Exploring Python: A Practical Introduction Through a Mini Programming Project

Jacob T. Kuchar

Post University

CIS216: Programming Principles

Professor Angela Scipio

August 10th, 2025

**Introduction**

For this project, I chose Python as my programming language. Python is one of the most popular programming languages in the world. Python is known for being easy to read, write, and learn. It is widely used in web development, automation, data science, artificial intelligence, and more. In this paper, I will discuss why I chose Python, describe its main features, and walk through a small Python program I created for this assignment. The program includes variables, a loop, a calculation, and the ability to write to a file. I have also included comments within the code to explain exactly how it works.

**Why I Chose Python**

I decided to use Python because it is straight forward and has a clean, readable syntax. This makes it easier to focus on solving problems instead of memorizing complicated rules. It is also highly versatile, which means it can be used for small scripts or large applicants. Another reason for my choice is Python’s strong community support and large collection of libraries. If I need to perform a task, chances are there’s already a library that can help me. This flexibility makes it ideal for both beginners and professionals.

**My Python Program**

The program I wrote is a simple tool that;

1. Asks the user for their name.
2. Asks how many hours they worked and their hourly wage.
3. Calculates their total pay.
4. Displays the result and saves it to a file.

Below is the code with comments explaining how each part works.

This program calculates total pay based on hours worked and hourly rate,

then saves the result to a text file.

**Step 1**: Create two variables to store user input

name = input("Enter your name: ") # Stores the user's name hours\_worked = float(input("Enter hours worked: ")) # Stores hours as a number hourly\_rate = float(input("Enter your hourly rate: ")) # Stores pay rate

**Step 2**: Perform a calculation to determine total pay

total\_pay = hours\_worked \* hourly\_rate # Multiplies hours by rate

**Step 3**: Use a loop to confirm the result before saving

while True: print(f"{name}, your total pay is: ${total\_pay:.2f}") confirm = input("Is this correct? (yes/no): ").lower() if confirm == "yes": break # Exit the loop if the user confirms else: # Allow the user to re-enter their data hours\_worked = float(input("Enter hours worked: ")) hourly\_rate = float(input("Enter your hourly rate: ")) total\_pay = hours\_worked \* hourly\_rate # Recalculate total pay

**Step 4**: Write the result to a file

with open("pay\_record.txt", "w") as file: file.write(f"Employee: {name}\n") file.write(f"Total Pay: ${total\_pay:.2f}\n")

**Step 5**: Let the user know the process is complete

print("Your pay record has been saved to 'pay\_record.txt'.")

**How the Program Meets the Requirements**

The program uses at least two variables: name, hours worked, and hourly rate. These store the user’s input so the program can use the information later in calculations. The line “total\_pay = hours\_worked \* hourly\_rate” multiplies the number of hours worked by the hourly rate to determine the total pay. The “while True:” loops allow the program to keep asking for confirmation until the users says”yes.” if the user says “no.” they can re-enter their data. The “with open(“pay\_record.txt”, “w”) as file:” block saves the users name and total pay into a text file. The “w” mode ensures the file is overwritten each time the program runs. Every step in the code has comments explaining what it does. This makes the program easier to understand for someone reading it for the first time.

**Why Python is a Good Fit**

Python is the ideal language for this project because it lets me accomplish all of these tasks with simple, clear syntax. Variables are easy to create, loops are straightforward, and writing to files takes only one or two lines. In other languages, these same tasks might require more complex code or additional setup. Pythons' simplicity makes it a great tool for learning programming concepts while still being powerful enough for real-world use.

**Conclusion**

This assignment allowed me to practice using Python to create a functional mini program while also reinforcing core programming concepts such as variables, loops, calculations, and file writing. Python's simplicity and flexibility made the process smooth, and its built-in features handled everything I needed without requiring extra libraries. I strongly believe Python is one of the best languages to start with because it is both beginner-friendly and capable of handling advanced projects. Completing this project has increased my confidence in writing Python code and understanding how these fundamental programming elements work together.

**References**

*Welcome to Python.org*. Python.org. (n.d.). <https://www.python.org/>

*The python language reference*. Python documentation. (n.d.). <https://docs.python.org/3/reference/index.html>

GeeksforGeeks. (2025a, July 23). *5 reasons why python is good for Beginners*. <https://www.geeksforgeeks.org/blogs/reasons-why-python-is-good-for-beginners/>