Discipline: Computer Science

Originator: Mark Lehr

# RIVERSIDE COMMUNITY COLLEGE DISTRICT INTEGRATED COURSE OUTLINE OF RECORD

COMPUTER SCIENCE 8

CSC-8 : Programming Concepts: Python

Lecture Hours: 54.000
Lab Hours: 18.000
TBA Option: Yes
Total Student Learning Hours: 72.000

## Course Description

**Prerequisite:** MAT-5 or MAT-12 or MAT-36 or

Introduction to the discipline of computer science incorporating problem definitions, algorithm development, and structured programming logic for business, scientific and mathematical applications. The Python language will be used for programming projects.

## Short Description for Class Schedule

## Entrance Skills:

Before entering the course, students should be able to demonstrate the following skills:

## Student Learning Outcomes:

Upon successful completion of the course, students should be able to demonstrate the following skills:

1. **Describe the principles of structured programming and be able to design, implement and test structured programs.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
   - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.
2. **Explain what an algorithm is and its importance in computer programming.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
   - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
3. **Use pseudocode, flowcharts, and a programming language to implement, test, and debug algorithms for solving problems. Identify the information requirements, synthesize the algorithmic steps needed to transform the data input into the required output information, and organize the output format to facilitate user communication.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to

explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.

- **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.

4. **Create computer programs using the principles of structured programming and demonstrate the use of an IDE with appropriate libraries. Design, implement, test, and debug programs that use fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and functions.**
    - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
    - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
5. **Apply the principles of logical and programming concepts to develop solutions for gaming, business, scientific and mathematical problems.**
    - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
    - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
    - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.

---

## Course Content:

Focus of the course topics is on the fundamentals of programming, problem solving, and software design. Models of applications from the areas of gaming, business, science and mathematics are presented throughout the course.

1. Algorithms and problem-solving The intent is to: a) discuss the importance of algorithms in the problem-solving process, b) identify the necessary properties of good algorithms, c) create algorithms for solving simple problems, d) use pseudocode, flowcharts, and a programming language implementation process, and e) describe strategies useful in debugging.
    a. Problem solving strategies,
    b. The role of algorithms in the problem solving process,
    c. Implementation strategies for algorithms,
    d. Debugging strategies,
    e. The concept and properties of algorithms.
2. Introduction to Python
    a. Coverage of topics by instructors should include the components of programs such as key words, variables, operators, and punctuation. Programming design tools, such as pseudocode, flowcharts, and hierarchy charts are presented.
    b. Students will be guided through Python data types, identifiers, variable declarations, constants, comments, program output, and simple arithmetic operations. Programming style conventions and good programming style are modeled. Students will write programs that input and handle numeric, character. The use of arithmetic operators and the creation of mathematical expressions are covered in detail, with emphasis on operator precedence

   c. The intent amongst other things is to; a) explain the value of declaration models, especially with respect to programming-in-the-large, b) identify and describe the properties of a variable such as its associated address, value, scope, persistence and size, c) discuss type incompatibility, d) demonstrate different forms of binding, visibility, scoping, and lifetime management, and e) defend the importance of types and type-checking in providing abstraction and safety.
3. Decision Structures
   a. Relational operators, relational expressions and how to control the flow of a program with the if, if/else, and if/else if statements are explained.
   b. Crucial applications of the conditional operator and the switch statement are covered, such as menu-driven programs and the validation of input.
4. Iteration Structures
   a. Repetition control structures, such as the while loop, and for loop are taught, along with the common uses for these structures.
   b. Counters, accumulators, running totals, sentinels, and other application-related topics are modeled.
5. Functions
   a. Students learn how and why to modularize programs.
   b. Argument passing is covered.
6. Tuples, List, Dictionaries, Arrays
   a. Students will create single and multidimensional arrays. Examples of array processing are provided including examples illustrating how to find the sum, average, highest and lowest values in an array and how to sum the rows, columns, and all elements of a two-dimensional array.
   b. Programming techniques using parallel arrays are demonstrated and students are shown how to use a data file as an input source to populate an array.
   c. The bubble sort, selection sort, linear search, and binary search algorithms are modeled.
7. File and I/0 Operations The intent for 4.-10. is to; a) analyze and explain the behavior of programs involving fundamental programming constructs, b) create, modify and expand programs that use standard conditional, iterative control structures and functions, c) design implement, test, debug programs that use the above programming constructs, d) choose the appropriate construct for the solution required, e) apply structure decomposition to break programs into smaller pieces (divide and conquer), f) describe and implement the mechanics of functional parameter passing.
   a. Sequential access, random access, text, and binary file use is modeled.
   b. The various modes for opening files are discussed, as well as the many methods for reading and writing file contents.
   c. Advanced output formatting is also covered.
8. Classes
   a. The object-oriented paradigm is introduced.
   b. Member variables and functions are discussed. The student learns about private and public access specifications, and reasons to use each.
   c. The topics of constructors, overloaded constructors, and destructors are presented. Modeling classes with UML, and how to define the classes in a particular problem is discussed.

**Additional Laboratory Content**

This course includes 54 hours of lab instruction. There will be assignments or projects to be submitted on the major topics such as:

1. Algorithm Development
   a. Use of flow-charting and pseudo-code
2. Components of a program
   a. Creation of templates for program creation
   b. Variables, Keywords, Operators, Syntax
3. Decision Structures
   a. Ternary Operator
   b. Independent If, and Dependent If
4. Iteration Structures
   a. Increment, decrement, and in-Range
   b. While, and For-loop
5. Functions
   a. Parameter passing
   b. Return values
6. Primitive Data Types
   a. Built-in
   b. Casting
7. File I/O

   a. Serial vs. Random access
   b. Get, put, read, write
   c. Binary I/O
   d. Reinterpret Cast
8. Introduction to Classes
   a. Structures - Global Data
   b. Data members vs. public interfaces

## Methods of Instruction:

Methods of instruction used to achieve student learning outcomes may include, but are not limited to, the following activities:

- Presentation of class lectures/discussions/demonstrations in order to clarify computer programming, computer problem solving, and software design concepts.
- Presentation of class lectures/discussions/demonstrations in order to clarify the principles of structured programming.
- Web-based/web-enhanced/online/distance learning tasks/activities to reinforce understanding of concepts related to computer programming skills, computer problem solving, and software design.
- Online and Laboratory activities and application assignments in order to address areas of improvement in computer programming, computer problem solving, and software design.
- Projects in order to facilitate and demonstrate the acquisition of skills required to create computer programs.
- Collaborative projects/cooperative learning tasks in order to encourage students to develop and apply computer programming, computer problem solving, software design, and team work skills.

## Methods of Evaluation:

Students will be evaluated for progress in and/or mastery of student learning outcomes using methods of evaluation which may include, but are not limited to, the following activities:

- Computer programs designed to demonstrate the acquisition of computer programming, computer problem solving, and software design concepts and skills.
- Quizzes/examinations designed to measure students' degree of mastery of fundamental computer programming and software design concepts and terminology.
- Collaborative projects designed to demonstrate successful understanding and application of computer programming, computer problem solving, software design, and team work skills.
- Computer Laboratory assignments/projects designed to clarify students' individual computer programming, computer problem solving, software design strengths and areas of improvement related to these skills.
- Common final project designed to evaluate students' overall achievement of course objectives in computer programming, computer problem solving, and software design concepts.

## Sample Assignments:

**Outside-of-Class Reading Assignments**

The primary assignments for this course involve the creation of programs using Python. To support that, students will be assigned textbook reading and/or other resource reading that covers programming concepts and demonstrates usage of Python.

**Outside-of-Class Writing Assignments**

- Assignments for this course involve writing Python programming statements to form complete programs that carry out specific tasks. The programs are compiled, debugged and executed to display solutions.
- Students will also be asked to document their work with flowcharts and written explanations that clarify their programming code.

**Other Outside-of-Class Assignments**

The primary assignments for this course involve the creation of programs using Python. Additional exercises in the computer lab will entail testing software and troubleshooting coded solutions to problems.

## Course Materials:

All materials used in this course will be periodically reviewed to ensure that they are appropriate for college level instruction. Possible texts include the following:

Lambert, Kenneth A., Osbourne, Martin. *Fundamentals of Python: First Programs.* Second Edition Cengage, 2019.
Matthes. *Python: A Hands-On Project-Based Introduction to Programming.* 2nd Edition No Starch Press, 2019.
Anaconda. Software. 2020. https://www.anaconda.com/products/individual, https://docs.anaconda.com/anaconda/install/ Individual free-license for Scientific Computing.
Draw.io: Online Diagram Software and Flow Chart Software. Software. 2020. https://draw.io, Free online - Flowcharting and UML Diagramming..
Git and Github. Software. 2020. https://git-scm.com/ and https://github.com/, Software Development Repositories. Free online software and accounts available over the internet..

## Codes/Dates:

**Board of Trustees Approval Date:** None
**COR Rev Date:** None

Generated on: 9/8/2020 7:13:30 AM