Python Learning Path: Project-Based Approach

Prepared by Jack Massey

Learning Strategy

The goal is to shift from passive learning to active coding. Each task builds on the previous one and introduces key Python fundamentals step-by-step. The learner is expected to code, test, and reflect on their work.

Step-by-Step Progression

- **Mini ATM — Deposit and Withdraw (from scratch)**
  - Ask the user what they want to do: Deposit / Withdraw / Quit.
  - Update the balance according to the action.
  - Print the current balance after each action.
  - Loop until user quits.
- **Add Transaction History**
  - Store each transaction in a list as a string, e.g. `"Deposit: +50€"`.
  - When the user quits, print the list of transactions.
- **Vending Machine**
  - Show a menu of items and their prices.
  - Ask the user to select one and enter their money.
  - If enough: print a success message and return change.
  - If not: print an error message.

Coding Template for Task 1

The following scaffold can help the learner get started:

Listing 1: Basic ATM Structure

```python
balance = 0

while True:
    print("What do you want to do?")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Quit")
    choice = input("> ")

    if choice == "1":
        # TODO: Ask for amount and add to balance
        pass

    elif choice == "2":
        # TODO: Ask for amount, check if enough, subtract
        pass

    elif choice == "3":
        print("Goodbye!")
        break

    else:
        print("Invalid option.")

    # TODO: Print current balance
```

Guiding Questions

Encourage Thinking

- What information does your program need to remember?
- What kind of input do you expect from the user?
- What conditions should trigger each part of the logic?
- How can you organize repeated actions inside a loop?
- What should happen if the user makes a mistake?

Next Steps

Once Task 1 and Task 2 are mastered, gradually introduce:

- **Saving data to a file** (with `open()`, `write()`, `readlines()`)
- **Datetime stamps** in transaction history
- **Modular functions** like `deposit()`, `withdraw()`
- (Later) object-oriented structure with a simple `ATM` class