Spring 2017

CST1101–Problem Solving with Computer Programming, Syllabus

Elena Filatova
CUNY New York City College of Technology

NEW YORK CITY COLLEGE OF TECHNOLOGY/CUNY
DEPARTMENT OF COMPUTER SYSTEMS TECHNOLOGY

CST1101–PROBLEM SOLVING WITH COMPUTER PROGRAMMING
(4 hours – 3 credits)

Instructor:
Name: Prof.
Office:
e-mail:
Office Hours:

Course Description:
This course introduces students to concepts of problem solving using constructs of logic inherent in computer programming languages. Students study the nature of problems, common solution approaches and analysis techniques. Students use a flowchart interpreter to diagram problem solutions. Students learn the basics of computer programming by learning Python. Both Python scripts and flowcharts enable students to construct solutions to common algorithmic problems. The major emphasis is on teaching the student to identify solutions to a problem and translate them into various forms that will enable the computer to perform some of the steps in a solution of an actual problem instance. These forms include flowcharting tool, viewing generated software code and the basics of debugging the code. At the end of the class students will write a project Python scripts that demonstrates the students' knowledge of all the basic programming concepts discussed in class (e.g., variables, conditions, loops, functions).

Course Objectives:
Upon successful completion of the course, students should be able to:

1. Demonstrate understanding of the steps required to solve a problem using a computer.
2. Demonstrate understanding of flowcharting techniques to solve an algorithm.
3. Demonstrate the knowledge or Boolean algebra (AND, OR, NOT operations).
4. Demonstrate understanding of the major programming notions: variables, decision statements, repetition/loop statements (both count- and event-controlled), arrays/lists, modules/functions, classes and objects and their use for basic problem solving.
5. Demonstrate understanding of the two major programming paradigms: procedural and object-oriented.
6. Install and run the IDLE Python programming environment.
7. Design and implement basic Python scripts.
8. Demonstrate broad problem-solving experience by referring to solutions from a problem bank covered during class.

General Education Outcomes:

- **SKILLS/Inquiry/Analysis:** Students will employ scientific reasoning and logical thinking.

- **SKILLS/Communication:** Students will communicate in diverse settings using oral (both speaking and listening) and visual means.

- **VALUES, ETHICS, RELATIONSHIPS / Professional/Personal Development:** Students will have access to on-line materials and solutions to programming problems and will be required to process those materials and solutions, understand them, use the ideas from them without passing others' ideas as their own.

Prerequisite – CUNY certification in mathematics, reading and writing. General knowledge of a personal computer is needed. Students may enroll in a workshop at the Academic Learning Center, located in the Atrium.

This is an OER (Open Educational Resources) course. All the required reading materials are free. The OER page for the course can be viewed here:
https://openlab.citytech.cuny.edu/cst1101-problemsolvingpython

Software Download (free, online)

- Python official site that includes documentation, downloads (IDLE for Python 3.6), news:

  https://www.python.org

- Flowchart interpreter

  http://www.flowgorithm.org/

Required Reading (free, online)

- Think Python, 2nd Edition by Allen B. Downey
  

  http://greenteapress.com/wp/think-python-2e/

- How to Think Like a Computer Scientist: Interactive Edition

  http://interactivepython.org/runestone/static/thinkcspy/index.html

Recommended reading (free, online)

- Algorithmic Problem Solving with Python by John B. Schneider, Shira Lynn Broschat, and Jess Dahmen.

  

  http://www.eecs.wsu.edu/~schneidj/swan/index.php

- How to Think Like a Computer Scientist by Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers

  http://www.openbookproject.net/thinkcs/python/english3e/

- Python Bibliotheca: http://www.openbookproject.net/pybiblio/

Tentative Evaluation and Grading Distribution (the exact grade distribution is defined by the instructors teaching individual sections)

Lab, Homework Assignments, quizzes	30%
Test1	15%
Test2	15%
Project	10%
Final Exam (cumulative)	30%
	=====
Total	100%

Grade System:

Letter Grade	A	A-	B+	B	B-	C+	C	D	F
Numerical Grade	93-100	90-92.9	87-89.9	83-86.9	80-82.9	77-79.9	70-76.9	60-69.9	<=59.9

The grade distribution follows the information in the NYCCT Student Handbook (p.43).

During the course of the class you are required to follow the NYCCT Academic Integrity Standards described in the Student Handbook (pp.95 – 99)

NYCCT Student Handbook can be downloaded here: http://www.citytech.cuny.edu/current-student/docs/StudentHandbook.pdf.

Storage Media -- You must have a USB storage media.

Tentative schedule

- **the changes in the schedule can be made to address the pace of an individual class**
- **Reading assignment for the topics can be checked on the CST 1101 OER site:** https://openlab.citytech.cuny.edu/cst1101-problemsolvingpython/assignments/

	Topic name
1 2 3	Topic 01 Class logistics; Introduction: • What is problem solving? • Why Python? • Why Flowcharts? Computer problem solving: • Solution = program / algorithm • Well-defined set of steps • Examples of problems solved using sets of steps: cooking recipes, puzzles Computer problem solving Elementary program examples
4 5	Topic 02 IDLE introduction, installation tips, Python 3.6 My first "Hello, World" program Code readability and comments, Introduction of two modes for Python IDLE (interactive and shell). Saving Python scripts and flowcharts. How saved scripts and flowcharts can be called and run/executed. Why interactive mode is not enough?
6 7	Topic 03 Variables, types, and data input/output The idea of a variable is introduced and the dynamics of the assignment statement are detailed. Three basic types are illustrated through examples: integers, floats, strings, Boolean. Type conversion
8 9 10	Topic 04 Boolean logic. Conditional execution (if-else) Boolean type is introduced together with three Boolean operations: and, or, not. Program structure and program flow. Demonstration of branching using flowcharts. Conditions/selections in Python: If If-else If-elif

11 12 13	Topic 05 Modules/functions Why creating modules within a program? Examples of modules (functions) Parameters / arguments Passing parameters
14	Test1
15 16	Topic 06 While loop Condition controlled loop
18	Topic 07 Lists
18 19 20	Topic 08 For loop and lists For vs While loop For loop with Range: different settings.
21	Test 2
22 23 24	Topic 09 String as a special case of a list Strings and iteration Importing modules
25	Topic 10 Introduction of the OOP paradigm.
26 27	Topic 11 Turtle graphics library. Turtles-objects. Use Turtle Graphics to review the notion of an object and basic programming tools: condition and selection.
28	Additional topics based on the professor interest / leftover material / repetition of the topics that caused most problems and questions during the semester.
29	Review for the final
30	Final (cumulative)

Assessment Criteria

For the successful completion of this course a student should be able to:	Evaluation methods and criteria

1. Demonstrate understanding of the steps required to solve a problem using a computer.	Students will describe problem, identify inputs, processes and desired outcomes in laboratory assignments, class work and tests.
2. Demonstrate understanding of flowcharting techniques to solve an algorithm.	Students will solve problems using the flowchart interpreter software and Python 2.7 in laboratory assignments, class work and tests.
3. Demonstrate the knowledge or Boolean algebra (AND, OR, NOT operations)	Students will solve Boolean algebra problems in laboratory assignments, class work and tests and incorporate these solutions in flowcharts and Python scripts.
4. Demonstrate understanding of the major programming notions: variables, decision statements, repetition/loop statements (both count- and event-controlled), arrays/lists, modules/functions, classes and objects and their use for basic problem solving.	Students will create algorithms for problem solving using the basic programming notions in laboratory assignments, class work and tests.
5. Demonstrate understanding of the two major programming paradigms: procedural and object-oriented.	Students will create new classes and objects of these classes in laboratory assignments, class work and tests.
6. Install and run the IDLE Python programming environment.	To complete homework assignments and practice programming skills outside the college students will install the IDLE Python environment on their own computers.
7. Design and implement basic Python scripts.	Students will use the knowledge of Boolean Algebra, problem solving paradigms and basic programming notions to write Python scripts in laboratory assignments, class work and tests.
8. Demonstrate broad problem-solving experience by referring to solutions from a problem bank covered during class	Students will demonstrate problem-solving ability in laboratory assignments, class work and tests.

General Education Outcomes and Assessment:

Learning Outcomes	**Assessment Method**
SKILLS/Inquiry/Analysis Students will employ scientific reasoning and logical thinking.	Students will describe problem, identify inputs, processes and desired outcomes

	in laboratory assignments, class work and tests. Students will solve problems using the flowchart interpreter software and Python in laboratory assignments, class work and tests. Students will identify coding paradigms in Laboratory Assignments, Class work and tests
SKILLS/Communication Students will communicate in diverse settings using oral (both speaking and listening) and visual means.	Students will discuss various problems and approaches towards solving these problems in class
VALUES, ETHICS, RELATIONSHIPS / Professional/Personal Development Students will have access to on-line materials and solutions to programming problems and will be required to process those materials and solutions, understand them, use the ideas from them without passing others' ideas as their own.	Students will learn to respectfully use the code generated by other programmers giving.