An Introduction to Interactive Programming in Python (Part 1)

Upcoming session: Dec 12 — Jan 23.

Commitment

5 weeks of study, 7-10 hours/week

About the Course

This two-part course is designed to help students with very little or no computing background learn the basics of building simple interactive applications. Our language of choice, Python, is an easy-to learn, high-level computer language that is used in many of the computational courses offered on Coursera. To make learning Python easy, we have developed a new browser-based programming environment that makes developing interactive applications in Python simple. These applications will involve windows whose contents are graphical and respond to buttons, the keyboard and the mouse.

In part 1 of this course, we will introduce the basic elements of programming (such as expressions, conditionals, and functions) and then use these elements to create simple interactive applications such as a digital stopwatch. Part 1 of this class will culminate in building a version of the classic arcade game "Pong".

Recommended Background - A knowledge of high school mathematics is required. While the class is designed for students with no prior programming experience, some beginning programmers have viewed the class as being fast-paced. For students interested in some light preparation prior to the start of class, we recommend a self-paced Python learning site such as codecademy.com.

WEEK 1

Week 0 - Statements, expressions, variables

Understand the structure of this class, explore Python as a calculator

Video · Introduction

Video · CodeSkulptor

Video · Arithmetic Expressions

Reading · Practice Exercises for Expressions (optional)

Video · Variables

Video · Saving in CodeSkulptor

Reading · Practice Exercises for Variables and Assignments (optional)

Quiz · Quiz 0

Video · Mini-project Video

Reading · Mini-project Description

Reading · Code Clinic Tips
Practice Peer Review · "We want... a shrubbery!"
WEEK 2
Week 1 - Functions, logic, conditionals
Learn the basic constructs of Python programming, create a program that plays a variant of Rock-Paper-Scissors
Video · Functions
Video · Visualizing Functions
Video · More Operations
Reading · Practice Exercises for Functions (optional)
Video · Logic and Comparisons
Video · Conditionals
Video · Programming Tips - 1
Reading · Practice Exercises for Logic and Conditionals (optional)
Quiz · Quiz 1
Video · Mini-project Video
Reading · Mini-project Description
Reading · Practice Mini-project: Mystical Octosphere (optional)
Reading · Code Clinic Tips
Peer Review · Rock-paper-scissors-lizard-Spock
WEEK 3
Week 2 - Event-driven programming, local/global variables
Learn the basics of event-driven programming, understand difference between local and global variables, create an interactive program that plays a simple guessing game
Video · Event-Driven Programming
Video · Local vs. Global Variables
Video · SimpleGUI
Reading · Practice Exercises for Interactive Applications (optional)
Quiz · Quiz 2a
Video · Buttons
Video · Input Fields
Video · Visualizing Events
Video · Programming Tips - 2
Reading · Practice Exercises for Button and Input Fields (optional)
Quiz · Quiz 2b
Video · Mini-project Video
Reading · Mini-project Description
Reading · Practice Mini-project: Magical Octosphere Reloaded (optional)
Reading · Code Clinic Tips
Peer Review · "Guess the Number!"
WEEK 4

Week 3 - Canvas, drawing, timers
Create a canvas in Python, learn how to draw on the canvas, create a digital stopwatch
Video · Canvas and Drawing
Video · String Processing
Video · Interactive Drawing
Reading · Practice Exercises for Drawing (optional)
Quiz · Quiz 3a
Video · Timers
Video · Visualizing Drawing and Timers
Video · Programming Tips - 3
Reading · Practice Exercises for Timers (optional)
Quiz · Quiz 3b
Video · Mini-project Video
Reading · Mini-project Description
Reading · Code Clinic Tips
Peer Review · Stopwatch: The Game
WEEK 5
Week 4 - Lists, keyboard input, the basics of modeling motion
Learn the basics of lists in Python, model moving objects in Python, recreate the classic arcade game "Pong"
Video · Lists
Video · Keyboard Input
Video · Motion
Video · Collisions and Reflections
Reading · Practice Exercises for Lists (optional)
Quiz · Quiz 4a
Video · Velocity Control
Video · Visualizing Lists and Mutation
Video · Programming Tips - 4
Reading · Practice Exercises for Keyboard (optional)
Quiz · Quiz 4b
Video · Mini-project Video
Reading · Mini-project Description
Reading · Code Clinic Tips
Peer Review · Pong

COURSE 2

An Introduction to Interactive Programming in Python (Part 2)

Upcoming session: Dec 12 — Jan 16.

Commitment
4 weeks of study, 7-10 hours/week

About the Course
This two-part course is designed to help students with very little or no computing background learn the basics of building simple interactive applications. Our language of choice, Python, is an easy-to learn, high-level computer language that is used in many of the computational courses offered on Coursera. To make learning Python easy, we have developed a new browser-based programming environment that makes developing interactive applications in Python simple. These applications will involve windows whose contents are graphical and respond to buttons, the keyboard and the mouse.

In part 2 of this course, we will introduce more elements of programming (such as list, dictionaries, and loops) and then use these elements to create games such as Blackjack. Part 1 of this class will culminate in building a version of the classic arcade game "Asteroids". Upon completing this course, you will be able to write small, but interesting Python programs. The next course in the specialization will begin to introduce a more principled approach to writing programs and solving computational problems that will allow you to write larger and more complex programs.

WEEK 1
Week 5 - Mouse input, list methods, dictionaries
Read mouse input, learn about list methods and dictionaries, draw images
Video · Introduction
Video · Mouse input
Video · List Methods
Video · List Examples
Video · Iteration
Reading · Practice Exercises for Mouse and List Methods (optional)
Quiz · Quiz 5a
Video · Dictionaries
Video · Images
Video · Visualizing Iteration
Video · Programming Tips - 5
Reading · Practice Exercises for Dictionaries and Images (optional)
Quiz · Quiz 5b
Video · Mini-project Video
Reading · Mini-project Description
Reading · Code Clinic Tips
Peer Review · Memory
WEEK 2

Week 6 - Classes and object-oriented programming
Learn the basics of object-oriented programming in Python using classes, work with tiled images
Video · Object-oriented Programming - 1
Video · Object-oriented Programming - 2
Video · Working with Objects
Video · Classes for Blackjack
Reading · Practice Exercises for Classes (part 1) (optional)
Reading · Practice Exercise for Avatar class (optional)
Quiz · Quiz 6a
Video · Tiled Images
Video · Visualizing Objects
Video · Programming Tips - 6
Reading · Practice Exercises for Classes (part 2) (optional)
Quiz · Quiz 6b
Video · Mini-project Video
Reading · Mini-project Description
Reading · Code Clinic Tips
Peer Review · Blackjack
WEEK 3
Week 7 - Basic game physics, sprites
Understand the math of acceleration and friction, work with sprites, add sound to your game
Video · Acceleration and Friction
Video · Spaceship Class
Video · Sound
Quiz · Quiz 7a
Video · Sprite Class
Video · Programming Tips - 7
Reading · Practice Exercises for Sprites and Sound (optional)
Quiz · Quiz 7b
Video · Mini-project Video
Reading · Mini-project Description
Reading · Code Clinic Tips
Peer Review · Spaceship
WEEK 4
Week 8 - Sets and animation
Learn about sets in Python, compute collisions between sprites, animate sprites
Video · Sets
Video · Collisions for Sprites
Reading · Practice Exercises for Sets and Collisions (optional)

Video · Sprite Animation
Video · Programming Tips - 8
Quiz · Quiz 8
Video · Mini-project Video
Reading · Mini-project Description
Reading · Code Clinic Tips
Peer Review · RiceRocks
Video · Beyond CodeSkulptor
Video · Class Wrap-up

COURSE 3

Principles of Computing (Part 1)

Upcoming session: Dec 12 — Jan 23.
Commitment
4 weeks of study, 7-10 hours/week

About the Course
This two-part course builds upon the programming skills that you learned in our Introduction to Interactive Programming in Python course. We will augment those skills with both important programming practices and critical mathematical problem solving skills. These skills underlie larger scale computational problem solving and programming. The main focus of the class will be programming weekly mini-projects in Python that build upon the mathematical and programming principles that are taught in the class. To keep the class fun and engaging, many of the projects will involve working with strategy-based games.

In part 1 of this course, the programming aspect of the class will focus on coding standards and testing. The mathematical portion of the class will focus on probability, combinatorics, and counting with an eye towards practical applications of these concepts in Computer Science.

Recommended Background - Students should be comfortable writing small (100+ line) programs in Python using constructs such as lists, dictionaries and classes and also have a high-school math background that includes algebra and pre-calculus.

WEEK 1
Required Python knowledge, coding standards, and machine grading
This week, we will introduce you to the structure and standards of the Principles of Computing courses.
Video · Principles of Computing
Video · Introduction

Video · Required Python Knowledge
Reading · Required Python Knowledge
Video · Coding Style and Standards
Reading · Guidelines for Coding Style
Video · Python Modules
Reading · Importing Custom Modules in Python
Video · Machine Grading
Quiz · Homework 1
Video · Mini-project Video
Reading · Mini-project Description
Reading · 2048
Other · Assignment: 2048 (Merge)
Other · Assignment Submission History
Reading · Math Expressions for Homework
Reading · Math Notes on Functions
Reading · Practice Mini-project - Solitaire Mancala
Video · CodeSkulptor
Video · Saving in CodeSkulptor
Video · Beyond CodeSkulptor (15:12) (optional video)
Reading · Python Development Environments
WEEK 2
Testing, plotting, and grids
This week, we will explain the importance of testing. We will also learn to solve problems with grids.
Video · The Importance of Testing
Video · Testing
Reading · Building Tests for Python Programs
Video · Plotting
Video · Grids
Quiz · Homework 2
Video · Mini-project Video
Reading · Mini-project Description
Other · Assignment: 2048 (Full)
Reading · Math Notes on Grid Representations
Reading · Practice Activity - Testing Solitaire Mancala
WEEK 3
Probability, randomness, and objects/references
This we will learn how to use probability and randomness to solve problems.
Video · The Importance of Probability
Video · Basics of Probability
Video · Expected Value

Video · Monte Carlo Methods
Video · Objects and References
Quiz · Homework 3
Video · Mini-project Video
Reading · Mini-project Description
Reading · TTTBoard Class
Other · Assignment: Tic-Tac-Toe (Monte Carlo)
Reading · Math Notes on Basic Probability
Reading · Math Notes on Expected Value
Reading · Practice Mini-project - Nim (Monte Carlo)
Reading · Practice Activity - The Monty Hall Problem
WEEK 4
Combinatorics, generators, and debugging
This week, we will learn how to use combinatorics to solve problems.
Video · The Importance of Combinatorics
Video · Enumeration
Video · Permutations and Combinations
Video · Combinatorics and Password Breaking
Video · Debugging
Quiz · Homework 4
Video · Mini-project Video
Reading · Mini-project Description
Other · Assignment: Yahtzee
Reading · Math Notes on Enumeration
Reading · Math Notes on Permutations and Combinations
Reading · Practice Activity - Analyzing a Simple Dice Game
Reading · Practice Activity - Counting Game States
WEEK 5
Counting, growth of functions, higher-order functions
This week, we will explain the importance of counting in solving complex problems.
Video · The Importance of Counting
Video · Counting and Sums
Video · Functions: Finding the Max
Video · Higher-order Functions
Video · Plotting Statement Counts
Quiz · Homework 5
Video · Mini-project Video
Reading · Mini-project Description
Reading · BuildInfo Class
Other · Assignment: Cookie Clicker
Reading · Math Notes on Arithmetic Sums

Reading · Math Notes on Logarithms and Exponentials
Reading · Math Notes on Growth Rates of Functions
Reading · Practice Activity - Modeling the Growth of Functions
Reading · Practice Activity - The Case of the Greedy Boss

COURSE 4

Principles of Computing (Part 2)

Upcoming session: Dec 12 — Jan 16.

Commitment
4 weeks of study, 7-10 hours/week

About the Course
This two-part course introduces the basic mathematical and programming principles that underlie much of Computer Science. Understanding these principles is crucial to the process of creating efficient and well-structured solutions for computational problems. To get hands-on experience working with these concepts, we will use the Python programming language. The main focus of the class will be weekly mini-projects that build upon the mathematical and programming principles that are taught in the class. To keep the class fun and engaging, many of the projects will involve working with strategy-based games.

In part 2 of this course, the programming portion of the class will focus on concepts such as recursion, assertions, and invariants. The mathematical portion of the class will focus on searching, sorting, and recursive data structures. Upon completing this course, you will have a solid foundation in the principles of computation and programming. This will prepare you for the next course in the specialization, which will begin to introduce a structured approach to developing and analyzing algorithms. Developing such algorithmic thinking skills will be critical to writing large scale software and solving real world computational problems.

WEEK 1
Searching and Data Structures
This week, we will explain the importance of searching. We will also explore various data structures and learn about inheritance.
Video · Introduction
Video · The Importance of Searching
Video · Generators
Video · Stacks and Queues
Video · Inheritance
Video · Grid Class
Video · Grid search
Quiz · Homework 1

Video · Mini-project Video
Reading · Mini-project Description
Other · Assignment: Zombie Apocalypse
Other · Assignment Submission History
Reading · Math Notes on Growth Rates of Functions
Reading · Math Notes on Grid Representations
Reading · Math Notes on Breadth-First Search
Reading · Practice Activity - Sorting strings
Reading · Practice Activity - Working with Distance Fields
Video · CodeSkulptor (optional video from our previous class)
Video · Saving in CodeSkulptor (optional video from our previous class)
Video · Beyond CodeSkulptor (optional video from our previous class)
WEEK 2
Recursion
This week, we will explain the importance of recursion.
Video · The Importance of Recursion
Video · Recursion
Video · Binary Search
Video · Visualizing Recursion
Video · Recurrences
Video · Reading Files
Quiz · Homework 2
Video · Mini-project Video
Reading · Mini-project Description
Other · Assignment: Word Wrangler
Reading · Math Notes on Recurrence Relations
Reading · Practice Activity - Recursion
Reading · Practice Activity - Binary representations for numbers
Reading · Practice Activity - Visualizing recurrences
WEEK 3
Trees
This week, we will explain the importance of trees. We will also explore how to set up game trees so that we can efficiently search them.
Video · The Importance of Trees
Video · Lambda
Video · Trees
Video · Illustration of Trees
Video · Minimax
Quiz · Homework 3
Video · Mini-project Video
Reading · Mini-project Description

Reading · TTTBoard Class
Other · Assignment: Tic-Tac-Toe (Minimax)
Reading · Math Notes on Trees
Reading · Math Notes on Minimax
Reading · Practice Activity - Nim (Tree search)
Reading · Practice Activity - Drawing trees
WEEK 4
Modeling, Assertions, and Invariants
This week, we will explain the importance of modeling. We will also explore how to use assertions and invariants to ensure that our models are always consistent and correct.
Video · The Importance of Modeling
Video · Assertions
Video · Invariants
Video · Modeling
Video · Software Development
Reading · The Basics of the Fifteen Puzzle
Quiz · Homework 4
Video · Mini-project Video
Reading · Mini-project Description
Other · Assignment: The Fifteen Puzzle
Reading · Math Notes on Invariants
Reading · Practice Activity - Solitaire Tantrix
Video · What is Algorithmic Thinking?

COURSE 5

Algorithmic Thinking (Part 1)

Upcoming session: Dec 12 — Jan 16.
Commitment
4 weeks of study, 7-10 hours/week

About the Course
Experienced Computer Scientists analyze and solve computational problems at a level of abstraction that is beyond that of any particular programming language. This two-part course builds on the principles that you learned in our Principles of Computing course and is designed to train students in the mathematical concepts and process of "Algorithmic Thinking", allowing them to build simpler, more efficient solutions to real-world computational problems.

In part 1 of this course, we will study the notion of algorithmic efficiency and consider its application to several problems from graph theory. As the central part of the course, students will implement several important graph algorithms in Python and then use

these algorithms to analyze two large real-world data sets. The main focus of these tasks is to understand interaction between the algorithms and the structure of the data sets being analyzed by these algorithms.

Recommended Background - Students should be comfortable writing intermediate size (300+ line) programs in Python and have a basic understanding of searching, sorting, and recursion. Students should also have a solid math background that includes algebra, pre-calculus and a familiarity with the math concepts covered in "Principles of Computing".

WEEK 1

WEEK 1
Module 1 - Core Materials
What is Algorithmic Thinking?, class structure, graphs, brute-force algorithms
Video · What is Algorithmic Thinking?
Video · Class structure
Video · Pseudo-code
Video · The small-world problem
Video · Graphs and representation
Video · Paths and distances
Video · Brute force
Video · What Is algorithm efficiency?
Video · Measuring efficiency
Video · Efficiency of brute force distance
Video · Number of steps of brute force distance
Quiz · Homework #1
Reading · Class notes
Reading · Coding notes
Video · Coding styles and standards - PoC
Video · Machine grading - PoC
Video · Plotting data - PoC
Video · Peer assessment - "We want a shrubbery!" - IIPP
WEEK 2
Modules 1 - Project and Application
Graph representations, plotting, analysis of citation graphs
Reading · Project #1 Description
Other · Assignment: Degree Distribution for Graphs
Other · Project Submission History
Reading · Application #1 Description
Peer Review · Analysis of Citation Graphs
Reading · Application #1 Solution

WEEK 3
Module 2 - Core Materials
Asymptotic analysis, "big O" notation, pseudocode, breadth-first search
Video · Orders of growth
Video · Asymptotics
Video · Illustrating "Big O"
Video · Illustrating BFS
Video · Queues and boundary cases
Video · Pseudocode
Video · BFS running time - loose analysis
Video · BFS running time - tighter analysis
Video · BFS-based distance distribution
Quiz · Homework #2
WEEK 4
Module 2 - Project and Application
Connected components, graph resilience, and analysis of computer networks
Reading · Project #2 Description
Other · Assignment: Connected Components and Graph Resilience
Reading · Application #2 Description
Peer Review · Analysis of a Computer Network
Reading · Application #2 Solution

COURSE 6

Algorithmic Thinking (Part 2)

Upcoming session: Dec 12 — Jan 16.
Commitment
4 weeks of study, 7-10 hours/week

About the Course
Experienced Computer Scientists analyze and solve computational problems at a level of abstraction that is beyond that of any particular programming language. This two-part class is designed to train students in the mathematical concepts and process of "Algorithmic Thinking", allowing them to build simpler, more efficient solutions to computational problems.

In part 2 of this course, we will study advanced algorithmic techniques such as divide-and-conquer and dynamic programming. As the central part of the course, students will implement several algorithms in Python that incorporate these techniques and then use these algorithms to analyze two large real-world data sets. The main focus of these tasks is to understand interaction between the algorithms and the structure of the data sets being analyzed by these algorithms.

Once students have completed this class, they will have both the mathematical and programming skills to analyze, design, and program solutions to a wide range of computational problems. While this class will use Python as its vehicle of choice to practice Algorithmic Thinking, the concepts that you will learn in this class transcend any particular programming language.

WEEK 1
Module 3 - Core Materials
Sorting, searching, big-O notation, the Master Theorem
Video · What is Algorithmic Thinking?
Video · The sorting problem
Video · A simple quadratic algorithm
Video · Illustrating MergeSort
Video · The recurrence for MergeSort
Video · The Master Theorem and MergeSort efficiency
Video · Linear vs. binary search
Video · Efficiency of binary search
Quiz · Homework #3
Video · Class structure (from part 1)
Reading · Class notes
Reading · Coding notes
Video · Coding styles and standards - PoC
Video · Testing and machine grading - PoC
Video · Plotting data - PoC
Video · Peer assessment - "We want a shrubbery!" - IIPP
WEEK 2
Module 3 - Project and Application
Closest pairs of points, clustering of points, comparison of clustering algorithms
Reading · Project #3 Description
Reading · Tests and Tips for Implementing the Clustering Methods
Other · Assignment: Closest Pairs and Clustering Algorithms
Other · Project Submission History
Reading · Application #3 Description
Peer Review · Comparison of Clustering Algorithms
Reading · Application #3 Solution
WEEK 3
Module 4 - Core Materials
Dynamic programming, running time of DP algorithms, local and global sequence alignment
Video · The RNA secondary structure problem

Video · A dynamic programming algorithm
Video · Illustrating the DP algorithm
Video · Running time of the DP algorithm
Video · DP vs. recursive implementation
Video · Global pairwise sequence alignment
Video · Local pairwise sequence alignment
Quiz · Homework 4
WEEK 4
Module 4 - Project and Application
Computation of sequence alignments, applications to genomics and text comparison
Reading · Project #4 Description
Other · Assignment: Computing Alignments of Sequences
Reading · Application #4 Description
Peer Review · Applications to Genomics and Beyond
Reading · Application #4 Solution
Video · Class wrap-up

COURSE 7

The Fundamentals of Computing Capstone Exam

Upcoming session: Feb 6 — Feb 20.
react-text: 172 /react-text
About the Capstone Project
While most specializations on Coursera conclude with a project-based course, students in the "Fundamentals of Computing" specialization have completed more than 20+ projects during the first six courses of the specialization. Given that much of the material in these courses is reused from session to session, our goal in this capstone class is to provide a conclusion to the specialization that allows each student an opportunity to demonstrate their individual mastery of the material in the specialization.

With this objective in mind, the focus in this Capstone class will be an exam whose questions are updated periodically. This approach is designed to help insure that each student is solving the exam problems on his/her own without outside help. For students that have done their own work, we do not anticipate that the exam will be particularly hard. However, those students who have relied too heavily on outside help in previous classes may have a difficult time. We believe that this approach will increase the value of the Certificate for this specialization.

WEEK 1
Fundamentals of Computing Capstone Exam

Complete a 25 question exam to demonstrate your mastery of the material in the Specialization
Video · Class Overview
Reading · Exam Preparation
Reading · Frequently Asked Questions
Reading · Honor Code (IMPORTANT)
Quiz · Capstone Exam
Video · Class Wrap Up