Yet Another Python Course: Python Developer Roadmap

Course Description

This course, tailored for non-developer professionals, follows the Python Developer Roadmap from roadmap.sh to teach Python programming from scratch. It emphasizes practical skills like automation, data processing, and web development, delivered through interactive Marimo notebooks. The course is divided into basic, intermediate, and advanced modules, with hands-on exercises and projects to build real-world proficiency.

Learning Objectives

By the end of this course, you will be able to:

- Write Python programs using basic syntax, data structures, and control flow.
- Create reusable code with functions, modules, and object-oriented programming.
- Automate tasks using file handling, web scraping, and APIs.
- Manage Python libraries and environments with package managers.
- (Optional) Build simple web applications and perform asynchronous programming or testing.

Course Structure

The course is organized into three main categories, each containing modules aligned with the Python Developer Roadmap. All materials are available in the "roadmap-sh" branch of the course repository.

Basic Modules

These modules cover foundational Python concepts, ideal for beginners.

1. **Basic Syntax**
    - Topics: Comments, print statements, indentation, basic operators.
    - Exercises: Print messages, use comments, perform calculations.
    - Project: Create a program that prints a personalized greeting based on user input.
2. **Variables and Data Types**
    - Topics: Integers, floats, strings, booleans, None.

- Exercises: Create variables, convert types, check types with `isinstance()`.
- Project: Build a program that summarizes user input (name, age, height).

3. **Functions**
   - Topics: Defining functions, parameters, return statements.
   - Exercises: Write functions for squares, positivity checks, string concatenation.
   - Project: Develop a calculator function for basic arithmetic operations.
4. **Built-in Functions**
   - Topics: `len()`, `input()`, `type()`, type conversion functions.
   - Exercises: Use `len()`, convert inputs, check types.
   - Project: Create a program that repeats a user's name a specified number of times.
5. **Conditionals**
   - Topics: If, elif, else statements, comparison operators.
   - Exercises: Check number signs, categorize numbers, validate strings.
   - Project: Write a program that assigns letter grades based on scores.
6. **Lists**
   - Topics: Creating lists, indexing, slicing, list methods.
   - Exercises: Access list items, modify lists, slice subsets.
   - Project: Build a to-do list program for adding and removing tasks.
7. **Tuples**
   - Topics: Creating tuples, immutability, slicing, concatenation.
   - Exercises: Access tuple items, concatenate tuples, test immutability.
   - Project: Calculate the distance between coordinates stored in a tuple.
8. **Sets**
   - Topics: Creating sets, set operations (union, intersection), methods.
   - Exercises: Add items, perform set operations, handle removals.
   - Project: Find common elements between two lists using sets.
9. **Dictionaries**
   - Topics: Key-value pairs, accessing values, dictionary methods.
   - Exercises: Create dictionaries, add pairs, loop through items.
   - Project: Develop a contact manager for storing and displaying contacts.
10. **Regular Expressions (RegEx)**

- Topics: `re` module, patterns, `match()` , `search()` , `findall()` .
- Exercises: Find words, match dates, extract numbers.
- Project: Extract email addresses from a text.

Intermediate Modules

These modules build on the basics, introducing more complex concepts for practical applications.

1. **Data Structures and Algorithms**
   - Topics: Stacks, queues, linear/binary search, bubble sort.
   - Exercises: Implement stacks, write search functions, create queues.
   - Project: Build a task scheduler using a queue.
2. **Iterators**
   - Topics: Iterator protocol, custom iterators, for loops.
   - Exercises: Use iterators, create custom iterators, loop through them.
   - Project: Generate Fibonacci numbers with a custom iterator.
3. **Decorators**
   - Topics: Function decorators, wrapper functions, `@decorator` syntax.
   - Exercises: Create timing decorators, log calls, apply to functions.
   - Project: Log inputs and outputs of a user-input processing function.
4. **Lambda Functions**
   - Topics: Lambda syntax, `map()` , `filter()` , sorting.
   - Exercises: Write lambdas, use with `map()` and `filter()` .
   - Project: Process user inputs (e.g., uppercase conversion) with lambdas.
5. **Object-Oriented Programming (OOP)**
   - Topics: Classes, objects, inheritance, encapsulation.
   - Exercises: Create classes, implement inheritance, use encapsulation.
   - Project: Model a library system for books and users.
6. **Modules**
   - Topics: Importing modules, standard library, custom modules.
   - Exercises: Use `math` , create custom modules, work with `datetime` .
   - Project: Handle file operations with a custom module.
7. **Built-in Methods**

- Topics: String, list, dictionary, set methods.
- Exercises: Convert strings, sort lists, use dictionary `get()` .
- Project: Process user inputs with string and list methods.

8. **Sorting Algorithms**

   - Topics: Bubble sort, selection sort, built-in `sort()` .
   - Exercises: Implement bubble sort, selection sort, compare with `sorted()` .
   - Project: Sort a list of names alphabetically.

Advanced Modules (Optional)

These modules cover advanced topics for learners interested in deeper Python skills. They are optional due to their complexity.

1. **Package Managers**

   - Topics: pip, conda, virtual environments, requirements files.
   - Exercises: Create environments, install packages, generate requirements.
   - Project: Fetch and display API data using `requests` in a virtual environment.

2. **Frameworks**

   - Topics: Flask, Django, routes, REST APIs.
   - Exercises: Create Flask routes, build multi-route apps, explore Django.
   - Project: Build a Flask app for form submission and data display.

3. **List Comprehensions**

   - Topics: Syntax, filtering, nested comprehensions.
   - Exercises: Generate cubes, filter numbers, create matrices.
   - Project: Filter words from user inputs using list comprehensions.

4. **Generator Expressions**

   - Topics: Syntax, memory efficiency, use with functions.
   - Exercises: Yield even numbers, sum squares, compare memory usage.
   - Project: Process a large dataset with generator expressions.

5. **Asynchronous Programming**

   - Topics: `async/await` , `asyncio` , `aiohttp` , event loops.
   - Exercises: Write async functions, use `asyncio.gather()` , fetch data.
   - Project: Fetch data from multiple APIs concurrently with `aiohttp` .

6. **Testing**

- Topics: Unit testing, pytest, assertions, test discovery.
- Exercises: Write tests, create test cases, run with pytest.
- Project: Write a test suite for a calculator function using pytest.

How to Use This Course

1. **Install Marimo**: Follow the instructions at Marimo Documentation.
2. **Clone the Repository**: Access the "roadmap-sh" branch:

```
git clone -b roadmap-sh https://github.com/pesnik/yet-another-python-course.git
```

3. **Run Exercises and Projects**: Each module contains `exercises.py` and `project.py` files. Open them in the Marimo editor:

```
marimo edit basic/basic_syntax/exercises.py
```

   Run as apps (hiding code):

```
marimo run basic/basic_syntax/exercises.py
```

4. **Progress Through Modules**: Start with Basic, move to Intermediate, and explore Advanced modules as desired. Complete exercises and projects to reinforce learning.

Prerequisites

- A computer with internet access.
- Basic familiarity with a text editor or IDE (e.g., VS Code).
- No prior programming experience required.

Assessment

Students will be assessed through the completion of exercises and projects in each module. These hands-on tasks demonstrate proficiency in applying Python concepts to practical scenarios.

Support

For questions or assistance, open an issue in the course repository or contact the instructor at [hasanrakibul.masum@gmail.com].

Additional Resources

- Python Official Documentation
- Marimo Documentation
- GeeksforGeeks Python Tutorial
- Real Python Tutorials
- Roadmap.sh Python Developer Roadmap