

Python developer (fresher)

Company: [Company name]

Location: [Location] | Work mode: [Work mode]

Employment type: Full-time | CTC: [CTC for the full-time role]

Joining: [Joining date] | Apply by: [Apply by]

About [Company name]

[About your company]

About the role

[Company name] is looking for a fresher Python developer to join the [Team] team. You will write and test Python code for backend features, internal scripts and integrations with other systems, starting with small tasks that a senior developer reviews. You will learn our framework, database and release process on real work. You will fit well if you like clear code, enjoy automating repetitive tasks and want to find out why a program misbehaves.

What you will do

- Write Python functions, classes and modules for backend features, following PEP 8 and the team's code style
- Build and change REST API endpoints in Django, Flask or FastAPI, including routes, input validation and serialisation, with review
- Read and write data through the ORM, and write plain SQL for reports and one-off data fixes
- Write scripts that automate repeat work, such as cleaning CSV or Excel files, generating daily reports and sending scheduled summaries
- Call third-party APIs with the requests library, handling timeouts, error responses and rate limits, and convert the JSON into the format the application needs
- Handle bad input and failures with proper exceptions and logging, so a script reports the problem instead of stopping halfway
- Write unit tests with pytest or unittest, and add a test that proves each bug fix
- Maintain virtual environments and the requirements file so the project runs the same way on every machine
- Debug issues from application logs and stack traces, and note the cause in the ticket
- Document functions and endpoints with docstrings, a README and the framework's API docs page

What we are looking for

- Core Python: data types, lists, dictionaries and sets, comprehensions, functions, classes and inheritance, exceptions and file handling
- At least one small application or API built with Django, Flask or FastAPI
- SQL basics (joins and grouping) on MySQL or PostgreSQL, and some use of an ORM
- Understanding of REST APIs and JSON, and use of the requests library or similar
- Comfort with pip, virtual environments and Git
- Ability to read a stack trace and work out which line and which value caused the error
- A Python project you can run and explain, with the code in a repository or shown in a walkthrough if the work is private

Good to have

- Pandas or NumPy for cleaning and reshaping data
- Writing tests with pytest
- Type hints and basic async and await
- Linux command line and basic Docker
- Web scraping with BeautifulSoup or Scrapy
- Exposure to task queues such as Celery

Who can apply

Final-year students and graduates of B.E./B.Tech in computer science, IT or related streams fit well, as do BCA, B.Sc (computer science, IT, maths, statistics or electronics), MCA and M.Sc candidates. Candidates from other streams who learnt Python through structured courses and projects can also apply.

How to apply

Apply through this job post by [\[Apply by\]](#). Shortlisted candidates will hear from us about the next steps.

Python developer intern

Company: [Company name]

Location: [Location] | Work mode: [Work mode]

Employment type: Internship | Duration: [Internship duration] | Stipend: [Stipend for the internship]

Joining: [Joining date] | Apply by: [Apply by]

About [Company name]

[About your company]

About the role

[Company name] is looking for a Python intern to work with a mentor on real scripts and backend tasks. You will write small tools, fix bugs, add tests and see how a Python service is built, tested and deployed. At the end you will present a working script or API that you built.

What you will do

- Write a script that cleans and merges CSV files and reports the rows it could not process
- Build one small API endpoint with your mentor and write tests for it
- Write a function that calls a third-party API and handles failures properly
- Fix small bugs from the issue list and add a test for each fix
- Add docstrings and a README so the next person can run your code
- Present the script or endpoint you built to the team

What we are looking for

- Pursuing a degree in computer science, IT or a related subject, or recently graduated
- Comfortable with loops, functions, lists, dictionaries and reading and writing files in Python
- Has run a Flask or Django application, even from a tutorial, and can say what each file does
- Has used pip and a virtual environment
- Can share or describe at least one Python project, even if the code is private

Who can apply

Final-year students and graduates of B.E./B.Tech in computer science, IT or related streams fit well, as do BCA, B.Sc (computer science, IT, maths, statistics or electronics), MCA and M.Sc candidates. Candidates from other streams who learnt Python through structured courses and projects can also apply.

How to apply

Apply through this job post by [Apply by]. Shortlisted candidates will hear from us about the next steps.

Screening questions to add to your job post

1. **A function receives a list and changes it inside the function. Does the caller's list change, and why?**
Tests understanding of mutable objects and references, a common source of Python bugs. A confident one-line answer is enough.
2. **A script reads a CSV file and crashes on one bad row. How would you change it so the other rows still get processed?** Checks exception handling and thinking about messy real data. Look for try and except, a log of the bad row and a summary at the end.
3. **In a project using Django, Flask or FastAPI, how does one request reach your code and become a response?** Look for a clear explanation of routing, input handling, application logic and the response. Discuss database access only if the project uses it.
4. **What is the difference between a list and a tuple, and when did you choose each in your own code?**
A basic question made useful by asking for a real example. Memorised answers rarely have one.
5. **Which libraries outside the standard library have you used, and for what?** Reveals practical exposure. Look for requests, pandas, a web framework or testing tools, and a description of the task each one solved.
6. **How would you call an API that is sometimes slow or fails?** Look for timeouts, limited retries, status code checks and logging, which are everyday needs in integration work.