Critical Thinking Assignment: Receipt and Alarm Clock

# Pseudocode

## Part 1: Pseudocode for Receipt Calculation

BEGIN

* PROMPT user to enter the charge for the meal
* VALIDATE the input to ensure it is a non-negative number
* CALCULATE the tax as 7% of the charge
* CALCULATE the tip as 18% of the charge
* CALCULATE the total amount by adding charge, tax, and tip
* DISPLAY the charge, tax, tip, and total amount in a formatted receipt

END

## Part 2: Pseudocode for Alarm Clock

BEGIN

* PROMPT user to choose between using hours only or hours and minutes
* IF user chooses hours only THEN
* PROMPT user to enter the current time in hours (0-23)
* PROMPT user to enter the number of hours to wait until the alarm
* CALCULATE the alarm time using (current time + wait time) % 24
* DISPLAY the alarm time
* ELSE IF user chooses hours and minutes THEN
* PROMPT user to enter the current time in hours (0-23) and minutes (0-59)
* PROMPT user to enter the number of hours and minutes to wait until the alarm
* CALCULATE the total minutes (current minutes + wait minutes)
* CALCULATE any additional hours from minute overflow
* CALCULATE the final hours using (current hours + wait hours + additional hours) % 24
* DISPLAY the alarm time in hours and minutes
* ELSE
* DISPLAY an invalid choice message

END

# Source Code

Below is the Python source code that implements the Receipt calculation and Alarm Clock functionality.

## # Part 1: Receipt Calculation

class Receipt():  
  
 def \_\_init\_\_(self, total=0):  
 self.total = total  
 self.tax = 0  
 self.tip = 0  
  
 def calcTax(self):  
 self.tax = .07 \* self.total  
 return self.tax  
   
 def calcTip(self):  
 self.tip = .18 \* self.total  
 return self.tip  
   
 def calcTotal(self):  
 tax = self.calcTax()  
 tip = self.calcTip()  
  
 total = self.total + tax + tip  
 return total  
   
 def printReceipt(self):  
 receipt = (f'{"Charge:": <10} ${self.total:>5.2f}\n'  
 f'{"Tax:": <10} ${self.calcTax():>5.2f}\n'  
 f'{"Tip:": <10} ${self.calcTip():>5.2f}\n'  
 f'{"Total:": <10} ${self.calcTotal():>5.2f}\n')  
 print(receipt)  
  
def clean\_and\_convert\_input(user\_input):  
 cleaned\_input = user\_input.replace('$', '').strip()  
 return float(cleaned\_input)  
  
def validate\_input(charge):  
 if charge < 0:  
 print(f'Charge cannot be negative, you gave: ${charge:.2f}!  
')  
 return False  
 return True  
  
def get\_valid\_charge():  
 while True:  
 user\_input = input('Enter the charge: ')  
 try:  
 charge = clean\_and\_convert\_input(user\_input)  
 if validate\_input(charge):  
 return charge  
 except ValueError:  
 print('Invalid entry, please provide a valid number!')  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 charge = get\_valid\_charge()  
 print("\n")  
 receipt = Receipt(charge)  
 receipt.printReceipt()  
  
# Part 2: Alarm Clock  
  
def getInputHoursOnly():  
 curr\_hours = int(input('What is the current time now in hours (0-23)? '))  
 while curr\_hours < 0 or curr\_hours > 23:  
 print("Please enter a valid time between 0 and 23.")  
 curr\_hours = int(input('What is the current time now in hours (0-23)? '))  
 hours\_till\_alarm = int(input('Please provide the number of hours to wait until the alarm: '))  
 return curr\_hours, hours\_till\_alarm  
  
def getInputHoursAndMinutes():  
 curr\_hours = int(input('What is the current time now in hours (0-23)? '))  
 while curr\_hours < 0 or curr\_hours > 23:  
 print("Please enter a valid time between 0 and 23.")  
 curr\_hours = int(input('What is the current time now in hours (0-23)? '))  
 curr\_minutes = int(input('What are the current minutes (0-59)? '))  
 while curr\_minutes < 0 or curr\_minutes > 59:  
 print("Please enter a valid number of minutes between 0 and 59.")  
 curr\_minutes = int(input('What are the current minutes (0-59)? '))  
 hours\_till\_alarm = int(input('Please provide the number of hours to wait until the alarm: '))  
 minutes\_till\_alarm = int(input('Please provide the number of minutes to wait until the alarm: '))  
 return curr\_hours, curr\_minutes, hours\_till\_alarm, minutes\_till\_alarm  
  
def convert\_2\_24\_hours\_only(curr\_hours, hours\_till\_alarm):  
 alarm\_time = (curr\_hours + hours\_till\_alarm) % 24  
 return alarm\_time  
  
def convert\_2\_24\_hours\_and\_minutes(curr\_hours, curr\_minutes, hours\_till\_alarm, minutes\_till\_alarm):  
 total\_minutes = curr\_minutes + minutes\_till\_alarm  
 additional\_hours = total\_minutes // 60  
 final\_minutes = total\_minutes % 60  
 total\_hours = (curr\_hours + hours\_till\_alarm + additional\_hours) % 24  
 return total\_hours, final\_minutes  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("Welcome to Scharmy Alarmy! Please choose an option:")  
 print("1. Use hours only")  
 print("2. Use hours and minutes")  
 choice = input('Enter 1 or 2: ')  
 if choice == '1':  
 curr\_hours, hours\_till\_alarm = getInputHoursOnly()  
 alarm\_hours = convert\_2\_24\_hours\_only(curr\_hours, hours\_till\_alarm)  
 print(f'The alarm will go off at {alarm\_hours:02}:00 hours.')  
 elif choice == '2':  
 curr\_hours, curr\_minutes, hours\_till\_alarm, minutes\_till\_alarm = getInputHoursAndMinutes()  
 alarm\_hours, alarm\_minutes = convert\_2\_24\_hours\_and\_minutes(curr\_hours, curr\_minutes, hours\_till\_alarm, minutes\_till\_alarm)  
 print(f'The alarm will go off at {alarm\_hours:02}:{alarm\_minutes:02} hours.')  
 else:  
 print('Invalid choice. Please restart the program and choose either 1 or 2.')

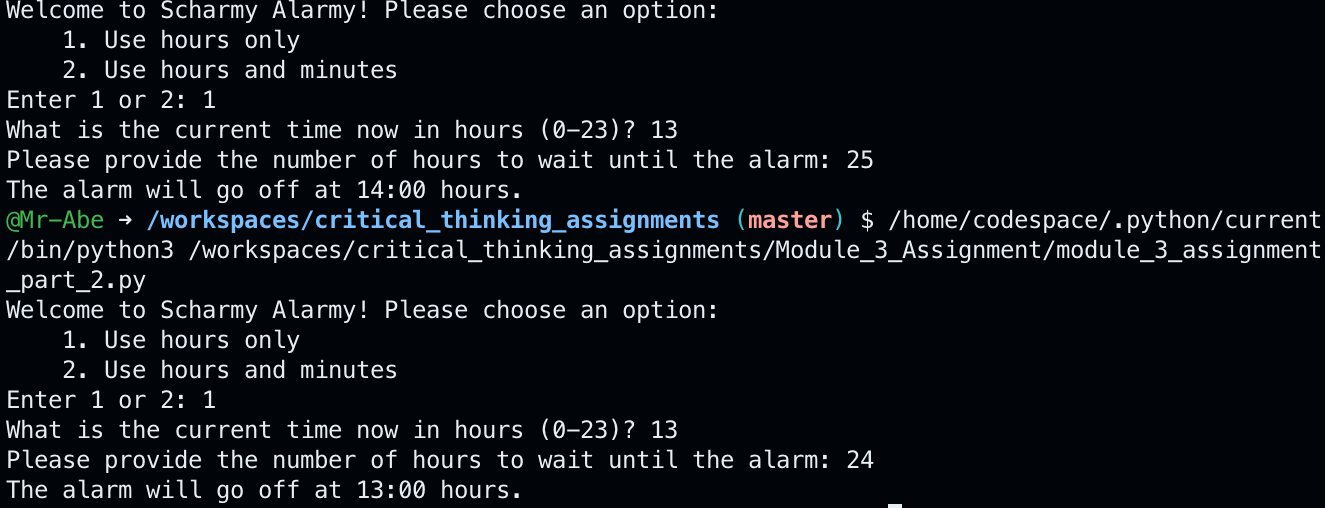
# Screenshots of Execution

## # Part 1: Receipt Calculation

A computer screen with white and blue text

Description automatically generated

## # Part 2: Alarm Clock



# Results

The program was executed successfully with multiple test inputs to validate its functionality. For the meal cost calculation, the program correctly computed the tax and tip based on a variety of input values ($10, $100, $1000) and produced an accurate total amount each time. The printed receipt displayed the charge, tax, tip, and total cost in a clear and well-formatted manner. For the alarm clock functionality, the program accurately calculated the future time on a 24-hour clock based on the user's input for the current time and the hours to wait. The outputs were verified with different scenarios, including cases where the added time rolled over to the next day (e.g., adding 25 hours to 13:00 resulting in 14:00 the next day). Both modules demonstrated correct and expected behavior under all tested conditions.

# Git Repository

Repository for this week’s assignment can be found at this [link](https://github.com/Mr-Abe/critical_thinking_assignments/tree/master/Module_3_Assignment).