, recurrence relations, sorting and searching algorithms, binary search, merge sort, randomized quicksort, dictionaries, memoization, classes and object oriented programming, stacks and queues, applications, and selected topics.

2. Course Instructor

Professor Louay Bazzi
ECE department, AUB

3. TA office hours

The course teaching assistants are available online Monday to Saturday to answer any questions related to the material or the assignments. The office hours schedule will be posted on Moodle.

4. Course delivery format

The course will be delivered in an asynchronous mode.

- **How is EECE 230 structured?** The course is divided into three modules, each module is divided into topics, each topic is divided into lessons, and each lesson is divided into prerecorded interactive short videos. Each topic has supporting programming assignments.
- **Asynchronous material: short videos and supporting material.** The duration of each short video is around 5 minutes on the average. They are short since according to studies, students tend to lose focus when watching long videos. These short videos form the main asynchronous material in EECE 230X. Supporting asynchronous material include sets of slides each associated with a topic, combined videos each associated with a lesson, code, and handouts.
- **Learning by discovering**. Each short video typically ends with a question whose answer is in the following video. You are strongly encouraged to give the question as much time as possible before moving to the next video. The reason is that learning by discovering strongly enhances your problem solving skills. It moves you from the receiving end of the instructional methodology to the active learner's end. For some questions, figuring out a complete answer on your own is not simple and may take a time. It is enough to think about for some time depending on your time constraints to prepare your mind for the answer in the following video. It doesn't really matter if you end up figuring out the answer because identifying what doesn't work is equally important. A main objective of this course is to learn to think like computer scientist. To do so, it is vital to understand what doesn't work to value what works. . The questions were compiled from previous offerings of the course in a classical delivery mode. Due to time limitation, students were given few minutes to come up with solutions and suggestions. The advantage of asynchronous format is that now you can take as much time as needed, ranging from a few minutes to hours depending on your time constraints.

5. **Programming Assignments**

Programming assignments are key component of the course. In assignments, you will acquire coding skills as well as problem solving skills. The problems in assignments are of varying difficulty levels. Some of them have hints. For some problems, hints are written in small font and backward to encourage your to think about the problem before reading the hint. On the average, it takes 3 - 5 hours to solve the assignment. It is essential that you work hard on the assignments on your own to achieve the course learning outcomes. Discussing your work on assignments with your classmates and teaching assistants is helpful, but you are expected to submit your own work. The assignments solutions will be posted immediately after the due dates. Please keep in mind that there is a big difference between solving a problem or at least trying to solve it and reading the solutions when they are out. To acquire the skill of problem solving, your are expected to first work on the assignment and then compare your work with the posted solutions.

Assignment will be graded by an automatic online judge aubsharifjudge.app on which you could submit each problem as many times you wish. If your submission is not fully correct, the online judge will give you feedback about the tests which failed.

When done with debugging assignment on the online judge, you are expected to submitted on Moodle and as soon as you submit it on Moodle, you will have access to its solutions.

6. Final Exam

To get a certificate of completion, you need to take the final exam which will be held online. The date of the final exam will be announced on Moodle. To pass the final exam, you are expected to complete the course asynchronous material and the assignments.

7. **Textbook(s) and/or required materials**

Guttag, John. *Introduction to Computation and Programming Using Python: With Application to Understanding Data, Second Edition*. MIT Press, 2016

8. Brief list of topics Covered

Topic No.	*Topics*	*Number of weeks*
Unit I: Foundations		6
1,2,3	Introduction to computation using Python, data types, selection, repetitions, and bisection method	Weeks 1,2
4,5	Lists, tuples, strings, and functions	Weeks 3,4
6,7	Files, exception handling, plotting, and Monte Carlo simulation	Week 5
8	Introduction to program efficiency and asymptotic analysis, binary search, and insertion sort	Week 6

Unit II: Recursion, memoization, searching and sorting, divide and conquer, recurrence relations, data structures, and applications		4
9,10	Recursion: elementary examples, memoization, merge sort, divide and conquer algorithms, recurrence relations, recursion tree method	Weeks 7,8
11	Elementary data structures: two-dimensional lists, dictionaries, and stacks	Week 9
12	Applications: randomized quick sort, recursive enumeration, maze depth-first traversal	Week 10
Unit III: Object Oriented Programming with applications and selected topics		3
13	Object Oriented Programming, classes, and inheritance	Weeks 11,12
14	Implementation of stacks and queues	Week 12
15	Graphs: representation, depth-first search, and breadth-first search	Week 13

9. Course Learning Outcomes

At the end of the course, students will be able to:

- Apply the principles of functional programming
- Implement searching and sorting algorithms
- Solve computational problems using searching and sorting
- Analyze of the efficiency of elementary algorithms
- Solve computational problems using recursion
- Solve computational problems using elementary data structures such as two-dimensional lists, dictionaries, stacks, and queues
- Apply the principles of object-oriented programming

10. Collaboration Policy and Cheating

Students are expected to complete all work with the highest standard of integrity in line with AUB's Student Code of Conduct. Plagiarism, forgery, cheating or any form of academic misconduct will not be tolerated and will automatically result in a failing grade. In this course, cheating is defined as follows:

- Copying or partially copying an assignment code from any person or source as is or with minor changes such as:
  - Editing the name and Id in the beginning of the file
  - Rephrasing/adding comments in the code
  - Changing variable names
  - Reordering some instructions
  - Rewriting.
- Getting help without acknowledgement for the sake of merely finishing the assignment.

- Sharing your code with anyone who might copy your code and submit it under their name.
- Providing or receiving step-by-step coaching on solving the assignment problems without ensuring that the learner fully understands what is going on.