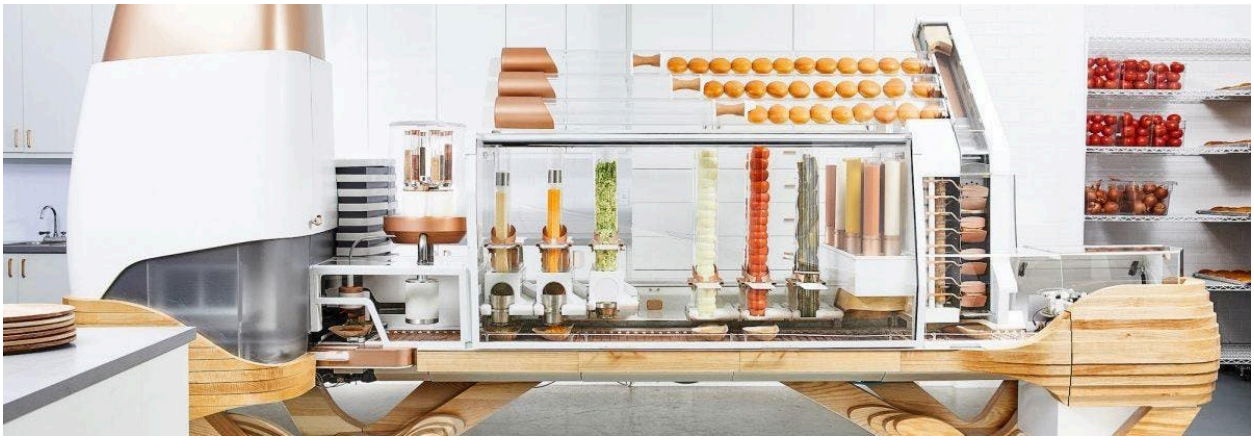
PROGRAMMING ASSIGNMENT 1: BurgerBot (33 POINTS)

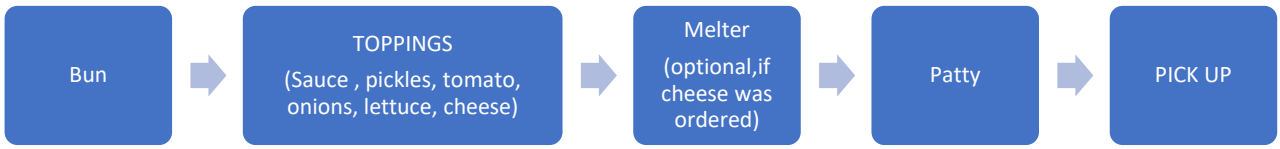


Background

In 2018, Forbes Magazine reported that a restaurant in San Francisco introduced Burger Bot, a fully automated assembly line style robot for preparing hamburgers. The assembly line has several stages. Watch it work in this video. The 14-foot machine contains 350 sensors, 20 computers and 50 actuator mechanisms to accomplish the burger making process. The Burger Bot drops each topping onto the bun as it passes through different stations on the conveyor belt. After the cheese station, the burger passes through a melter station, and stays there long enough for the cheese to melt, only if the burger has cheese on it. The beef is ground, formed into a patty, and cooked fresh for each person.

PROGRAM SPECIFICATIONS:

In this program you will simulate taking a customer's order and calculating the amount of time required to prepare the burger. The diagram illustrates the stages of burger assembly line:



Input:

The program should ask user to enter:

- the customer's name,
- 0 or 1 for each of the 6 toppings (sauce, pickles, tomato, onions, lettuce, cheese), where 1 indicates inclusion, and 0 – omission of the topping.

Processing:

The program must compute the time to produce the burger, and its cost according to the following rules.

Time: The time to produce the burger equals the sum of times it spends at the Assembly line, according to these specifications:

1. The Burger Bot starts by **dropping the bun** onto a burger plate. That takes 6 seconds.
2. The stop at each **topping** station (sauce, pickles, tomato, onions, lettuce, cheese) takes 4 seconds, if the topping is ordered, and 0, if it is not.
3. If you order cheese, the assembly also stops at the **melter**, which takes 45 seconds. If you don't order cheese, the order travels through the melter, but does not stop there.
4. The time required to **reheat the beef patty and place** it on the bun is 2.5 minutes. The burger starts cooking when the assembled bun reaches the burger station.
5. The time required to travel **between each of 10 stations** (including the pick-up area) is 3 seconds.

Cost: The cost of a burger as $6 plus 33 cents per each of *pickles, tomatoes, onions,* and *lettuce*, plus 50 cents for *cheese*.

Output:

- Production time in minutes and seconds. The program must include the customer's name in the output as shown.
- Cost of the burger in dollars, rounded to 2 decimal places. Don't worry if your program leaves out the ending 0, e.g., prints $7.5 and not $7.50. We will learn about formatting later.

SAMPLE RUNS

Consider two sample runs below. Values entered by the user are shown in boldfaced font, as in: **Sophie**

Sample run 1

```
Welcome to Burger Bot! Please enter details of your order.
Please enter customer name: Sophie
Sauce? 1 for yes, 0 for no: 1
Pickles? 1 for yes, 0 for no: 1
Tomatoes? 1 for yes, 0 for no: 0
Onion? 1 for yes, 0 for no: 0
Lettuce? 1 for yes, 0 for no: 1
Cheese? 1 for yes, 0 for no: 1

Sophie, your burger will be ready in 244 seconds, which is 4 minutes and 4
seconds.
The cost is $ 7.16
Enjoy your burger!
```

Sample run 2

```
Welcome to Burger Bot! Please enter details of your order.
Customer name: Greg
Sauce? 1 for yes, 0 for no: 1
Pickles? 1 for yes, 0 for no: 1
Tomatoes? 1 for yes, 0 for no: 1
Onion? 1 for yes, 0 for no: 1
Lettuce? 1 for yes, 0 for no: 1
```

```
Cheese? 1 for yes, 0 for no: 1

Greg, your burger will be ready in 252 seconds, which is 4 minutes and 12
seconds.
The cost is $ 7.82
Enjoy your burger!
```

HINTS

1. The easiest way to deal with times and time intervals when calculating the time to assemble the burger is to convert all time values to seconds. After determining the number of seconds required to assemble a burger, calculate the corresponding number of minutes and seconds using the operations of integer division and remainder (mod).

   For instance, in the first interaction, the total time of production is comprised of the time the bun drop takes (6 sec), the time spent for placing each topping (4* 4 sec = 16 sec), the cheese melting time (45 sec), the patty cooking time (150 sec), plus the time order travels through 10 stations (9 * 3 sec = 27 sec).

2. Use each value of 0 or 1 to account for whether a topping is included by multiplying it by the time at each station or cost for that topping. Then, if the topping is NOT included, you'll be multiplying by zero, so that number of seconds or cents will not be included in the total time required to assemble and prepare the burger. *For example*, if cheese is not ordered, numcheese would be 0, and cost of cheese (numcheese*.50) would be 0; if numcheese = 1, the cost of cheese will be .50.

   To illustrate this further, consider computation of the cost for the first sample run: there are 2 toppings charged 33 cents (note that sauce is not charged), and cheese costs extra 50 cents, so the total comes to $7.16. Note that if `numpickles = 1, numtomato = 0, numonion = 0, numlettuce = 1, and numcheese` = 1, the amount due for the order is `6 +.33 * (numpickles + numtomato + numonion + numlettuce) + .5* numcheese`

REQUIREMENTS

For full credit, you must follow these requirements:

- Assume all user inputs are as expected. You do not have to validate inputs.
- It is important that the order of inputs and outputs match the sample output as closely as possible with the shown formatting of numbers.
- Do not use any if statements, loops, or other Python libraries. Do not use any format strings. You should be able to write this program only using input statements, assignment statements, and printing the output. Be sure to use appropriate data types.

Programming Style requirements: apply these to all your code

- Include introductory comments (docstrings) listing your name, the date the program was created, a brief description of the program and**, in case you used generative AI tools, a complete reference, including the prompts used**.
- Provide comments within the program code for any part of code that would benefit from summarizing or explanatory text.
- Use symbolic constants named in ALL UPPERCASE LETTERS in your program to represent parameters of computation that are not entered by the user, e.g. time between sections, base price of the burger, topping, etc. Do not hard-code any literal values in your code where a variable or constant is appropriate.
- Use variable names that reflect the purpose of the variable and add comments if more information would be helpful (for example, units of measurement or valid values).
- Use "white space" – blank lines – to visually group related statements and make your code more readable.

SUBMISSION AND GRADING

Always test your program to make sure it works using a variety of cases. Once you have completed your homework submit your Python file, and, **if you used generative AI, a complete transcript of the sessions, including your prompts and the tool's output, in a separate document or as a link to a publicly accessible document.**

Your program should compile without syntax errors to receive any credit. If a part of your program is working, you will receive partial credit, but only if the program compiles without syntax errors. **Your grade will be zero on this assignment if it contains any syntax errors.**

#	Criteria	Points
1	Intro comments - docstring	2
2	Proper use of symbolic constants	2
3	Meaningful variable names	2
4	Correct number and order of inputs	3
5	Correct calculation of production time	10
6	Correct calculation of cost	10
8	Correct number and format of the output values	4
	Total	33

REMINDERS

This project requires knowledge of only very basic mathematical operations on numeric values. When working on a project always make sure you understand the program requirements first, and then think about the algorithm that you will use and write out the steps. Only after you have thought through the details of the algorithm and verified it on a few test cases, should you start working on its implementation in Python. Be sure to test your program to ensure the output matches the assignment. When working on the program it is important to learn to develop it gradually by **implementing one logical step of the algorithm at a time and testing the program after implementing each step**.