straighterline

Syllabus

CS104 | **Introduction to Programming in Python**

Course Text

There is no text for this course. All materials are included in the course fee.

Course Description

This course provides a comprehensive foundation in Python programming for beginners. Students learn core programming concepts including variables, data types, control structures, and functions before advancing to object-oriented programming with classes and inheritance. The course covers essential data structures like lists and dictionaries, file handling, exception management, and modular programming. Advanced topics include recursion, algorithm design for searching and sorting, and data visualization through plotting. Students gain practical experience building programs that solve real-world problems while developing computational thinking skills. By course completion, students will be proficient in Python fundamentals and prepared for intermediate programming courses.

Learning Outcomes

After completing this course, students will be able to:

1. Apply fundamental Python syntax and programming constructs to solve basic computational problems.
2. Analyze data types and their appropriate usage in different programming contexts.
3. Evaluate different conditional logic approaches to implement decision-making in programs.
4. Create iterative solutions using various loop structures to process data and solve repetitive tasks.
5. Design modular programs by implementing and utilizing functions with appropriate parameters and return values.
6. Manipulate string and collection data structures (lists and dictionaries) to store, access, and process information.
7. Construct object-oriented solutions using classes, inheritance, and encapsulation principles.
8. Implement robust error handling and program modularity through exceptions and module organization.
9. Synthesize file I/O operations with data processing techniques to create programs that interact with external data sources.
10. Compare and implement fundamental algorithms including recursive solutions, searching, and sorting techniques, while visualizing results through plotting.

Course Prerequisites

There are no prerequisites for this course.

Academic Integrity Statement

Academic integrity is the pursuit of scholarly activity in an honest, truthful and responsible manner. Violations of academic integrity include, but are not limited to, plagiarism, cheating, fabrication and academic misconduct. Failure to comply with the Academic Integrity Policy can result in a failure and/or zero on the attempted assignment/examination, a removal from the course, disqualification to enroll in future courses, and/or revocation of an academic transcript.

Course Completion Policy

In order for a course to be considered complete, **all required coursework must be attempted, submitted, and graded.** Required coursework consists of graded assignments. Any Academic Integrity Policy violations may prevent a course from being considered complete.

Assessment Types

StraighterLine courses may include any combination of the assessment types described below. Review the descriptions to learn about each type, then review the Course Evaluation Criteria to understand how your learning will be measured in this course.

Benchmarks
Benchmarks test your mastery of course concepts. You have 3 attempts, and your highest score counts.
Note: Cumulative Benchmarks (final exams) only allow 1 attempt.

Capstones
Capstones are project-based assessments that help you apply concepts to real-world scenarios. You have 2 attempts, and your highest score counts.

Checkpoints
Checkpoints are quick knowledge checks on important course concepts. All are open-book, and most have 1-3 attempts.

AI Use-Case Policies

StraighterLine Capstone assessments operate under one of three AI Use-Case Policies. These designations are selected intentionally to support learners in developing digital literacy, ethical reasoning, and authentic communication skills. Each model requires students to engage meaningfully with the course outcomes while adhering to academic standards.

Independent Work Requirement: Capstones with this designation must be completed independently without using AI tools. The goal is for learners to showcase their own understanding and skills without AI assistance. Students are expected to generate and submit original work developed solely through their own reasoning and effort.

AI-Assisted Planning Option: Capstones with this designation may allow AI tools to support brainstorming and assessment planning. If allowed, students will be asked to document any AI assistance by noting how it informed their work. Documentation must be included within the assignment or in a designated reflection field. Examples include describing how an AI tool helped organize an outline, generate ideas, or surface sources for further exploration.

AI-Integration Requirement: Capstones with this designation require AI tools as part of the learning process. Students will be asked to reflect upon their AI interactions and AI contributions to the assessment. Reflections must include which tools were used, how they were used, and what insights students gained from the process. This promotes transparency, ethical use, and metacognitive skill-building.

Course Evaluation Criteria

Your score provides a percentage score and letter grade for each course. A passing percentage is 70% or higher.

There are a total of 1000 points in the course:

Assessment	Points	Learning Outcomes
Checkpoint 1: Introduction to Python	20	1
Benchmark 1: Introduction to Python	42	1
Checkpoint 2: Variables and Expressions	20	1
Benchmark 2: Variables and Expressions	42	1
Checkpoint 3: Types	20	2
Benchmark 3: Types	42	2
Checkpoint 4: Branching	20	3
Benchmark 4: Branching	42	3
Checkpoint 5: Loops	20	4
Benchmark 5: Loops	42	4
Checkpoint 6: Functions	20	5
Benchmark 6: Functions	42	5
Checkpoint 7: Strings	20	6
Benchmark 7: Strings	42	6
Checkpoint 8: Lists and Dictionaries	20	6
Benchmark 8: Lists and Dictionaries	42	6
Checkpoint 9: Classes	20	7
Benchmark 9: Classes	43	7
Checkpoint 10: Exceptions	20	8
Benchmark 10: Exceptions	43	8
Checkpoint 11: Modules	20	8
Benchmark 11: Modules	43	8
Checkpoint 12: Files	20	9
Benchmark 12: Files	43	9
Checkpoint 13: Inheritance	20	7

Assessment	Points	Learning Outcomes
Benchmark 13: Inheritance	43	7
Checkpoint 14: Recursion	20	10
Benchmark 14: Recursion	43	10
Checkpoint 15: Plotting	20	10
Benchmark 15: Plotting	43	10
Checkpoint 16: Searching and Sorting Algorithms	20	10
Benchmark 16: Searching and Sorting Algorithms	43	10
Total	1000	

Course Roadmap

This roadmap provides an overview of the checkpoints and lessons covered in this course.

Checkpoint 1: Introduction to Python

- Programming (general)
- Programming using Python
- Basic input and output
- Errors
- Integrated development environment
- Computers and programs (general)
- Computer tour
- Language history
- Why whitespace and precision matter
- Python example: Salary calculation
- Additional practice: Output art

Checkpoint 2: Variables and Expressions

- Variables and assignments
- Identifiers
- Objects
- Numeric types: Floating-point
- Arithmetic expressions
- Python expressions
- Division and modulo
- Module basics
- Math module
- Random numbers
- Representing text
- Additional practice: Number games

Checkpoint 3: Types

- String basics
- String formatting
- List basics
- Tuple basics
- Set basics
- Dictionary basics
- Common data types summary
- Additional practice: Grade calculation
- Type conversions
- Binary numbers
- Additional practice: Health data

Checkpoint 4: Branching

- If-else branches (general)
- Detecting equal values with branches
- Detecting ranges with branches (general)
- Detecting ranges with branches
- Detecting ranges using logical operators
- Detecting ranges with gaps
- Detecting multiple features with branches
- Comparing data types and common errors
- Membership and identity operators
- Order of evaluation
- Code blocks and indentation
- Conditional expressions
- Additional practice: Tweet decoder

Checkpoint 5: Loops

- Loops (general)
- While loops
- More while loop examples
- Counting
- For loops
- Counting using the range() function
- While vs. for loops
- Nested loops
- Developing programs incrementally
- Break and continue
- Loop else
- Getting both index and value when looping: enumerate()
- Additional practice: Dice statistics

Checkpoint 6: Functions

- User-defined function basics
- Print functions
- Dynamic typing
- Reasons for defining functions
- Writing mathematical functions
- Function stubs
- Functions with branches/loops
- Functions are objects

- Functions: Common errors
- Scope of variables and functions
- Namespaces and scope resolution
- Function arguments
- Keyword arguments and default parameter values
- Arbitrary argument lists
- Multiple function outputs
- Help! Using docstrings to document functions
- Engineering examples

Checkpoint 7: Strings

- String slicing
- Advanced string formatting
- String methods
- Splitting and joining strings

Checkpoint 8: Lists and Dictionaries

- Lists
- List methods
- Iterating over a list
- List games
- List nesting
- List slicing
- Loops modifying lists
- List comprehensions
- Sorting lists
- Command-line arguments
- Additional practice: Engineering examples
- Dictionaries
- Dictionary methods
- Iterating over a dictionary
- Dictionary nesting

Checkpoint 9: Classes

- Classes: Introduction
- Classes: Grouping data
- Instance methods
- Class and instance object types
- Class example: Seat reservation system
- Class constructors
- Class interfaces
- Class customization
- More operator overloading: Classes as numeric types
- Memory allocation and garbage collection

Checkpoint 10: Exceptions

- Handling exceptions using try and except
- Multiple exception handlers
- Raising exceptions
- Exceptions with functions

- Using finally to clean up
- Custom exception types

Checkpoint 11: Modules

- Modules
- Finding modules
- Importing specific names from a module
- Executing modules as scripts
- Reloading modules
- Packages
- Standard library

Checkpoint 12: Files

- Reading files
- Writing files
- Interacting with file systems
- Binary data
- Command-line arguments and files
- The “with” statement
- Comma separated values files

Checkpoint 13: Inheritance

- Derived classes
- Accessing base class attributes
- Overriding class methods
- Is-a versus has-a relationships
- Mixin classes and multiple inheritance
- Testing your code: The unittest module

Checkpoint 14: Recursion

- Recursive functions
- Recursive algorithm: Search
- Adding output statements for debugging
- Creating a recursive function
- Recursive math functions
- Recursive exploration of all possibilities

Checkpoint 15: Plotting

- Introduction to data science
- Data science life cycle
- Introduction to Python for data science
- Introduction to Jupyter Notebooks
- NumPy
- pandas
- Matplotlib

Checkpoint 16: Searching and Sorting Algorithms

- Searching and algorithms
- Binary search
- O notation
- Algorithm analysis
- Sorting: Introduction
- Selection sort
- Insertion sort
- Quicksort
- Merge sort

Related Courses

IT101
IT Fundamentals
View Course →

MAT102
Quantitative Reasoning
View Course →

MAT202
Introduction to Statistics
View Course →