CS 1110 Spring 2021, Assignment 2:
Frame and Object Notation: Book Titles

Updates:

- Thursday Mar 11, 12:30pm: In Section 3.2, stray references to using CMS to group have been removed.

CS 1110 Spring 2021, Assignment 2: Frame and Object Notation: Book Titles*

1 Estimate of Time Required

We anticipate that this assignment should take only 2-5 hours, including doing some worked examples beforehand, so that students should be comfortable with both completing this assignment and submitting revisions of A1 during overlapping time periods.

2 Worked Examples

We strongly recommend that you go through worked examples of diagramming variables, objects, and call frames out beforehand, and will be very happy to go over these with you at consulting/office hours[1]. Besides lectures slides, see A2s and solutions thereunto for previous semesters in the "Archive" section of our assignment advice and archive page[2].

Contents

*Authors: Anne Bracy, Lillian Lee, Steve Marschner, Stephen McDowell, Walker White, and surely the influence of David Gries.

[1] http://www.cs.cornell.edu/courses/cs1110/2021sp/staff

[2] https://www.cs.cornell.edu/courses/cs1110/2021sp/resources/doing-assignments.html

3 New Rules

3.1 One-shot Submission From Now On

There is no revise-and-resubmit for this or any subsequent assignment unless otherwise noted.

3.2 Groups Not Preserved Across Assignments; You Need To Re-Group (So To Speak)

You may work alone or with just one other person, who can be someone you've grouped with before in CS1110, or a different person. If you are partnering, ~~even if you were grouped in a previous assignment, **the two of you must still form a new group on CMS BEFORE submitting,** which will link your submission "portals"~~ see ~~More details are in Section~~ the directions for how this works in Gradescope in Section 8.

4 Rules From Before That Still Apply

See Sections 1.1-1.3 of Assignment 1[3].

5 Learning Objectives, Which Have Grading Implications

You will practice executing Python code on "paper" using the notation we have introduced in class. This notation constitutes a precise visual language for describing and understanding exactly what Python is doing when it executes code. In complex coding situations, we use these kinds of diagrams ourselves to figure out what's going on.

Concepts tested:

1. What statements create variables or change the values of what variables (including global variables, local variables, and object attributes).
2. That the result of a constructor expression is the ID of the new object that is created.
3. That frames summarize the state of the process of executing a function call. The variables they contain store information local to the corresponding function, and indicate what that function can affect; the program counter records what line number should be executed next.
    - *Parameters* are local variables whose purpose is to hold the input values that the function is supplied when called.
    - *Arguments* are the input values that are supplied to a function when the function is called, and are assigned to the corresponding parameter variables.
4. Which statements of an if-statement are executed in a given setting.

Given these learning objectives, some seemingly minor but actually tragic mistakes you can expect to lose points for committing are:

1. Drawing fewer or more objects than the number of constructor expressions that are evaluated during execution.
2. In the frame for a call to a function, not drawing a correspondingly named box for each parameter in that function's header.
3. Drawing non-existent variables (indicating that you believe in their existence).
4. Writing the name of a variable, say `x`, inside of the box for another variable, say a box named `y`, instead of a value [the problem: if `x`'s value is subsequently changed, you would predict the wrong value for `y`].
5. Writing a variable name on the tab of an object instead of a value [the problem: if the variable's value changes, you would incorrectly predict that the object's ID changes too].

Some seemingly minor notational mistakes that *could* indicate deeper misunderstandings and thus risk point deductions are:

1. Not having the correct sequence of crossed-out line numbers in the frame's program counter [the problem: you might be misunderstanding where the flow of execution goes next].

[3] https://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment1/a1.pdf

6 Notational Conventions (Some Differ From Previous Semesters)

1. Do not try to "show animation" in your diagrams.[4] That is, there should be one call-frame on your paper for one function call, no matter how many individual lines of that function are executed. As another example: each global variable should show up exactly once, not multiple times, on your paper.
2. Do not erase any values, objects, or frames. For values that are changed, the old value should be neatly crossed out such that we can see what the old value was; and the new value should be written next to it. Similarly, frames should be crossed out rather than erased, and objects should never be removed.
3. In the program counter (the place in call frames where you put line numbers and cross them out), do not enter numbers for lines corresponding to comments or docstrings.
4. When function execution ends, cross out the final value of the program counter. The series of crossed-out program-counter values is a record of which lines were executed during the function call.
5. If a function call returns some value v, write "RETURN: v" in the frame, e.g., "RETURN 3" for a function call that returned the integer 3. (Be sure you are writing a value, not a variable name.) If a function call explicitly or implicitly returns `None`, write "RETURN None" in the frame.
6. Place your frames so that their position reflects the order in which they were created; we recommend starting at the top and drawing each frame below the previous one.
7. Don't draw the objects (folders) for imported modules or for function definitions.
8. Don't draw call frames for built-in functions, such as `int()`.
9. Since we haven't covered class definitions in detail yet, assume that the creation of a new object and initialization of its attributes happen in one step, and no call frame is generated.[5]
10. `else`-statements should be treated as executable (and executed) lines.

7 Your Task

In Section 9, and also in file a2_objects.py[6] (which imports readers.py[7]) and file a2_calls.py[8], is the code you are to work with.

The definition for class `Book` (given in file `readers.py`) means that a constructor expression like `Book("Less", "Andrew Sean Greer")` creates a new `Book` object with a `title` attribute having the value "Less" and an `author` attribute having the value `"Andrew Sean Greer"`.

The definition for class `Reader` (given in file `readers.py`) means that a constructor expression like `Reader("Michiko Kakutani", b)` creates a new `Reader` object with a `name` attribute having the value `"Michiko Kakutani"` and a `fav` attribute with a value that is the id of a Book object stored in variable `b`.

Your submission should contain:

- A diagram (compliant with the notational conventions given in Section 6, lecture and the worked examples in Section 2) showing what happens during the execution of `a2_objects.py`;
- Your answers to Q1-Q5, where these questions appear as comments in `a2_objects.py`;
- A diagram (compliant with the notational conventions given in Section 6, lecture and the worked examples in Section 2) showing what happens during the execution of `a2_calls.py`.

In your diagrams, **use the line numbers given in Section 9**, and **choose consecutive ids for the objects you create.** This is *different* than what Python Tutor does! (More on this below.)

You are welcome to draw your diagrams by hand (and will need to do so for exams), but we also provide a template for depicting folders for Powerpoint[9] and Google Slides[10].

[4] This is a significant difference between Fall CS1110 and Spring CS1110 call-frame notation.
[5] That is, for now, don't worry about the `__init__` method in a class or draw a frame for it, even though Python Tutor does.
[6] http://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment2/a2_objects.py
[7] http://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment2/readers.py
[8] http://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment2/a2_calls.py
[9] http://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment2/diagram_template.pptx
[10] https://docs.google.com/presentation/d/1cgKNatTHH54W0y46DMddIRizyvBkUn-fRlhrnKE2UfA/edit?usp=sharing

7.1 Need Help? Try Python Tutor

You are allowed and encouraged to use the CS1110 version of Python Tutor[11]. But, first try the worked examples by hand, and attempt to do this assignment manually before checking with Python Tutor: One often learns more from making mistakes and being corrected than by just seeing someone else or some program solve a problem for us.

Be aware that some of Python Tutor's notational and display conventions differ from what we require on assignments and exams.

8 Turning in the Assignment

We highly recommend that before submission, you double-check your answers against the "Learning Objectives, Which Have Grading Implications" section above.

8.1 Documenting Your NetID(s)

Put the netids of *all* group members at the top of the page(s) you submit.

8.2 Creating a PDF For Submission (Plan Ahead To Make Time For This)

If you work on paper, there are scanners in Olin and/or Uris Library[12]. Other possibilities[13] and computing labs equipped with scanners can be found by selecting "Peripherals" in the sidebar and then "Scanners" here[14].[15] You can also scan your homework with a mobile device[16].

Your solution must be a single PDF file.[17] Your file must be less than 100MB in size, and ideally less than 10MB. This should not be a problem as long as the resolution is reasonable.[18]

8.3 Not Legible To Us? No Credit, Unfortunately

A caution for those who plan to take a photo of your work and convert that to pdf: We unfortunately must reserve the right to judge a submission to be illegible and thus assign zero credit, and have had to do so in the past. *Please* make sure the pdf file you submit can be read by the graders.[19] You can also download your submission from CMS to see how it looks.

8.4 Submit on Gradescope, not CMS

To facilitate grading for this assignment, we will use Gradescope[20] instead of CMS.

8.5 Timeline and Deadlines

1. If you are partnering: well before Fri Mar 19, decide which of you will make the submission for the 2pm checkpoint on Fri Mar 19. **This is important because on Gradescope, unlike CMS, partnerships are declared *after* first submission!**

2. By 2pm on Fri Mar 19, whoever is responsible for submitting — and only that person — makes the initial submission on Gradescope[21]. **If partnering,** then that person adds the partner to the submission; instructions at `https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members`.

[11] `http://cs1110.cs.cornell.edu/tutor`
[12] `https://olinuris.library.cornell.edu/collections/maps/services`
[13] `https://www.library.cornell.edu/about/collections/visual-resources/digitization`
[14] `http://mapping.cit.cornell.edu/publiclabs/map/alternate_search_view.cfm`
[15] Disclaimer: the CS1110 staff cannot ascertain for you or guarantee that a particular scanner is available or in working order.
[16] `https://help.gradescope.com/article/0chl25eed3`
[17] For pdf file merging, there is the program PDFtk for Windows (`https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/`) and the built-in application Preview for OS X.
[18] If your file is still too large, try SmallPDF (`https://smallpdf.com/compress-pdf`) to compress it.
[19] Former TA Anthony Poon recommends Genius Scan (`https://www.thegrizzlylabs.com/genius-scan/`), which processes images into a flat, perspective-corrected, and evenly-lit PDF.
[20] `https://www.gradescope.com/courses/230805`
[21] `https://www.gradescope.com/courses/230805`

3. By 11:59pm Ithaca time on Fri Mar 19, make your final submission on Gradescope[22].

The 2pm checkpoint provides you a chance to alert us during business hours of any submission problems. **Since you've been warned to submit early, do not expect that we will accept work that doesn't make it onto ~~CMS~~the submission server on time** for whatever reason, including server delays stemming from many other students trying to submit at the same time as you.[23]

Of course, if some special circumstances arise, contact the instructor(s) immediately.

9 Code to diagram

9.1 a2_objects.py

You can assume that execution starts at line 22; that is, assume that the expression in line 19 evaluates to True.

```
1  import readers
2
3  def recommend(source, target):
4      """Make the favorite Book of `target` be the same favorite Book of `source`.
5
6      Precondition: source has a favorite Book (it's not None). """
7      target.fav = source.fav
8
9  def sequel(original, newtitle):
10     """Returns: new Book by the same author as `original` but with new title
11     `newtitle`.
12     Preconditions:
13         original is a (non-empty) Book.
14         newtitle is a (possibly empty) string."""
15     return readers.Book(newtitle, original.author)
16
17
18
19 if __name__ == '__main__':
20
21
22     bsg = readers.Book("Black Swan Green", "David Mitchell")
23     fs = readers.Book("The Fifth Season", "N.K. Jemisin")
24     og = sequel(fs, "The Obelisk Gate")
25     ss = sequel(og, "The Stone Sky")
26     ic = readers.Book("Interior Chinatown", "Charles Yu")
27     book1 = fs
28
29     martha = readers.Reader("Martha", book1)
30     ryan = readers.Reader("Ryan", bsg)
31     denice = readers.Reader("Denice", book1)
32     # STUDENTS:
33     # What are all the readers' favorite books?
34     # Answer this by writing on your answer sheet:
35     #
36     #       Q1
37     #       martha.fav.title:  ...
38     #       ryan.fav.title:  ...
39     #       denice.fav.title:  ...
40     #
41     # and replace each "..." with the corresponding values.
42
43     recommend(martha, ryan)
44     # STUDENTS:
45     # Does this call to `recommend` change anything about any reader's
```

[22] https://www.gradescope.com/courses/230805

[23] There are no so-called "slipdays" and there is no "you get to submit late at the price of a late penalty" policy.

```
    # favorite book?
    # Answer this by writing on your answer sheet:
    #
    #       Q2
    #       martha.fav.title:  ...
    #       ryan.fav.title:  ...
    #       denice.fav.title:  ...
    #
    # and replace each "..." with the corresponding values.


    book1 = ic
    # STUDENTS:
    # Does changing book1 change anything about any reader's favorite book?
    # Answer this by writing on your answer sheet:
    #
    #       Q3
    #       martha.fav.title:  ...
    #       ryan.fav.title:  ...
    #       denice.fav.title:  ...
    #
    # and replace each "..." with the corresponding values.

    martha.fav = og
    # STUDENTS:
    # Does changing martha's favorite book change anything about any other
    # reader's favorite book?
    # The answer is No.
    # Show that you understand this by writing on your answer sheet:
    #
    #       Q4
    #       martha.fav.title:  ...
    #       ryan.fav.title:  ...
    #       denice.fav.title:  ...
    #
    # and replace each "..." with the corresponding values

    fs.title = "Broken Earth #1"
    # STUDENTS:
    # Does changing fs.title change anything about any reader's favorite book?
    # Answer this by writing on your answer sheet:
    #
    #       Q5
    #       martha.fav.title:  ...
    #       ryan.fav.title:  ...
    #       denice.fav.title:  ...
    #
    # and replace each "..." with the corresponding values.
```

9.2 a2_call.py

```
# a2_calls.py
# Prof. Lee (LJL2)
# Mar 11, 2021

WARNING = ""

def replace_first(s, target, rep):
    """Returns: a copy of string `s` with the FIRST instance of string
    `target` in `s` replaced by string `rep`.

    Example: replace_first('THanks', 'H', 'h') -> 'Thanks'
```

```
12
13      Preconditions: `target` has length >=1 and appears in `s`. As mentioned,
14                     all the arguments have type str.
15      """
16      pos = s.index(target)
17      before = s[:pos]
18      after = s[pos + len(target):]
19
20      return before + rep + after  # desired output
21
22  def replace2(s, target, rep):
23      """Replace first two occurrences of target in s, or just the 1st if only
24      one occurrence.
25      Preconditions: target occurs at least once in s. All inputs are strings.
26      target has length >= 1."""
27      before = s[:s.index(target)+len(target)]
28      after = s[s.index(target)+len(target):]
29      if s.count(target) > 1:
30          after = replace_first(after, target, rep)
31      else:
32          WARNING = "Warning: only one instance of " + target + " found"
33
34      return replace_first(before, target, rep) + after
35
36
37  x = "MiLLissippi"
38  y = replace2(x, 'L', 's')
39  z = replace2(x, 'LL', 'ss')
40
41  # STUDENTS: don't forget to include the global variable warning in your diagram.
```