**Stephanie Cogburn**

**Python Program (Parts 1 & 2)**

**4th of August, 2024**

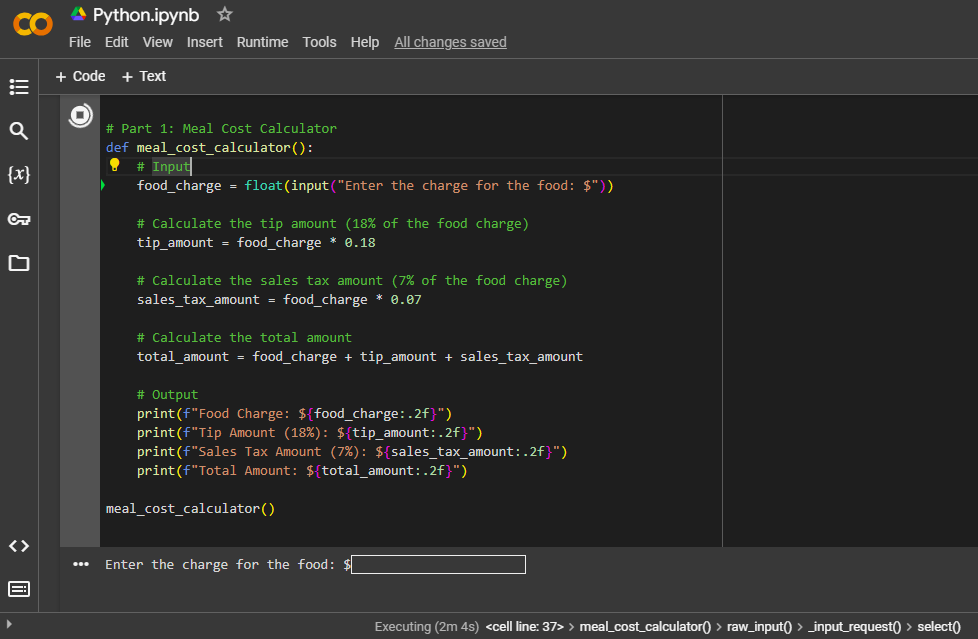
**Part 1: Meal Cost Calculator**

* Pseudocode:

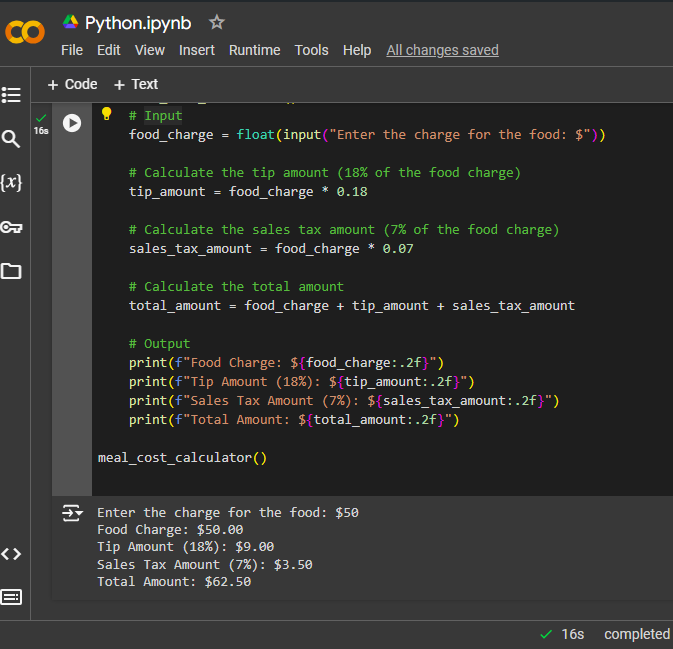
1. Prompt the user to enter the charge for the food.
2. Calculate the tip amount (18% of the food charge).
3. Calculate the sales tax amount (7% of the food charge).
4. Calculate the total amount (food charge + tip amount + sales tax amount).
5. Display the food charge, tip amount, sales tax amount, and total amount.

* Source code:
* #-------------------------------------------
* # Program Name: Meal Cost Calculator
* # Author: Stephanie Cogburn
* # Date: 04/08/2024
* #-------------------------------------------
* # Pseudocode:
* #   1. Prompt the user to enter the charge for the food
* #   2. Calculate the tip amount (18% of the food charge)
* #   3. Calculate the sales tax amount (7% of the food charge)
* #   4. Calculate the total amount (food charge + tip amount + sales tax amount)
* #   5. Display the food charge, tip amount, sales tax amount, and total amount
* #-------------------------------------------
* # Program Inputs: food\_charge
* # Program Outputs: tip\_amount, sales\_tax\_amount, total\_amount
* #-------------------------------------------
* # Part 1: Meal Cost Calculator
* def meal\_cost\_calculator():
* # Input
* food\_charge = float(input("Enter the charge for the food: $"))
* # Calculate the tip amount (18% of the food charge)
* tip\_amount = food\_charge \* 0.18
* # Calculate the sales tax amount (7% of the food charge)
* sales\_tax\_amount = food\_charge \* 0.07
* # Calculate the total amount
* total\_amount = food\_charge + tip\_amount + sales\_tax\_amount
* # Output
* print(f"Food Charge: ${food\_charge:.2f}")
* print(f"Tip Amount (18%): ${tip\_amount:.2f}")
* print(f"Sales Tax Amount (7%): ${sales\_tax\_amount:.2f}")
* print(f"Total Amount: ${total\_amount:.2f}")
* meal\_cost\_calculator()
* Screenshots:

***Code execution using Google Colab Python Notebook***



***Code result from Google Colab Python notebook***



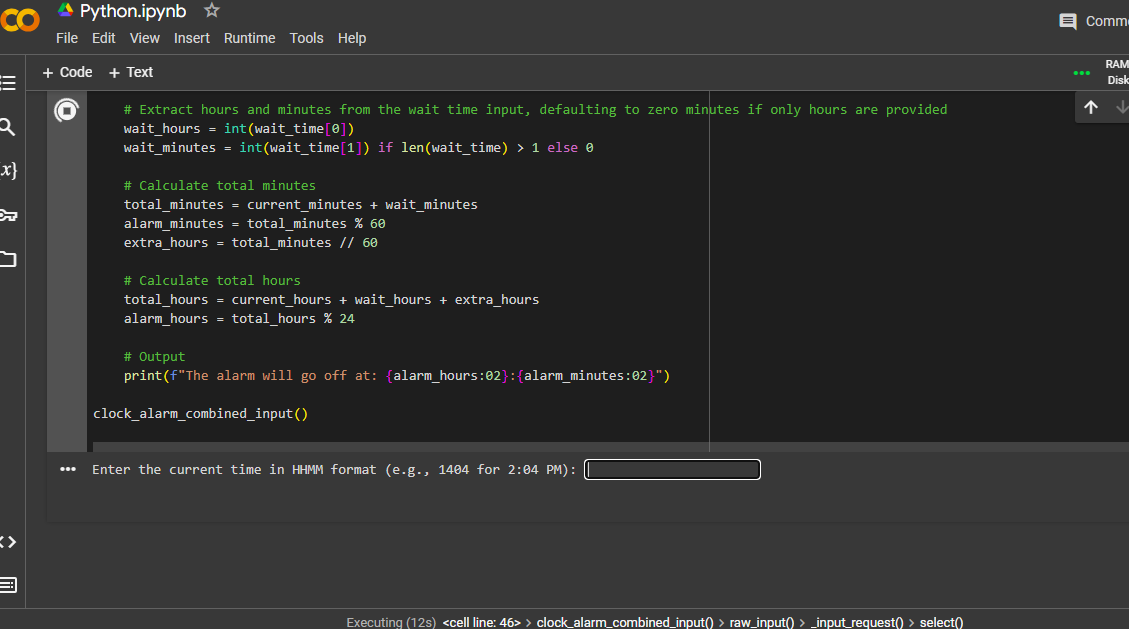
**Part 2: 24-Hour Clock Alarm**

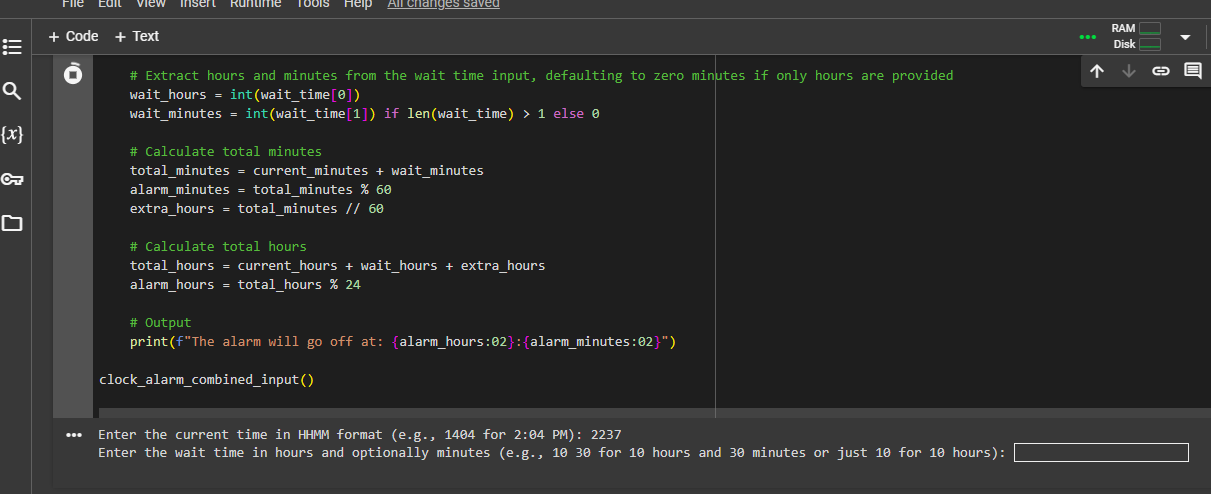
* Pseudocode:

1. Prompt the user to enter the current time in "HHMM" format
2. Extract hours and minutes from the current time input
3. Prompt the user to enter the wait time, allowing for either just hours or both hours and minutes in a single input prompt
4. Extract hours and minutes from the wait time input, defaulting to zero minutes if only hours are provided
5. Calculate the total minutes when the alarm will go off by adding the current time's minutes and the wait time's minutes
6. Calculate the total hours when the alarm will go off by adding the current time's hours and the wait time's hours plus any overflow from the minutes' calculation
7. Adjust the total hours and minutes to fit within a 24-hour format
8. Display the time when the alarm will go off

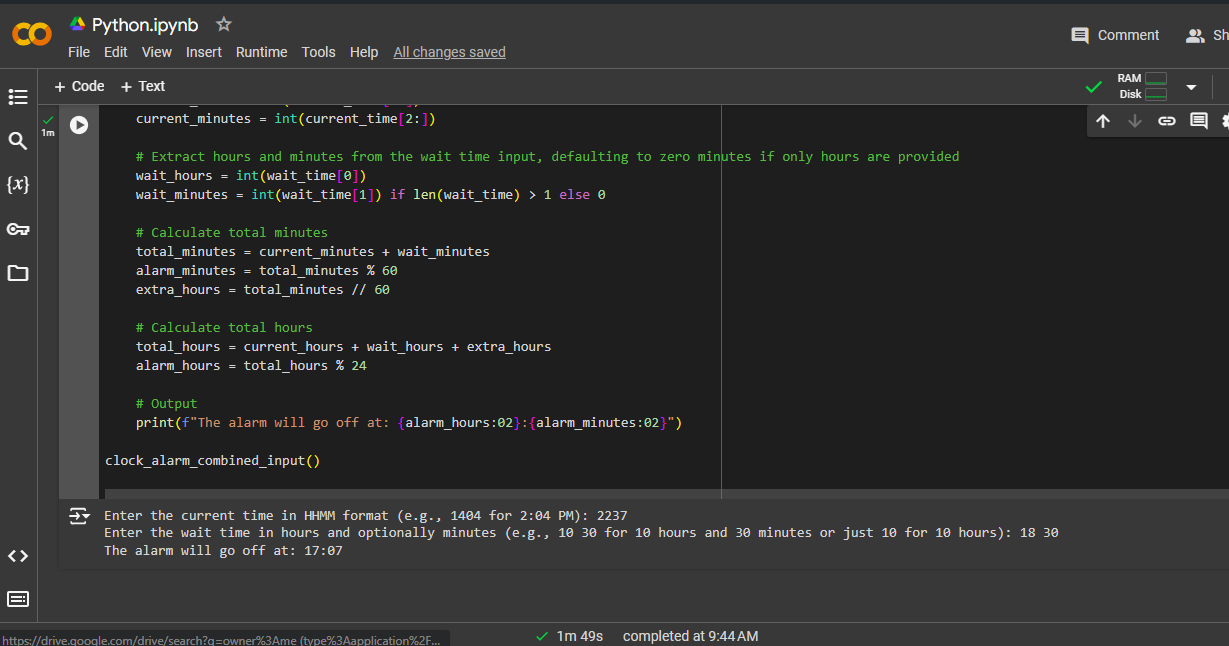
* Source code:
* #-------------------------------------------
* # Program Name: 24-Hour Clock Alarm
* # Author: Stephanie Cogburn
* # Date: 04/08/2024
* #-------------------------------------------
* # Pseudocode:
* #   1. Prompt the user to enter the current time in "HHMM" format
* #   2. Extract hours and minutes from the current time input
* #   3. Prompt the user to enter the wait time, allowing for either just hours or both hours and minutes in a single input prompt
* #   4. Extract hours and minutes from the wait time input, defaulting to zero minutes if only hours are provided
* #   5. Calculate the total minutes when the alarm will go off by adding the current time's minutes and the wait time's minutes
* #   6. Calculate the total hours when the alarm will go off by adding the current time's hours and the wait time's hours plus any overflow from the minutes' calculation
* #   7. Adjust the total hours and minutes to fit within a 24-hour format
* #   8. Display the time when the alarm will go off
* #-------------------------------------------
* # Program Inputs: current\_time, wait\_time
* # Program Outputs: alarm\_hours, alarm\_minutes
* #-------------------------------------------
* # Part 2: 24-Hour Clock Alarm with Flexible Wait Time Input
* def clock\_alarm\_combined\_input():
* # Input
* current\_time = input("Enter the current time in HHMM format (e.g., 1404 for 2:04 PM): ")
* wait\_time = input("Enter the wait time in hours and optionally minutes (e.g., 10 30 for 10 hours and 30 minutes or just 10 for 10 hours): ").split()
* # Extract hours and minutes from the current time input
* current\_hours = int(current\_time[:2])
* current\_minutes = int(current\_time[2:])
* # Extract hours and minutes from the wait time input, defaulting to zero minutes if only hours are provided
* wait\_hours = int(wait\_time[0])
* wait\_minutes = int(wait\_time[1]) if len(wait\_time) > 1 else 0
* # Calculate total minutes
* total\_minutes = current\_minutes + wait\_minutes
* alarm\_minutes = total\_minutes % 60
* extra\_hours = total\_minutes // 60
* # Calculate total hours
* total\_hours = current\_hours + wait\_hours + extra\_hours
* alarm\_hours = total\_hours % 24
* # Output
* print(f"The alarm will go off at: {alarm\_hours:02}:{alarm\_minutes:02}")
* clock\_alarm\_combined\_input()
* Screenshots:

***Code execution using Google Colab Python Notebook***





***Code result from Google Colab Python notebook***



**GIT Repository:**

Github link: