Mutable Data

Computer Science - Week 4  
Jul 23, 2022 - Version 0.0.2

Please make sure that all members of the group place their UD **email** AND **name** below.

Choose roles following the [instructions here](https://blockpy.cis.udel.edu/assignments/reading/bakery_appendix_pogil).

You should work in groups of 3. If you cannot find 3 group members, then work in groups of 2.

| **Role** | **Name** | **Email** |
| --- | --- | --- |
| **Manager** | Nicholas Ohaire | nohaire@udel.edu |
| **Speaker** | Mason Davis | [mcd@udel.edu](mailto:mcd@udel.edu) |
| **Recorder** | Zach Phillips | phillizr@udel.edu |

# 1) Mutability of Integers

The two functions below are both meant to multiply the given parameter by two (doubling it), returning the new value. However, the functions are implemented slightly differently. Read over the code, run it in Thonny, and then answer the questions.

| def double(number: int) -> int:  new\_number = number \* 2  return new\_number  def double\_mutate(number: int) -> int:  number = number \* 2  return number  price = 5  new\_price = double(price)  cost = 5  new\_cost = double(cost) |
| --- |

1. What will be the value of each of the following variables, after the code finishes executing?

| price | 5 |
| --- | --- |
| new\_price | 10 |
| cost | 5 |
| new\_cost | 10 |

2. When the program finishes, will the variable new\_number have a value? If yes, then what value will the variable have? If not, then why won’t the variable have a value?

| It does not have a value due to not being in the global scope. |
| --- |

3. Mutability determines whether a value actually changes, or if a variable holding the value merely changes which value is held. In terms of *mutability*, do the two functions double and double\_mutate have any meaningful difference in what they do?

| Double creates a new variable and returns that. Double\_mutate just redefines the number variable with a new value, so they don’t actually do anything different. |
| --- |

# 2) Mutability of Strings

The two functions below are both meant to capitalize a string and add an exclamation mark. However, the functions are implemented slightly differently. Read over the code, run it in Thonny, and then answer the questions.

| def shout(text: str) -> str:  new\_text = text.upper() + "!"  return new\_text  def shout\_mutate(text: str) -> str:  text = text.upper()  text += "!"  return text  word = "apple"  new\_word = shout(word)  message = "hello"  new\_message = shout\_mutate(message) |
| --- |

4. What will be the value of each of the following variables, after the code finishes executing?

| word | apple |
| --- | --- |
| new\_word | APPLE! |
| message | hello |
| new\_message | HELLO! |

5. In terms of *mutability*, do the two functions shout and shout\_mutate have any meaningful difference in what they do?

| There is no actual meaningful difference between the two. |
| --- |

6. Would the second line of shout\_mutate’s body work differently if it were instead:

text = text + "!"

| It would not work differently the ‘+=’ is the shorthand version of ‘text + “!”. |
| --- |

# 3) Mutability of Dataclasses

The two functions below are both meant to turn a lightbulb on or off. However, the functions are implemented slightly differently. Read over the code, run it in Thonny, and then answer the questions.

| from dataclasses import dataclass  @dataclass  class Lightbulb:  on: bool  color: str  def flip(bulb: Lightbulb) -> Lightbulb:  new\_bulb = Lightbulb(not bulb.on, bulb.color)  return new\_bulb  def flip\_mutate(bulb: Lightbulb) -> Lightbulb:  bulb.on = not bulb.on  return bulb  hallway = Lightbulb(True, 'white')  new\_hallway = flip(hallway)  kitchen = Lightbulb(True, 'white')  new\_kitchen = flip\_mutate(kitchen) |
| --- |

7. What will be the value of each of the following variables, after the code finishes executing?

Choose the appropriate boolean value from the dropdown.

| hallway | Lightbulb(True, "white") |
| --- | --- |
| new\_hallway | Lightbulb(False, "white") |
| kitchen | Lightbulb(False, "white") |
| new\_kitchen | Lightbulb(False, "white") |

8. The mutability of dataclasses is very different than the mutability of primitive data. In terms of mutability, what difference is there in what flip and flip\_mutate do?

| Flip returns a new Lightbulb after changing the value in the initial Lightbulb class. Flip\_mutate returns the same Lightbulb instance, but changes the ‘on’ bool to the opposite. |
| --- |

9. Why does the mutability of dataclasses matter? What kinds of bugs might occur if someone is careless with flip\_mutate?

| It matters because the variables inside the dataclasses can be changed in a global scope. It is possible, that while meaning to create a new variable, it actually mutates the variable itself. There could also be a bug where the program has changed the value, but the programmer doesn’t expect it to happen. |
| --- |

# 4) Mutability of Lists

The two functions below are both meant to add a zero to a list of numbers. However, the functions are implemented slightly differently. Read over the code, run it in Thonny, and then answer the questions.

| def add\_third(numbers: list[int]) -> list[int]:  new\_numbers = [numbers[0], numbers[1]]  new\_numbers.append(0)  return new\_numbers  def add\_third\_mutate(numbers: list[int]) -> list[int]:  numbers.append(0)  return numbers  values = [1, 2]  new\_values = add\_third(values)  integers = [1, 2]  new\_integers = add\_third\_mutate(integers) |
| --- |

10. What is the value of each of the following variables, after the code finishes executing?

| values | [1, 2] |
| --- | --- |
| new\_values | [1, 2, 0] |
| integers | [1, 2, 0] |
| new\_integers | [1, 2, 0] |

11. Like dataclasses, the mutability of lists is different from the mutability of primitive types like strings and integers. In terms of mutability, what is the difference in what add\_third and add\_third\_mutate do?

| Add\_third creates a new variable using the values argument. Add\_third\_mutate changes the actual integers value by appending a new argument while also creating a new variable. |
| --- |

# 5) Reflect and Review

Discuss among yourselves: what did you learn from this activity? What was surprising or interesting? If you didn’t learn anything, what do you think we were trying to teach you? How could this activity be improved?

| We learned that when the Wifi doesn’t work in the classroom, the class gets hard to teach. Though it does allow for some interesting work arounds, which is important to understand. |
| --- |
| We learned that we need to be careful with dataclass instances, since they can be changed on a global or local scope. |
| We learned how there is a difference between creating a new variable with parts of an old variable will not change the old variable. However, when a function doesn’t create a new variable and just changes the old variable, both variables can change. |

# Final Submission

When your group is happy with your answers for all the questions, download this file as a Word Document (docx) and upload the file to the appropriate assignment on Canvas.

Only one member of your group needs to submit.