Programming in Python

Week 1

[Introduction to the Course | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/3vpix/introduction-to-the-course) 3-16-2024 1640

This course provides a comprehensive introduction to the Python programming language. You'll learn foundational concepts, data structures, programming paradigms, testing techniques, and how to leverage libraries and tools for effective Python development.

[How is Python used in the real world? | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/cflqw/how-is-python-used-in-the-real-world) 3-16-2024 1645  
Software engineer talking about her job.

[Introduction to Programming | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/YZzIL/introduction-to-programming)[Why Python? | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/xbwgK/why-python) 3-16-2024 1652

* The video provides an overview of the history of programming, highlighting Charles Babbage's development of the difference engine and analytical engine as precursors to modern computing, and introduces the basics of programming, explaining that it involves writing instructions in a language that a computer can understand to perform tasks. It covers the transition from mechanical computing to binary code representation in modern computers, the role of compilers or interpreters in converting human-readable code to machine code, and emphasizes programming as both a skill and a creative process.
* Facts with Dates and Terms:
* Charles Babbage began working on improving calculating devices in 1822 at Cambridge University.
* The difference engine and analytical engine were early forms of mechanical computing devices developed by Babbage, with the analytical engine considered a basis for modern computing.
* Ada Lovelace, a friend of Babbage, is credited with publishing the first algorithm intended for processing by a machine.
* Computers operate on binary code, a system of representation using two digits, 0 and 1, corresponding to off and on electrical states, respectively.
* Transistors, tiny electrical conductors, are used within the CPU (Central Processing Unit) to represent binary code.
* Programming languages are compiled or interpreted to convert into human interaction with computers by abstracting binary complexity.

[Why Python? | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/xbwgK/why-python) 3-16-2024 1658

* Summary: Python is a versatile, easy-to-learn programming language with wide applications in web development, data analysis, artificial intelligence, and more. Its popularity stems from its English-like syntax, developer productivity, and high demand for Python skills.
* Terms:
* High-level programming language: A language designed for human readability, abstracting away low-level machine details.
* Syntax: The rules governing how a programming language is written.
* Frameworks: Collections of pre-written code to streamline development in specific areas.
* Libraries: Collections of reusable code modules.

[Visual Studio Code | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/yIlUR/visual-studio-code)

[Installing Python paths (Optional for Windows Users) | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/3FA7B/installing-python-paths-optional-for-windows-users)

[Installing Python paths (Optional for Mac users) | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/ItMxk/installing-python-paths-optional-for-mac-users)

[Required dependencies | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/7b4pc/required-dependencies) 3-16-2024 1718

* Xcode is a comprehensive development toolkit required on macOS for installing
* Homebrew, a popular package manager that fills the gap of macOS not having a native package manager like Linux.
* Windows installations of Python are straightforward as they don't require additional dependencies, unlike macOS, which necessitates installing Xcode and Homebrew to properly set up the development environment.

[Environment check for Windows | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/7qV0b/environment-check-for-windows) 3-15-2024 0700

* Summary: This video demonstrates how to set up Visual Studio Code (VS Code) for Python development on Windows. It covers checking the Python installation, creating a Python file, selecting the appropriate Python interpreter, and running the file to verify the setup.
* Terms/Acronyms:
* IDE: Integrated Development Environment (software for coding)
* VS Code: Visual Studio Code
* Interpreter: A program that executes code

[Environment check for Mac | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/zDEdn/environment-check-for-mac) 3-16-2024 0715

* Not reviewed

[Running code - Command line VS IDE | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/gFxuh/running-code-command-line-vs-ide) 3-16-2024 0719

* Not reviewed

[Python syntax, spaces matter | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/dvyoj/python-syntax-spaces-matter) 3-17-2024 0555

* The lesson covers how incorrect use of whitespace and indentation in Python can introduce syntax errors and other code issues. It stresses the significance of adhering to Python's whitespace and indentation rules, while also demonstrating how analyzing error messages can aid in locating and resolving such problems. <claude>
* Multiple print statements on the same line need to be separated by a semicolon and a space.
* The backslash (\) can be used for line continuation when combining lines with arbitrary whitespace.

[Python syntax cheat sheet | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/9w2n1/python-syntax-cheat-sheet) 3-17-2024 0555

* Extremely basic. I this is your first time learning python definitely focus on this, if your experienced skip it.

[Commenting code | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/cCh1k/commenting-code) 3-17-2024 0555

* Extremely basic. I this is your first time learning python definitely focus on this, if your experienced skip it.

[Variables | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/m8Xps/variables) 3-17-2024 0622

* The lesson focuses on introducing the concept of variables in Python programming and emphasizing the importance of using meaningful and consistent variable names. It also covers how to declare, assign values, reassign, and delete variables in Python. <claude>
* Key Points:
* Variables can be declared and assigned values of different data types, such as integers, strings, etc.
* The del command is used to delete a variable.
* Terms, Acronyms, and Commands:
* Variable: A named storage location in memory that holds a value.
* Camel Case: A naming convention where the first word is in lowercase, and subsequent words have their first letter capitalized (e.g., myVariableName).
* Snake Case: A naming convention where words are separated by underscores and all letters are lowercase (e.g., my\_variable\_name).
* del: A Python command used to delete a variable.

[Basic data types | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/CrJKk/basic-data-types) 3-17-2024 0633

* The lesson focuses on introducing the different data types available in Python programming and explaining how Python automatically assigns the appropriate data type to variables based on their assigned values. It emphasizes the importance of understanding data types for proper data interpretation and manipulation.

[Strings | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/8ctQ2/strings) 3-17-2024 0637

* The lesson covers the concept of strings in Python programming, explaining how they are sequences of characters enclosed in quotes and how to declare, manipulate, and access individual characters within strings. It also introduces the idea of string concatenation and the use of the len() function to determine the length of a string. <claude>

[Basic Data type and Function Cheatsheet | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/uLjKS/basic-data-type-and-function-cheatsheet) 3-17-2024 0643

* Basic

[Type casting | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/kyqAa/type-casting) 3-17-2024 0650

* Summary: Typecasting allows you to convert between different data types in Python. This can be done either implicitly (automatically by Python) or explicitly using built-in functions.
* Typecasting Functions
* str() Converts to string.
* int() Converts to integer.
* float() Converts to floating-point number.
* ord() Returns an integer representing the underlying unicode character.
* hex() Converts a given integer to a hexadecimal string.
* oct() Takes an integer and returns a string representing an octal number.
* tuple() Converts to a tuple.
* set() Converts to a set.
* list() Converts to a list.
* dict() Converts to a dictionary.
* Implicit vs Explicit Conversion
* Implicit conversion: Python can automatically convert between compatible data types. For example, it can convert an integer to a float if it encounters a decimal value.
* Explicit conversion: You can use the functions mentioned above to explicitly convert between data types, regardless of compatibility.

[User input, console output | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/OLqVN/user-input-console-output)

* Summary: Python provides input and output functions to interact with users. The input() function collects data from the user, while the print() function displays information on the screen.

[Type casting, a deeper look | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/DB2hx/type-casting-a-deeper-look)

* This lesson focuses on type casting or data type conversion in Python, which is the process of changing the data type of a value from one type to another. It highlights the importance of explicit type conversion, especially when working with user input or combining different data types in operations or output strings.

[Type casting input | Coursera](https://www.coursera.org/learn/programming-in-python/programming/M1Gzs/type-casting-input) <LAB> 3-17-2024 0708

Basic stuff I did not do it again. This is my second pass through the course.

[Knowledge check - Welcome to Python Programming | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/2TQYq/knowledge-check-welcome-to-python-programming)<Quiz> 3-17-2024 0710

5-5

[Additional resources | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/GknR9/additional-resources) 3-17-2024 0712

* [Python](https://docs.python.org/3/library/functions.html) Check out W3 Schools to learn more about coding and web development:
* [W3Schools](https://www.w3schools.com/python/default.asp) Check out W3 Schools to learn more about coding and web development
* [HackerRank](https://www.hackerrank.com/domains/python) Check out HackerRank to practice your new acquired Python skills

[Math and logical operators | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/X9AdI/math-and-logical-operators) 3-17-2024 0724

* Summary: Python provides mathematical operators (+, -, \*, /) for performing calculations, and logical operators (and, or, not) for evaluating conditions in decision-making. These operators are essential for controlling the flow of your programs.

[Control flow: If / else, else if | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/uRlis/control-flow-if-else-else-if) 3-17-2024 0728

basic

[Conditional statements | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/ydKG3/conditional-statements) 3-17-2024 0730

Basic

[Switch statement | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/iDVVy/switch-statement) 3-17-2024 0750

* Summary: The match statement offers a cleaner way to compare a variable against multiple possible values. It's a useful alternative to lengthy if-elif-else chains, especially when you have many conditions.
* Key Terms
* match statement: Introduced in Python 3.10 for pattern matching.
* case: Equivalent to an if or elif condition.
* \_ (underscore): The default case, similar to an else block.
* | (pipe symbol): Acts as an "or" between multiple patterns.
* keywords = ["match", "case", "pattern matching", "if-elif-else"

[Looping constructs | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/oDGCN/looping-constructs) 3-17-2024 0756

* Summary: Python offers two ways to repeat code blocks: the for loop iterates through sequences (like strings or lists), while the while loop continues as long as a specified condition remains true.
* Key Terms
* Loop: A programming construct that repeats a block of code.
* Sequence: An ordered collection of items (e.g., strings, lists).
* Iteration: One cycle through a loop's code block.
* Index: The position of an item within a sequence.
* enumerate(): A function used to get both index and value during a for loop.
* keywords = ["loop", "for loop", "while loop", "iteration", "sequence", "index", "enumerate"]

[Looping Constructs: Practical Examples | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/t1TZ5/looping-constructs-practical-examples)

* Basic

[Practicing control flow and loops | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/M7OTq/practicing-control-flow-and-loops) 3-17-2024 1140

* Did the code but wrote it first in notepad because VSC with CoPilot was not letting me learn. The auto complete is scary.

[Nested loops and the effect on algorithmic complexity | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/NqmUe/nested-loops-and-the-effect-on-algorithmic-complexity)

* Summary: Nested loops consist of an outer loop containing an inner loop. The inner loop executes completely for each iteration of the outer loop, potentially leading to longer execution times for large datasets.
* Terms, Acronyms, Concepts, Commands, Libraries
* Nested loops: Loops placed within other loops.
* Outer loop: The primary loop in a nested loop structure.
* Inner loop: The loop contained within the outer loop.
* Time complexity: A measure of how an algorithm's runtime increases with input size.
* range(): Function to generate a sequence of numbers.
* time module: Library for working with time-related functions.
* keywords = ["nested loops", "outer loop", "inner loop", "Python", "time complexity", "range", "time module"]

[Exercise: Use control flow and loops to solve a problem | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/kaRQQ/exercise-use-control-flow-and-loops-to-solve-a-problem) 3-17-2024 1226

* Summary: This exercise demonstrates how to use for loops, conditional statements (if, elif, else) and the enumerate function to manipulate and analyze a list of numbers. Control flow and loops are essential for solving programming problems that involve iterating over data.
* Acronyms, Terms, Commands
* for loop: A structure for iterating over sequences of data.
* if, elif, else: Conditional statements for controlling the flow of execution based on conditions.
* enumerate(): A function that adds an index counter to items in a sequence during iteration.
* keywords = ["for loop", "control flow", "conditional statement", "enumerate", "index", "iteration", "Python"]

[Use control flow and loops to solve a problem - solution | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/4UK39/use-control-flow-and-loops-to-solve-a-problem-solution) 3-17-2024 1232

Exemplar

[Self-review: Use control flow and loops to solve a problem | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/COT3y/self-review-use-control-flow-and-loops-to-solve-a-problem) 3-17-2024 1236

2 of 3. Guess I did not fully understand break. I thought it stopped code no matter where it is seen. It just breaks out of a for loop.

[Module quiz: Getting started with Python | Coursera](https://www.coursera.org/learn/programming-in-python/exam/T5Npi/module-quiz-getting-started-with-python)

* 9 of 10 then 10 0f 10
* Dynamically Typed: (Concept) In programming, it means the language automatically assigns data types to variables at runtime, rather than requiring them to be defined by the programmer.
* isinstance(): (Command) A standard function to check if an object is of a specified type.
* Enumerate Function: Provides both the index and the value of each item in a sequence, allowing for more informative loop iterations.

[Functions | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/jn7Yx/functions) 3-17-24 1302

* Basic

[Variable scope | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/ohNir/variable-scope) 3-17-24 1500

* Scope in Programming
* A diagram of a scope

  Description automatically generated
* 1. Local Scope
* Definition: Refers to variables declared inside a function or a code block (delimited by curly braces in many languages).
* Accessibility: Local variables are only accessible within the function or block where they're created.
* Lifetime: Local variables exist only while the function or block is executing. Once the function/block finishes, those variables are destroyed.
* 2. Enclosing Scope (Nested Scope)
* Definition: A scope created by a function within another function (nested functions). Variables from an outer function are accessible within the inner function.
* Accessibility: Variables from the enclosing scope are accessible by the inner function, forming a chain of nested scopes.
* Modification: Inner functions can typically read variables from the enclosing scope. Whether they can modify them depends on the programming language.
* 3. Global Scope
* Definition: Refers to variables declared outside of any function or block.
* Accessibility: Global variables are accessible from anywhere within the entire program file.
* Lifetime: Global variables persist throughout the entire program execution.
* Caution: Overuse of global variables can lead to naming conflicts and make code harder to maintain, so it's generally recommended to minimize their use.
* 4. Built-in Scope
* Definition: The scope containing pre-defined functions and objects provided by the programming language itself (e.g., print(), len(), built-in data types).
* Accessibility: Built-in names are always available in any part of your code.
* Block Scope: Some languages (like C++, Java) have block scope, where variables defined within a set of curly braces are only accessible within that block.
* Key Points about Scope
* Scope helps prevent naming conflicts and accidental modification of variables.
* It's generally good practice to keep variables in the smallest scope possible to enhance code readability and maintainability.

[Function and variable scope | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/PmZPi/function-and-variable-scope) 3-17-24 1505

Understanding variable scope in Python is crucial for effective programming, highlighting the distinction between local, enclosing, global, and built-in scope levels, which determine where and how variables and functions can be accessed within the code. Local scope is confined to the defining function, enclosing scope pertains to variables in a nested function's outer function, global scope encompasses variables declared outside any function, and built-in scope covers Python's predefined keywords and functions.

Four Scopes Defined

Local Scope: Variables defined within a function, accessible only inside that function.

Enclosing Scope: Pertains to a variable in an outer function that is accessible to its nested (inner) functions but not outside the outer function.

Global Scope: Variables defined at the top level of a script or module, accessible from anywhere in the code.

Built-in Scope: The scope of Python’s built-in functions and keywords, accessible globally from any part of the code.

[What are data structures? | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/tM8H9/what-are-data-structures) 3-17-24 1505

* Data structures in Python, such as lists, dictionaries, tuples, and sets, are designed to organize and manage complex collections of data, enabling efficient operations on them. They can be either mutable, allowing modifications like changes, updates, or deletions (e.g., lists), or immutable, preventing any changes once data is set (e.g., tuples).

[Lists | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/nbNBK/lists) 3-17-24 1515

* **Summary:** In Python, lists function as versatile arrays that can store elements of various data types. You can easily access, modify, add, and remove list items using built-in functions and indexing.
* **Terms, Commands, Acronyms, Ideas**
* **List:** An ordered collection of items.
* **Index:** The numerical position of an item in a list, starting from 0.
* **Nested List:** A list containing other lists.
* **insert():** Function to insert an item at a specific index.
* **append():** Function to add an item to the end of a list.
* **extend():** Function to add multiple items from another list.
* **pop():** Function to remove an item at a specific index (or the last item if no index is given).
* **del:** Keyword to delete an item at a specific index.
* **Iteration:** The process of looping through list items.
* keywords = ["Python", "list", "data structure", "index", "insert", "append", "extend", "pop", "delete", "iteration"]

[Tuples | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/HbrMT/tuples) 3-17-24 1529

* Basic

[Sets | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/ReZbS/sets) 3-17-2024 1541

* **Summary:** Sets in Python are unordered collections of unique elements. They offer useful methods for manipulating data, including mathematical set operations like union, intersection, difference, and symmetric difference.
* **Acronyms, Commands, Terms, Ideas**
* **Set:** An unordered collection of unique elements.
* **Curly Braces {}:** Used to define a set.
* **Duplicate Values:** Sets cannot contain duplicate elements.
* **add()**: Method to add an element to a set.
* **remove()**: Method to remove a specified element from a set.
* **discard()**: Method to remove a specified element (if it exists) from a set.
* **union() / |** : Operators to combine the elements of two sets into a new set.
* **intersection() / &**: Operators to find the common elements between two sets.
* **difference() / -**: Operators to find elements in one set that are not in another.
* **symmetric\_difference() / ^**: Operators to find elements that are in either of two sets, but not in both.
* keywords = ["Python", "set", "unordered", "unique", "add", "remove", "discard", "union", "intersection", "difference", "symmetric\_difference"]

[Dictionaries | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/DVkab/dictionaries) 3-17-2024 1600

* Summary: Dictionaries in Python are unordered collections of key-value pairs. They offer fast retrieval of values using keys, making them useful for storing and organizing data where quick access by key is important.
* Commands, Acronyms, Ideas
* Dictionary: A data structure that maps unique keys to associated values.
* Key: A unique identifier used to access a value in the dictionary.
* Value: The data associated with a key in the dictionary.
* Key-Value Pair: A single unit in a dictionary consisting of a key and its corresponding value.
* Curly braces {}: Used to define a dictionary.
* del: Keyword to delete a key-value pair from a dictionary.
* items(): Method to iterate through both keys and values of a dictionary.
* keywords = ["Python", "dictionary", "key", "value", "key-value pair", "unordered", "lookup", "data structure"]

[kwargs | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/tRj6Q/kwargs) 3-17-2024 1605

* Summary: In Python, \*args and \*\*kwargs provide flexibility when defining functions that can accept variable numbers of arguments. \*args allows you to pass in an arbitrary number of non-keyword arguments, while \*\*kwargs lets you pass an arbitrary number of keyword arguments.
* \*args collects all the positional arguments into a tuple, while \*\*kwargs collects all the keyword arguments into a dictionary.
* All \*args must be first then the \*\*kwargs for the values passed into a function.
* Commands, Acronyms, Important Info
* \*args: Used in a function definition to gather non-keyword arguments into a tuple.
* \*\*kwargs: Used in a function definition to gather keyword arguments (e.g., name=value) into a dictionary.
* keywords = ["Python", "functions", "arguments", "\*args", "\*\*kwargs", "variable arguments", "keyword arguments"]

[Functions, loops and data structures | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/PXZhd/functions-loops-and-data-structures)<quiz> 3-17-2024

* 5 of 5

[Choosing and using data structures | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/Ncam1/choosing-and-using-data-structures) 3-17-2024 1630

* Basic

[Visual Studio Code on Coursera | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/2IEyt/visual-studio-code-on-coursera) 3-17-2024 1630

* Shows how to reset a lab.
* Keywords = [‘lab’, ‘reset lab’]

[Functions, loops and data structures | Coursera](https://www.coursera.org/learn/programming-in-python/programming/tb6oS/functions-loops-and-data-structures) 3-17-2024 1645

* Lab that I chose not to repeat, but I did copy to my repo and played with the code.

[Knowledge check: Functions and Data structures | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/JDtVK/knowledge-check-functions-and-data-structures/attempt?redirectToCover=true) 3-17-2024 1650

* 5 of 5

[What are exceptions | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/Bv7pJ/what-are-exceptions) 3-18-2024 0535

* Summary: Errors and exceptions signal problems in Python code. Understanding how to identify and handle them is crucial for writing robust programs.
* Ideas, Acronyms, Definitions
* Errors: Problems with the code itself that prevent it from running.
* Syntax Error: Invalid syntax (e.g., missing colon, incorrect indentation).
* Exception: An error that occurs during program execution (e.g., dividing by zero).
* IDE: (Integrated Development Environment) Software like Visual Studio Code that assists in coding.
* keywords = ["Python", "error", "exception", "syntax error", "ZeroDivisionError", "IDE"]

[Exercise: Exceptions in Python | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/moVSY/exercise-exceptions-in-python) 3-18-2024 0540

* Basic

[Exceptions in Python | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/DdZ9w/exceptions-in-python/attempt?redirectToCover=true) 3-18-2024 0545

* Missed one. What is the base error in Python. I chose BaseExcexeption, but it was Exception.

[File handling in Python | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/oPHqM/file-handling-in-python)

[File handling in Python | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/oPHqM/file-handling-in-python) 3-18-2024 0944

* Summary: This video introduces the basics of file handling in Python. It explains how to use the open() and close() functions, along with different file modes, to read and write text and binary files.
* Terms, Concepts, and Definitions:
* File Handling: The process of working with files on a computer system.
* open() function: Used to open a file and establish a connection to it.
* close() function: Used to close the connection to an open file.
* Mode: An argument to the open() function, indicating the action you want to perform (read, write, create, etc.).
* r: Read mode (text format)
* rb: Read mode (binary format)
* w: Write mode (overwrites existing file)
* a: Append mode (adds to the existing file)
* Text Format: Human-readable file format.
* Binary Format: Non-human-readable format, more efficient for storage.
* with open(): Context manager that automatically closes the file upon block completion.
* keywords = "file handling", "open", "close", "mode", "read", "write", "text format", "binary format", "with open"]

[Creating Files | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/Rtpzt/creating-files) 3-18-2024 1042

* Summary: This video teaches how to create files in Python using the open() function with write ('w') and append ('a') modes. It also demonstrates inserting text content, managing newlines, and handling the FileNotFoundError exception for robust file creation.
* Terms, Commands, Libraries, Definitions:
* File Creation: The process of making a new file on a computer system.
* open() function: Python's built-in function to establish a connection to a file.
* Modes:
* 'w' (write): Creates a new file or overwrites an existing one.
* 'a' (append): Adds content to the end of an existing file.
* write() function: Inserts text into a file.
* readlines() function: Reads all lines of a file into a list.
* Newline (\n): Character used to represent the end of a line and start a new one.
* try...except: Python's error handling mechanism.
* FileNotFoundError: Specific exception raised when the specified file is not found.
* keywords = ["Python", "file creation", "open", "write", "append", "writelines", "newline", "exception handling", "FileNotFoundError"]
* I did a few exercises saved to ***file\_practice.py***

[Reading Files | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/EbEvm/reading-files) 3-18-2024 1120

Summary: This lesson teaches how to read file content in Python. It covers the read(), readline(), and readlines() functions, along with explanations of absolute and relative file paths.

Terms & Ideas:

* read() function: Reads the entire file as a single string. Can optionally read a specified number of characters.
* readline() function: Reads a single line of the file as a string. Can also read a specified number of characters.
* readlines() function: Reads the entire file into a list, where each list item is a line from the file.
* Absolute path: Specifies the complete location of a file from the root directory (e.g., "C:/Users/username/Documents/myfile.txt")
* Relative path: Specifies the file location relative to the current working directory (e.g., "myfile.txt")

keywords = ["Python", "file reading", "read", "readline", "readlines", "absolute path", "relative path", "with open"]

[Storing file contents in data structures | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/6DITb/storing-file-contents-in-data-structures) 3-18-2024 1140

* An exercise using open, read, and Random. Basic stuff. I did the exercise and saved it in ***file\_practice.py***

[Read in data, store, manipulate and output new data to a file | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/ICmRF/read-in-data-store-manipulate-and-output-new-data-to-a-file) <quiz> 3-18-2024 1149

5 of 5

[Read in data, store, manipulate and output new data to a file | Coursera](https://www.coursera.org/learn/programming-in-python/programming/2Frg6/read-in-data-store-manipulate-and-output-new-data-to-a-file)<LAB> 3-18-2024 1225

* This lab focuses on reading data from a file, manipulating this data in various ways, and then outputting new data to another file in Python. It emphasizes the use of file handling functions such as open, read, and write, and teaches how to store file contents in a list for different types of data manipulation including reading lines in sequential order, selecting specific lines, and reversing the line order.
* Commands, Acronyms, Concepts, and Definitions:
* open(): A built-in Python function used to open a file in a specified mode ('r' for reading, 'w' for writing, etc.).
* read(): Reads the entire contents of a file into a string.
* write(): Writes a specified string to a file.
* readlines(): Reads all the lines of a file into a list.
* Python List: A data structure in Python that is mutable, or changeable, and ordered. It can store a sequence of items.
* File Handling: The process of performing operations on a file, such as opening, reading, writing, and closing it.
* Sequential Order: Arranging items in a sequence, one after the other, typically starting from the first.
* Even-numbered lines: Lines of a file with an even number (2, 4, 6, etc.) when counting starts from 1.
* Reverse Order: Arranging items in the opposite sequence from their original order.
* keywords = ["open", "read", "write", "readlines", "File Handling", "Sequential Order", "Even-numbered lines", "Reverse Order"]

[Module summary: Basic Programming with Python | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/6mc36/module-summary-basic-programming-with-python) 3-18-2024 1226

Summary

[Module quiz: Basic Programming with Python | Coursera](https://www.coursera.org/learn/programming-in-python/exam/PXk6W/module-quiz-basic-programming-with-python) 3-28-2024

75% first try, 100% second try.

Week 3

[What is procedural programming? | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/16l4S/what-is-procedural-programming) 3-18-2024 1332

* Procedural programming involves breaking code into sequential procedures (or functions) that perform specific tasks. This approach promotes code reusability and maintainability, making it a good starting point for new developers.

[Algorithms | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/dP5ot/algorithms) 3-18-2024 1359

* Summary: Algorithms are step-by-step instructions designed to solve problems. This video uses a palindrome-checking example to illustrate how algorithms are created in code, breaking down the problem into smaller logical steps.
* The example given in the video is incorrect because it will not iterate through the string and will only ever check the first and last character. The code needed to increase the startIndex and reduce the endIndex at each loop.
* Terms, Concepts, & Definitions
* Algorithm: A finite sequence of well-defined steps to solve a problem.
* Palindrome: A word or phrase that reads the same forwards and backwards.
* Index: The numerical position of a character within a string (starting from 0).
* len() function: Python function to find the length of a string.
* keywords = ["algorithm", "problem-solving", "palindrome", "string", "index", "iteration", "for loop", "condition", "function"]

[Writing Algorithms | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/5Fzn9/writing-algorithms) 3-18-2024 1403

Basic stuff.

[Exercise: Make a cup of coffee | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/aPLwz/exercise-make-a-cup-of-coffee) 3-18-2024 1404

Basic stuff

[Algorithmic complexity | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/loqDb/algorithmic-complexity) 3-18-2024 1409

Basic stuff

[Intro to Big-O notation | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/o9U5O/intro-to-big-o-notation) 3-18-2024 1455

* Summary: Big O notation is a system for describing how the runtime of an algorithm scales with the size of its input. Understanding Big O notation allows you to compare algorithms, identify bottlenecks, and design code that performs well as datasets grow.
* Terms and Definitions:
* Big O Notation: Mathematical notation that describes the limiting behavior (worst-case or upper bound) of an algorithm's runtime as the input size increases.
* Time Complexity: Describes how the runtime of an algorithm changes in relation to the input size.
* Constant Time (O(1)): Runtime does not depend on input size.
* Linear Time (O(n)): Runtime increases linearly with input size.
* Quadratic Time (O(n^2)): Runtime increases with the square of the input size.
* Logarithmic Time (O(log n)): Runtime increases logarithmically with input size (very efficient).

[Knowledge check: Procedural Programming | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/HFv8c/knowledge-check-procedural-programming/attempt?redirectToCover=true)3-18-2024 1500

* 2 of 4. Definitely come back to this. I did some extra work trying to fully grasp. The wording of the questions threw me for a loop.
* I read this and hope I can retain it. [The Big O Notation - An Introduction - DEV Community](https://dev.to/sarah_chima/the-big-o-notation-an-introduction-34f7)
* Try this test again

[What is functional programming? | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/ZbsP2/what-is-functional-programming) 3-18-2024 1515

* Summary: Functional programming is a programming style that emphasizes the use of pure functions to create clean and maintainable code. Unlike traditional functions, pure functions always return the same output for a given input and avoid modifying external data.
* Terms, Acronyms, Concepts:
* Function: A reusable block of code that takes input, processes it, and returns output.
* Pure Function: A function that always produces the same output for the same input and has no side effects (doesn't modify anything outside its scope).
* Traditional Function: A function that can have side effects, potentially modifying data outside its scope.
* Global State: Data that can be accessed and modified from anywhere in the program.
* Local State Data that is only accessible within a specific function.
* Side Effects: Changes a function makes to the state of the program outside of its own scope.
* keywords = ["def", "return", "map", "sorted"]
* for fun I tried 4 or 5 different way to do the exercise. The shortest versionfound was by ChatGPT: ***print("\n".join([w[::-1] for w in coffee\_types]))***

[Pure functions | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/yyqzJ/pure-functions) 3-18-2024 1600

* Summary: Pure functions are a core concept in functional programming. They always return the same output for a given input and do not modify data outside their scope, promoting code that is predictable, easier to debug, and more extendable.
* Terms, Acronyms, Commands, Ideas:
* Pure Function: A function that doesn't change data outside its own scope and whose output depends solely on its input.
* Global Scope: The scope where variables are accessible from anywhere in the program.
* Local Scope: The scope where variables are only accessible within a specific function.
* Side Effect: Any change a function makes to the state of the program outside its own scope.
* .copy() method: Creates a shallow copy of a list.
* keywords = ["append()", "copy()"]

[Recursion | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/x6ZeP/recursion) 3-18-2024 1610

* Summary: Recursion is a programming technique where a function calls itself to break down a larger problem into smaller, self-similar subproblems. This approach can make code more elegant for tasks that have a naturally repetitive structure, but it's important to include a stopping condition to avoid infinite loops.
* Terms, Acronyms, and Ideas:
* Recursion: A function calling itself within its own definition.
* Base Case: The condition that stops the recursive calls, preventing an infinite loop.
* Factorial: The product of all positive integers less than or equal to a given number (e.g., factorial of 5 is 54321 = 120).
* keywords = [“recursion”]

[Reversing a string on Python | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/k2X68/reversing-a-string-on-python) 3-18-2024 1630

[Map & filter | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/ynhJp/map-filter) 3-19-2024 0500

* Summary
* This video tutorial explains how to use Python's map and filter functions to process lists. It demonstrates how to filter a list of coffee names to find those starting with the letter "C".
* Commands/Ideas
* map function: Applies a function to every element of a list, returning an iterator of results.
* filter function: Filters elements from a list based on a condition, returning an iterator of elements where the condition is true.
* iterator: An object that yields values one at a time, conserving memory.
* def: Keyword used to define a function.
* if statement: Used for conditional execution of code.
* keywords = ['map', 'filter', 'def', 'if']
* Map Object Consumption
* Once you've used an iterator produced by map (for example, by iterating over it in a loop), the iterator is considered "consumed". Trying to use it again will result in an empty output, as the values have already been yielded.

[Comprehensions | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/vBzIL/comprehensions) 3-19-2024 1100

* This reading introduces the concept of comprehensions in Python, which are compact ways to create new lists, dictionaries, sets, and generators from existing sequences. It covers the following types.
* List comprehensions: Used to create new lists by applying operations to elements of an existing list.
* Dictionary comprehensions: Used to create dictionaries, with keys and values derived from existing lists or sequences.
* Set comprehensions: Used to create sets, which store unique elements, based on elements from a sequence.
* Generator comprehensions: Similar to list comprehensions, but produce values on demand, making them more memory-efficient for large datasets.
* Acronyms, Commands, and Terms
* Comprehension: A concise way of creating sequences in Python.
* List Comprehension: Creates a new list.
* Dictionary Comprehension: Creates a new dictionary.
* Set Comprehension: Creates a new set.
* Generator Comprehension: Creates a generator object.
* map() function: Applies a function to each element of a sequence, returning a map object.
* if-else condition: Used to filter elements based on specific criteria.
* Important Lessons
* Comprehensions offer cleaner and more readable code than traditional for loops.
* Use the correct type of comprehension for your desired output (list, dictionary, set, or generator).
* Generator comprehensions are ideal for conserving memory with large datasets.
* keywords = ['comprehension', 'list comprehension', 'dictionary comprehension', 'set comprehension', 'generator comprehension', 'map', 'filter', 'if-else']

[Mapping key values to dictionary data structures | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/n6RAb/mapping-key-values-to-dictionary-data-structures/attempt?redirectToCover=true)<quiz> 3-19-2024 1100

* I missed 1 of 3. Try this again in a week

[Mapping key-values to Dictionary data structures | Coursera](https://www.coursera.org/learn/programming-in-python/programming/kzMv1/mapping-key-values-to-dictionary-data-structures) 3-19-2024 1100

* A lab I am not doing again.

[Knowledge check: Functional Programming | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/Np9Ie/knowledge-check-functional-programming) 3-19-2024 1215

* Missed 2 of 4 so need to redo. But I think I understand better. One question I was just lazy and didn’t fully read.
* [Recursion in Python: An Introduction – Real Python](https://realpython.com/python-recursion/#what-is-recursion)

[Introduction to Object Oriented Programming | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/XmPiS/introduction-to-object-oriented-programming) 3-19-2024 1215

* Object-oriented programming (OOP) is a programming paradigm that simplifies complex coding by modeling real-world concepts as classes and objects. OOP relies on four key concepts: inheritance, polymorphism, encapsulation, and abstraction, to enhance code reusability and organization.
* Acronyms, Commands, and Definitions
* OOP: Object-Oriented Programming, a programming paradigm.
* Class: A blueprint for creating objects.
* Object: An instance of a class, with its own attributes and behaviors.
* Method: A function defined within a class, determining object behavior.
* Inheritance: Creating a new class (subclass) by deriving from an existing one (parent class).
* Polymorphism: The ability of a function to adapt to different object types.
* Encapsulation: Bundling variables and methods within a class to control access.
* Abstraction: Hiding implementation details to simplify usage.
* keywords = ['class', 'object', 'method', 'inheritance', 'polymorphism', 'encapsulation', 'abstraction']

[OOP Principles | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/4VaFD/oop-principles) 3-19-2024 1230

* Object-oriented programming (OOP) utilizes principles to organize code and make it reusable. Key principles of OOP include encapsulation (bundling data and methods within classes), polymorphism (functions adapting to different data types), inheritance (creating new classes from existing ones), and abstraction (hiding implementation details).
* Key Terms, Ideas, and Methodologies
* Encapsulation: Controlling access to data and methods within a class to establish clear boundaries.
* Information Hiding: Protecting internal data representation, often using access modifiers like underscores in Python.
* Polymorphism: The ability of functions/operators to behave differently depending on the object they work with.
* Inheritance: Deriving new classes (subclasses) from existing ones (parent classes) to reuse code and create hierarchies.
* Method Resolution Order (MRO): Rules Python uses to determine the order of inheritance in complex class structures.
* Abstraction: Simplifying complexity by hiding implementation details and providing essential interfaces.
* Abstract Base Class (ABC): A blueprint for creating abstract classes in Python using the abc module.
* Acronyms
* OOP: Object-Oriented Programming
* MRO: Method Resolution Order
* ABC: Abstract Base Class
* keywords = ['class', 'object', 'method', 'encapsulation', 'information hiding', 'polymorphism', 'inheritance', 'MRO', 'abstraction', 'ABC']

[Exercise: Define a Class | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/hq9io/exercise-define-a-class) 3-19-2024 1430

* Basic

[Self-review: Define a Class | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/b7is1/self-review-define-a-class)3-19-2024 1430

Missed one because I rushed to answer without fully thinking. Not because I did not know the answer.

[Instantiate a custom Object | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/DDfnl/instantiate-a-custom-object) 3-19-2024 1550

[Exercise: Instantiate a custom Object | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/jxAD4/exercise-instantiate-a-custom-object) 3-19-2024 1600

* This is a practice scenario instantiating an object. I do not fully grasp this and failed in my first attempt. I finally got it, but only with the help of CoPilot.
* I will need to do this again

[Instance methods | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/KiDE8/instance-methods) 3-20-2024 0430

* This video introduces instance variables and instance methods in Python object-oriented programming. These concepts allow you to store data (instance variables) and define behaviors (instance methods) that are specific to each instance (object) of a class, enabling you to manage the state of individual objects independently.
* Acronyms, Commands, Concepts, and Definitions
* Instance Variable: An attribute (piece of data) associated with a specific instance of a class. Each instance can have its own unique values for instance variables.
* Instance Method: A function defined within a class that operates on the data (instance variables) of a specific instance.
* State of an Object: The current values of an object's instance variables. Instance methods can modify an object's state.
* \_\_init\_\_() method: The special constructor method in a class, responsible for initializing instance variables when an object is created.
* self: A keyword within class methods referring to the specific instance on which the method is called.
* keywords = [‘instance methods’]
* I did two simple challenges given by Gemini that forced me to create a class and instances and use them. I wrote them in notepad first.

[Parent classes vs. child classes | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/H2mB8/parent-classes-vs-child-classes) 3-20-2024 0600

* Summary: This video explains inheritance, a fundamental concept in object-oriented programming. In Python, inheritance allows child classes (subclasses) to inherit properties and behaviors from parent classes (superclasses), promoting code reusability and organization.
* Acronyms, Ideas, and Commands:
* Inheritance: A mechanism where a child class derives attributes and methods from a parent class.
* Parent class (superclass, base class): The class from which attributes and behaviors are inherited.
* Child class (subclass, derived class): The class that inherits from the parent class.
* super() function: Used within a child class to access inherited methods and properties from the parent class.
* keywords = ["Python", "OOP", "inheritance", "class", "parent class", "child class", "super()", "object"]
* I completed a challenge from Gemini based on the lesson plan.

[Inheritance and Multiple Inheritance | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/ZUR2d/inheritance-and-multiple-inheritance) 3-20-2024 0700

* This reading explores inheritance in Python, a key concept in object-oriented programming that allows for code reusability and a hierarchical structure. It covers the following types of inheritance:
* Simple Inheritance: A child class derives attributes and methods from a single parent class.
* Multiple Inheritance: A child class inherits from multiple parent classes.
* Multi-level Inheritance: A child class inherits from a parent class, which in turn inherits from its own parent class (creating a chain).
* Important Terms, Acronyms, Ideas, and Commands
* Inheritance: Mechanism for deriving new classes (child classes) from existing ones (parent classes).
* Parent Class (Base Class, Superclass): The class from which attributes and behaviors are inherited.
* Child Class (Derived Class, Subclass): The class that inherits from a parent class.
* issubclass(): Built-in function to check if a class is a subclass of another class.
* isinstance() Built-in function to check if an object is an instance of a particular class.
* super() Built-in function used within a child class to access methods and properties from its parent class(es).
* keywords = ["Python", "OOP", "inheritance", "class", "parent class", "child class", "multiple inheritance", "multi-level inheritance", "issubclass", "isinstance", "super"]

[Abstract classes and methods | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/rh0lN/abstract-classes-and-methods) 3-20-2024 1200

* Abstract classes provide a blueprint for derived classes, ensuring consistent interfaces and avoiding code duplication. Python achieves abstraction using the ABC module and requires derived classes to implement abstract methods defined in the abstract base class.
* Acronyms/Commands:
* ABC: Abstract Base Class (the Python module for abstraction)
* Ideas/Concepts:
* Abstract class: A class that cannot be directly instantiated, serving as a template for derived classes.
* Abstract method: A method declared in the abstract class but lacking implementation; derived classes must provide the implementation.
* Instantiation: The process of creating an object (instance) of a class.
* Decorator: A function that modifies the behavior of another function (in this case, the @abstractmethod decorator marks a method as abstract).
* keywords = ["abstract class", "abstract method", "inheritance", "interface", "blueprint", "Python", "ABC module", "decorator"]
* I did a challenge generated from Gemini. You can see in the code what happens with the decorator and without. This is a flaw in this lesson, the instructors do not really communicate what happens when you don’t use the decorator.
* Link to code: [Meta-Back-End-Developer-Professional-Certificate/programming\_in\_python/week3/gemini-challenge-for-abstract.py at master · Inouye165/Meta-Back-End-Developer-Professional-Certificate (github.com)](https://github.com/Inouye165/Meta-Back-End-Developer-Professional-Certificate/blob/master/programming_in_python/week3/gemini-challenge-for-abstract.py)

[Abstract classes and methods | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/yhRl6/abstract-classes-and-methods) 3-20-2024 1253

I did horrible and obviously am not grasping the lesson fully. RETAKE

[Method Resolution Order | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/p4jBX/method-resolution-order) 3-20-2024 1327

* Python uses Method Resolution Order (MRO) to determine the inheritance search path for methods and attributes. MRO becomes crucial in complex inheritance scenarios (e.g., multiple inheritance) to avoid ambiguity and ensure predictable behavior.
* Important Information:
* MRO: Method Resolution Order – the order Python follows to find methods or attributes in a class hierarchy.
* Inheritance types: Simple, multiple, multi-level, hierarchical, hybrid.
* Linearization: The process of determining the MRO.
* C3 Linearization: The algorithm used by Python 3 for MRO.
* DFS: Depth-First Search - an algorithm used in older Python versions for MRO.
* .mro() function: Displays the MRO of a class in Python.
* help() function: Provides detailed information about a class, including its MRO.
* keywords = ["MRO", "inheritance", "linearization", "C3 linearization", "DFS", "multiple inheritance", "multi-level inheritance", "hierarchical inheritance", "hybrid inheritance", ".mro()", "help()"]

MRO (Method Resolution Order) in Python – By Gemini

Purpose:

MRO is the set of rules Python follows to determine the order in which it searches for a method or attribute within a class hierarchy, especially when dealing with multiple inheritance.

Why it matters: In complex inheritance scenarios, multiple classes might define the same method or attribute. The MRO ensures a predictable and unambiguous way for Python to decide which implementation to use.

How it works (simplified):

Depth-first, left-to-right: Python generally searches the current class first, then its parent classes, and so on, moving from the most specific subclass to the most general base class. Within a given level of the hierarchy, it proceeds from left to right.

C3 Linearization Algorithm: Modern Python uses a specific algorithm called C3 linearization to calculate the MRO. This algorithm provides consistent and logical method resolution, even in complex diamond inheritance patterns.

Key Points:

MRO is crucial for maintainable code: Understanding MRO helps you predict how your classes will interact and avoids unexpected behavior in inheritance hierarchies.

Tools: You can use the .mro() attribute on a class or the help() function to inspect the MRO of a class.

[Working with Methods: Examples | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/7XKNQ/working-with-methods-examples) 3-20-2024 1345

* Multiple inheritance in Python allows a class to inherit from several parent classes. The Method Resolution Order (MRO) dictates the order in which Python searches for methods and attributes within complex inheritance hierarchies, ensuring predictable behavior.
* Important Information
* MRO: Method Resolution Order – The sequence Python follows to resolve inheritance conflicts in multiple inheritance.
* C3 Linearization: The algorithm modern Python uses to calculate the MRO.
* Diamond Inheritance: A pattern where a class inherits from two classes that both inherit from a common class, leading to potential ambiguity.
* .mro() function: Displays the Method Resolution Order of a class.
* keywords = ["multiple inheritance", "MRO", "C3 linearization", "diamond inheritance", ".mro()"]

[Exercise: Working with Methods | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/RDQDT/exercise-working-with-methods) 3-20-2024 1400

* Basic example

[Working with Methods - solution | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/KF8D3/working-with-methods-solution) 3-20-2024 1430

[Module summary: Programming paradigms | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/jRb8c/module-summary-programming-paradigms) 3-20-2024 1500

* I did horribly. Return to this multiple times.

[Module quiz: Programming Paradigms | Coursera](https://www.coursera.org/learn/programming-in-python/exam/LXRUu/module-quiz-programming-paradigms) 3-20-24 1520

Missed part of one question, because I didn’t realize it was pick all that apply and selected first I saw and moved on.

Week 4

[What is a module in Python? | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/qi5Vh/what-is-a-module-in-python) 3-20-2024 1529

* Introduction to the new week

[Accessing modules | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/L3XYl/accessing-modules) 3-20-2024 1615

* Python modules, reusable code stored in files, can be accessed from various locations. These locations include the current directory, built-in module directories, and directories defined in the Python path environment variable. Importing modules allows you to leverage pre-written code, promoting code efficiency and organization. To explore a module's documentation in detail, you can right-click on its name in your code editor (like Visual Studio Code) and choose an option like "Go to definition" or "View documentation."
* Important Information
* Module: A Python file containing reusable code.
* Built-in modules: Core modules that come pre-installed with Python.
* User-defined modules: Modules that you create yourself.
* sys.path: A list of directories the Python interpreter searches for modules.
* import statement: Used to bring modules into your code's namespace.
* Within VSC, gain quick insights into libraries by hovering over their names while holding Ctrl (Windows) or Cmd (macOS).
* keywords = ["module", "built-in module", "user-defined module", "sys.path", "import"]

[The import statement | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/Ol5YW/the-import-statement) 3-20-2024 1635

* Modules as Files: Every Python file (.py) is inherently a module. This emphasizes the modular structure of Python.
* Main Module: The script you directly execute is considered the "main module."
* Current Working Directory (Scope): Importing modules from within the same directory is the simplest approach.
* Built-in Modules: Python's standard library provides a rich set of pre-installed modules (e.g., json). No additional installation is needed.
* Python Package Index (PyPI): The central repository for finding and installing third-party Python packages.
* pip: The standard package installer for Python; used to manage packages from PyPI.
* init.py Files: These files are essential to signal to Python that a directory should be treated as a package.
* sys.path: A list of directories where Python searches for modules. You can modify it to import modules from other locations.

[Writing import statements | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/WJKCe/writing-import-statements) 3-20-2024 1643

* Very basic

[Namespacing and scoping | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/iiaP3/namespacing-and-scoping) 3-20-2024 1709

* Python namespaces provide a way to organize names (e.g., variables, functions) within a program. Scopes define the visibility and accessibility of these names, with LEGB rule (Local, Enclosed, Global, Built-in) dictating the order Python searches for them.
* Important Terms
* Namespace: A mapping between names and objects within a Python program. Modules themselves are a form of namespace.
* Scope: The region of code where a namespace is directly accessible.
* **LEGB** Rule: The order Python follows: **L**ocal, **E**nclosed (nested functions), **G**lobal, **B**uilt-in.
* Implicit Declaration: Python variables are created when you first assign a value to them.
* locals() and globals(): Built-in functions that display the contents of local and global namespaces, respectively.
* global and nonlocal Keywords: Keywords used to modify variable scope within functions.
* Access Modifiers: While not explicit in Python, conventions exist (like using underscores) to signal intended privacy levels (e.g., \_protectedVariable, \_\_privateVariable). This is important for code organization and maintainability in large projects.
* Concurrency: Python utilizes the Global Interpreter Lock (GIL), which impacts how multiple threads are handled. Understanding concurrency models (threading, multiprocessing, asyncio) is crucial for writing efficient parallel code in Python.
* Memory Allocation: Python has automatic memory management (garbage collection). While often seamless, being aware of memory allocation patterns can help optimize performance in resource-intensive applications.
* keywords = ["namespace", "scope", "LEGB rule", "implicit declaration", "locals", "globals", "global", "nonlocal",” Access Modifiers”,” Concurrency”,” Memory Allocation”]

[Import and Scope | Coursera](https://www.coursera.org/learn/programming-in-python/programming/ev3oz/import-and-scope)<lab> 3-20-2024 1727

* Just a lab. Not doing it again, but moved it to GitHub and played with it a bit.

[reload() function | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/RB8hh/reload-function) 3-20-2024 1800

* I’m probably missing why this was important, but seemed silly and basic.

[Module Use-cases | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/4tG6q/module-use-cases) 3-20-2024 1615

* Basic

[Knowledge check: Modules | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/NpeaX/knowledge-check-modules) 3-20-2024 1627

* OMG I don’t get some of these questions. Retry

[Popular packages: NumPy, pandas, Matplotlib, etc | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/CVxZA/popular-packages-numpy-pandas-matplotlib-etc) 3-20-2024 1634

* Python packages are organized collections of modules that provide specialized functionality. The vast Python Package Index (PyPI) offers packages for various domains, including data science, machine learning, web development, and more, making Python a versatile language for diverse projects.

[Popular Packages: Examples | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/j4eGx/popular-packages-examples) 3-20-2024 1635

* Basic. Covers some packages so good to read at least once.

[Data analysis packages | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/Kahiq/data-analysis-packages) 3-21-2024 0450

Short summary: This video provides an overview of essential Python libraries for data science. It introduces NumPy, Pandas, Scikit-learn, and Matplotlib, highlighting their roles in data analysis, manipulation, machine learning, and visualization.

Acronyms:

* CSV: Comma-Separated Values (a file format for tabular data)
* JSON: JavaScript Object Notation (a data exchange format)
* SVM: Support Vector Machine (a machine learning algorithm type)

Software packages:

* NumPy: Provides efficient array structures and numerical computation tools.
* Pandas: Offers data structures and tools for analysis and manipulation.
* Scikit-learn: Contains various machine learning algorithms for tasks like classification and regression.
* Matplotlib: A powerful library for creating various visualizations.

Commands/modules:

* import pandas as pd: Imports the Pandas library with the common alias 'pd'.
* import NumPy as np: Imports the NumPy library with the common alias 'np'.
* import matplotlib.pyplot as plt: Imports the pyplot module from Matplotlib for plotting.

Ideas/techniques:

* Data analysis: Examining and cleaning data to extract insights.
* Data manipulation: Transforming and organizing data for analysis.
* Machine learning: Algorithms that allow computers to learn from data.
* Visualization: Creating graphs and charts to represent data.

keywords = ['Python', 'data science', 'data analysis', 'NumPy', 'Pandas', 'Scikit-learn', 'Matplotlib', 'CSV', 'JSON', 'ND array', 'pyplot', 'machine learning', 'visualization']

Note: Generated by Gemini

[Machine learning, deep learning and AI: PyTorch, TensorFlow | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/xVEp9/machine-learning-deep-learning-and-ai-pytorch-tensorflow) 3-21-2024 0530

* No real learning just talks about libraries.

[Big Data and Analysis with Python | Coursera](https://www.coursera.org/learn/programming-in-python/supplement/J17qh/big-data-and-analysis-with-python) 3-21-2024 0540

* No real learning just talks about libraries.

[Python web frameworks | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/tFVsp/python-web-frameworks) 3-21-2024 0540

* Short summary This video explains web frameworks, software that helps build web applications efficiently. It covers Python frameworks like Django and Flask, full-stack and microframeworks, and how they simplify development.
* Acronyms:
* HTTP: Hypertext Transfer Protocol (rules for sharing information on the web)
* WSGI: Web Server Gateway Interface (standard for Python web apps to talk to servers)
* API: Application Programming Interface (a way for different software parts to communicate)
* Technical words with definitions:
* Framework: A pre-made structure to build software on, providing basic tools and patterns.
* Deployment: Making a web application live and accessible to users.
* Scalability: The ability of an application to handle growing amounts of work.
* Template: A reusable format for creating web pages with dynamic content.
* Authentication: The process of verifying a user's identity to log them in.
* Rarely used words with definitions:
* Redundant: Unnecessary or repetitive.
* Asynchronous: Tasks happening out of order or at the same time, not step-by-step.
* Concurrent: Multiple things happening simultaneously.
* Templating: Using templates to create dynamic website content.
* Integration: Combining different software components to work together.
* Software packages with definitions:
* Django: A full-stack Python framework for building complex web applications, with lots of built-in features.
* Flask: A microframework in Python, smaller and more flexible for simple websites or APIs.
* Web2py: Another Python full-stack framework emphasizing ease of use.
* Pyramid: A versatile Python framework, good for both small and large projects.
* Bottle: A lightweight microframework, ideal for tiny web projects.
* Dash: Python framework focused on building analytical web applications.
* CherryPy: A minimalist Python framework known for its speed and simplicity.
* Growler: Python asynchronous framework for handling many connections at once.
* AIOHTTP: Python library for building asynchronous web services and clients.
* Sanic: Python framework built for speed and handling lots of requests quickly.
* Ideas/techniques with definitions:
* Web development: The process of designing and building websites and applications that run on the web.
* Full-stack development: Building both the front-end (what the user sees) and back-end (server and database) parts of a website.
* Microframework development: Building web apps using smaller frameworks with only essential features.
* Asynchronous programming: A style of coding where tasks don't have to wait for each other in a strict order.
* API development: Creating interfaces that allow different software components to interact.
* Keywords: keywords = ['Python', 'web development', 'framework', 'Django', 'Flask', 'full-stack', 'microframework', 'asynchronous', 'API', 'HTTP', 'WSGI']

[Knowledge check: Popular Packages, Libraries and Frameworks | Coursera](https://www.coursera.org/learn/programming-in-python/quiz/D4sbY/knowledge-check-popular-packages-libraries-and-frameworks) 3-21-2024 0610

* Missed one part of a question. Retake 100%

[What is testing? | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/33pF7/what-is-testing) 3-21-2024 0616

* Just a description of testing, no real learning.

[Types of testing | Coursera](https://www.coursera.org/learn/programming-in-python/lecture/ZFo42/types-of-testing)