CS161 – Introduction to Computer Science I
Spring 2021, CRN: 41666, Credits: 4
Syllabus (tentative)

Instructor: Joseph Jess
Web: http://cf.linnbenton.edu/bcs/cs/jessj/web.cfm?pgID=5404
email: jessj@linnbenton.edu

Initial class note: This may be an introduction course to computer science (CS), but that does not mean it will always be easy. This is a beginning of learning to program, design, test, and implement concepts using the Python programming language as a tool. We should expect to do some reading, much practicing and searching, and much discussion of the topics we cover.

Extra note for 2021: I understand how busy this year has been even for those in good situations, just be sure to contact me as early as you can when things are stressful or anything unexpectedly comes up :)!

1. **LBCC catalog course description, including pre-requisites/co-requisites**:
   Introduces the principles of computer programming using an object-oriented language. Includes topics related to problem-solving concepts, verification and validation, representation of data, sources of error, debugging techniques, conditional statements, loops, and arrays. Introduces both command line applications and those with graphical user interfaces.

   Prerequisite: MTH 095 Intermediate Algebra or higher and CS 160 Orientation to Computer Science, both with a grade of "C" or better.

2. **Class Time-space**:
   2.1 Lecture + demo + lab: Online, Remotely delivered, check Moodle for details!

3. **Course Learning Outcomes:**
   Upon successful completion of this course, students will be able to:
   3.1 Demonstrate an understanding of the difference between primitive data types and objects and their representation,
   3.2 Demonstrate the use of good program development, debugging techniques and documentation,
   3.3 Write object-oriented code that includes iterative and decision-making structures,
   3.4 Develop programs that accept input from both the keyboard and files on disk and which writes output to either the screen or file,
   3.5 Write, compile and run simple web-based and desktop-based graphical user interface applications using components and containers,
   3.6 Write simple, user-designed classes that demonstrate an understanding of encapsulation.

4. **Learning resources:**
   4.1 **Note:** All required class materials and storage will be available in a digital format
   4.2 The Python language – available through several media. We will discuss this some in class
   4.3 A Python interpreter (3.6 or better and accessible from the command line) – I will use the interpreter available from https://www.python.org/. We can discuss this more in class
   4.4 An integrated development environment (IDE) recommended in our other Python courses includes Mu or VIdle, I will plan to use plain text editor with some scripts (likely Notepad++ if I have my way), but will also show off some smart IDE features of IDEs such as VS Code
      4.4.1 We will discuss some capabilities of smart code editors during the course.
   4.5(**strongly recommended**) A desire to learn, experiment, design, test, and problem solve with code (both on and off of a computer).
5. **Other course materials required**:
   5.1 Internet Access
   5.2 An LBCC-provided email account,

6. **Other Learning resources that you might consider using:**
   (these are not required, but you might find helpful, along with many other resources we could talk about in class)

   6.1 *The Practice of Computing Using Python*,3rd Ed.,
   Authors: Punch and Enbody
   ISBNs: 978-0134379760, 978-0134380315 (“student value edition”)
   Note: Available from the LBCC Campus Store as a 180-day rental. This same textbook is used as a resource in CS162 so

renting it in fall or winter can cover CS162 in the following term as well.

6.2A Byte of Python (free and openly accessible!)(in case you need some of the basics!): https://python.swaroopch.com/

7. Grading:

7.1 Scores for assignments will be available when the instructor gets to it… usually within the week of the due date. Grades will be made available through the Moodle gradebook.

7.2 Students will be required to turn in all coursework items before 23:59 (Pacific Time Zone) on the date that they are due (generally the first meeting day of the week in my courses)... though I have an extremely forgiving option for making up for missed work at the end of the term.

7.3 To earn a passing grade in this course you must pass each of the following coursework categories:

7.3.1 **Demonstration**: Discussion, quizzes, and weekly assignments – 50%

7.3.1.1 There are a number of discussion questions (usually turned in as a part of the programming projects), quizzes, and programming projects to be completed for this class, designed to challenge and solidify design, coding, and testing skills. Projects are generally graded based on the following rubric:

A. **Program Design** (20%)

Rating	Criteria
20	Solution well thought out
10	Solution partially planned out
0-5	ad hoc solution; program was “designed at the keyboard” or no design submitted

B. **Program Execution** (20%)

Rating	Criteria
20	Program runs very well under a variety of conditions, as submitted
10	Program runs much of the time, may be missing required files or instructions for libraries used
0-5	Program runs very poorly, not at all, or requires several modifications or files before it runs

C. **Specification Satisfaction** (20%)

Rating	Criteria
20	Program satisfies specification completely and correctly
10	Important parts of the specification not implemented
0-5	Program poorly satisfies specification, or not at all

D. **Coding Style** (20%)

Rating	Criteria
20	Well-formatted, understandable code and appropriate use of language capabilities
10	Code difficult to follow in one reading or poor use of language capabilities
0-5	Incomprehensible code, poor use of language capabilities, or a need to scroll up and down repeatedly

E. **Comments** (20%)

Rating	Criteria
20	Concise, meaningful, and well-formatted comments and docstrings
10	Partial, poorly written, or poorly formatted comments
0-5	Wordy, unnecessary, incorrect, badly written or formatted, or none or nearly no comments

---

Note: careful design, systematic testing, consistent style, and readability of code are important software quality factors (all of which are subject to interpretation but graded by the instructor based on the spirit and letter of the requirements, so be sure to explain your decisions).

Note well: Your submission should be explained and able to be compiled and run from just your submitted files in your final submission for that project. This means that you need to include any files provided to you that are necessary for your project to compile or run.

Note very well: Source code and related documents submitted must be designed and implemented by the student submitting the work and any code must compile and run on one of the instructor's machines in order to be graded. (to create a working program quickly: get it working simply, then add to it; if at some point it stops compiling you will better know where an error was introduced)

7.3.2 **Final**: Final project – 50%

7.3.2.1 There will be a final project to test the overall ability to understand, design, implement, test, and reflect on the problem solving and programming knowledge and skills covered in the class.

7.3.2.2 The final project will be a mix of “in-class” (initial design, testing, and implementation discussion) and take-home (finalizing design, testing, and implementation) elements.

7.3.3 **Make-up Work**: I do not care when you learn it, as long as you learn it. To support this idea I allow missing work, even after the term for many, to be made up for in the final project (just be sure I know you are trying to make it up in the final project submission!).

7.3.4 Final grades will be given out based on the following based on score in the class:
90-100%: A
80-89%: B
70-79%: C
60-69%: D
00-59%: F

7.4 Reminder: A passing grade in order to count for course requirements for CS classes is generally a C or above.

8. Course Topics:

Week	Topics
1	Course Intro, tools, the study of computer science, beginnings
2	using the command line, docstrings, control, algorithms and program development
3	Git and GitHub, working with strings
4	functions (quick start), files and exceptions part 1
5	lists and tuples, more on functions
6	dictionaries and sets, more program development
7	intro to classes
8	discuss and start final projects
9	continue final projects
10	finish final projects?
11	Final projects due by 2359 (Pacific time), Tuesday 08 December, 2020

9. Other Administrative Information:

9.1 For a list of general administration information (note that this list is not intended to be exhaustive), such as:
9.1.1 contacting me,
9.1.2 accessibility resources,
9.1.3 expectations of student conduct,
9.1.4 communications,
9.1.5 student assistance,
9.1.6 miscellany,
9.1.7 nondiscrimination & nonharrasment,
(each section contains a number of sub-sections and is not meant to be exhaustive of all situations)

see my administrative information document: *administrative information* document.