

Python Programming 2: Algorithms, modeling, and data processing

Fall 2019 Syllabus:The Community College of Allegheny County.

Jump to a section:

[Catalog Description](#)
[Course Scope](#)
[Learning Outcomes](#)
[Listed Topics](#)
[Learning Cycle](#)
[Assessment](#)
[Student Resources](#)
[Outside References](#)
[Syllabus Author](#)

Course Catalog Description

Building on language foundations developed in Python 1, this second semester python course focuses on the language's powerful file processing and data manipulation tools. Students will explore core libraries that allow programs to access operating system services, manipulate data of many types, interact with the user through graphical user interfaces (GUIs), and crunch out data metrics. This fast-paced course is project-focused and builds not only python programming skills but also best practices in object-oriented software design.

[back to top](#)

Course Scope

The following flow chart depicts the scope of this proposed second course in python programming. Three semesters of python coursework is included in the chart to provide context for the four course components under study in this course--these are the colored rectangles.

The knowledge blocks shown in the diagram are organized into learning components symbolized by the rectangles with thick borders and bold text. These larger, umbrella-like components are broken down into smaller learning modules which progress in a sequence shown with arrows flowing throughout the diagram.

Learning to program in any computer language is an iterative process in which more advanced learning depends on mastery of language fundamentals. The core of the python language is encapsulated in the Language Core and Object Oriented Python components in the diagram. Together, these two components represent the scope of a first course in Python.

Once the core components of a language are internalized, carrying out more complex computations with a variety of data types and design goals becomes possible, and these are the "middle four" components in the diagram below. They may be taught in any order, but a suggested sequence is shown in with arrows.

[back to top](#)

Learning Outcomes

File manipulation and data storage

1. Identify and load a python library suitable for processing files of a given type
2. Integrate a python file processing library into a program to accomplish application objectives
3. Integrate an operating system process into a given program, making use of core python OS-related objects

Graphical User Interfaces (GUIs)

1. Create instances of the core python graphical user interface (GUI) components: buttons, text boxes, select boxes, and images
2. Use data-display related GUI components to convey meaningful information extracted from a simple data set
3. Conduct a user interview to determine design requirements for a GUI and implement those findings into a working program
4. Implement a user-centered design in python and gather user feedback to a prototype

Algorithm Design

1. Model the core phases of smart algorithm design with a simple, non-technical design problem
2. Convert a given algorithm written in English into working python code and test its functionality
3. Design a new algorithm to solve a technical problem
4. Implement the new algorithm in python and test its functionality

Simulation

1. Implement the classic Monto Carlo simulation method to a given competitive modeling scenario
2. Creatively design and implement a simulation of a given human or system interaction using best practices in design phases

General

1. Using a version control system, like git, curate an online portfolio of working and documented python code from at least 2 course projects
2. Effectively discuss their python skills and their applications to an employer during a practice interview

[back to top](#)

Listed Topics

File manipulation and data storage

- File types and python object adapters
- Looping through files with dictionaries
- File-based data stores
- Operating system interaction

Graphical User Interfaces (GUIs)

- User-interface GUI components
- Data display GUI components
- GUI Design through user interview
- User-centered design

Algorithm Design

- Top-down design approach
- Psuedocode versions of algorithms
- Algorithm implementation in python
- Searching, sorting, and traversal algorithms

Simulation

- Monte carlo simulations
- Simulation design phases
- Model and unit testing

General

- Technical interview preparation

[back to top](#)

Assessment of learning objectives

Assessment Philosophy

As a lab-like course built around using python code to solve non-trivial, business-related problems, course assessments in python 2 are based on fully-baked student work products. In relation to the course learning model diagrammed above, student work projects emerge at the end of each module and at the conclusion of the component's culminating project.

The instructor provides incremental feedback to students during the course of the module's individual project work time--often called formative assessment. Small misunderstandings or trouble spots that emerge inside a module can be ironed out before they impede the larger learning goals of the component. After all modules are mastered and a final project completed the instructor offers additional, formal feedback concerning the project's alignment to its design specifications is provided.

Students complete the following steps in advance of their presentation and feedback session for their culminating project:

- Program design specifications
- Program flow diagram adjusted to reflect actual implementation
- Thoughtful responses to "heart-of-the-matter" questions

[back to top](#)

Using design criteria alignment in place of rubrics

The best assessment tools are those with which the students directly engage in creating and using. This can take the form of a class-generated project rubric, for example. As students create assessment criteria prior to implementing a project, the resulting work is both more likely to align to the assessment criteria and meaningfully assist students in completing their work. When that rubric is then used by the students to assess their own work, valuable mental processes are underway which tend to naturally improve skill and confidence.

Rubrics are widespread and useful tools for many types of student work outside of the technical design realm. In a coding class, such as this python 2 course, the process of assessing student code against initial design requirements organically takes the place of rubric-based assessment without displacing its generic value as a teaching tool.

[back to top](#)



Mapping project performance to course letter grades

The following table serves as a possible correlation guide between module and component project assessment and the formal course letter grades instructors assign to each student at the conclusion of the semester:

Course Letter grade	Student performance criteria
A	Independent practice for each model is completed and documented . Culminating projects for each component meet all specified design criteria. Component reflections show evidence of synthesis with other technical learning domains.
B	Independent practice for each module has been attempted but not consistently documented to reveal command of the code. Culminating projects for each component meets some but not all design criteria. Component reflections show moderate thought, limited to current learning topics .
C	Independent practice for 1/2 to 2/3 of modules has been attempted but not consistently documented. Culminating projects for each component meets some but not all design criteria. Component reflections show low levels of thought relative to A and B work.
D	Independent practice for less than 1/2 of modules has been attempted but not consistently documented. Culminating projects for each component meets few, if any design criteria. Component reflections are incomplete .
F	Independent practice for 1/4th or fewer of modules has been attempted and not consistently documented. Culminating projects were not meaningfully attempted . Component reflections were not attempted.

[back to top](#)





Recommended student resources

- Print Book: Python Programming: An Introduction to Computer Science, 2nd Edition, by John Zelle, Franklin, Beedle, and Associates Independent Press. [Amazon link](#). 
- Online learning tool: Code Academy's Interactive Python programming course
- Python Software Foundation's [Python 3 Language reference](#) 

[back to top](#)

References for syllabus design

The following resources were consulted in the design of this course syllabus:

- University of Washington's [Python 100 course syllabus](#) 
- Harvard University's CSCI E-7 [Introduction to Python course syllabus and resource list](#) 
- Python Programming: An Introduction to Computer Science, 2nd Edition, by John Zelle, Franklin, Beedle, and Associates Independent Press. [Amazon link](#). 
- Think Python: How to Think Like a Computer Scientist, 2nd edition, by Allen Downey, Green Tea Press. [Link to PDF of book online](#). 

Accommodations for Individuals with Disabilities:

The college recognizes its responsibility to provide academic and nonacademic services and programs equally to individuals with and without disabilities. To this end, the college provides reasonable accommodations for qualified students and employees with documented disabilities consistent with the requirements of the Americans with Disabilities Act, sections 503 and 504 of the Rehabilitation Act and other federal, state and local laws and regulations. The college maintains an Office of Supportive Services at each campus location to receive, review and evaluate requests from students who require an accommodation with respect to their educational program. Students' requesting reasonable accommodations due to a documented disability must first register with their campus' Supportive Services Office and obtain an official letter identifying approved accommodations to be distributed to their faculty members.

Attendance Procedure for Pregnancy and Pregnancy Related Conditions:

In accordance with Title IX of the Education Amendments of 1972, absences due to pregnancy or related conditions, including recovery from childbirth, shall be excused for as long as the absences are determined to be medically necessary. Students will be provided with the opportunity to make up any work missed as a result of such absences, if possible. For more information or requests for accommodations, students should inform their instructor(s) and/or contact the Civil Rights Compliance Officer/Title IX Coordinator, Sumana Misra-Zets, at 412.237.4535 or smisra@ccac.edu.

Attendance Procedure for Religious Observance

The college will make reasonable efforts to accommodate students who must be absent from classes or miss scheduled exams in order to observe a religious holiday or participate in some other form of religious observance. Students shall be provided, whenever possible, reasonable opportunity to make up academic assignments missed due to such absences, unless doing so would create or impose an undue burden on other students or the College. It shall be the students' responsibility to provide written notice via the Request for Accommodation for Religious Observances Form (accessible at <https://www.ccac.edu/nondiscrimination/>) to every instructor for each course in which an accommodation is being requested. For more information contact the Civil Rights Compliance Officer/Title IX Coordinator, Sumana Misra-Zets, at 412.237.4535 or smisra@ccac.edu.

Chosen First Name Procedure for Students

Many individuals use names other than their legal first name to identify themselves for a variety of personal and/or cultural reasons. The college seeks to provide an inclusive and non-discriminatory environment by making it possible for students to use a chosen first name on college records when a legal name is not required. Chosen first names may not be applicable in certain programs due to the requirements of accreditation organizations and clinical sites. For more information, please see the [Student Handbook](#)

Syllabus Author

Eric Darsow, Faculty, Computer Information Technology department, Community College of Allegheny County