Assignment Description  
1. Readme

**Part 1**:  
  
This program defines a Person class and a Customer subclass. It demonstrates the use of the Customer class by creating an instance and displaying its attributes.  
  
**Part 2**:  
  
This program defines an Employee class and a ProductionWorker subclass. It creates an instance of ProductionWorker, prompts the user for input, and displays the stored information using accessor methods.  
  
  
  
2. Source Code of All Files

Part 1:  
  
"""

Author: Paul Sommers

Date written: 12/5/2024

Assignment: Module 07 Programming Assignment 1

Short Desc: This program defines a Person class and a Customer subclass. It demonstrates the use of the Customer class by creating an instance and displaying its attributes.

"""

# Define the Person class

class Person:

def \_\_init\_\_(self, name, address, telephone):

self.name = name

self.address = address

self.telephone = telephone

# Define the Customer class, a subclass of Person

class Customer(Person):

def \_\_init\_\_(self, name, address, telephone, customerNumber, mailingList):

# Call the superclass (Person) constructor

super().\_\_init\_\_(name, address, telephone)

self.customerNumber = customerNumber

self.mailingList = mailingList

# Example customer class

def main():

# Create an instance of the Customer class

customer = Customer(

name="John Doe",

address="123 Main St, Anytown, USA",

telephone="555-1234",

customerNumber=1001,

mailingList=True

)

# Display the customer's details

print("Customer Details:")

print(f"Name: {customer.name}")

print(f"Address: {customer.address}")

print(f"Telephone: {customer.telephone}")

print(f"Customer Number: {customer.customerNumber}")

print(f"On Mailing List: {'Yes' if customer.mailingList else 'No'}")

# Run the program

main()

Part 2:

"""

Author: Paul Sommers

Date written: 12/5/2024

Assignment: Module 07 Programming Assignment 2

Short Desc: This program defines an Employee class and a ProductionWorker subclass. It creates an instance of ProductionWorker,

prompts the user for input, and displays the stored information using accessor methods.

"""

# Define the Employee class

class Employee:

def \_\_init\_\_(self, name, number):

self.\_\_name = name

self.\_\_number = number

# Accessor (getter) methods

def getName(self):

return self.\_\_name

def getNumber(self):

return self.\_\_number

# Mutator (setter) methods

def setName(self, name):

self.\_\_name = name

def setNumber(self, number):

self.\_\_number = number

# Define the ProductionWorker class, a subclass of Employee

class ProductionWorker(Employee):

def \_\_init\_\_(self, name, number, payRate, shift):

# Call the superclass (Employee) constructor

super().\_\_init\_\_(name, number)

self.\_\_payRate = payRate

self.\_\_shift = shift

# Accessor (getter) methods

def getPayRate(self):

return self.\_\_payRate

def getShift(self):

return self.\_\_shift

# Mutator (setter) methods

def setPayRate(self, payRate):

self.\_\_payRate = payRate

def setShift(self, shift):

self.\_\_shift = shift

# Main program

def main():

# Prompt the user for ProductionWorker details

name = input("Enter employee name: ")

number = input("Enter employee number: ")

payRate = float(input("Enter hourly pay: "))

shift = int(input("Enter shift number (1 for day, 2 for night): "))

# Create a ProductionWorker

worker = ProductionWorker(name, number, payRate, shift)

# Display the details

print("\nEmployee Details:")

print(f"Name: {worker.getName()}")

print(f"Employee Number: {worker.getNumber()}")

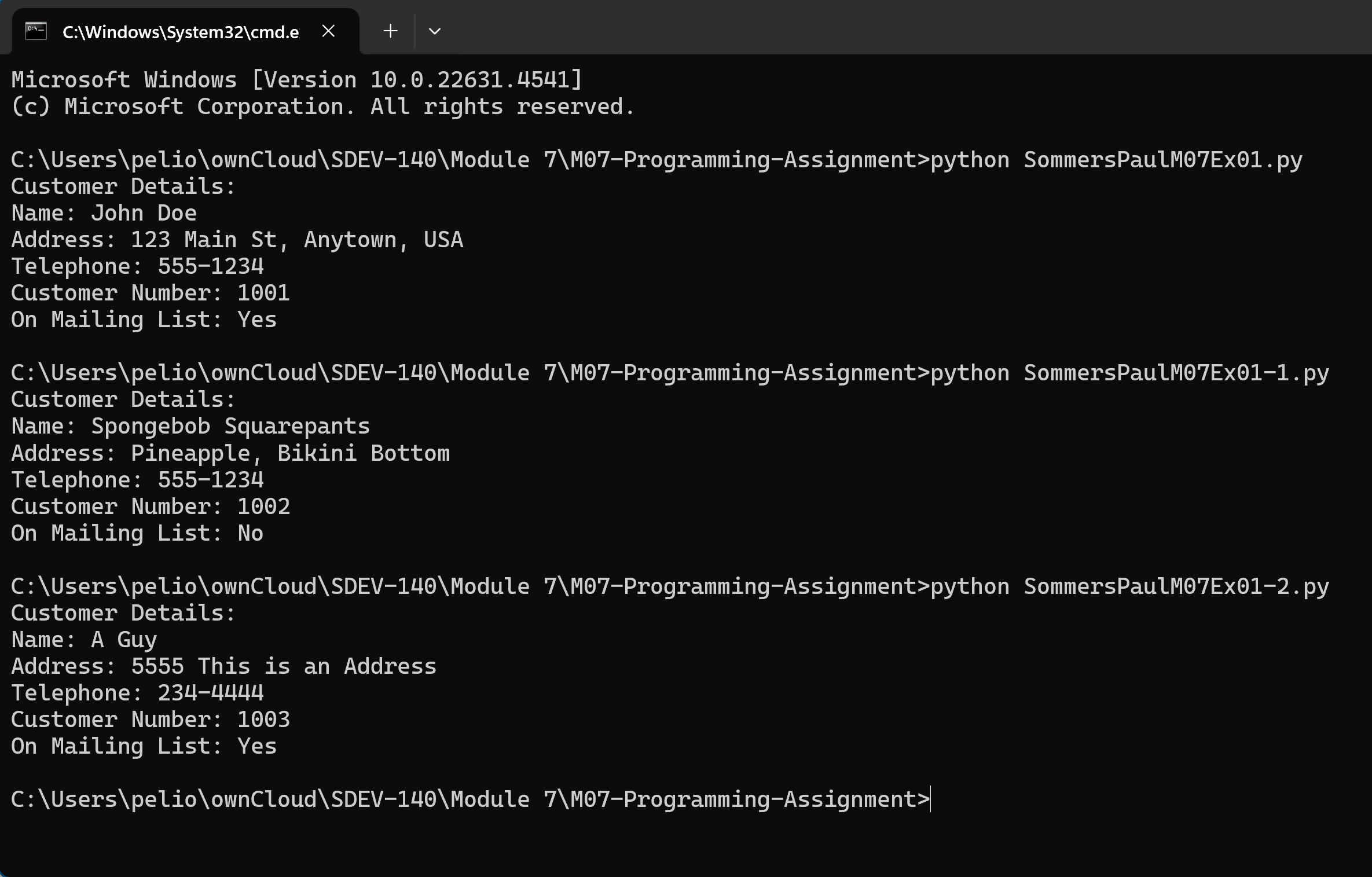
print(f"Hourly Pay Rate: ${worker.getPayRate():.2f}")

print(f"Shift: {'Day' if worker.getShift() == 1 else 'Night'}")

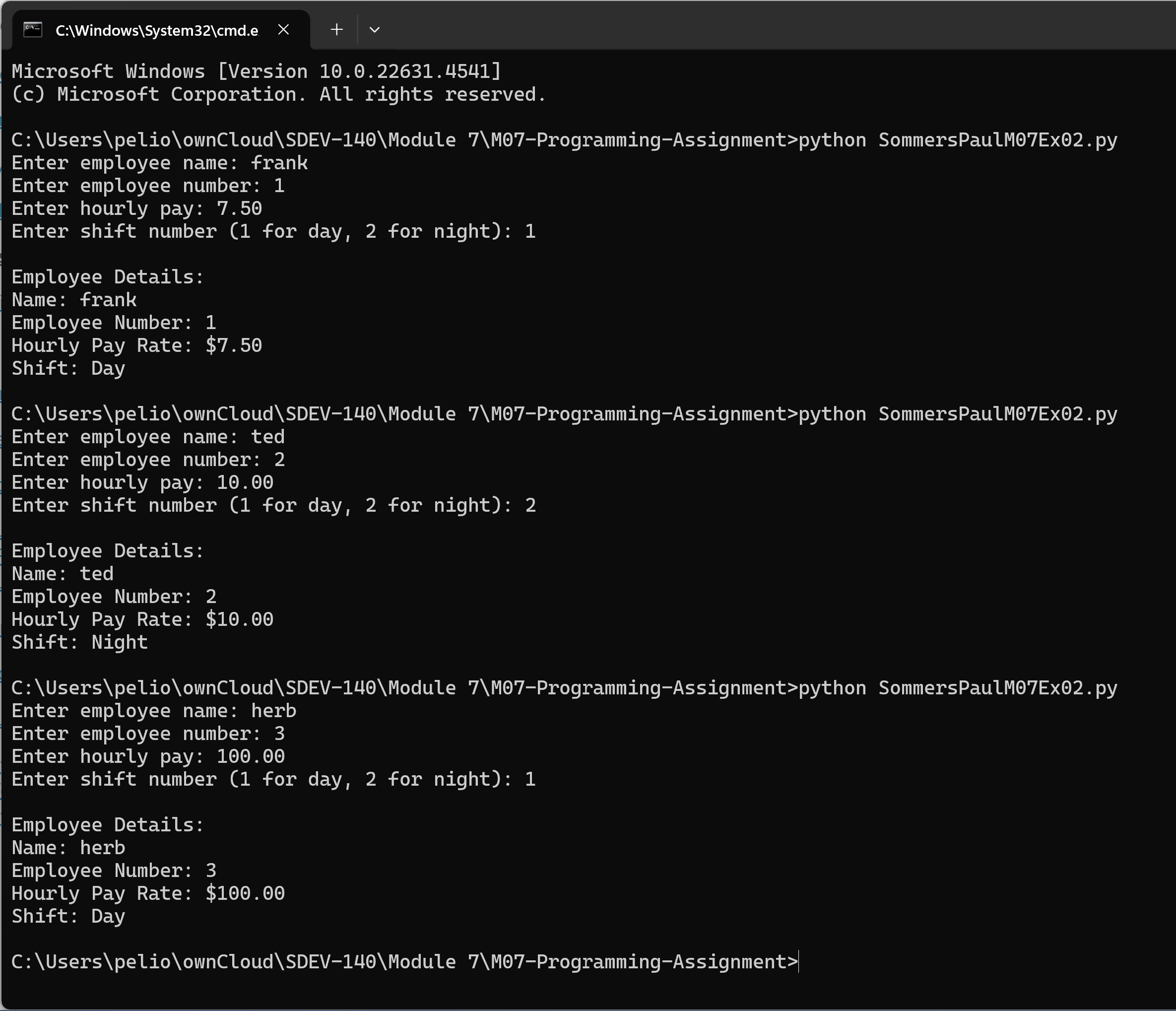
# Run the program

main()

3. Three Use Case Screen Shots

**Part 1:**  


**Part 2:**



4. GitHub Url  
  
<https://github.com/PaulSommers/SDEV140-M07-Programming-Assignment>