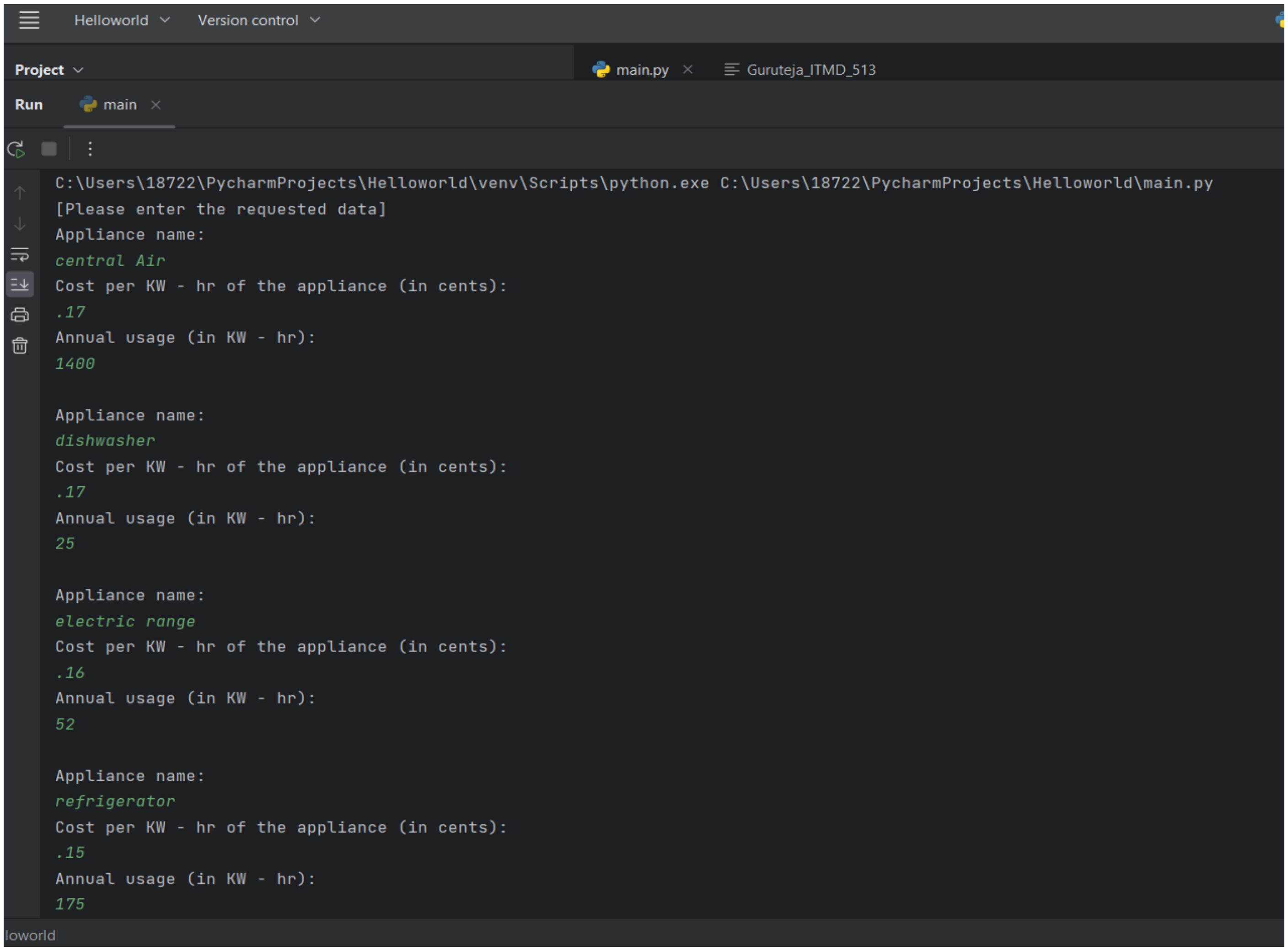
GURUTEJA KANDERI

ITMD-513-LAB_01

A20526883
06/09/2023

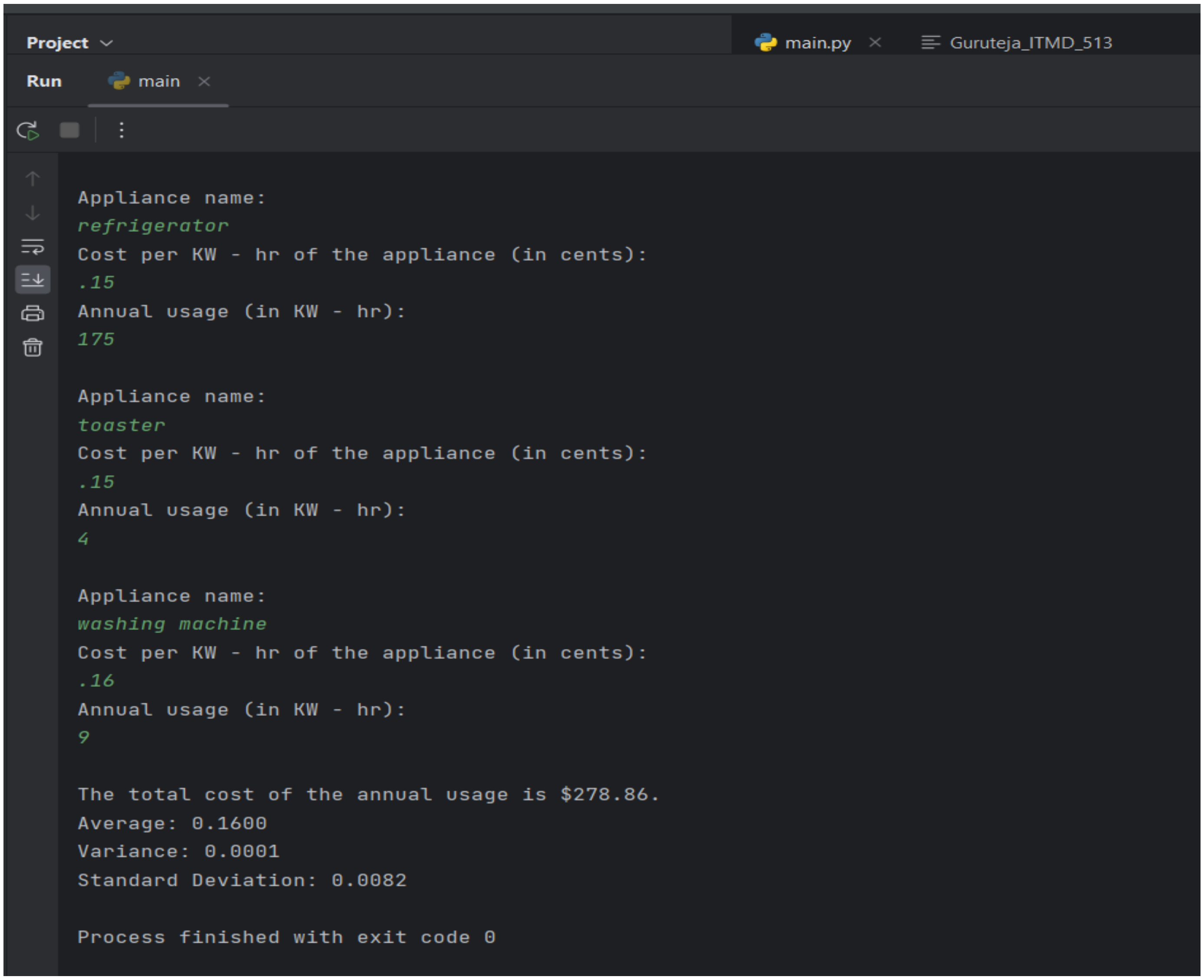
OUTPUT:



```
C:\Users\18722\PycharmProjects\Helloworld\venv\Scripts\python.exe C:\Users\18722\PycharmProjects\Helloworld\main.py
[Please enter the requested data]
Appliance name:
central Air
Cost per KW - hr of the appliance (in cents):
.17
Annual usage (in KW - hr):
1400

Appliance name:
dishwasher
Cost per KW - hr of the appliance (in cents):
.17
Annual usage (in KW - hr):
25

Appliance name:
electric range
Cost per KW - hr of the appliance (in cents):
.16
Annual usage (in KW - hr):
52

Appliance name:
refrigerator
Cost per KW - hr of the appliance (in cents):
.15
Annual usage (in KW - hr):
175
```

GURUTEJA KANDERI
A20526883
06/09/2023

ITMD-513-LAB_01



```
Appliance name:
refrigerator
Cost per KW - hr of the appliance (in cents):
.15
Annual usage (in KW - hr):
175

Appliance name:
toaster
Cost per KW - hr of the appliance (in cents):
.15
Annual usage (in KW - hr):
4

Appliance name:
washing machine
Cost per KW - hr of the appliance (in cents):
.16
Annual usage (in KW - hr):
9

The total cost of the annual usage is $278.86.
Average: 0.1600
Variance: 0.0001
Standard Deviation: 0.0082

Process finished with exit code 0
```

Program source Code:

```
# Local variable declarations
total_cost = 0.0  # Declare variable as a float type to accumulate total
charges
num_appliances = 6  # Number of appliances to calculate

# Declare lists to store appliance details
appliance_names = []
cost_per_kwh = []
annual_usage = []
```

GURUTEJA KANDERI **ITMD-513-LAB_01**

A20526883
06/09/2023

```
# Prompt the user to enter the requested data for each appliance
print("[Please enter the requested data]")

for i in range(num_appliances):
    print("Appliance name:")
    appliance_name = input()
    appliance_names.append(appliance_name)

    print("Cost per kWh of the appliance (in cents):")
    cost = float(input())
    cost_per_kwh.append(cost)

    print("Annual usage (in kWh):")
    usage = float(input())
    annual_usage.append(usage)

    # Calculate the annual cost for the current appliance
    annual_cost = cost * usage
    total_cost += annual_cost

    print()  # Print an empty line for better readability

# Calculate the average cost
average_cost = sum(cost_per_kwh) / num_appliances

# Calculate the variance
variance = sum((average_cost - cost) ** 2 for cost in cost_per_kwh) /
num_appliances

# Calculate the standard deviation
standard_deviation = variance ** 0.5

# Display the total cost of the annual usage and the additional stats
print("The total cost of the annual usage is $%.2f." % total_cost)
print("Average cost per kWh: %.4f" % average_cost)
print("Variance: %.4f" % variance)
print("Standard Deviation: %.4f" % standard_deviation)
```

(1) What is meant by Sequential Program Control?

Sequential Program Control means executing the program line by line from the first line of program to bottom line. It follows the linear order ensuring the predictable flow of control. Sequential control can be combined with other control structures like conditionals and loops to create more complex program logic when needed.

(2) Without using selection control or repetitive control, how would you modify the program to account for a coupon, for a new energy - saving appliance, that the program user can implement to lower the total cost?

- Request the user to enter the total cost of the energy-saving appliance, as mentioned.
- Inquire whether the user possesses a valuable coupon to avail additional discounts.
- If the user indeed possesses a coupon, courteously ask them to provide the coupon amount.
- Diligently subtract the coupon amount from the original total cost of the appliance.
- Finally, graciously present the updated total cost, embracing the applied coupon's benevolent impact.

(3) What is the purpose of adding comment statements?

- Documentation: They explain the code's purpose, behavior, and usage, serving as helpful documentation for developers like us.
- Readability: Comments enhance code comprehension by providing context and clarifications, making it easier for us to understand and maintain the code.
- Algorithmic Explanation: We can use comments to give a high-level overview or detailed explanations of complex algorithms or logic, helping us follow the code's execution flow more effectively.
- Cross-Referencing: Comments can include references to related code sections or external resources, allowing us to navigate between different parts of the codebase and find additional information conveniently.
- Code Review: When reviewing code, we can use comments to address specific points or questions, encouraging discussion and providing guidance for improvements or potential issues.
- Debugging: During the debugging process, comments can assist us by temporarily disabling code or offering helpful guidance to identify and resolve issues.
- Collaboration: Comments facilitate effective communication and collaboration among team members, enabling us to share knowledge and work together efficiently.
- History: Comments also serve as a record of code changes and decisions over time, giving us insights into the codebase's evolution and helping us understand past modifications.

(4) What is the function of the interpreter?

- Code Execution: As an interpreter, it executes your code directly, line by line, converting it into machine instructions on the fly during runtime.

- Translation: It performs real-time translation of your high-level code into machine code or bytecode, ensuring efficient execution of your program.
- Rapid Development and Testing: With an interpreter, you can experience faster development and testing cycles since code changes are immediately observed without the need for separate compilation steps. This boosts your productivity.
- Debugging and Error Reporting: The interpreter provides you with detailed error messages and robust debugging capabilities, helping you identify and resolve issues swiftly during the development process.
- Portability: Enjoy the advantage of portability as an interpreter allows your code to run on different platforms. Its flexibility enables compatibility across various operating systems and architectures, making it convenient for your specific needs.

(5) Why is it important to test your program?

- Identify and Resolve Issues: Testing helps uncover errors, bugs, and unexpected behavior in the program, allowing developers to address them before deployment and ensure the program functions correctly.
- Enhance Reliability: Thorough testing boosts the reliability of the program by validating its behavior in different scenarios, ensuring consistent performance and reducing the risk of failures or crashes.
- Improve Quality: Testing plays a crucial role in improving the overall quality of the program by identifying areas that require optimization or refinement, leading to a higher-quality end product.
- Ensure User Satisfaction: Testing validates that the program meets user expectations and requirements, ensuring it behaves as intended and provides a positive user experience, leading to increased user satisfaction.
- Cost and Time Efficiency: Detecting and resolving issues during testing is more cost-effective and time-efficient compared to dealing with them after deployment. Testing early in the development cycle helps prevent extensive troubleshooting and rework, saving time and resources.