uCertify

# Course Outline

## Ultimate Python Programming

04 Aug 2025

1. Exercises, Quizzes, Flashcards & Glossary

   Number of Questions
2. Expert Instructor-Led Training
3. ADA Compliant & JAWS Compatible Platform
4. State of the Art Educator Tools
5. Award Winning Learning Platform (LMS)
6. Chapter & Lessons

   Syllabus

   Chapter 1: Preface

   Chapter 2: Introduction to Python

   Chapter 3: Getting Started

   Chapter 4: Strings

   Chapter 5: Lists and Tuples

   Chapter 6: Dictionaries and Sets

   Chapter 7: Conditional Execution

   Chapter 8: Loops

   Chapter 9: Looping Techniques

   Chapter 10: Comprehensions

   Chapter 11: Functions

   Chapter 12: Modules and Packages

   Chapter 13: Namespaces and Scope

   Chapter 14: Files

   Chapter 15: Object Oriented Programming

   Chapter 16: Magic Methods

   Chapter 17: Inheritance and Polymorphism

   Chapter 18: Iterators and Generators

   Chapter 19: Decorators

   Chapter 20: Lambda Expressions and Functional Programming

Chapter 21: Exception Handling

Chapter 22: Context Managers

Videos and How To

## 1. Quiz

Quizzes test your knowledge on the topics of the exam when you go through the course material. There is no limit to the number of times you can attempt it.


## 2. Expert Instructor-Led Training

uCertify uses the content from the finest publishers and only the IT industry's finest instructors. They have a minimum of 15 years real-world experience and are subject matter experts in their fields. Unlike a live class, you can study at your own pace. This creates a personal learning experience and gives you all the benefit of hands-on training with the flexibility of doing it around your schedule 24/7.

## 3. ADA Compliant & JAWS Compatible Platform

uCertify course and labs are ADA (Americans with Disability Act) compliant. It is now more accessible to students with features such as:

- Change the font, size, and color of the content of the course
- Text-to-speech, reads the text into spoken words
- Interactive videos, how-tos videos come with transcripts and voice-over

- Interactive transcripts, each word is clickable. Students can clip a specific part of the video by clicking on a word or a portion of the text.

JAWS (Job Access with Speech) is a computer screen reader program for Microsoft Windows that reads the screen either with a text-to-speech output or by a Refreshable Braille display. Student can easily navigate uCertify course using JAWS shortcut keys.

## 4. State of the Art Educator Tools

uCertify knows the importance of instructors and provide tools to help them do their job effectively. Instructors are able to clone and customize course. Do ability grouping. Create sections. Design grade scale and grade formula. Create and schedule assessments. Educators can also move a student from self-paced to mentor-guided to instructor-led mode in three clicks.

## 5. Award Winning Learning Platform (LMS)

uCertify has developed an award winning, highly interactive yet simple to use platform. The SIIA CODiE Awards is the only peer-reviewed program to showcase business and education technology's finest products and services. Since 1986, thousands of products, services and solutions have been recognized for achieving excellence. uCertify has won CODiE awards consecutively for last 7 years:

- **2014**
    1. Best Postsecondary Learning Solution

- **2015**
    1. Best Education Solution
    2. Best Virtual Learning Solution
    3. Best Student Assessment Solution
    4. Best Postsecondary Learning Solution
    5. Best Career and Workforce Readiness Solution
    6. Best Instructional Solution in Other Curriculum Areas
    7. Best Corporate Learning/Workforce Development Solution

- **2016**
    1. Best Virtual Learning Solution
    2. Best Education Cloud-based Solution
    3. Best College and Career Readiness Solution
    4. Best Corporate / Workforce Learning Solution
    5. Best Postsecondary Learning Content Solution
    6. Best Postsecondary LMS or Learning Platform
    7. Best Learning Relationship Management Solution

- **2017**
    1. Best Overall Education Solution
    2. Best Student Assessment Solution
    3. Best Corporate/Workforce Learning Solution
    4. Best Higher Education LMS or Learning Platform

- **2018**
    1. Best Higher Education LMS or Learning Platform
    2. Best Instructional Solution in Other Curriculum Areas
    3. Best Learning Relationship Management Solution

- **2019**
    1. Best Virtual Learning Solution
    2. Best Content Authoring Development or Curation Solution
    3. Best Higher Education Learning Management Solution (LMS)

- **2020**
    1. Best College and Career Readiness Solution
    2. Best Cross-Curricular Solution
    3. Best Virtual Learning Solution

## 6.  Chapter & Lessons

uCertify brings these textbooks to life. It is full of interactive activities that keeps the learner engaged. uCertify brings all available learning resources for a topic in one place so that the learner can efficiently learn without going to multiple places. Challenge questions are also embedded in the chapters so learners can attempt those while they are learning about that particular topic. This helps them grasp the concepts better because they can go over it again right away which improves learning.

Learners can do Flashcards, Exercises, Quizzes and Labs related to each chapter. At the end of every lesson, uCertify courses guide the learners on the path they should follow.

# Syllabus


## Chapter 1: Preface

## Chapter 2: Introduction to Python

- What makes Python so popular
- Python implementation
- Installing Python
- Python Interactive Mode
- Executing a Python Script
- IDLE
- Getting Help

## Chapter 3: Getting Started

- Identifiers
- Python Types

- Objects
- Variables and assignment statement
- Multiple and Pairwise Assignments
- Deleting a name
- Naming convention for constants
- Operators
- Augmented assignment statements
- Expressions
- Order of operations: Operator Precedence and Associativity
- Type Conversion
- Statements
- Printing Output
- Getting user input
- Complete programs
- Comments
- Indentation in Python
- Container types
- Mutable and Immutable Types

- Functions and methods
- Importing
- Revisiting interactive mode
- Errors
- PEP8
- Exercise

## Chapter 4: Strings

- Indexing
- Strings are immutable
- String Slicing
- String Concatenation and Repetition
- Checking membership
- Adding whitespace to strings
- Creating multiline strings
- String methods
- Case-changing methods
- Character classification methods
- Aligning text within strings

- Removing unwanted leading and trailing characters
- Searching and replacing substrings
- Chaining method calls
- String comparison
- String conversions
- Escape Sequences
- Raw string literals
- String formatting
- String formatting using the format() method of string class
- Representation of text - character encodings
- Exercise

## Chapter 5: Lists and Tuples

- Accessing individual elements of a list by indexing
- Getting parts of a list by slicing
- Changing an item in a list by index assignment
- Changing a Portion of the list by slice assignment
- Adding an item at the end of the list by using append()

- Adding an item anywhere in the list by using insert()
- Adding multiple items at the end by using extend() or +=
- Removing a single element or a slice by using the del statement
- Removing an element by index and getting it by using pop()
- Removing an element by value using remove()
- Removing all the elements by using clear()
- Sorting a List
- Reversing a List
- Finding an item in the list
- Comparing Lists
- Built-in functions used on lists
- Concatenation and Replication
- Using a list with functions from the random module
- Creating a list
- Using range to create a list of integers
- Using the repetition operator to create a list of repeated values
- Creating a list by splitting a string
- Converting a list of strings to a single string using join()

- List of Lists (Nested lists)
- Copying a list
- Shallow copy and deep copy
- Repetition operator with nested lists
- Tuples
- Tuple packing and unpacking
- Exercise

## Chapter 6: Dictionaries and Sets

- Dictionaries
- Adding new key-value pairs
- Modifying Values
- Getting a value from a key by using the get() method
- Getting a value from a key by using the setdefault() method
- Getting all keys, all values, and all key-value pairs
- Checking for the existence of a key or a value in a dictionary
- Comparing dictionaries
- Deleting key-value pairs from a dictionary
- Creating a Dictionary at run time

- Creating a dictionary from existing data by using dict()
- Creating a dictionary by using the fromkeys() method
- Combining dictionaries
- Nesting of dictionaries
- Aliasing and Shallow vs. Deep Copy
- Introduction to sets
- Creating a set
- Adding and Removing elements
- Comparing sets
- Union, intersection, and difference of sets
- Frozenset
- Exercise

## Chapter 7: Conditional Execution

- if statement
- else clause in if statement
- Nested if statements
- Multiway selection by using elif clause

- Truthiness
- Short circuit behavior of operators and and or
- Values returned by and and or operators
- if else operator
- Exercise

## Chapter 8: Loops

- while loop
- for loop
- Nesting of Loops
- Premature termination of loops using the break statement
- continue statement
- else block in Loops
- pass statement
- for loop vs. while loop
- Exercise

## Chapter 9: Looping Techniques

- Iterating in sorted and reversed order

- Iterating over unique values
- Index-Based for loops
- Making in-place changes in a list while iterating
- Skipping some items while iterating
- Using range and len combination to shuffle a sequence
- enumerate function
- Iterating over multiple sequences using zip
- Modifying a collection while iterating in a for loop
- Infinite loop with break
- Avoiding complex logical conditions using break
- Exercise

## Chapter 10: Comprehensions

- List Comprehensions
- if clause in list comprehension
- Ternary operator in list comprehension
- Modifying a list while iterating
- Getting keys from values in a dictionary using list comprehension
- Using list comprehensions to avoid aliasing while creating lists of lists

- Multiple for clauses and Nested list Comprehensions
- Extracting a column in a matrix
- Dictionary Comprehensions
- Inverting the dictionary
- Set Comprehensions
- When not to use comprehensions
- Exercise

## Chapter 11: Functions

- Function Definition
- Function call
- Flow of control
- Parameters and Arguments
- No type checking of arguments
- Local Variables
- return statement
- Returning Multiple Values
- Semantics of argument passing

- Default Arguments
- Default arguments that could change over time
- Positional and Keyword Arguments
- Unpacking Arguments
- Variable number of positional arguments
- Variable number of keyword arguments
- Keyword-only arguments
- Positional-Only Arguments
- Multiple Unpackings in a Python Function Call
- Arguments and Parameters summary
- Function Objects
- Attributes of a function
- Doctrsings
- Function Annotations
- Recursive Functions
- Exercise

## Chapter 12: Modules and Packages

- Modules

- Types of modules
- Exploring modules
- Creating and naming a new module
- Importing a module
- Importing all names from a module
- Restricting names that can be imported
- Importing individual names from a module
- Using an alias while importing
- Documenting a module
- Module search Path
- Module object
- Byte compiled version of a module
- Reloading a module
- Scripts and modules
- Packages
- Importing a package and its contents
- Subpackages
- Relative imports

- Exercise

## Chapter 13: Namespaces and Scope

- Namespaces
- Inspecting namespaces
- Scope
- Name Resolution
- global statement
- nonlocal statement
- Exercise

## Chapter 14: Files

- Opening a File
- File opening modes
- Buffering
- Binary and Text Files
- Closing a file
- with statement
- Random Access

- Using seek in text mode
- Calling seek in append mode
- Reading and writing to the same file
- Reading a File using read()
- Line oriented reading
- Writing to a file
- Redirecting output of print to a file
- Example Programs
- File Related Modules
- Command Line Arguments
- Storing and Retrieving Python objects using pickle
- Exercise
- Project : Hangman Game

## Chapter 15: Object Oriented Programming

- Programming Paradigms
- Introduction to object-oriented programming
- Defining Classes and Creating Instance Objects
- Adding methods to the class

- Adding instance variables
- Calling a method inside another method
- Common pitfalls
- Initializer
- Data Hiding
- Class Variables
- Class and object namespaces
- Changing a class variable through an instance
- Class Methods
- Creating alternative initializers using class Methods
- Static Methods
- Creating Managed Attributes using properties
- Designing a class
- Exercise
- Project : Quiz creation
- Project : Snakes and Ladders Game
- Project : Log in system

## Chapter 16: Magic Methods

- Overloading Binary Arithmetic operators
- Reverse methods
- In-place methods
- Magic Methods for comparison
- Comparing objects of different classes
- String representation of an instance object
- Construction and destruction of objects
- Making instance objects callable
- Overloading type conversion functions
- List of magic methods
- Exercise
- Project : Date Class

## Chapter 17: Inheritance and Polymorphism

- Inheriting a class
- Adding new methods and data members to the derived class
- Overriding a base Method
- Invoking the base class methods

- Multilevel Inheritance
- object class
- Multiple Inheritance
- Method Resolution Order (MRO)
- super and MRO
- Polymorphism
- Abstract Base classes
- Composition
- Exercise

## Chapter 18: Iterators and Generators

- Iterables
- Iterators
- for loop Iteration Context – How for loop works
- Iteration Tools
- Iterator vs Iterable
- Creating your own Iterator
- Making your class Iterable
- Some More Iterators

- Lazy evaluation
- itertools Module
- Generators
- Generator function vs Normal function
- Generator expressions
- Exercise

## Chapter 19: Decorators

- Prerequisites for understanding decorators
- Introduction to decorators
- Writing your first decorator
- Applying your decorator to multiple functions
- Automatic decoration syntax
- Decorator Example: Timer
- Decorator Example: Logger
- Decorator Example: Counting function calls
- Applications of decorators
- Decorating functions that take arguments

- Returning values from decorated functions
- Decorator Example: Checking return values
- Decorator Example: Checking argument values
- Applying Multiple Decorators
- Preserving metadata of a function after decoration
- General template for writing a decorator
- Decorators with parameters
- General template for writing a decorator factory
- Decorator factory example
- Applying decorators to imported functions
- Decorating classes
- Class Decorators
- Class Decorators with parameters
- Exercise

## Chapter 20: Lambda Expressions and Functional Programming

- Lambda expression
- Comparing def statement and lambda expression
- Examples of lambda expressions

- Using Lambda expressions
- Using lambda expressions for returning function objects
- Lambda expressions as closures
- Creating jump tables using lambda functions
- Using lambda expressions in sorted built-in function
- Functional programming
- map
- map with multiple iterables
- filter
- Reducing an iterable
- Built-in reducing functions
- operator module
- Exercise

## Chapter 21: Exception Handling

- Types of Errors
- Strategies to handle exceptions in your code
- Error Handling by Python (Default exception handling)

- Built-in Exceptions: Python Exceptions Class Hierarchy
- Customized Exception Handling by using try…except
- Catching multiple exceptions using multiple except handlers and single except handler
- How to handle an exception
- Guaranteed execution of finally block
- else Block
- Why do we need an else block
- How to get exception details
- Nested try statements
- Raising Exception
- Re-raising Exception
- Chaining Exceptions
- Creating your own exceptions in Python (Custom exceptions)
- Assertions
- Exercise

## Chapter 22: Context Managers

- with statement
- Implementing our own context manager

- Exception raised inside with block
- Why we need with statement and context managers
- Runtime context
- Example: Sending output of a portion of code to a file
- Example : Finding time taken by a piece of code
- Using context managers in the standard library
- Nested with statements and multiple context Managers
- Implementing a context manager by using a decorator on a generator
- Exercise

You can’t stay away! Get

3187 Independence Drive
Livermore, CA 94551,
United States

+1-415-763-6300

support@ucertify.com

www.ucertify.com