

**Project Title:**

OER Designs and Development for CSCI120

August 2022

Alireza Davoodi

[adavoodi@columbiacollege.ca](mailto:adavoodi@columbiacollege.ca)

**Description:**

The main objective of the project is to create an OER package for CSCI120 course in Columbia College.

**Deliverables:**

* Designing course modules and outline
* Creating lecture notes
* Lecture notes final review and modification
* Creating sample questions + video recording
* Creating the course website (Github) and making the course available on different platform
* Reviewing/editing the Chapters (if needed)
* Testing the source codes used in the textbook
* Final reporting

**OER Package:**

* **For Students:**
  + Lecture Note (Non-editable)
  + Sample Questions/Answers
  + Sample Midterm + Answers
  + Sample Final exam + Answers
* **For Instructors:**
  + Course timeline
  + Lecture Note (Editable)
  + Labs
  + Assignments + Source Code
  + Sample Quizzes
  + Sample Midterm Exam
  + Sample Final Exam
  + Sample Project + Project Starter + Implementation (Source Code)

**Definition**

* **Lecture Note:** Is PowerPoint files prepared for the instructors and students to review the topic of the chapter
* **Assignments:** Is set of questions assigned to students to do as homework. It is can be done individually or in group. Assignments usually contain 5-10 questions.
* **Labs:** The labs are designed for students to practice the last lecture topics (Designed for 1-2 hours)- Labs are usually 3-5 questions.
* **Sample Practices:** Sample is extra/complementary examples for students to learn more.
* **Quiz:** Quiz is mainly multiple-choice questions to practice the subject.
* **Exam:** Exam is questions for the midterm and final exam.
* **Project:** The project is defined for the course.
* **Source Code:** Source code is code for the assignments, labs, tutorials, and sample practices.

**Learning Objectives**

The learning Objectives of the course, based on the course outline and objectives designed for this course in Columbia College.

1. To learn fundamental concepts and terminology of computing science, acquire elementary skills for programming in a high-level language and be exposed to diverse fields within, and applications of computing science
2. To recognize compile-time and run time-errors
3. To describe an algorithm with pseudocode
4. To understand how information is represented in computer
5. To understand how to convert integers to binary and vice versa
6. To define and use variables and constants
7. To appreciate the importance of comments and good code layout
8. To write arithmetic expressions and assignment statements
9. To create programs that read and process inputs, and display the results
10. To learn how to use and process Python strings
11. To implement decisions using if statements
12. To develop strategies for testing your programs
13. To validate user input
14. To implement while and for loops
15. To understand nested loops
16. To be able to implement functions
17. To become familiar with the concepts of parameter passing
18. To develop strategies for decomposing complex tasks into simpler ones
19. To be able to determine the scope of a variable
20. To create simple graphics programs
21. To collect elements using lists and tuples
22. To use the for loop for traversing lists, tuples, and strings
23. To use lists, tuples, and strings with functions
24. To work with a dictionary container
25. To work with a dictionary for table lookups
26. To build and use a set container
27. To learn common set operations for processing data
28. To read and write text files
29. To raise and handle exceptions
30. To learn how to think recursively
31. To understand the relationship between recursion and iteration
32. To analyze programs that are much easier to solve by recursion than by iteration
33. To study several soring and searching algorithms
34. To understand big-Oh notation
35. To estimate and compare the performance of algorithms
36. To understand the concepts of classes, object, and encapsulation
37. To implement instance variables, methods, and constructors
38. To be able to design, implement, and test your own classes
39. To understand the behavior of object references

**Proposed Chapters** (And Covered Learning Objectives)

**Chapter 1:** Introduction to Computer and Problem Solving

* To learn fundamental concepts and terminology of computing science, acquire elementary skills for programming in a high-level language and be exposed to diverse fields within, and applications of computing science
* To recognize compile-time and run time-errors
* To understand how information is represented in computer
* To understand how to convert integers to binary and vice versa
* To describe an algorithm with pseudocode

**Chapter2:** Variables, Expressions and Statements

* To define and use variables and constants
* To appreciate the importance of comments and good code layout
* To write arithmetic expressions and assignment statements
* To create programs that read and process inputs, and display the results

**Chapter3:** Conditional Statement

* To learn how to use and process Python strings
* To implement decisions using if statements
* To develop strategies for testing your programs
* To validate user input

**Chapter4:** Functions

* To be able to implement functions
* To become familiar with the concepts of parameter passing
* To develop strategies for decomposing complex tasks into simpler ones
* To be able to determine the scope of a variable

**Chapter5:** Iterations

* To implement while and for loops
* To understand nested loops

**Chapter6:** Strings

* To learn how to use and process Python strings

**Chapter7:** Files

* To read and write text files
* To raise and handle exceptions

**Chapter8:** Lists

* To collect elements using lists and tuples

**Chapter9:** Dictionary

* To work with a dictionary container
* To work with a dictionary for table lookups
* To build and use a set container
* To learn common set operations for processing data

**Chapter10:** Tuples

* To collect elements using lists and tuples

**Chapter11:** Recursion

* To learn how to think recursively
* To understand the relationship between recursion and iteration
* To analyze programs that are much easier to solve by recursion than by iteration

**Chapter12:** Object-Oriented Programming

* To implement instance variables, methods, and constructors
* To be able to design, implement, and test your own classes
* To understand the behavior of object references

**Chapter13:** Complexity Analysis

* To estimate and compare the performance of algorithms
* To understand big-Oh notation

**Chapter14:** Data Structures

* To study several soring and searching algorithms

**Proposed Timetable**

The following timetable is for a 13-week course.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Week** | **Lectures (2hr)** | **Labs (2hrs)** | **Assignments** | **Quizzes** | **Topic** |
| **1** | Chapter1 | Lab1 | - | - | Introduction, Problem Solving |
| **2** | Chapter2-3 | Lab2 | - | - | Variables, Conditional |
| **3** | Chapter5-6 | Lab3 | Assignment1 | - | Iterations, String |
| **4** | Chapter4 | Lab4 | - | - | Functions |
| **5** | Chapter8 | Lab5 | - | Quiz1 | Lists |
| **6** | Chapter9-10 | Lab6 | Assignment2 | - | Dictionary, Tuples |
| **7** | Chapter7 | Lab7 | - | - | Files |
| **8** | Chapter11 | Lab8 | - | - | Recursion |
| **9** | Chapter12 | Lab9 | - | Quiz2 | OOP |
| **10** | Lab10 | Assignment3 |  |
| **11** | Chapter13 | Lab11 | - | - | Complexity Analysis |
| **12** | Chapter14 | Lab12 | - | Quiz3 | Data Structure |
| **13** | - | - |

**Proposed Evaluation Scheme**

|  |  |
| --- | --- |
| **Evaluation Methods** | **%** |
| assignments | 12 |
| Labs | 25 |
| quizzes | 10 |
| Mid-term exam | 20 |
| final exam | 25 |
| Project | 8 |
| **Total** | **100%** |