

CS202: Coding in Python 2

Course Syllabus

Course Description

This course continues the in-depth introduction to coding in [Python](#) from CS201. During the course, students will master fundamental data structures such as lists, tuples, and dictionaries. Students will also gain proficiency with advanced topics including for-each loops, string operations, web APIs, and user-defined functions.

In addition, students will learn industry practices such as pair programming, code reviews, and role-based project development. Throughout the course, students will continuously demonstrate their knowledge through both traditional assessments and real-world coding projects, crafting the foundations of their professional portfolio. This course will also prepare students to complete the Python Institute's Certified Entry-Level Python Programmer certification exam.

Course Outline

| Unit 4: Lists | Unit 5: Data Structures | Unit 6: Functions |
|---|--|--|
| Lesson 4.1 Lists and For-Each Loops | Lesson 5.1 Tuples | Lesson 6.1 User-Defined Functions |
| Lesson 4.2 List Operations | Lesson 5.2 Dictionaries | Lesson 6.2 Scope and References |
| Unit 4 Quiz | Unit 5 Quiz | Industry Practice Docstrings |
| Research Project Private Data | Research Project Computer Networks | Unit 6 Quiz |
| Lesson 4.3 Advanced List Operations | Lesson 5.3 Nested Collections | Research Project Preventing Cyberattacks |
| Lesson 4.4 Strings as Collections | Lesson 5.4 Web APIs | Lesson 6.3 Advanced Parameters |
| Lesson 4.5 String Operations | Unit 5 Test | Unit 6 Test |
| Unit 4 Test | Industry Practice Code Review | Unit 6 Project |
| Industry Practice Pair Coding | Unit 5 Project | |
| Unit 4 Project | | |

Learning Objectives

| Unit 4: Lists | |
|---------------------------------------|---|
| 4.1: Lists and For-Each Loops | |
| Comprehension Objectives | <ul style="list-style-type: none"> ● Define a list as a changeable collection of ordered information ● Identify the valid and invalid indexes in a list ● Describe the properties of elements contained in a list ● Describe how the loop variable of a for-each loop changes when iterating over a list ● Compare and contrast different types of loops |
| Application Objectives | <ul style="list-style-type: none"> ● Create a new list of one or more elements ● Iterate over the elements in a list using a loop ● Retrieve the length of a list ● Reference an element in a list by its index |
| 4.2: List Operations | |
| Comprehension Objectives | <ul style="list-style-type: none"> ● Give examples of problems that can be solved by querying a list ● Give examples of how a list can change during the execution of a program ● Determine the most appropriate method to add or remove elements from a list ● Describe how a new list can be created by analyzing the elements of an existing list ● Describe problems caused by adding or removing an item from a list while iterating over that list |
| Application Objectives | <ul style="list-style-type: none"> ● Query a list about the presence, number, or location of an element ● Add or remove an element from a list ● Map a list to another list using a loop ● Filter a list into another list using a loop |
| Research Project: Private Data | |
| Comprehension Objectives | <ul style="list-style-type: none"> ● Describe tradeoffs between allowing information to be public and keeping it private and secure ● Describe ways in which data can be collected that might not be obvious to the data's subject |

| | |
|--------------------------------------|--|
| | <ul style="list-style-type: none"> • Give examples of malicious or controversial usage of private data |
| Application Objectives | <ul style="list-style-type: none"> • Debate laws and regulations around data privacy • Gather information from a variety of sources • Evaluate the accuracy and bias of sources • Provide citations for sources used |
| 4.3: Advanced List Operations | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Give examples of when to use a sorted list over an unsorted list • Describe the benefits of using randomness with collections • Explain the difference between equality and identity in lists • Compare and contrast the use of <code>+</code> and <code>*</code> operators on lists with their use on other data types |
| Application Objectives | <ul style="list-style-type: none"> • Change the order of elements in a list by sorting and reversing • Find the minimum or maximum value in a list • Apply functions from the random library to lists • Check whether two variables refer to the same list or two lists with equal values • Concatenate and repeat lists |
| 4.4: Strings as Collections | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Compare and contrast strings, as a collection of characters, to lists • Identify which collection operations can and cannot be used on strings • Identify which elements of an existing collection will be selected by a slice operation • Identify the default values for collection slicing when no values are given • Recognize that slicing a collection with a negative step value results in a new reversed collection |
| Application Objectives | <ul style="list-style-type: none"> • Use previously explored collection operations on |

| | |
|---------------------------------------|--|
| | <p>strings and substrings</p> <ul style="list-style-type: none"> • Create a new collection by slicing an existing collection |
| 4.5: String Operations | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Identify and differentiate between spaces, tabs and newlines • Describe how an escape sequence makes it easier to display certain characters • Identify when a list of substrings is preferred over a string of the same characters and vice versa • Choose which string operation is most appropriate for a given task • Explain why an intermediate step is needed to swap two existing values |
| Application Objectives | <ul style="list-style-type: none"> • Create a string with one or more escape sequences • Convert a string to a list and vice versa • Create a new string by replacing all occurrences of a substring with a new substring • Convert the letters of a string to the same case • Check whether a string contains only the same type of characters |
| Industry Practice: Pair Coding | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Define pair coding as a process where two people write the same lines of code together • Describe the how professionals work together when pair coding • Explain the benefits and challenges of pair coding |
| Application Objectives | <ul style="list-style-type: none"> • Build a program with another student by pair coding as both a driver and a navigator |

| | | |
|--------------------------------|--------------------------|--|
| Unit 5: Data Structures | 5.1: Tuples | |
| | Comprehension Objectives | <ul style="list-style-type: none"> • Define a tuple as an unchangeable collection of ordered information • Explain why it can be useful to use a tuple instead |

| | |
|--|---|
| | <p>of a list</p> <ul style="list-style-type: none"> ● Identify which collection operations can and cannot be used on tuples ● Identify which data types a collection can be casted to or from ● Predict which values will be assigned to each variable when unpacking a tuple |
| Application Objectives | <ul style="list-style-type: none"> ● Create a new tuple of one or more elements ● Use previously explored collection operations on tuples ● Cast a collection from one data type to another ● Assign the elements of a collection to one or more variables in a single unpacking statement |
| 5.2: Dictionaries | |
| Comprehension Objectives | <ul style="list-style-type: none"> ● Define a dictionary as a changeable collection of unordered key-value pairs ● Describe the properties of key-value pairs in dictionaries ● Given a set of data, choose which collection type would be most appropriate for storing it ● Identify which dictionary operation is most appropriate to solve a given problem |
| Application Objectives | <ul style="list-style-type: none"> ● Create a new dictionary of zero or more key-value pairs ● Reference a value in a dictionary by a key ● Use previously explored collection operations on dictionaries ● Update a dictionary with the contents of another dictionary ● Iterate over a dictionary using a for-each loop ● Retrieve a list of the keys, values, or items in a dictionary |
| Research Project: Computer Networks | |
| Comprehension Objectives | <ul style="list-style-type: none"> ● Define the properties of a computer network ● Give real-world examples of different types of computer networks |

| | |
|--------------------------------|---|
| | <ul style="list-style-type: none"> ● Give examples of protocols used in computer networks |
| Application Objectives | <ul style="list-style-type: none"> ● Diagram the topology of a computer network ● Model how data is transmitted via a protocol ● Evaluate the reliability and scalability of a computer network |
| 5.3: Nested Collections | |
| Comprehension Objectives | <ul style="list-style-type: none"> ● Differentiate between the way lists and dictionaries organize nested collections of data ● Identify that collections can be nested to any depth ● Determine which nested collections would best model a given set of data ● Describe how sequential bracket notation is used to reference data in a nested collection ● Identify the most effective strategy to reference data within a nested collection |
| Application Objectives | <ul style="list-style-type: none"> ● Create a new heterogeneous nested collection ● Create a new homogeneous nested collection ● Use sequential bracket notation to reference data in a nested collection ● Use iteration to build a nested collection and to reference its data |
| 5.4: Web APIs | |
| Comprehension Objectives | <ul style="list-style-type: none"> ● Explain the essential requirements and workflow for HTTP client-server communication ● Explain the tradeoffs between hard-coding information into a program versus requesting it from a server ● Compare and contrast requesting information from a server to calling a function in a program ● Explain additional requirements and complications that may occur with HTTP client-server communication |
| Application Objectives | <ul style="list-style-type: none"> ● Create an HTTP client that communicates with an HTTP server |

| | |
|---------------------------------------|---|
| | <ul style="list-style-type: none"> • Send a complex request to an HTTP server that includes parameters and API keys • Check the status code of a response from an HTTP server • Locate, read, and understand the documentation for a web API |
| Industry Practice: Code Review | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Describe the process of code reviews • List the benefits of code reviews |
| Application Objectives | <ul style="list-style-type: none"> • Review code with other students • Revise code based on code reviews |

| | | |
|------------------------------|------------------------------------|---|
| Unit 6: Functions | 6.1: User-Defined Functions | |
| | Comprehension Objectives | <ul style="list-style-type: none"> • Explain the benefits of organizing code inside a function • Trace the code execution from a function call to the function's code and back • Identify what arguments a function requires based on its signature • Identify the rules for how a function finishes and returns to the code that called it |
| | Application Objectives | <ul style="list-style-type: none"> • Create and call user-defined functions • Use a user-defined function as or within an expression • Identify and refactor appropriate sections of code by abstracting them into functions • Validate input arguments at the start of a function |
| | 6.2: Scope and References | |
| | Comprehension Objectives | <ul style="list-style-type: none"> • Define scope as the area of a program where a variable or function name is valid • Describe how Python searches through a program's namespaces looking for a valid name • Give examples of common side effects from a function call • Differentiate between modifying a value and |

| | |
|--|---|
| | re-assigning a reference |
| Application Objectives | <ul style="list-style-type: none"> • Correct scope errors when accessing a variable or parameter • Produce function side effects related to modifying values |
| Industry Skill: Docstrings | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Define a docstring as a string literal that provides documentation for the code that follows • Describe the benefits of documenting functions • Differentiate between commenting and documentation |
| Application Objectives | <ul style="list-style-type: none"> • Write a properly-formatted docstring for a function • Call an unfamiliar function by referencing its docstring documentation |
| Research Project: Preventing Cyberattacks | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Give examples of common cyberattack methods • Identify common issues that may make data vulnerable to attack • Give examples of existing cybersecurity measures • Discuss tradeoffs for various security measures, such as ethics, efficiency, feasibility, etc. |
| Application Objectives | <ul style="list-style-type: none"> • Give examples of the damage that could be caused to a system or its users based on a description of a vulnerability or attack • Suggest appropriate responses to given types of cyberattacks |
| 6.3: Advanced Parameters | |
| Comprehension Objectives | <ul style="list-style-type: none"> • Predict the initial value of all function parameters when the function is called • Differentiate between function parameters that do and do not have default values • Identify function parameters that can be assigned a collection of zero or more arguments |

| | | |
|--|------------------------|--|
| | Application Objectives | <ul style="list-style-type: none"> ● Define functions with parameters that have default values ● Choose whether to call a function with an argument for a parameter that has a default value ● Call functions with arguments in non-positional order by referencing parameter names ● Define functions with a parameter that collects variable-length arguments ● Call functions that assigns zero or more arguments to a parameter as a collection |
|--|------------------------|--|

This course also includes an optional test prep component for students preparing to take the [Certified Entry-Level Python Programmer \(PCEP\)](#) certification exam.

| PCEP | | Optional Test Prep |
|-------------|--------------------------|--|
| | Comprehension Objectives | <ul style="list-style-type: none"> ● Describe how the global keyword changes the scope of a variable ● Describe how the yield keyword interacts with functions ● Describe the function of generators in Python ● Give examples of problems that might be created by the lack of float precision in Python ● Compare bitwise operators to logical operators ● List uses of bitwise operators ● Give examples of non-decimal number systems |
| | Application Objectives | <ul style="list-style-type: none"> ● Use the global keyword in appropriate locations ● Trace the execution of a recursive function call ● Fix infinite recursion problems ● Use a generator to create a list ● Use alternative print methods, including the end= and sep= optional arguments ● Predict the result of using bitwise operators ● Give the decimal value of numbers written in binary |

Standards and Certifications

Upon completion of CS201: Coding with Python 1 and CS202: Coding with Python 2, students will be prepared to take the [Certified Entry-Level Python Programmer \(PCEP\)](#) certification exam.

Additionally, students who complete CS201, CS202, and CS203: Coding with Python 3 will be prepared to take the [Microsoft Technology Associate \(MTA\): Introduction to Programming Using Python](#) certification exam.

| All Units | |
|----------------|---|
| CSTA Standards | <ul style="list-style-type: none">● 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.● 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.● 2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.● 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.● 2-AP-17: Systematically test and refine programs using a range of test cases.● 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.● 2-AP-19: Document programs in order to make them easier to follow, test, and debug.● 2-DA-09: Refine computational models based on the data they have generated.● 3A-AP-13: Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.● 3A-AP-14: Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.● 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and |

| | | |
|--|--|--|
| | | <p>drawbacks of choices made.</p> <ul style="list-style-type: none"> ● 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. ● 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. ● 3A-DA-10: Evaluate the tradeoffs in how data elements are organized and where data is stored. ● 3A-IC-26: Demonstrate ways a given algorithm applies to problems across disciplines. ● 3A-IC-27: Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields. ● 3B-AP-10: Use and adapt classic algorithms to solve computational problems. ● 3B-AP-12: Compare and contrast fundamental data structures and their uses. ● 3B-AP-15: Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution. ● 3B-DA-07: Evaluate the ability of models and simulations to test and support the refinement of hypotheses. |
|--|--|--|

| Unit 4: Lists | |
|----------------------|--|
| CSTA Standards | <ul style="list-style-type: none"> ● 2-IC-23: Describe tradeoffs between allowing information to be public and keeping information private and secure. ● 2-NI-05: Explain how physical and digital security measures protect electronic information. ● 3A-IC-24: Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices. ● 3A-IC-29: Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to |

| | |
|--------------------|--|
| | <p>users.</p> <ul style="list-style-type: none"> ● 3A-IC-30: Evaluate the social and economic implications of privacy in the context of safety, law, or ethics. ● 3A-NI-05: Give examples to illustrate how sensitive data can be affected by malware and other attacks. ● 3B-AP-18: Explain security issues that might lead to compromised computer programs. ● 3B-IC-25: Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society. ● 3B-IC-28: Debate laws and regulations that impact the development and use of software. |
| PCEP Certification | <ul style="list-style-type: none"> ● simple strings: indexing, immutability ● building loops: for, in ● iterating through sequences ● expanding loops: for-else ● simple lists: indexing, the len() function ● lists in detail: slicing, basic methods (append(), insert(), index()) and functions (sorted(), etc.), iterating lists with the for loop, initializing, in and not in operators, list comprehension ● strings in detail: escaping using the \ character, quotes and apostrophes inside strings, multiline strings, basic string functions. |
| MTA Certification | <ul style="list-style-type: none"> ● Determine the sequence of execution based on operator precedence: containment ● Perform data and data type operations: construct data structures; perform indexing and slicing operations |

Unit 5: Data Structures

CSTA Standards

- **2-AP-15:** Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
- **2-CS-02:** Design projects that combine hardware and software components to collect and

| | |
|--------------------|--|
| | <p>exchange data.</p> <ul style="list-style-type: none"> ● 2-CS-03: Systematically identify and fix problems with computing devices and their components. ● 2-DA-08: Collect data using computational tools and transform the data to make it more useful and reliable. ● 2-NI-04: Model the role of protocols in transmitting data across networks and the Internet. ● 3A-IC-27: Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields. ● 3A-NI-04: Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. ● 3B-AP-23: Evaluate key qualities of a program through a process such as a code review. ● 3B-NI-03: Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology). |
| PCEP Certification | <ul style="list-style-type: none"> ● lists in lists: matrices and cubes ● lists in lists: matrices and cubes ● tuples: indexing, slicing, building, immutability ● tuples vs. lists: similarities and differences, lists inside tuples and tuples inside lists ● dictionaries: building, indexing, adding and removing keys, iterating through dictionaries as well as their keys and values, checking key existence, keys(), items() and values() methods |
| MTA Certification | <ul style="list-style-type: none"> ● Perform data and data type operations: construct data structures |

Unit 6: Functions

CSTA Standards

- **2-AP-14:** Create procedures with parameters to organize code and make it easier to reuse.

- **2-IC-20:** Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
- **2-IC-23:** Describe tradeoffs between allowing information to be public and keeping information private and secure.
- **2-NI-05:** Explain how physical and digital security measures protect electronic information.
- **3A-AP-18:** Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
- **3A-IC-29:** Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.
- **3A-IC-30:** Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.
- **3A-NI-05:** Give examples to illustrate how sensitive data can be affected by malware and other attacks.
- **3A-NI-06:** Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.
- **3A-NI-07:** Compare various security measures, considering tradeoffs between the usability and security of a computing system.
- **3A-NI-08:** Explain tradeoffs when selecting and implementing cybersecurity recommendations.
- **3B-AP-14:** Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
- **3B-AP-18:** Explain security issues that might lead to compromised computer programs.
- **3B-IC-28:** Debate laws and regulations that impact the development and use of software.

| | |
|--------------------|---|
| | <ul style="list-style-type: none"> ● 3B-NI-04: Compare ways software developers protect devices and information from unauthorized access. |
| PCEP Certification | <ul style="list-style-type: none"> ● defining and invoking your own functions ● return keyword, returning results ● the pass instruction ● the None keyword ● parameters vs. arguments ● positional keyword and mixed argument passing, ● default parameter values ● name scopes, name hiding (shadowing), the global keyword |
| MTA Certification | <ul style="list-style-type: none"> ● Construct and analyze code segments that include function definitions: call signatures; default values; return; def; pass |

| PCEP Prep | |
|--------------------|---|
| CSTA Standards | <ul style="list-style-type: none"> ● 3B-AP-13: Illustrate the flow of execution of a recursive algorithm. |
| PCEP Certification | <ul style="list-style-type: none"> ● numeral systems (binary, octal, decimal, hexadecimal) ● bitwise operators: ~ & ^ << >> ● accuracy of floating-point numbers ● formatting print() output with end= and sep= arguments ● recursion ● defining and invoking your own generators ● yield keyword ● converting generator objects into lists using the list() function |
| MTA Certification | (none) |