School of Computing
UNIVERSITY OF GEORGIA

Course Information Sheet

CSCI 1300—1300L

Introduction to Programming with Python

Brief Course Description
(50-words or less)

Introduction to algorithmic problem solving using the Python programming language. Basic techniques of program development and supportive software tools. Programming projects.

Extended Course Description / Comments

This course is an introduction to algorithmic problem solving using the Python programming language: objects, values, types, expressions, simple statements, compound statements, inputs, and outputs. This course includes programming projects that incorporate algorithm design and implementation in Python 3 or above. This course is intended for anyone who is interested in learning how to program. Python is the most widely used programming language in Data Science, Open Science, and Web application development.

Pre-Requisite

None

Required, Elective or Selected Elective

Elective Course

Approved Textbooks
(if more than one listed, the textbook used is up to the instructor's discretion)

Authors:	Al Sweigart
Title:	Automate the Boring Stuff with Python - Practical Programming for Total Beginners
Edition:	2nd Edition
ISBN-13:	978-1593279929
OER:	https://automatetheboringstuff.com/

Specific Learning Outcomes (Performance Indicators)

This course presents basic programming topics in the Python programming language. The term "develop" should be interpreted as following the software development life cycle (SDLC), including reading, testing, executing, and basic debugging. At the end of the semester, all students will be able to:

1. Develop and implement algorithms that use basic programming concepts like expressions, variables, operators, input, and output.
2. Develop and implement algorithms that use Boolean expressions, control flow statements, and functions from the standard library to intelligently respond to different conditions.
3. Organize code into more manageable chunks by developing programs that define and use functions; and develop programs that handle potential errors.
4. Develop and implement maintainable programs that organize and manipulate data using Python data types like lists and dictionaries.
5. Develop and implement algorithms that manipulate strings to produce desired program output.
6. Develop and implement algorithms that combine a majority of the other learning outcomes to produce a solution to a set of problems.

ABET Learning Outcomes

Graduates of the program will have an ability to:

A. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
B. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
C. Communicate effectively in a variety of professional contexts.

D. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
E. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
F. Apply computer science theory and software development fundamentals to produce computing-based solutions.

NOTE: In the construction of the student learning outcomes for this course, the instructors interpreted "computing requirements" in (B) as the functional requirements for a software solution and not as specific hardware requirements for the target platform; likewise, the phrase "[a]pply computer science theory" in (F) was interpreted as using computer science principles.

This course is not assessed for ABET.

Relationship Between Student Outcomes and Learning Outcomes

	ABET Learning Outcomes						
Specific Learning Outcomes		A	B	C	D	E	F
	1		●				
	2		●				●
	3		●				●
	4		●				
	5		●				
	6		●				
	7		●				

Major Topics Covered

1. **Python Basics** (Knowledge level: Usage)
   a) Construct expressions involving variables and operators.
   b) Assign user input to variables.
   c) Display evaluated expressions on the screen.
   d) Use the Python interactive shell and development environments such as IDLE and Jupyter to write code.

2. **Control Flow and Algorithms** (Knowledge level: Usage)
   a) Construct Boolean expressions involving variables, Boolean operators, and comparison operators.
   b) Use if, elif, and else statements to control the flow of program execution.
   c) Use while, break, and continue statements to control the flow of program execution.
   d) Use for statements to control the flow of program execution.
   e) Import and use functions from the Python standard library.

3. **Functions and Algorithms** (Knowledge level: Usage)
   a) Construct basic functions using def statements and value-returning functions using def and return statements.
   b) Establish the scope of a variable as either local or global.
   c) Use try and except statements to handle lines of code that could potentially have an error.
   d) Use the open() function and methods like read() and close() to interact with files.

4. **Lists** (Knowledge level: Usage)

a) Construct expressions involving list expressions.
b) Use index values and slices to get or change values in a list.
c) Use del statements to remove values from a list.
d) Use for statements to traverse a list.
e) Construct Boolean expressions involving lists using the in and not in operators.
f) Call methods on lists, including index(), append(), insert(), remove(), and sort().
g) Modify a single list using multiple variables containing the same list reference value; pass list references into functions.
h) Use the copy module’s copy() and deepcopy() functions to make shallow and deep copies of a list.

5. **Dictionaries** (Knowledge level: Usage)
   a) Construct expressions involving dictionary expressions.
   b) Use keys to get or change values in a dictionary.
   c) Call methods on dictionaries, including keys(), values(), items(), get(), and setdefault().
   d) Construct Boolean expressions involving dictionaries.
   e) Use the pprint module’s pprint function to pretty print a dictionary.
   f) Use a dictionary to model real-world objects.

6. **String Manipulation** (Knowledge level: Usage)
   a) Construct simple string literals.
   b) Construct string literals involving escape characters.
   c) Construct Boolean expressions involving strings.
   d) Call methods on strings.

Knowledge Levels

The following is the ACM’s categorization of different levels of mastery: Assessment, Usage, and Familiarity. Note that Assessment encompasses both Usage and Familiarity, and Usage encompasses Familiarity.

Familiarity: The student understands what a concept is or what it means. This level of mastery concerns a basic awareness of a concept as opposed to expecting real facility with its application. It provides an answer to the question “What do you know about this?”

Usage: The student is able to use or apply a concept in a concrete way. Using a concept may include, for example, appropriately using a specific concept in a program, using a particular proof technique, or performing a particular analysis. It provides an answer to the question “What do you know how to do?”

Assessment: The student is able to consider a concept from multiple viewpoints and/or justify the selection of a particular approach to solve a problem. This level of mastery implies more than using a concept; it involves the ability to select an appropriate approach from understood alternatives. It provides an answer to the question “Why would you do that?”

Course Master

Dr. Michael E. Cotterell

Modified

- 11/27/2019 by Dr. Michael Cotterell
- 02/09/2024 by Dr. Michael Cotterell