# Module 3: Critical Thinking Assignment

Maurice Henning

Colorado State University Global

CSC500-1: Principals of Programming

Dr. Steven Evans

April 9, 2024

**Creating Python Programs: Part 1**

In this two-part module, I was asked to write a program that would calculate the cost of a

meal at a restaurant using the cost of a meal and arriving at a final cost once you add an 18%

tip and 7% sales tax. My approach to this was fairly simple. Broken down into human

terminology for me to understand what is requested followed by a step-by-step approach

translation for computer code comprehension.

The process began as this: First step I created a comment to inform the user what to expect with

the following code line using a # sign followed by instructions to input cost of food. I then

created a string, a sequence of characters enclosed in either single quotes (‘’) or double quotes

(“”) according to Elhouseny (2023), and called that string food\_charge. I created that string to

equal or represent the cost of food however in the statement I requested the amount to be input as

a figure with a decimal point as opposed to a whole number. My statement requests for the cost

of food to be entered with and printed as such; ‘Enter the amount of the food: $’. This is printed

because the elements are in single quotes

I then created another comment to add tip and tax in the following strings labeled as tip and

sales\_tax. . The string tip is equal to 18% and printed in decimal format as 0.18 which is then

multiplied by the cost of food better known as the string food\_charge. The second statement

similarly asks for the sales tax represented by the string sales\_tax which is then multiplied by the

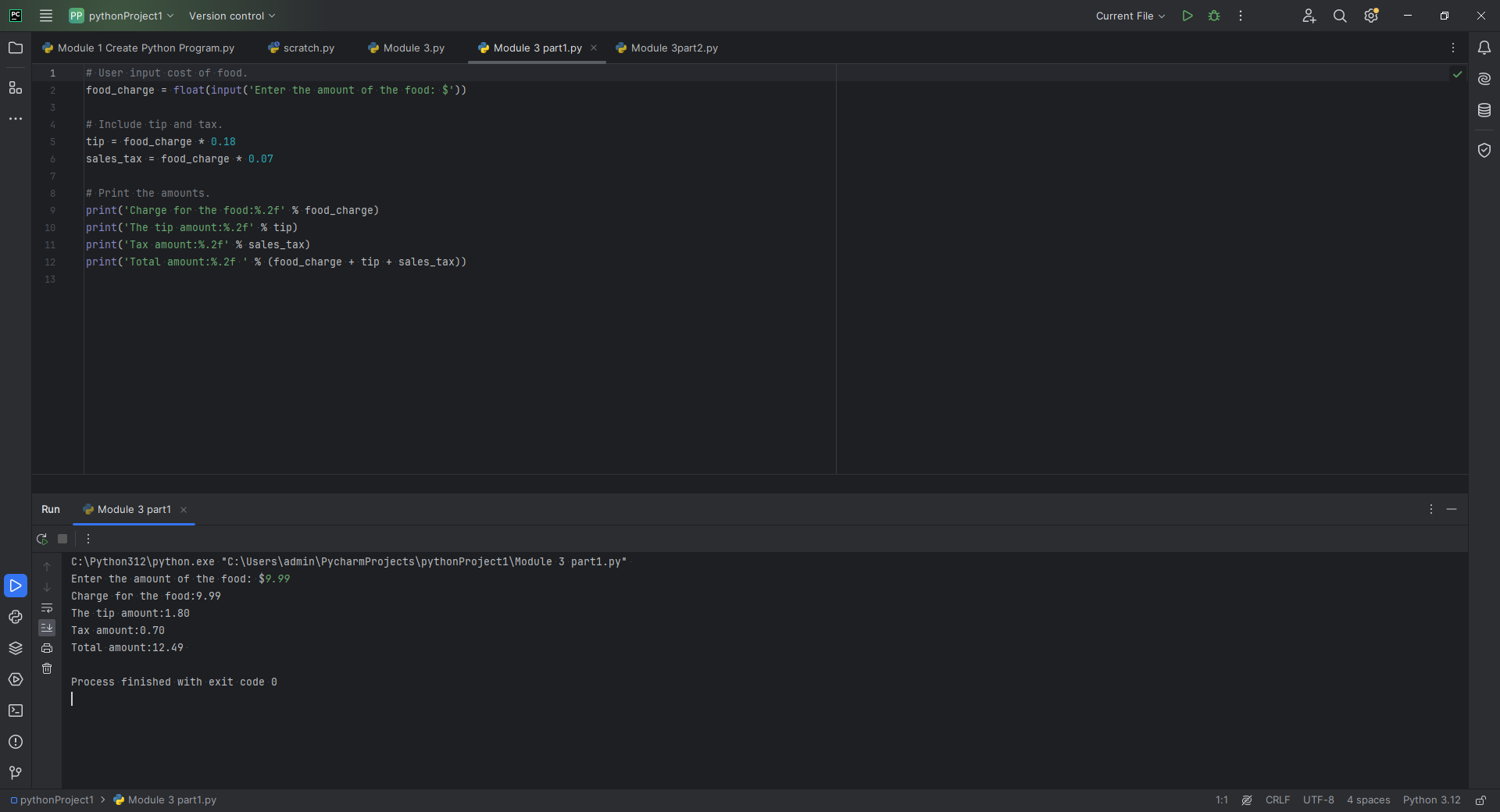
food\_charge string.

And finally, I comment (#) informing the user what sentences will be printed as a result of

entering food costs. The following sentences would then be printed in the following order: 1.

Charge for the food, 2. The tip amount, 3. Tax amount, 4. The total amount. The program will

execute the result of each equation. See EG below



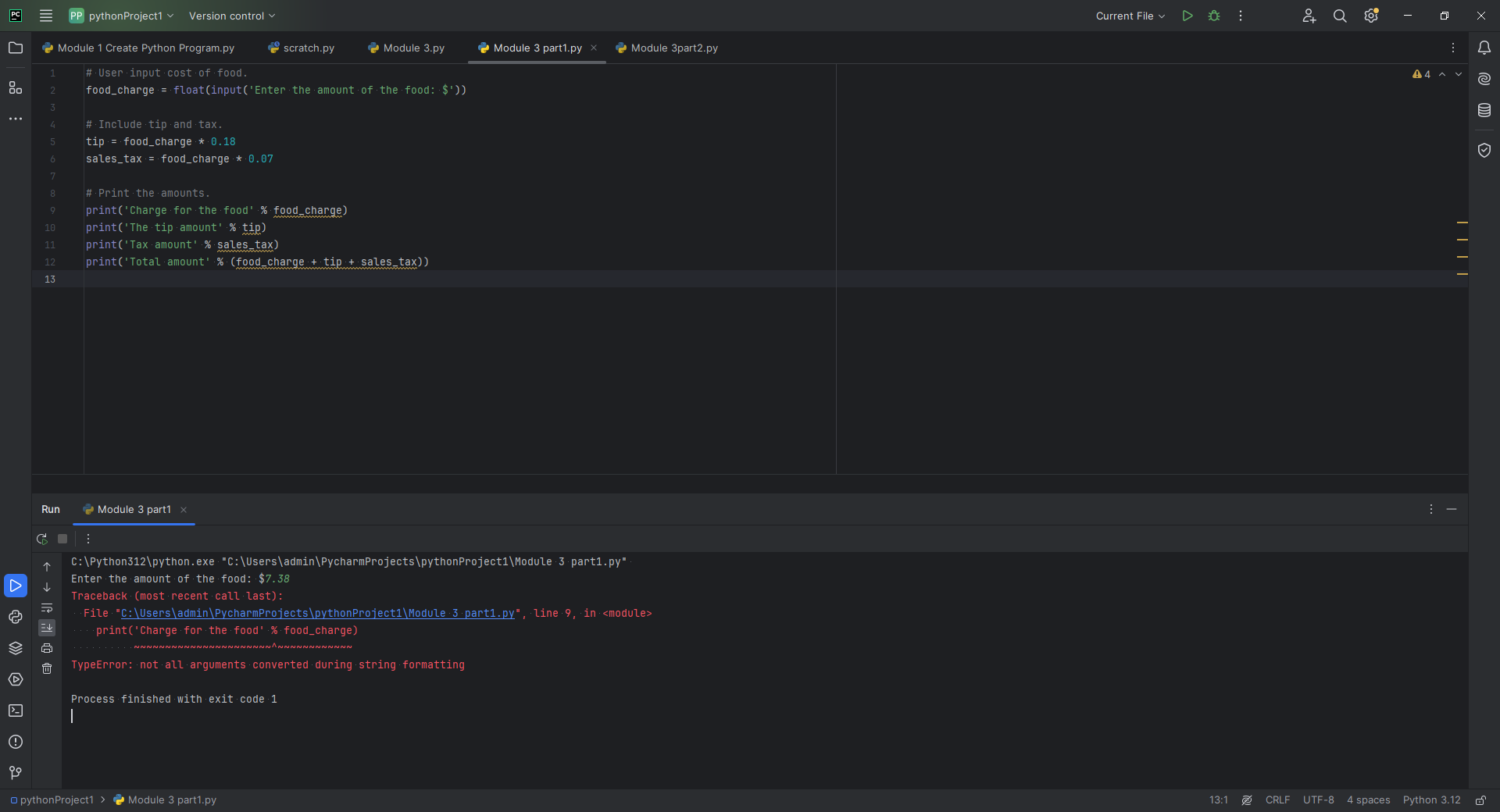
One of the challenges I experienced with this program was executing the statement without using

a formatter to format the floating values. The %f is included in the statement ('Charge for the

food:%.2f' % food\_charge) places the decimal point two places over from the whole number. If

you attempt to execute this statement without the %f you will receive the error: TypeError: not

all arguments converted during string formatting. See below



**Creating Programs: Part 2**

In the second portion of the module, I was asked to write a program requesting the input of

current time in hours, similar to military time followed by the number of hours before an alarm

rings. The output should be in the 24-hour clock. The following was my approach.

I first made a comment # explaining that the user should enter the current time. On my second

line, I created a variable labeled as current\_time. Current\_time would be equal to the value

which should be entered as a whole number labeled as an integer followed by the sentence

“Enter current time (military time). Next line I created another comment # asking for user input

for the number of hours before alarm rings. This is followed by the next line with a statement

variable I called hours\_to\_wait. This statement would hold the value number for hours before

alarm, alongside the statement “ Enter the number of hours for alarm”)

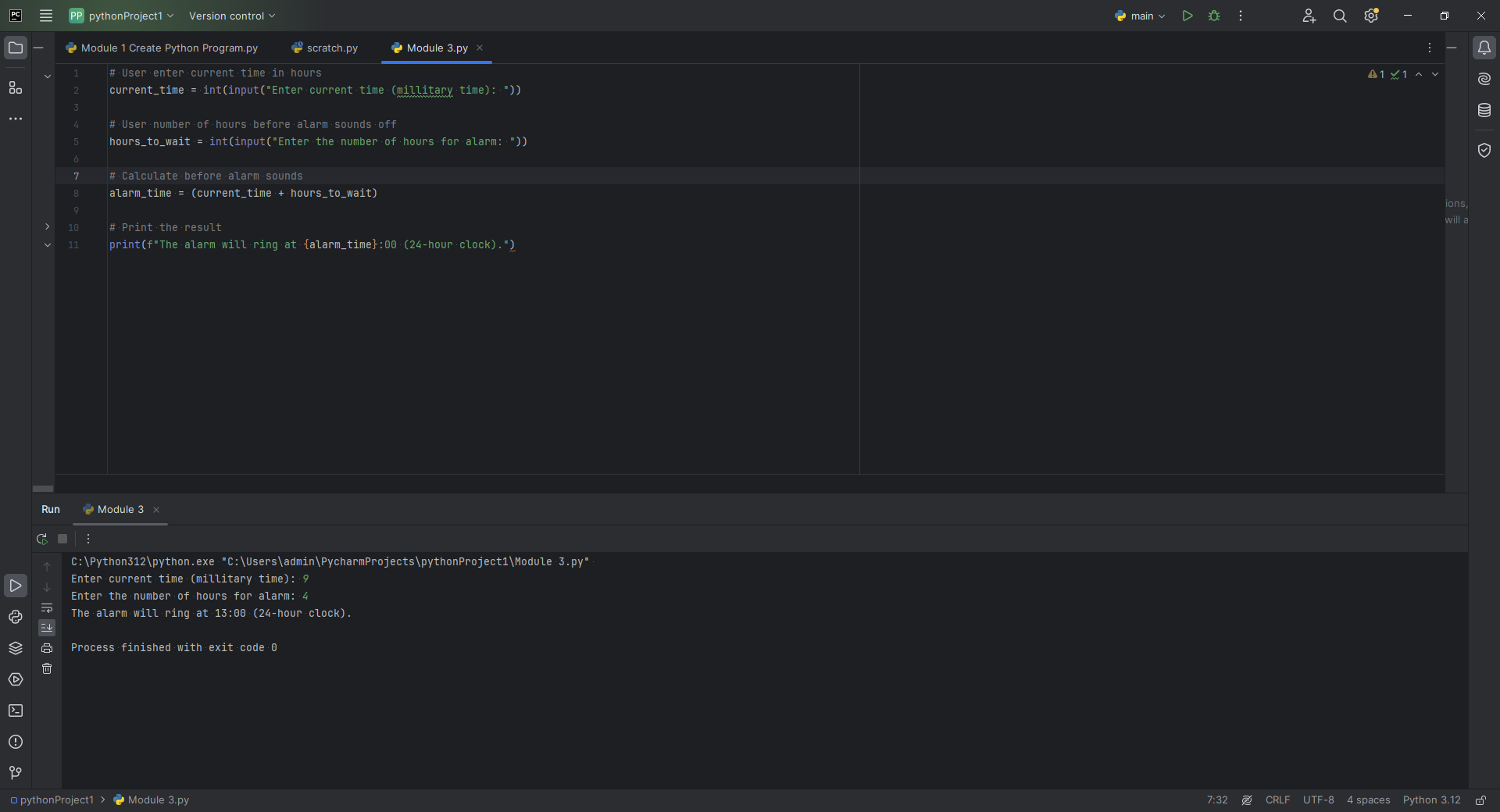
On the next line, I created the comment # to perform calculations before alarm sounds. Line 8 I

created the variable labeled alarm\_time. This variable will serve as the new variable that comes

as result of the variable current\_time plus hours\_to\_wait are added together.

I then create my final # comment line telling the system to print the sentence “The alarm will

ring at the value that is equated with the variable alarm\_time. See EG below



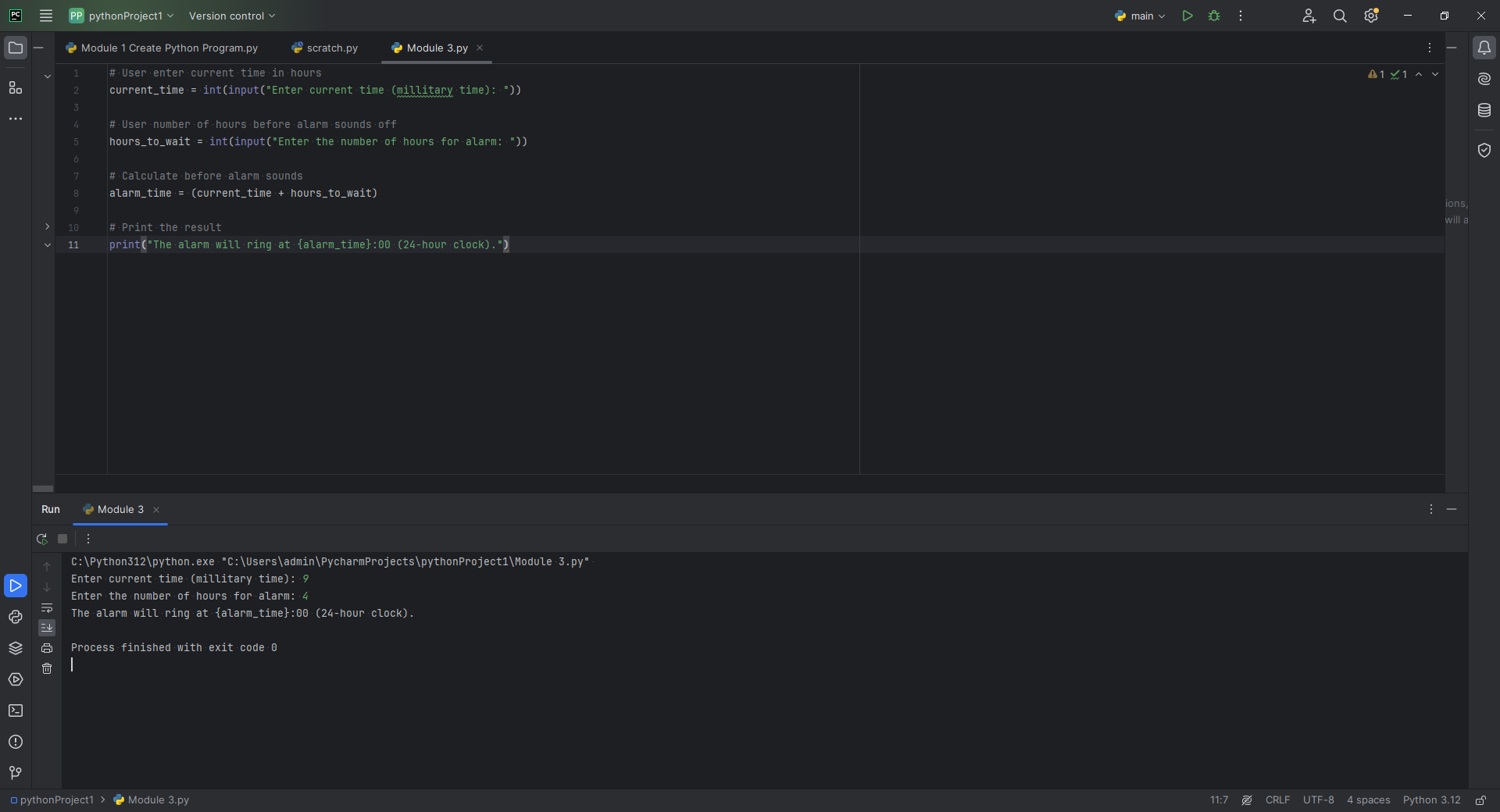
One of the challenges I encountered with creating this program was improper formatting.

My original print statement was print ("The alarm will ring at {alarm\_time}:00 (24-hour

clock)."). As a result, I would get this in return: *The alarm will ring at {alarm\_time}:00 (24-*

*hour clock).* After further research, I discovered that for proper output I needed to insert a f-

string which would then expand the sentence to include the full-time clock. See EG below



**Pseudocode**

**Part 1:**

**1: Obtain** cost of food **2: Calculate** tip of 18% (auto) **3. Calculate** tax 7% (auto) **4. Display** food charge **5. Display** tip amount **6. Display** tax  **7. Display** total of food, tip, and tax

**Part 2:**

**1: Obtain** current time **2. Obtain** hours before the alarm goes off **3. Calculate** the current time plus hours to wait **4. Display** result

**References**

ElHousieny, R. (2023) Introduction to Strings in Python

[**https://ranyel.medium.com/introduction-to-strings-in-python-18caf6e02152**](https://ranyel.medium.com/introduction-to-strings-in-python-18caf6e02152)