Project Report:

Critical Thinking Module 3 – Meal Bill Calculator & Alarm Clock

Alexander Ricciardi

Colorado State University Global

CSC500: Principles of Programming

Professor: Dr. Brian Holbert

September 23, 2025

Project Report:

Critical Thinking Module 3 – Meal Bill Calculator & Alarm Clock

This documentation is part of the Critical Thinking Assignment Module 3 from CSC500: Principles of Programming at Colorado State University Global. This Project Report is an overview of the program's functionality and testing scenarios, including console output screenshots. The program is coded in Python 3.11, and it is called Critical Thinking Module 3 – Meal Bill Calculator & Alarm Clock.

The Assignment Direction:

Creating Python Programs
Part 1:
Write a program that calculates the total amount of a meal purchased at a restaurant. The program should ask the user to enter the charge for the food and then calculate the amounts with an 18 percent tip and 7 percent sales tax. Display each of these amounts and the total price.

Part 2:
Many people keep time using a 24-hour clock (11 is 11am and 23 is 11pm, 0 is midnight). If it is currently 13 and you set your alarm to go off in 50 hours, it will be 15 (3pm). Write a Python program to solve the general version of the above problem. Ask the user for the time now (in hours) and then ask for the number of hours to wait for the alarm. Your program should output what the time will be on a 24-hour clock when the alarm goes off.

Submission:
Compile and submit your pseudocode, source code, and screenshots of the application executing the code from Parts 1 and 2, the results and GIT repository in a single document (Word is preferred).

My Program Description:

The program is a small terminal app. that has two parts:

- Part 1 (Restaurant Bill): Total meal calculator based on food charges/tips/taxes
- Part 2 (24-Hour Alarm): Alarm time calculator based on military time (24hours) and (added functionality) labels the day the alarm will go off (Today, Tomorrow, or In N days).

It also includes input validation.

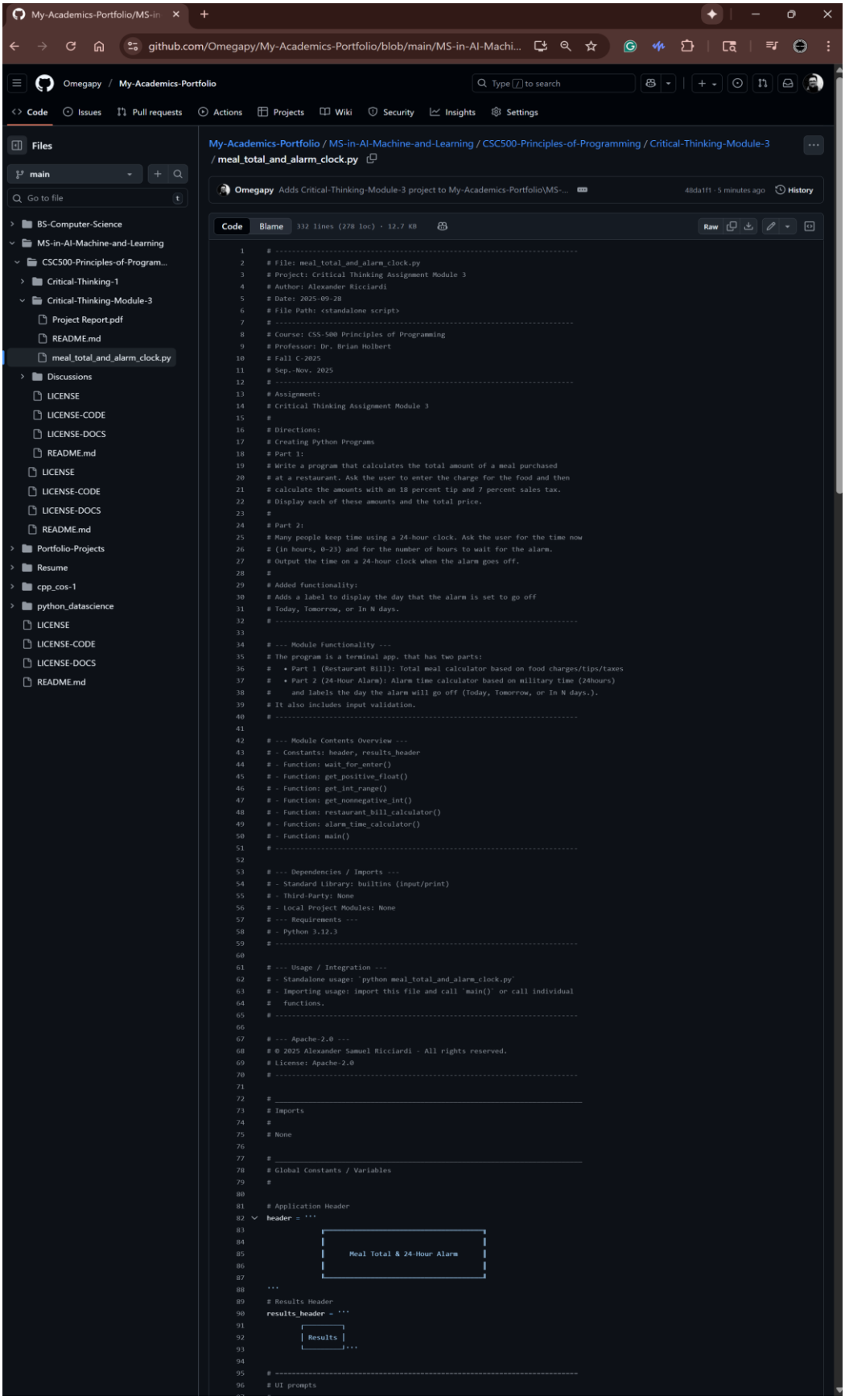
Git Repository:

I use GitHub as my Distributed Version Control System (DVCS).
The following is a link to my GitHub profile, Omega.py.

The link to this specific assignment is:
https://github.com/Omegapy/My-Academics-Portfolio/tree/main/MS-in-AI-Machine-and-Learning/CSC500-Principles-of-Programming/Critical-Thinking-Module-3

Figure-1

Code on GitHub



Code Snippet 1

Project Code in VS Code

```
# ------------------------------------------------------------------------
# File: meal_total_and_alarm_clock.py
# Project: Critical Thinking Assignment Module 3
# Author: Alexander Ricciardi
# Date: 2025-09-28
# File Path: <standalone script>
# ------------------------------------------------------------------------
# Course: CSS-500 Principles of Programming
# Professor: Dr. Brian Holbert
# Fall C-2025
# Sep.-Nov. 2025
# ------------------------------------------------------------------------
# Assignment:
# Critical Thinking Assignment Module 3
#
# Directions:
# Creating Python Programs
# Part 1:
# Write a program that calculates the total amount of a meal purchased
# at a restaurant. Ask the user to enter the charge for the food and then
# calculate the amounts with an 18 percent tip and 7 percent sales tax.
# Display each of these amounts and the total price.
#
# Part 2:
# Many people keep time using a 24-hour clock. Ask the user for the time now
# (in hours, 0–23) and for the number of hours to wait for the alarm.
# Output the time on a 24-hour clock when the alarm goes off.
#
# Added functionality:
# Adds a label to display the day that the alarm is set to go off
# Today, Tomorrow, or In N days.
# ------------------------------------------------------------------------

# --- Module Functionality ---
# The program is a terminal app. that has two parts:
#   • Part 1 (Restaurant Bill): Total meal calculator based on food charges/tips/taxes
#   • Part 2 (24-Hour Alarm): Alarm time calculator based on military time (24hours)
#     and labels the day the alarm will go off (Today, Tomorrow, or In N days.).
# It also includes input validation.
# ------------------------------------------------------------------------

# --- Module Contents Overview ---
# - Constants: header, results_header
```

```
# - Function: wait_for_enter()
# - Function: get_positive_float()
# - Function: get_int_range()
# - Function: get_nonnegative_int()
# - Function: restaurant_bill_calculator()
# - Function: alarm_time_calculator()
# - Function: main()
# ------------------------------------------------------------------------

# --- Dependencies / Imports ---
# - Standard Library: builtins (input/print)
# - Third-Party: None
# - Local Project Modules: None
# --- Requirements ---
# - Python 3.12.3
# ------------------------------------------------------------------------

# --- Usage / Integration ---
# - Standalone usage: `python meal_total_and_alarm_clock.py`
# - Importing usage: import this file and call `main()` or call individual
#   functions.
# ------------------------------------------------------------------------

# --- Apache-2.0 ---
# © 2025 Alexander Samuel Ricciardi - All rights reserved.
# License: Apache-2.0
# ------------------------------------------------------------------------

# ________________________________________________________________________
# Imports
#
# None

# ________________________________________________________________________
# Global Constants / Variables
#

# Application Header
header = '''
            ╔═══════════════════════════════════╗
            ╠                                   ╠
            ╠      Meal Total & 24-Hour Alarm     ╠
            ╠                                   ╠
            ╚═══════════════════════════════════╝
'''
```

```
# Results Header
results_header = '''
        ┌─────────┐
        │ Results │
        └─────────┘'''


# ============================================================================
# UI prompts
# ============================================================================

# ---------------------------------------------------------------------------- wait_for_enter()
def wait_for_enter() -> None:
    """Pause program until the user presses Enter.

    Allows the user time to read the screen before continuing.

    Args:
        None

    Returns:
        None

    Side Effects:
        Awaiting user input.
    """
    input("\n        Press Enter to continue...")
# ---------------------------------------------------------------------------- end wait_for_enter()

# ---------------------------------------------------------------------------- get_positive_float()
def get_positive_float(prompt: str) -> float:
    """Prompt the user for a positive floating-point number.

    Loops until a valid non-negative numeric input is entered.

    Args:
        prompt (str): The prompt shown to the user.

    Returns:
        float: The non-negative number inputted by the user.

    Notes:
        Values are converted using float(...) and negative values are rejected.
    """
    while True:
        try:
```

```
            value = float(input(prompt))
            if value < 0:
                print("        Error: Please enter a non-negative value!\n")
                continue
            return value
        except ValueError:
            print("        Error: Please enter a valid number!\n")
# ------------------------------------------------------------------ end get_positive_float()

# ------------------------------------------------------------------------- get_int_range()
def get_int_range(prompt: str, min_value: int, max_value: int) -> int:
    """Prompt for an integer within an inclusive range.

    Args:
        prompt (str): Display to the user.
        min_value (int): Minimum value.
        max_value (int): Maximum value.

    Returns:
        int: A validated integer in [min_value, max_value].
    """
    while True:
        try:
            value = int(input(prompt))
            if value < min_value or value > max_value:
                print(f"        Error: Enter a value between {min_value} and {max_value}!\n")
                continue
            return value
        except ValueError:
            print("        Error: Please enter an integer value!\n")
# ------------------------------------------------------------------- end get_int_in_range()

# ---------------------------------------------------------------- get_nonnegative_int()
def get_nonnegative_int(prompt: str) -> int:
    """Prompt the user for a non-negative integer (>= 0).

    Args:
        prompt (str): The prompt

    Returns:
        int: The validated non-negative integer.
    """
    while True:
        try:
            value = int(input(prompt))
```

```
            if value < 0:
                print("        Error: Please enter a non-negative integer!\n")
                continue
            return value
        except ValueError:
            print("        Error: Please enter an integer value!\n")
# ------------------------------------------------------------------ end get_nonnegative_int()

# ============================================================================
# Part 1: Restaurant Bill Calculator
# ============================================================================

# ------------------------------------------------------------ restaurant_bill_calculator()
def restaurant_bill_calculator() -> None:
    """Part 1: Calculate and display meal totals with tip and tax.

    Workflow:
      1. Displays program header
      2. Captures total food charges
      3. Computes 18% tip and 7% sales tax on the total food charges
      4. Displays the breakdown bill and the final total
    """
    header_part1 = '''
        ┌─────────────────────────────────────────────┐
        │ PART 1: Restaurant Bill (Tip & Sales Tax)   │
        └─────────────────────────────────────────────┘'''
    print(header_part1)

    # Input
    food_charge = get_positive_float("        Enter the charge for the food: $")

    # Compute
    TIP_RATE: float = 0.18
    TAX_RATE: float = 0.07
    tip = food_charge * TIP_RATE
    tax = food_charge * TAX_RATE  # tax applies to food charge only
    total = food_charge + tip + tax

    # Output
    print(results_header)
    print(f"        Food charge:     ${food_charge:,.2f}")
    print(f"        18% tip:         ${tip:,.2f}")
    print(f"        7% sales tax:    ${tax:,.2f}")
    print("        " + "-" * 28)
    print(f"        Total:           ${total:,.2f}")
```

```
    wait_for_enter()
# ------------------------------------------------------------ end restaurant_bill_calculator()
# ============================================================================
# Part 2: 24-Hour Alarm Time Calculator
# ============================================================================

# ------------------------------------------------------------------ alarm_time_calculator()
def alarm_time_calculator() -> None:
    """Part 2: Compute and display the alarm time on a 24-hour clock,
    and label it as Today, Tomorrow, or In N days.
    """
    header_part2 = '''
        ┌──────────────────────────────────────────────┐
        │ PART 2: 24-Hour Alarm Time Calculator        │
        └──────────────────────────────────────────────┘'''
    print(header_part2)

    current_hour = get_int_range("        What is the current hour (0-23)? ", 0, 23)
    wait_hours = get_nonnegative_int("        How many hours to wait? ")

    total_hours = current_hour + wait_hours
    # using modulo operator (remainder after division) -> 3 % 2 = 1
    alarm_hour = total_hours % 24 # 24 hours -> 1 full day
    # using integer floor division (quotient floored after division) -> 7 // 3 = 2
    num_days = total_hours // 24

    # Today / Tomorrow / In N days
    if num_days == 0:
        day_label = "Today"
    elif num_days == 1:
        day_label = "Tomorrow"
    else:
        day_label = f"In {num_days} days"

    print(results_header)
    print(f"        Current hour: {current_hour:02d}:00")
    print(f"        Wait hours:   {wait_hours}")
    print("        " + "-" * 33)
    # Keep spacing before the label so it lines up like your example
    print(f"        Alarm at:     {alarm_hour:02d}:00     {day_label}")
    wait_for_enter()
# ------------------------------------------------------------ end alarm_time_calculator()

# ============================================================================
```

```
# Program Entry / Menu Loop
# ============================================================================

# ------------------------------------------------------------------------------ main()
def main() -> None:
    """Entry point for the program's menu.

    Presents a looped menu with three choices:
      1. Part 1: Restaurant Bill (Food, Tip, and Tax)
      2. Part 2: 24-Hour Alarm
      3. Exit

    Side Effects:
        Prints a menu, captures user input, runs the operations,
        and loops until the user exits
    """
    print(header)

    while True:
        menu = '''
        ┌──────────────────────────────────┐
        | Menu:                            |
        | 1. Restaurant Bill (Part 1)      |
        | 2. 24-Hour Alarm Time (Part 2)   |
        | 3. Exit                          |
        └──────────────────────────────────┘
        '''
        print(menu)
        # .strip() string method that removes whitespace
        # from the beginning and end of the user entered input
        choice = input("        Enter 1, 2, or 3: ").strip()

        if choice == '1':
            restaurant_bill_calculator()
        elif choice == '2':
            alarm_time_calculator()
        elif choice == '3':
            print("\n        Bye! 👌\n")
            break
        else:
            print("\n        Error: Please enter 1, 2, or 3.")
            wait_for_enter()
# ------------------------------------------------------------------------------ end main()

# ____________________________________________________________________________
```

```
# Module Initialization / Main Execution Guard
# This code runs only when the file is executed
# ------------------------------------------------------------------------------ main_guard
if __name__ == "__main__":
    main()
# ---------------------------------------------------------------------------- end main_guard

# ____________________________________________________________________________
# End of File
#
```

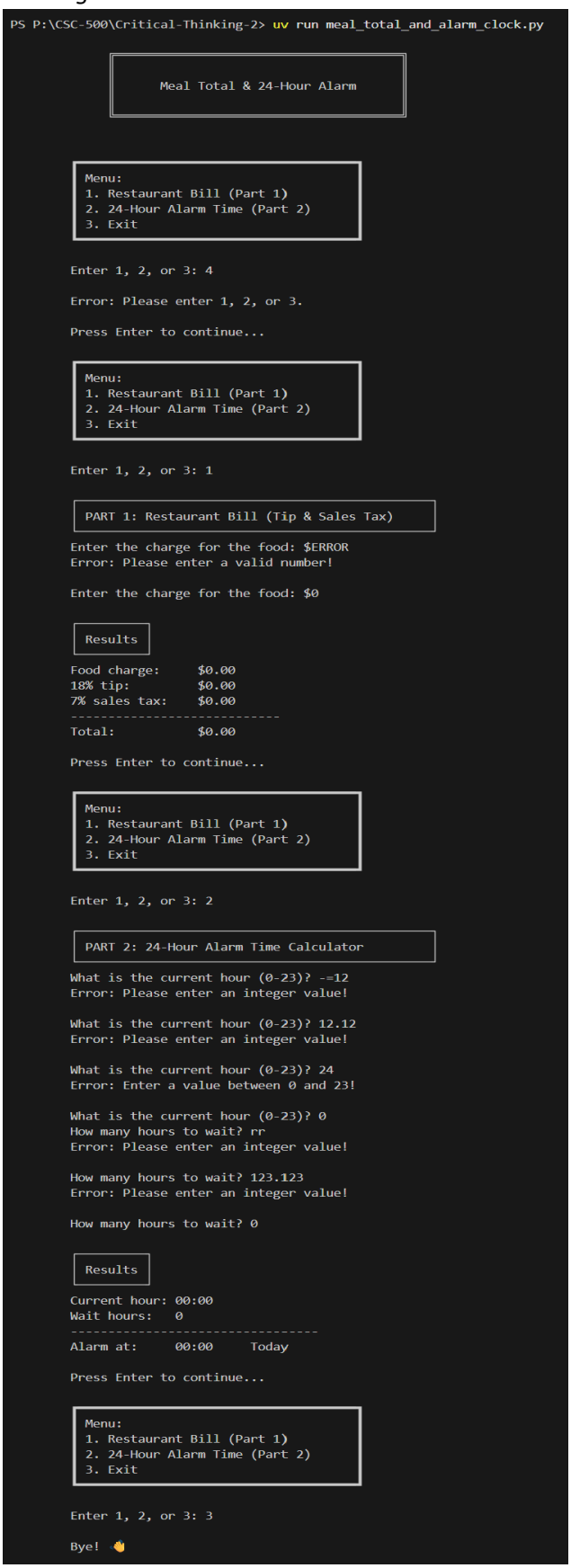
Continue next page

Figure 2

Project Outputs in the VS Code Terminal

Figure 3

Project Outputs Troubleshooting in the VS Code Terminal



As shown in Figures 2, Figure 3, and Code Snippet 1, the program runs without any issues, displaying the correct outputs as expected.