Detailed Roadmap: Python for Digital Design Verification Scripting

Python is becoming increasingly essential in digital design verification, especially for automating repetitive tasks, running regressions, parsing logs, and analyzing coverage. This roadmap will help you move from beginner to project-ready Python usage tailored specifically for verification roles in companies like Intel, NVIDIA, TI, or Qualcomm.

# Step 1: Learn Core Python – Verification-Focused

Skip general-purpose Python topics like GUI apps or web dev. Focus only on parts used in EDA and chip design workflows.  
Key areas:  
- File handling: `open`, `with`, `os.path`  
- CLI tools: `sys.argv`, `argparse`  
- Shell integration: `subprocess.run`, `os.system`  
- Pattern matching: `re` for parsing logs  
- Data processing: `json`, `csv`, `yaml`  
- Collections: `defaultdict`, `Counter`, `namedtuple`

Resources:

• Python Official Docs – https://docs.python.org/3/

• W3Schools Python – https://www.w3schools.com/python/

• Automate the Boring Stuff with Python – https://automatetheboringstuff.com/

# Step 2: Focus on Verification Use Cases

Don’t waste time learning everything. Instead, directly implement:  
✔ Testbench control scripts  
✔ Simulator automation (run testcases, collect logs)  
✔ Log analyzers (PASS/FAIL extraction, error summaries)  
✔ Coverage report readers  
✔ Waiver file generators  
✔ FSDB/SAIF analysis tools (wave extraction, toggling reports)

# Step 3: Build Practical Tools You Can Use & Showcase

Start small:  
- Regression runner script with YAML input  
- Log parser to extract FAIL/ERROR messages from simulation outputs  
- Coverage merger from multiple JSON reports  
  
Then grow:  
- HTML summary dashboards (with Python + Jinja2)  
- Visualization scripts (pandas + matplotlib/seaborn)  
- Scripts that generate or modify SystemVerilog code or testbenches

# Step 4: Integrate Python into Real DV Workflows

This is where you gain the edge:  
- Use Python to wrap simulators (e.g., `vcs`, `xrun`, `vsim`)  
- Use it to auto-trigger regressions from Jenkins or cron  
- Parse coverage XMLs and generate summaries  
- Add script hooks in UVM testbench (e.g., to post-process logs or generate reports)  
- Interface with waveform viewers (e.g., Verdi/PyVerdi)

# Step 5: Document and Share on GitHub

Keep your code clean, documented, and version-controlled. Create:  
- A README with use cases and setup instructions  
- Example input/output files  
- Screenshots or sample log outputs  
- A simple license (MIT)

Example GitHub project: https://github.com/<your-username>/dv-python-utils

# Step 6: Prepare for DV Interviews with Scripts

Many interviews test automation skills. You can practice:  
- Writing a script to grep failures from log directories  
- Generating 100 testcases from templates  
- Counting toggle rates in VCD files

⚠ Tip: Focus less on syntax and more on solving \*real problems\*. Interviewers and leads value automation and initiative over just Python syntax memorization.

# 🧠 Problem Solving Practice: Python for Verification

Modern verification roles increasingly emphasize your ability to automate and problem-solve using Python, especially with AI-assisted workflows and smart tooling. Below are curated problems that mimic real-world scenarios in digital verification environments.

## 📂 File & Log Handling Problems

1. Write a Python script that scans all log files in a directory and extracts lines containing 'ERROR', 'FAIL', or 'FATAL'.  
2. Implement a script that renames all waveform dump files with a timestamp suffix.  
3. Create a tool that compares two simulation logs and highlights differences in test results.

## 📊 Coverage & Report Analysis Problems

1. Parse multiple JSON or XML coverage reports and merge them into a master coverage summary.  
2. Create a bar chart of coverage percentage per module using `matplotlib`.  
3. Build a script that extracts top 5 unhit functional coverpoints from a coverage report.

## ⚙️ Automation & Regression Management Problems

1. Implement a YAML-driven regression manager that runs testcases in parallel and outputs a summary table.  
2. Add email notification to your regression script if more than 3 tests fail.  
3. Simulate UVM test control via Python: generate a list of test names with unique seeds and pass them to a simulator.

## 📈 Waveform & FSDB Analysis Problems

1. Write a script that reads signal transitions from a VCD file and finds the signal with highest toggling.  
2. Automate the generation of signal trace plots for key nets from a VCD dump using `PyVerdi` or `pyvcd`.  
3. Extract reset pulse duration from an FSDB or VCD file for a specified signal.

## 🧪 Bonus: UVM & Python Integration Challenges

1. Write a script to auto-generate UVM testcase stubs from a list of test names in Excel.  
2. Create a tool that analyzes `uvm\_info` messages and categorizes them into INFO, WARNING, ERROR buckets.  
3. Build a testbench file tree visualizer using Python's `os.walk()` and `graphviz`.

💡 Pro Tip: Store your solutions with sample logs/configs on GitHub. Recruiters will be impressed not just by your syntax—but your \*\*tool-building mindset\*\*.