The Python Workflow: From Code to GitHub

AiCE Software Engineering Foundations — Python Programming

You now know how to write Python code, run it, and read what happens when it goes wrong. This closing lesson ties those pieces into a single repeatable cycle — the exact working pattern you'll follow for the rest of this program — and shows how to make pushing your work to GitHub a consistent habit rather than an afterthought.

1. The Full Working Cycle

Every lesson in this program follows the same pattern. Internalize it until it becomes automatic:

Step	Action	Command
1	Create or open a Python file in Vim	vim filename.py
2	Switch to Insert Mode	Press i
3	Write your Python code	Type the code
4	Return to Command Mode	Press Escape
5	Save and quit	:wq
6	Run the Python file	python3 filename.py
7	Review the output	Read the terminal
8	Push your work to GitHub	git add . → git commit → git push

This cycle — write, run, observe, push — is the cycle of a professional developer. The terminal is your canvas, Python is your language, and GitHub is your archive.

2. Pushing Your Python Work to GitHub

At natural checkpoints — typically once you've written and tested a small batch of files — you push your work to GitHub. This keeps your code backed up, tracked, and visible as part of your developer portfolio.

The complete push sequence, once you're inside your working repository directory, is always the same three commands:

```
git add .
git commit -m "Add hello.py — first Python program"
git push
```

After pushing, open your browser, navigate to your aice-python-beginners repository on GitHub, and refresh the page. You'll see the file listed there with the exact code you wrote.

The same sequence scales naturally as your work grows. After writing several files in a single session — say, greetings.py, facts.py, numbers.py, and compare.py — you can stage and push all of them together with one commit that describes the batch:

```
git add .
git commit -m "Add practice scripts: greetings, facts,
numbers, compare"
git push
```

> **Your GitHub Is Your Reputation**
> *In the African tech industry — and globally — a GitHub profile with a consistent history of meaningful commits speaks louder than a CV. Every file you push in this program is an investment in your professional reputation. Start building that reputation from lesson one, and keep the story consistent.*

3. Writing Commit Messages That Mean Something

A commit message is a short, present-tense description of what the commit adds or changes. "Add hello.py — first Python program" tells a future reader exactly what happened without needing to open the file. As your commits accumulate, this habit turns your repository into a readable timeline of your own progress — rather than a folder of unlabeled changes.

Key Takeaways

The eight-step cycle — open the file, insert, write, escape, save, run, review, push — is the loop you'll repeat for every piece of Python code in this program, and eventually for every professional project after it. Push regularly, write commit messages that describe what actually changed, and treat your GitHub history as a running record of your growth as a developer. With your environment, your error-reading instincts, and your workflow all in place, you're ready to move on to the real substance of the language: variables and data types, covered next.