Python Roadmap

Python Roadmap

First e Roadmap chusemundhu , Asal nuv python nduku nerchukovali anukuntunavo decide avu ,
Ade andru easy antunaru so nerchukundam ani kadhu , ne end goal enti ? web applications build chyala ? data analyst ? Data Science Engineer ? lekunte enti ? so first end goal nti ani decide ayi language nerchuko
Evi anni nak ndhuk chptunavu ani anukokandi .Just oka salaha , endukante nenu me stage nunde vachanu

RESOURCES

Official Documentation - Meku coding Experience Unte Documentation Follow Avandi
Documentation navalla kadu anukune valu kinda videos chuseyandi
English
Hindi
Django - English

Which one to choose ? Django or Flask ?

Usually Django pedha Applications build cheyadaniki use avutundi , built in authentication & authorization system untundi , adhe flask chinna applications build cheyadaniki use avutundi , authentication kosam third party libraries use cheyali
No matter which resource you pick, set a goal to finish it by a certain date. If you're a beginner, spending 2–3 hours each day, you can finish the course in about 10–11 weeks. Once you've finished the basics, move on to exploring more about Python. Remember, there's a lot to learn, so take your time and have fun exploring Python on your own once you finish the basics.

1. Basics of Python:

- Installation and Setup:
  - Install Python on your system.
  - Set up a virtual environment for project isolation.
- Hello World:
  - Write your first Python program to print "Hello, World!".
- Data Types and Variables:
  - Understand basic data types: int, float, str, bool.
  - Learn about variables and dynamic typing.
- Operators:
  - Explore arithmetic, comparison, logical, and assignment operators.
- Control Flow:
  - Learn about if statements, loops (for, while), and conditional expressions.
- Functions:
  - Define and call functions.
  - Understand function parameters and return values.

2. Data Structures:

- Lists and Tuples:
  - Explore lists and tuples for storing ordered collections.
  - Learn about indexing and slicing.
- Dictionaries:
  - Understand dictionaries for key-value pairs.

Python Roadmap

- Learn about sets and their use cases.
- Explore set operations.

3. Advanced Concepts:

- List Comprehensions:
  - Use list comprehensions for concise and readable code.
  - Understand the syntax and benefits.
- Lambda Functions:
  - Explore lambda functions for anonymous, small-scale operations.
  - Understand the syntax and use cases.
- Generators:
  - Learn about generators for lazy evaluation.
  - Understand the yield statement.

4. Object-Oriented Programming (OOP):

- Classes and Objects:
  - Understand the principles of OOP.
  - Define classes and create objects.
- Inheritance:
  - Learn about inheritance and code reuse.
  - Explore super() and method overriding.
- Encapsulation and Abstraction:
  - Understand encapsulation and abstraction principles.
  - Use properties and methods to hide implementation details.

5. File Handling:

- Reading and Writing Files:
  - Learn how to open, read, and write to files.
  - Explore file modes and handling.
- Working with JSON and CSV:
  - Understand how to work with JSON and CSV data.
  - Use the json **and** csv **modules.**

6. Exception Handling:

- Try-Except Blocks:
  - Handle exceptions using try and except blocks.
  - Learn about the final **block.**
- **Custom Exceptions:**
  - **Create custom exception classes.**
  - **Raise and catch custom exceptions.**

7. Modules and Packages:

- Importing Modules:
  - Learn how to import built-in and external modules.
  - Understand module namespaces.
- Creating and Using Packages:

Python Roadmap

8. Working with APIs:

- HTTP Requests:
  - Use the requests **library to make HTTP requests.**
  - **Understand HTTP methods (GET, POST, etc.).**
- **JSON Parsing:**
  - **Parse JSON data from API responses.**
  - **Understand serialization and deserialization.**

9. Database Connectivity:

- Database Basics:
  - Understand basic database concepts.
  - Learn about relational databases.
- Using SQLite with Python:
  - Connect to SQLite databases using the sqlite3 **module.**
  - **Execute SQL queries and fetch results.**

10. Web Development :

- Flask or Django Framework:
  - Explore web frameworks for Python.
  - Build simple web applications.

Django Roadmap

1. Getting Started:

- Install Django:
  - Use pip to install Django on your system.
  - Set up a virtual environment for your project.
- Create a Django Project:
  - Use the django-admin tool to create a new Django project.
  - Explore the project structure.
- Run the Development Server:
  - Start the development server to see the default Django welcome page.
  - Understand how Django handles requests and responses.

2. Django Models:

- Define Models:
  - Create Django models to represent database tables.
  - Explore different field types (CharField, IntegerField, ForeignKey, etc.).
- Migrations:
  - Learn about database migrations in Django.
  - Use makemigrations and migrate commands.
- Admin Interface:
  - Register models with the Django admin for easy data management.
  - Customize the admin interface.

3. Django Views and Templates:

- Create Views:
  - Define views to handle requests.
  - Understand function-based views and class-based views.
- Templates:

Python Roadmap

- Set up URL patterns to map URLs to views.
- Understand regular expressions in URL patterns.

4. Django Forms:

- Build Forms:
  - Create Django forms for user input.
  - Explore different form fields and widgets.
- Form Handling:
  - Handle form submissions in views.
  - Validate and process form data.
- Class-Based Views with Forms:
  - Combine class-based views with forms for more structured code.

5. Django Authentication:

- User Authentication:
  - Implement user authentication in Django.
  - Use the built-in authentication views and forms.
- User Registration:
  - Create custom views and forms for user registration.
  - Implement email confirmation if needed.

6. Django ORM Queries:

- Retrieve Data:
  - Learn to query the database using Django ORM.
  - Use methods like filter(), get(), and exclude().
- Advanced Queries:
  - Explore more complex queries using Q objects.
  - Understand the use of annotate() and aggregate().
- Query Optimization:
  - Optimize database queries to improve performance.

7. Django REST Framework (Optional):

- Introduction to DRF:
  - Learn about Django REST Framework for building APIs.
  - Explore serializers and views.
- API Views and Authentication:
  - Implement API views and handle authentication.
  - Understand different authentication classes.
- Pagination and Viewsets:
  - Explore pagination options for APIs.
  - Use viewsets for more concise API views.

8. Middleware and Django Settings:

- Django Middleware:
  - Understand the concept of middleware in Django.
  - Create custom middleware for your project.
- Django Settings:
  - Explore Django settings for configuring your project.
  - Use settings to manage various aspects of your application.

- Learn about PyTest for writing unit tests.
- Understand test fixtures and assertions.

DSA Important Topics (Python)

1. Data Structures:

- Arrays:
  - Understanding arrays, indexing, and manipulation.
  - Implementing dynamic arrays and amortized analysis.
- Linked Lists:
  - Singly and doubly linked lists.
  - Operations like insertion, deletion, and traversal.
- Stacks and Queues:
  - Implementing stacks and queues.
  - Solving problems using these structures.
- Hashing:
  - Understanding hash functions and collision resolution.
  - Implementing hash tables and solving problems using hashing.
- Trees:
  - Binary Trees and Binary Search Trees (BST).
  - Tree traversal (in-order, pre-order, post-order).
  - Balancing trees (AVL, Red-Black).
- Heaps:
  - Understanding min and max heaps.
  - Heap operations (insertion, deletion, heapify).
  - Priority queues.
- Graphs:
  - Representing graphs (adjacency matrix, adjacency list).
  - Depth-First Search (DFS) and Breadth-First Search (BFS).
- Trie:
  - Implementing and using trie data structure.

2. Algorithms:

- Searching:
  - Linear search and binary search.
  - Hash-based searching.
- Sorting:
  - Bubble sort, selection sort, insertion sort.
  - Merge sort, quicksort, heapsort.
  - Sorting in Python (using `sorted()` and `sort()`).
- Recursion:
  - Understanding recursion and solving problems recursively.
  - Memoization and dynamic programming.
- Divide and Conquer:
  - Solving problems using the divide-and-conquer strategy.
  - Examples like merge sort and binary search.
- Greedy Algorithms:
  - Solving optimization problems using greedy algorithms.
  - Huffman coding, Dijkstra's algorithm.

- Graph Algorithms:
  - Depth-First Search (DFS) and Breadth-First Search (BFS).
  - Shortest path algorithms (Dijkstra, Bellman-Ford).
  - Minimum Spanning Tree (Prim's, Kruskal's).
- String Algorithms:
  - String manipulation and pattern matching.
  - Knuth-Morris-Pratt (KMP) algorithm.
- Mathematical Algorithms:
  - Euclidean algorithm for GCD.
  - Sieve of Eratosthenes for prime numbers.
  - Binary exponentiation.

3. Python-Specific Topics:

- List Comprehensions:
  - Utilizing concise list comprehensions for iterative operations.
- Dictionaries and Sets:
  - Leveraging Python dictionaries and sets for efficient data manipulation.
- Lambda Functions:
  - Understanding and using lambda functions.
- Iterators and Generators:
  - Implementing iterators and generators for efficient memory usage.
- Collections Module:
  - Exploring specialized data structures in Python's `collections` **module.**
- Functional Programming:
  - Applying functional programming concepts in Python.