Carnegie Mellon University
Computer Science Academy

# 15-111, College-Level Programming and Computer Science Syllabus

## 1. Course Description

A technical introduction to the fundamentals of programming with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, informal analysis, and effective testing and debugging. Starting from first principles, we will cover a large subset of the Python programming language, including its standard libraries and programming paradigms.

This is a rigorous and fast-paced course that requires at least one year of prior programming experience at the high school level. It is equivalent to Carnegie Mellon University's 15-112 course and should substitute for any first-semester college programming course as a result.

NOTE: students must achieve a C or better on the proctored final to satisfy the prerequisite for any subsequent Computer Science course at Carnegie Mellon such as 15-122 or 15-150.

## 2. Learning Objectives

At the end of the course, students should be able to:

**1) Write clear, robust, and efficient code in Python using:**

- sequential, conditional, and loop statements
- strings, lists, tuples, sets, and dictionaries
- objects and classes
- recursive approaches
- graphics and interaction

**2) Develop programs to effectively solve medium-sized tasks by:**

- employing modular, top-down design in program construction
- demonstrating an effective programming style based on established standards, practices, and guidelines
- proactively creating and writing test cases to test and debug code
- applying computational problem-solving skills to new problems
- explaining and analyzing the efficiency of algorithms, particularly by predicting the Big-O running time of small pieces of code

**3) Design and write a substantial (500-1500 line) program in Python with minimal guidance**

## 3. Topic List:

- **1 Basic Programming Constructs**
  - **1.1 Getting Started**
  - **1.2 Data, Expressions, and Variables**
    - 1.2.1 Types
    - 1.2.2 Constants
    - 1.2.3 Variables
    - 1.2.4 Operators
    - 1.2.5 More About Operators
    - 1.2.6 Code Tracing
  - **1.3 Functions**
    - 1.3.1 Intro
    - 1.3.2 Helpful Functions
    - 1.3.3 Writing Your Own Functions
    - 1.3.4 Variable Scope
    - 1.3.5 Print Versus Return
    - 1.3.6 Test Functions
    - 1.3.7 Helper Functions
    - 1.3.8 Console IO Functions
    - 1.3.9 Code Tracing with Functions
  - **1.4 Conditionals**
    - 1.4.1 if Statements
    - 1.4.2 if-else Statements
    - 1.4.3 Nested Conditionals
    - 1.4.4 if-elif-else
    - 1.4.5 if-else Expressions
    - 1.4.6 Short-Circuit Evaluation
    - 1.4.7 Improper Use of Conditionals
    - 1.4.8 Code Tracing with Conditionals
- **2 Loops And Strings**
  - **2.1 Getting Started**
  - **2.2 Loops**
    - 2.2.1 for loops and range
    - 2.2.2 Nested for loops

        - 6.4.7 Efficiency Practice
- **7 Recursion**
    - **7.1 Getting Started**
    - **7.2 Code Tracing with Recursion**
    - **7.3 Recursion Practice**
    - **7.4 More Recursion Practice**
    - **7.5 Some Recursion Issues**
    - **7.6 Some Recursion Classics**
        - 7.6.2 Greatest Common Divisor (gcd)
        - 7.6.3 Towers of Hanoi
        - 7.6.4 Flood Fill
        - 7.6.5 Quick Sort
        - 7.6.6 Merge Sort
- **8 Object Oriented Programming (OOP)**
    - 8.1 Getting Started
    - 8.2 Writing Classes
    - 8.3 Writing Methods
    - 8.4 Equality Testing
    - 8.5 Using Instances in Sets and Dictionaries
    - 8.6 Class Attributes
    - 8.7 Additional Topics
    - 8.8 Wrapping Up
    - 8.9 Code Tracing with OOP