Tricky Tracing

Computer Science - Week 6  
Jul 29, 2022 - Version 0.0.1

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** | Dakota kelley | dkelley@udel.edu |
| **Speaker** | Zach Phillips | phillizr@udel.edu |
| **Recorder** | Zach Phillips | phillizr@udel.edu |

# 1) Read Code

The Manager should download the following Python code and open it in Thonny:

<https://gist.githubusercontent.com/acbart/d5aee622b23972c0f97c85cc9bf06966/raw/10cbe8a42f3f62ebb1186d4b867dac401e100676/tricky_tracing.py>

Together, your group should read through the code.

1. Based on the do\_everything function's body, what does this program do? Don't explain *how* the code works, explain *what* the code does. Be concise but specific; you should only need 1-3 sentences.

| First, the function finds the first odd number in the inputted list. It then reassigns the integer list after doubling all the elements in the list then removes all items in the list that come before the specified number (beta). It then finds the last even number and compares it to the first odd number. Returns True if the odd number is higher than the last even number or False if the opposite is true. |
| --- |

2. In what order will the functions be called? You only need to give the names of the functions, not their arguments. Make sure you also include any builtin functions that are called, but do NOT include append method calls.

| 1. do\_everything() 2. find\_first\_odd() 3. double() 4. remove\_after\_high() 5. find\_last\_even() 6. print() |
| --- |

3. What improvements could you suggest for the iteration variables' and iteration lists' names? Justify your choice of names. NOTE: You can reuse the same variable name more than once, as long as you justify why you are doing so.

| We would change the variables in remove\_after\_high(). Instead of being items and item, we would use nums and num or numbers in numbers. Items is not a good variable name because it is not very specific. Integers might also not be a good variable name since it is close to the int keyword if you use it during a for loop. |
| --- |

4. Does the variable alpha violate a scope rule? If so, which rule, and why is it okay that the rule was violated in that case? If not, then why not?

| Yes, Variables outside a local scope should not be used inside that scope. But it is okay to violate it since it is being used to pass into a function, and nowhere else. |
| --- |

5. Does the variable beta violate a scope rule? If so, which rule, and why is it okay that the rule was violated in that case? If not, then why not?

| Yes, Variables outside a local scope should not be used inside that scope. But it is being used once. In doing so, you have the ability to change the variable once, at the top of the program since it is a global constant. |
| --- |

6. Do you think that the comment on line 9 is necessary? Why or why not?

| It might be necessary if someone doesn’t understand the modulo operator, but it shouldn’t be necessary since it’s one of the basic operators. |
| --- |

# 2) Trace Code

7. You are going to fill out a Trace Table for this program. We have already provided an appropriate number of stack frames. Use the table below these three subquestions. We strongly recommend you run the code in Thonny using the Debugger so you can step through each line and see the data at each step.

7.A) Label each stack frame with an appropriate **title** based on its function name. Remember that stack frames are added below existing frames; so the first function called will be directly below the Global Frame, and the last function called will be in the last box. Refer to your answer for question 2.

7.B) Replace the word VARIABLE with the **names** of the variables that live in that stack frame’s scope. We have provided the appropriate number of boxes for each variable. Note how the Global Frame’s box is already done for you.

7.C) Fill in the boxes with the **values** that the given variable takes on over the course of the program. We have marked the final box in green to indicate that that is the last value. You only need to write in a new box when the value changes, not when it stays the same. Observe how the third frame’s number variable’s history is traced, as an example.

When tracing lists that are being appended to, recall that the lists are updated mutably. Therefore, you should not have multiple boxes for those lists; each time a new value is appended to the list, simply add another value to the list shown in the box.

However, keep in mind that lists returned by these functions are NEW lists, so in those cases a new list will need to be put into the next box (HINT: this is relevant for the do\_everything() frame!).

| Global frame   | alpha | | [4, 2, 1, 5, 8, 9, 2] | | --- | | | --- | --- | --- | | beta | | 8 | | --- | | | answer | | False | | --- | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| do\_everything()   | integers | | [4, 2, 1, 5, 8, 9, 2] | | --- | | [8, 4, 2, 10, 16, 18, 4] | | [8, 4, 2] | | | --- | --- | --- | --- | --- | | first\_odd | | 1 | | --- | | | last\_even | | 2 | | --- | | |
| Find\_first\_odd frame   | numbers | | [4, 2, 1, 5, 8, 9, 2] | | --- | | | --- | --- | --- | | number | | 4 | | --- | | 2 | | 1 | | |
| Double frame   | values | | [4, 2, 1, 5, 8, 9, 2] | | --- | | | --- | --- | --- | | new\_value | | [8, 4, 2, 10, 16, 18, 4] | | --- | | | value | | 4 | | --- | | 2 | | 1 | | 5 | | 8 | | 9 | | 2 | | |
| Remove\_after\_high frame   | items | | [8, 4, 2, 10, 16, 18, 4] | | --- | | | --- | --- | --- | | item | 8   | 4  2 | | --- | | | taking | | True | | --- | | False | | | taken | | [] | | --- | | [8] | | [8, 4] | | [8, 4, 2] | |  | |  | | [8,4,2] | | |
| Find\_last\_even frame   | nums | | [8, 4, 2] | | --- | | | --- | --- | --- | | result | | -1 | | --- | | 8 | | 4 | | 2 | | | num | | 8 | | --- | | 4 | | 2 | | |

# 3) 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 were attempting to learn how all the different frames are updated step by step throughout the program. We followed the data, instead of the lines of code, and how they changed throughout the program. |
| --- |
| Gave a more vivid picture of how python actually runs codes and by doing this it allows us to understand more about python and make it easier to trace back our own code. |
| We learned more about step over, step into, and step out of the debugger in Thonny. |

# Final Submission

When your team 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 team needs to submit.