Contents

Contents

[Contents 1](#_Toc201267902)

[**🔶 Stage 1: Python Core & Data Structures** 2](#_Toc201267903)

[**📌 Goals:** 2](#_Toc201267904)

[**✅ To-Do:** 2](#_Toc201267905)

[**🎯 Target Duration: 2 weeks** 2](#_Toc201267906)

[**🔶 Stage 2: DSA Preparation (Interview-Focused)** 3](#_Toc201267907)

[**📌 Goals:** 3](#_Toc201267908)

[**✅ Structured Plan:** 3](#_Toc201267909)

[**🎯 Target Duration: 2–3 months (daily 1–2 hrs)** 3](#_Toc201267910)

[**🔶 Stage 3: CS Fundamentals** 3](#_Toc201267911)

[**📌 Goals:** 3](#_Toc201267912)

[**✅ Learn This Much:** 4](#_Toc201267913)

[**🧠 Resources:** 4](#_Toc201267914)

[**🎯 Target Duration: 3–4 weeks** 4](#_Toc201267915)

[**🔶 Stage 4: System Design (LLD + HLD)** 4](#_Toc201267916)

[**📌 Goals:** 4](#_Toc201267917)

[**✅ LLD (Low-Level Design)** 4](#_Toc201267918)

[**✅ HLD (High-Level Design)** 4](#_Toc201267919)

[**🧠 Resources:** 5](#_Toc201267920)

[**🎯 Target Duration: 1 month** 5](#_Toc201267921)

[**🔶 Stage 5: Resume & Project Portfolio** 5](#_Toc201267922)

[**📌 Goals:** 5](#_Toc201267923)

[**✅ Resume Tips:** 5](#_Toc201267924)

[**✅ Project Ideas (If building new ones)** 5](#_Toc201267925)

[**🧠 Resume Review Checklist:** 5](#_Toc201267926)

[**🎯 Target Duration: 1–2 weeks** 5](#_Toc201267927)

[**🔶 Stage 6: Job Hunting & Interviewing** 5](#_Toc201267928)

[**📌 Goals:** 5](#_Toc201267929)

[**✅ Platforms:** 6](#_Toc201267930)

[**✅ Cold Outreach Templates:** 6](#_Toc201267931)

[**✅ Mock Interviews:** 6](#_Toc201267932)

[**🎯 Target Duration: Parallel to other stages (30 mins daily)** 6](#_Toc201267933)

[**🔚 Final Advice:** 6](#_Toc201267934)

# **🔶 Stage 1: Python Core & Data Structures**

**📌 Goals:**

* Build intuition on how Python handles memory and data structures under the hood.
* Understand time and space complexity of built-in operations.

**✅ To-Do:**

1. **Master Built-in Types**
   * list, tuple, set, dict, str
   * Learn operations: slicing, searching, insertion, deletion, sorting
2. **Understand Complexity**
   * Time/space for:
     + list.append(), list.insert(), dict.get(), set.add(), etc.
   * Use: [Big-O cheat sheet](https://www.bigocheatsheet.com/)
3. **Dive into collections module**
   * Learn: defaultdict, Counter, OrderedDict, deque, namedtuple
4. **Useful Standard Libraries**
   * itertools, heapq, bisect, functools, math, operator

**Asymptotic notations**

**Amortized analysis**

**Mutable and non-mutable**

**Dictionaries (Hash Maps)**

* Get/Set operations: O(1) average case, O(n) worst case
* Key lookup and insertion are fundamental for many interview problems
* Understanding hash collision handling is crucial

**Sets**

* Add/Remove/Contains: O(1) average case
* Set operations (union, intersection): O(len(s1) + len(s2))
* Essential for deduplication and membership testing

**Tuples**

* Immutable sequences with O(1) access time
* Memory efficient compared to lists for fixed data

**Time and Space Complexity Analysis**

Master the Big O notation for common operations. Focus on understanding how your code performance scales with input size. For lists with 1,000 times more data: O(1) shows no change, O(log n) shows ~10x slowdown, O(n) shows 1,000x slowdown, and O(n²) shows 1,000,000x slowdown.

**Collections Module Mastery**

The collections module provides specialized container datatypes that are frequently tested in interviews:

**Counter**

* Counting hashable objects with O(n) time complexity for creation
* Most common operations and arithmetic operations on counts

**deque (Double-ended queue)**

* O(1) append/pop from both ends vs O(n) for list operations at the beginning
* Essential for BFS implementations and sliding window problems

**defaultdict**

* Eliminates KeyError exceptions and reduces code complexity
* Particularly useful for grouping and counting problems[7](https://stackoverflow.com/questions/5900578/collections-defaultdict-difference-with-normal-dict)

**namedtuple**

* Provides readable, immutable data structures[8](https://docs.python.org/3/library/collections.html)
* Memory efficient alternative to classes for simple data containers[9](https://imarticus.org/blog/collections-in-python/)

**OrderedDict and ChainMap**

* OrderedDict maintains insertion order (less relevant since Python 3.7)[4](https://www.pickl.ai/blog/python-collections-module/)
* ChainMap creates single view of multiple mappings

**🎯 Target Duration: 2 weeks**

# **🔶 Stage 2: DSA Preparation (Interview-Focused)**

**📌 Goals:**

* Build problem-solving skills using Python
* Learn patterns over memorizing solutions

**✅ Structured Plan:**

1. **Core Topics (1–2 weeks each):**
   * Arrays & Strings
   * Hashing
   * Sliding Window / Two Pointers
   * Linked Lists
   * Trees (BFS, DFS, Binary Trees, BSTs)
   * Recursion & Backtracking
   * Graphs (DFS, BFS, Union-Find)
   * Heaps, Tries, Stacks, Queues
   * Greedy & Sorting
   * Dynamic Programming
2. **Best Practice Platforms**
   * 🧠 NeetCode 150 (Start here)
   * LeetCode: Blind 75, Top Interview
   * InterviewBit (structured)
   * Striver DSA Sheet (for revision)
3. **Python Tips**
   * Use list comprehensions, generators
   * Memorization: @lru\_cache
   * Custom comparators: functools.cmp\_to\_key
   * Write clean, readable code

**🎯 Target Duration: 2–3 months (daily 1–2 hrs)**

# **🔶 Stage 3: CS Fundamentals**

**📌 Goals:**

* Cover core subjects from an interview perspective (not full syllabus)

**✅ Learn This Much:**

1. **Operating Systems**
   * Threads vs processes, memory management, scheduling
2. **Networking**
   * HTTP/HTTPS, DNS, Load Balancing, REST, basics of TCP/IP
3. **DBMS**
   * Normalization, Joins, Indexing, Transactions, SQL vs NoSQL
4. **OOP**
   * Classes, inheritance, polymorphism, abstraction
   * SOLID principles

**🧠 Resources:**

* GFG, OSNotes, Tech Dummies YouTube
* “System Design Primer” GitHub repo

**🎯 Target Duration: 3–4 weeks**

# **🔶 Stage 4: System Design (LLD + HLD)**

**📌 Goals:**

* Prepare for ML + backend system design questions

**✅ LLD (Low-Level Design)**

* Concepts: OOP, UML, SOLID
* Practice designing: Cache, Rate Limiter, Parking Lot, etc.
* Use Python to implement class-based designs

**✅ HLD (High-Level Design)**

* Focus on ML-centric systems:
  + Recommendation Engines
  + Feature Stores
  + Vector DB Pipelines
  + LLM Serving Architectures
* Learn: Load balancing, scalability, database sharding, queues, caching

**🧠 Resources:**

* Gaurav Sen (YouTube), SystemDesign.dev
* “Designing Data-Intensive Applications” by Kleppmann (select chapters)

**🎯 Target Duration: 1 month**

# **🔶 Stage 5: Resume & Project Portfolio**

**📌 Goals:**

* Build a crisp, results-focused resume that showcases AI/LLM strengths

**✅ Resume Tips:**

* Use **STAR format** for each project
* Focus on impact: “Improved inference latency by 30%”
* Add GitHub & LinkedIn links

**✅ Project Ideas (If building new ones)**

* LLM fine-tuning pipeline
* Multi-modal AI tool (e.g., image + text input)
* ML System with real-time inference (use FastAPI + ONNX)

**🧠 Resume Review Checklist:**

* One page only
* Tailored for AI/ML jobs
* Clear sectioning: Summary, Skills, Projects, Experience

**🎯 Target Duration: 1–2 weeks**

# **🔶 Stage 6: Job Hunting & Interviewing**

**📌 Goals:**

* Build network + pipelines for getting interviews

**✅ Platforms:**

* **LinkedIn**, **Wellfound (AngelList)**, **Triplebyte**
* **Levels.fyi**, **Hired**, **AI Exchange**, **Uplink**, **Turing**

**✅ Cold Outreach Templates:**

text

CopyEdit

Hi [Name], I'm an AI/ML engineer with hands-on LLM experience and currently preparing for FAANG interviews. I came across your work at [Company] and would love to connect and possibly get advice or referral for [role]. Appreciate your time!

**✅ Mock Interviews:**

* Try **Pramp**, **Interviewing.io**, or paid FAANG mocks on **Experty** or **IGotAnOffer**

**🎯 Target Duration: Parallel to other stages (30 mins daily)**

# **🔚 Final Advice:**

* Focus on consistency (daily 1–2 hours is enough)
* Mix DSA with System Design and AI projects weekly
* Don't wait for 100% prep — start applying after DSA + resume are 70% ready