PYTHON ROADMAP

120 concepts to become a Python developer

8 areas, 15 concepts each. Every concept explained in one line, the order to study each area, and two practice questions to test yourself.

How to use it: one area at a time, tick each concept only when you can explain it out loud in under a minute, then answer the two practice questions without notes.

@chalkesahil · chalkesahil.ai

AREA 01 / 08

Python basics

Study order: Code a little every day until the syntax disappears.

01	**Variables and types**	int, float, str, bool.
02	**Input and output**	print and input.
03	**Conditions**	if, elif, else.
04	**Loops**	for, while, break, continue.
05	**Functions**	def, arguments and return.
06	**Default and keyword args**	Flexible functions.
07	**Scope**	Local vs global.
08	**Strings**	Slicing and methods.
09	**f-strings**	Clean formatting.
10	**Modules**	import and your own files.
11	**pip**	Install packages.
12	**Virtual environments**	One per project.
13	**Error messages**	Read tracebacks.
14	**PEP 8**	Python's style guide.
15	**VS Code setup**	Run and debug easily.

Practice

1. Print the FizzBuzz sequence up to 100.
2. Check whether a string is a palindrome.

AREA 02 / 08

Data structures

Study order: Know which built-in to reach for; that's half of Python.

01	**Lists**	Ordered and changeable.
02	**Tuples**	Ordered and fixed.
03	**Sets**	Unique values.
04	**Dictionaries**	Key-value lookups.
05	**List comprehensions**	Build lists in one line.
06	**Dict comprehensions**	Build dicts in one line.
07	**Slicing**	Parts of lists and strings.
08	**Sorting**	sorted with key functions.
09	**enumerate and zip**	Cleaner loops.
10	**collections.Counter**	Count things fast.
11	**defaultdict**	Dicts with defaults.
12	**deque**	Fast queue.
13	**heapq**	Priority queue.
14	**Time complexity**	Which operations are fast.
15	**Mutability**	Why lists change under you.

Practice

1. Find the most common word in a paragraph.
2. Group a list of names by first letter.

AREA 03 / 08

OOP

Study order: Most real Python codebases are built from classes.

01	**Classes and objects**	Blueprint and instance.
02	**__init__**	Set up a new object.
03	**Instance vs class attributes**	Per object vs shared.
04	**Methods**	Functions inside classes.
05	**Inheritance**	Reuse behaviour.
06	**super()**	Call the parent class.
07	**Polymorphism**	Same method, different classes.
08	**Encapsulation**	Underscore conventions.
09	**Dunder methods**	__str__, __len__, __eq__.
10	**Class and static methods**	Not tied to one object.
11	**Properties**	Getters and setters the Python way.
12	**Abstract classes**	abc module.
13	**Dataclasses**	Less boilerplate.
14	**Composition**	Has-a over is-a.
15	**Design patterns**	Singleton and factory in Python.

Practice

1. Build a BankAccount class with deposit and withdraw.
2. When would you use a dataclass?

AREA 04 / 08

Advanced Python

Study order: These topics come up in every Python interview.

01	**Iterators**	Objects you can loop over.
02	**Generators**	Lazy values with yield.
03	**Decorators**	Wrap functions with extra behaviour.
04	**Context managers**	with blocks and cleanup.
05	**Lambda**	Small anonymous functions.
06	**map, filter, reduce**	Functional helpers.
07	***args and **kwargs**	Flexible arguments.
08	**Type hints**	Clearer code, fewer bugs.
09	**Exceptions**	try, except, finally, raise.
10	**Custom exceptions**	Errors with meaning.
11	**GIL**	Why threads don't run Python in parallel.
12	**Threading**	Good for waiting on I/O.
13	**Multiprocessing**	Use all CPU cores.
14	**asyncio**	Many network calls at once.
15	**Memory and references**	Copy vs deepcopy.

Practice

1. Write a decorator that times a function.
2. Read a 10 GB log file line by line with a generator.

AREA 05 / 08

Files and APIs

Study order: Most Python jobs move data from one place to another.

01	**Reading and writing files**	open and with.
02	**CSV**	csv module and DictReader.
03	**JSON**	Load and dump.
04	**pathlib**	Clean file paths.
05	**Regular expressions**	Find patterns in text.
06	**requests**	Call web APIs.
07	**Status codes and errors**	Handle failed calls.
08	**Pagination**	Fetch all pages.
09	**Authentication**	API keys and tokens.
10	**Web scraping**	BeautifulSoup basics.
11	**Automation scripts**	Rename, move and report.
12	**Scheduling**	Cron or schedule library.
13	**Environment variables**	Keep secrets out of code.
14	**Logging module**	Useful logs instead of prints.
15	**CLI tools**	argparse or Typer.

Practice

1. Pull data from a paginated API and save it as CSV.
2. Rename 500 files based on their dates.

AREA 06 / 08

Django and FastAPI

Study order: Learn one framework deeply; Django and FastAPI are the most hired.

01	**How web apps work**	Request, response, routes.
02	**Django project structure**	Apps, settings, urls.
03	**Django models**	Tables as Python classes.
04	**Django admin**	Free admin panel.
05	**Django views and templates**	Pages from data.
06	**Django REST Framework**	APIs in Django.
07	**FastAPI basics**	Fast, typed APIs.
08	**Path and query params**	Inputs to endpoints.
09	**Pydantic models**	Validate request data.
10	**Dependency injection**	FastAPI's Depends.
11	**Async endpoints**	Many requests at once.
12	**Auth**	JWT login.
13	**Middleware**	Code around every request.
14	**Auto docs**	Swagger UI for free.
15	**Deployment**	Gunicorn, Uvicorn and a server.

Practice

1. Build a FastAPI CRUD API for tasks.
2. Django vs FastAPI: when would you pick each?

AREA 07 / 08

Databases

Study order: Every Python backend talks to a database.

01	**SQL basics**	SELECT, WHERE, JOIN.
02	**PostgreSQL**	The common choice.
03	**sqlite3**	A database in one file.
04	**SQLAlchemy**	Python's main ORM.
05	**Django ORM**	Queries without SQL.
06	**Migrations**	Change tables safely.
07	**Relationships**	One-to-many and many-to-many.
08	**Indexes**	Faster queries.
09	**Transactions**	All-or-nothing writes.
10	**N+1 queries**	The classic slowdown.
11	**select_related**	Fix N+1 in Django.
12	**Connection pooling**	Reuse connections.
13	**Redis with Python**	Caching.
14	**MongoDB with Python**	pymongo.
15	**Backups**	Protect the data.

Practice

1. Design tables for a course platform.
2. A Django page makes 200 queries. Fix it.

AREA 08 / 08

Testing and interviews

Study order: Clean, tested code is what gets you past the first job.

01	**pytest**	Simple tests.
02	**Fixtures**	Reusable test setup.
03	**Mocking**	Fake APIs and databases.
04	**Test coverage**	What's not tested.
05	**Debugging with pdb**	Step through code.
06	**Git workflow**	Branches and pull requests.
07	**Code review**	Read others' code.
08	**Linting**	ruff or flake8.
09	**Formatting**	black.
10	**Docker basics**	Ship the app anywhere.
11	**Python interview questions**	GIL, decorators, generators.
12	**DSA in Python**	Arrays, hashing, two pointers.
13	**Projects**	One API, one automation, one data tool.
14	**Resume**	Projects with results.
15	**AI tools**	Copilot helps, you still own the code.

Practice

1. Write tests for a function that calls an external API.
2. Explain the GIL in simple words.