

SKILLS CHECKLIST

Python Syllabus Checklist

Tick off what you can do, try the task that proves it, and find the course and chapter that teaches it.

1 Chapters | Philip Osztromok | Generated with Claude

Contents

Chapter 1 Python Checklist 2

Python Syllabus Checklist

What you can learn here, and where

✅ Your Python Checklist

This page lists what the Python courses on this site teach, as things you can *do*. Tick an item when you can do it, use the "Where to learn it" column to find the course and chapter, and use the "Prove it" column for a short task to try yourself. Print the PDF to keep a copy that you can tick with a pen.

> **HOW TO USE THIS PAGE**
>
> The sections follow the order of the courses: **Python Fundamentals** (beginner), **Python Intermediate**, **Python Advanced**, the **Beginner Projects** and some **extra practice** lessons. The "Prove it" tasks are short prompts for you to try, not graded exercises. Ticks on the website are not saved when you leave the page, so the printable PDF is the way to keep your progress.

1. Python Fundamentals (beginner)

Nine chapters: the basics of the language.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	install Python, use the REPL and run a script	**Python Fundamentals**, Chapter 1 (Getting Started)	Save a file that prints a greeting and run it from the command line.
☐	use the basic data types and convert between them	**Python Fundamentals**, Chapter 2 (Variables & Basic Data Types)	Turn the string `"42"` into a number and add 8. Print the type of `3.5` and of `True`.
☐	make decisions with operators and `if` / `elif` / `else`	**Python Fundamentals**, Chapter 3 (Operators & Control Flow)	Write a function that returns "fizz" for multiples of 3, "buzz" for multiples of 5 and the number otherwise.
☐	repeat actions with `for` and `while` loops	**Python Fundamentals**, Chapter 4 (Loops)	Print the numbers 1 to 10, skipping 5.
☐	slice and format strings	**Python Fundamentals**, Chapter 5 (Strings & String Formatting)	Print the last word of a sentence in capitals, then print a price to two decimal places with an f-string.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	use lists, tuples and sets	**Python Fundamentals**, Chapter 6 (Lists, Tuples & Sets)	Remove the duplicates from `[3, 1, 3, 2, 1]` and print the rest in order.
☐	use dictionaries	**Python Fundamentals**, Chapter 7 (Dictionaries)	Count how many times each letter appears in `"banana"`.
☐	write functions with parameters and return values	**Python Fundamentals**, Chapter 8 (Functions)	Write `greet(name, greeting="Hello")` and call it with and without the second argument.
☐	import modules and set up a virtual environment	**Python Fundamentals**, Chapter 9 (Modules & Packages)	Create a virtual environment, then import `math` and print the square root of 16.

2. Python Intermediate

Eight chapters: classes, files, errors, testing and working with data.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	write a class with attributes and methods	**Python Intermediate**, Chapter 1 (Object-Oriented Programming)	Write a `Dog` class with a name and a `speak()` method.
☐	use inheritance and the special (dunder) methods	**Python Intermediate**, Chapter 2 (OOP Deep Dive)	Give a `Point` class `__str__` and `__eq__` so two equal points compare equal.
☐	handle errors with `try` / `except`	**Python Intermediate**, Chapter 3 (Exception Handling)	Write a function that turns text into a number, or returns nothing if the text is not a number.
☐	read and write files	**Python Intermediate**, Chapter 4 (File I/O)	Write three lines to a file with `with open(...)` and read them back with `pathlib`.
☐	use comprehensions and generators	**Python Intermediate**, Chapter 5 (Comprehensions & Generators)	Write a generator that yields the squares of 1 to 5, and a list comprehension of the even squares.
☐	write and use decorators	**Python Intermediate**, Chapter 6 (Decorators)	Write a decorator that prints how long a function took.
☐	work with JSON, CSV and dates	**Python Intermediate**, Chapter 7 (Working with Data)	Write a dictionary to JSON and read it back, then print today's date as YYYY-MM-DD.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	test my code with pytest	**Python Intermediate**, Chapter 8 (Testing with pytest)	Write a `test_add` function that asserts `add(2, 3) == 5` and run it with pytest.

3. Python Advanced

Eight chapters: advanced classes, concurrency, typing, packaging and performance.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	use abstract base classes and multiple inheritance	**Python Advanced**, Chapter 1 (Advanced OOP)	Define an abstract `Shape` with an `area()` method and two subclasses; show that `Shape()` cannot be created.
☐	write custom iterators and context managers	**Python Advanced**, Chapter 2 (Iterators & Context Managers Deep Dive)	Write a context manager that prints “enter” and “exit” around a `with` block.
☐	run work in threads and processes, and explain the GIL	**Python Advanced**, Chapter 3 (Concurrency)	Run three `time.sleep(1)` calls in threads and show that the total time is about one second.
☐	write asynchronous code with `async` / `await`	**Python Advanced**, Chapter 4 (Async Python)	Run three coroutines that `await asyncio.sleep(1)` with `asyncio.gather` and show they finish together.
☐	add type hints and check them	**Python Advanced**, Chapter 5 (Type Hints & Static Typing)	Add type hints to a function and to a list variable, then run mypy on the file.
☐	package a project	**Python Advanced**, Chapter 6 (Packaging & Distribution)	Write a minimal `pyproject.toml` with a name, a version and a command-line script entry.
☐	measure and improve performance	**Python Advanced**, Chapter 7 (Performance)	Time two ways of summing 1000 numbers with `timeit`, then profile one with `cProfile`.
☐	combine these skills in a real command-line tool	**Python Advanced**, Chapter 8 (Capstone: Building a Real CLI Tool)	Write a tool with `argparse` that takes a filename and prints its number of lines.

4. Beginner Projects

Eight small projects that put the Fundamentals to work. The prove-it task is the core of each project, not the whole thing.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	build a number guessing game	**Python Projects (Beginner)**, Chapter 1 (Number Guessing Game)	Pick a random number from 1 to 100 and answer "higher", "lower" or "correct" to a guess.
☐	build a simple calculator	**Python Projects (Beginner)**, Chapter 2 (Simple Calculator)	Write a function that adds, subtracts, multiplies and divides, and reports division by zero instead of crashing.
☐	build a password generator	**Python Projects (Beginner)**, Chapter 3 (Password Generator)	Make a 12-character random password from letters and digits with the `secrets` module.
☐	build a to-do list app that saves to a file	**Python Projects (Beginner)**, Chapter 4 (To-Do List CLI App)	Save a list of tasks to a JSON file and load it back.
☐	build a quiz game	**Python Projects (Beginner)**, Chapter 5 (Quiz Game)	Store questions and answers in a dictionary and count how many a player got right.
☐	build a contact book	**Python Projects (Beginner)**, Chapter 6 (Contact Book)	Add a contact and delete another in a JSON file.
☐	build a simple web scraper	**Python Projects (Beginner)**, Chapter 7 (Simple Web Scraper)	Fetch a page with `requests` and list its links with BeautifulSoup.
☐	build an expense tracker	**Python Projects (Beginner)**, Chapter 8 (Capstone: Expense Tracker)	Total a list of (category, amount) expenses by category.

5. Extra practice

The weekly lessons and the Preply lessons are single topic pages, not numbered chapters, so the "Where to learn it" column names the lessons.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	control how output is displayed	**Python weekly lessons**: Displaying Output, and Displaying Output (advanced), with their exercises	Print two words separated by a dash with no newline at the end.
☐	predict truthy and falsy values	**Python weekly lessons**: Boolean Logic, Truthy and Falsy	Predict `bool(x)` for `0`, `""`, `[]` and `"False"`, then check.
☐	choose between `for` and `while`	**Python weekly lessons**: For Loops and Range, and For vs While Loops, with their exercises	Write the same loop over 0, 2, 4, 6, 8 with `for` and with `while`.

	I CAN…	WHERE TO LEARN IT	PROVE IT
☐	avoid and stop infinite loops	**Python weekly lessons**: Infinite Loops	Write a `while True` loop that stops by itself after three passes.
☐	loop over a dictionary	**Python weekly lessons**: Iterating Over a Dictionary with a For Loop, with its exercises	Print each key and value of a dictionary as "key: value".
☐	loop over strings and tuples	**Python weekly lessons**: Iterating Through Strings, and Iterating Through Tuples with a For Loop, with their exercises	Print each letter of a word, then double each number in a tuple.
☐	update a tuple of records	**Python weekly lessons**: Updating a Tuple of Records, and Iterating Tuples, Lists and Dictionaries with a While Loop	Change one record in a tuple of tuples (tuples cannot be edited, so build a new one).
☐	use the starter topics from the Preply lessons	**Python (Preply Lessons)**: Introduction, Variables, Conditionals, Lists and Loops, Lists of Lists, Tuples, Dictionaries	Make a 3 by 3 list of lists and print its middle value.

6. Not yet on this site

Some things are not taught yet, so this checklist does not claim them. This list is based on the chapter topic lines, so some of these may appear inside a chapter.

- Recursion, regular expressions, dataclasses, logging and a tour of the standard library (collections, itertools).
- Debugging with a debugger, and version control.
- Databases from Python, web frameworks and data-science libraries. Courses that use Python for something else (robotics, scripting, web frameworks) are not part of this checklist.

ABOUT ACCURACY

The checklist items come from the chapters' banner and topic lines, and nothing was checked against the official Python documentation. The "Prove it" tasks were each tried with a reference solution on one computer (Python 3.15), except the pytest task, the web scraper task and the mypy part of the type-hints task, which need extra packages that were not installed, and the packaging task, where only the configuration file was checked, not an actual build or upload. If a task does not work for you, check the chapter and the current Python documentation.

KEEP GOING

Revisit a chapter whenever an item on this list is still unticked. The chapters have worked examples you can copy and change.