CS 1110 Spring 2020, Assignment 2: Frame and Object Notation — Payment Services

Updates:

- Feb 21, 5pm: students should group and submit on **CMS**. (The course staff will handle moving CMS A2 files to Gradescope, which facilitates grading "visual" material. Superseded references to Gradescope have been changed, with replacement text marked in orange.)

CS 1110 Spring 2020, Assignment 2: Frame and Object Notation — Payment Services*

1 Prefatory Note

We anticipate that this assignment should take only 2-3 hours, so that students should be comfortable with both completing this assignment and submitting revisions of A1 during overlapping time periods.

2 Worked Examples

First, to help solidify the concepts that are exercised in this assignment, we provide some worked examples of diagramming variables, objects, and call frames. We strongly recommend that you try these examples out as soon as possible, and will be very happy to go over these with you at consulting/office hours.

Spring 2014 That semester had a different convention about where to indicate the return value, but otherwise the notation is the same.

Here is a small piece of code and three different corresponding diagrams: one done as a video by Prof. Anne Bracy, one by Prof. Lillian Lee and one by Prof. Steve Marschner. We give independent solutions to indicate the kinds of variations in notation we don't care about.[1]

Other Spring assignments These semesters have used essentially the same conventions as we are this semester: assignment and solutions; assignment and solutions; assignment and solutions.

3 Motivation

There are a variety of payment services, such as Venmo, Zelle, and so on, that enable people to transfer money to each other, for a fee. One possible fee scheme is that the person receiving the money pays the service either a percentage of the amount received, or a fixed minimum amount, whichever is greater.

Certainly one imagines these services are "computerized"; this assignment involves code that represents a very simple implementation of a system in which people might be using a variety of different services.

Our implementation doesn't allow a transfer to go through if either the sender tries to send more money than they have, or the recipient can't afford the transaction fee even with the influx of cash.

Contents

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

[1] Examples: whether or not you draw a box around the class name in the upper-right of an object; or whether you put your global variables in a column, a row, or in a cluster.

4 New Rules

4.1 One-shot Submission From Now On

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

4.2 Partnerships Are Not Preserved Across Assignments — You Need To Re-Group (So To Speak)

If you are partnering, regardless of whether you were grouped in a previous assignment, **the two of you must still form a new group on CMS**.

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, regardless of whether you were grouped in a previous assignment, **the two of you must form a group on CMS BEFORE submitting, which will link your submission "portals"**. More details are in Section 9.

5 Rules From Before That Still Apply

The text in this section is identical to that in Assignment 1.

5.1 What Collaborations Are (Dis-)Allowed And How To Document Them

The full policy is on the course Academic Integrity page, but we re-state here **the main principles**, where "you" means you and, if there is one, your one CMS-registered group partner,

1. Never look at, access or possess any portion of another group's work in any form. (This includes work written on a whiteboard.)
2. Never show or share any portion of your work in any form to anyone except a member of the course staff.
3. Never request solutions from outside sources, for example, on online services like StackOverflow.
4. DO specifically acknowledge by name all help you received, whether or not it was "legal" according to (1)-(3).

Rule (4) means "cite your sources": the top lines of what you submit must accurately describe the entire set of people and sources[2] that contributed to the code that is submitted.

Example:[3]

[2] Other than the course staff or course materials.

[3] We aren't asking for your names, on the principle that grading an assignment shouldn't require knowing the identity of the author(s). But we do want to have NetID identifiers so we don't get confused when grading many files at once.

```
CS 1110 Assignment 2
By KDF4 and LJL2
Sources/people consulted: discussed strategy with Walker White
Feb 12, 2020
```

If your partnership dissolves, follow the instructions in our Academic Integrity policies' item on the "group divorce scenario".

6 Learning Objectives, Which Have Grading Implications

In this assignment, 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).

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

1. 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`].
2. 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].
3. 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].

7 Notational Conventions (Some Differ From Previous Semesters)

1. Do not draw multiple versions of the same thing.[4] So, for example, there should only be one call frame on your paper for one function call, no matter how many individual lines of that function are executed.
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. 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.
4. 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. value 3. (Be sure you are writing a value, not a variable name.)
5. 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. Bare `else` statements should be treated as executable (and executed) lines.

8 Your Task

On the last page of this document, and also in file a2.py (which imports payments.py), is the code you are to work with.

The definition for class `Person` (which can be found in the file `payments.py`) means that a constructor expression like `Person(16.0, svc)` creates a new `Person` object with an `acct` attribute having the value 16.0 and a `service` attribute having the value of whatever is in variable `svc`.

The definition for class `Service` (which can be found in the file `payments.py`) means that a constructor expression like `Service("iPay", 123.99, .05, 3.0)` creates a new `Service` object with a `name` attribute having the value `"iPay"`, an `acct` attribute with value `123.99`, a `rate` attribute with value `.05`, and a `min` atttribute with value 3.0.

Your submission should consist of a diagram, compliant with the notational conventions given in Section 7, lecture, and the worked examples in Section 2, showing what happens during the execution of it. Use the line numbers given on the last page of this document.

Choose consecutive ids for the objects you create. This is *different* than what Python Tutor does! (More on this below.)

8.1 Need Help? Try Python Tutor

You can catch many errors by comparing your on-paper results to the results of Python Tutor. 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.

You can copy the `a2.py` code into the main tab of Python Tutor and the contents of any files it imports into separate tabs, making sure to change the tab names to the corresponding module names; Prof. Fan has posted a video demonstrating the process. Be aware that some of Python Tutor's notational and display conventions differ from what we require on assignments and exams.

[4] This is the one 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.

9 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.

9.1 Documenting Your NetID(s)

Put the netids of *all* group members at the top of the page(s) you submit.[6]

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

If you work on paper, there are scanners in Olin and Uris Library. Other possibilities and computing labs equipped with scanners can be found by selecting "Peripherals" in the sidebar and then "Scanners" here.[7] You can also scan your homework with a mobile device.

Your solution must be a single PDF file.[8] It 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. Do not scan at a higher resolution than 300 dpi.[9]

9.3 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.[10] You can also download your submission from CMS to see how it looks.

9.4 ~~Submit on Gradescope, not CMS~~

~~Submit your assignment to Gradescope (not CMS). An account will be created for you with your Cornell NetID email address by Monday February 24th, and with luck earlier than that. (You will receive an email from Gradescope when accounts are set up; check your spam folder if you cannot find it. If you still cannot find it, you can perform a password reset using your Cornell NetID email address.)~~

9.5 Due dates

1. If you are partnering: well before submission, follow the instructions in the "How to form a group" section on the course CMS usage guide, linked to on our Resources page. Both parties need to act on CMS (http://cmsx.cs.cornell.edu) in order for the grouping to take effect.

2. By 2pm on Fri February 28, submit whatever you have done at that point to CMS, following steps 1-3 in the "Updating, verifying, and documenting assignment submission" section of our CMS usage guide. It is OK if you haven't finished working on the files yet; CMS lets you update submissions until the final deadline.[11] [12]

3. By **11:59pm on Fri February 28**, make your final submission.[13]

---

[6] ~~This will help us make sure we transfer numerical scores to CMS correctly, when we eventually transition the scores from Gradescope to CMS.~~

[7] Disclaimer: the CS1110 staff cannot ascertain for you or guarantee that a particular scanner is available or in working order.

[8] For pdf file merging, there is the program PDFtk for Windows and the built-in application Preview for OS X.

[9] If your file is still too large, try SmallPDF to compress it.

[10] Former TA Anthony Poon recommends Genius Scan, which processes images into a flat, perspective-corrected, and evenly-lit PDF.

[11] The 2pm checkpoint on Fri February 28 provides you a chance to alert us during business hours if any problems arise. Since you've been warned to submit early, do not expect that we will accept work that doesn't make it onto 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.

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

[13] The 2pm checkpoint on Fri February 28 provides you a chance to alert us during business hours if any problems arise. Since you've been warned to submit early, do not expect that we will accept work that doesn't make it onto the submission server on time, for whatever reason. There are no so-called "slipdays" and there is no "you get to submit late at the price of a late penalty" policy. Of course, if some special circumstances arise, contact the instructor(s) immediately.

```
# a2.py
# Prof. Lee, cs1110-prof@cornell.edu

import payments

def transfer(source, recip, amt):
    fee = compute_fee(recip, amt)
    if amt <= source.acct and recip.acct + amt - fee >= 0:
        source.acct -=  amt
        recip.acct += (amt - fee)
        recip.service.acct += fee
        return 1
    return 0


def compute_fee(payee, amt):
    rate = payee.service.rate
    return max(rate*amt, payee.service.min)

service1 = payments.Service("Palpay", 1000.0, .02, 7.0)
service2 = payments.Service("VenMoMoney", 400.0, .01, 10.0)
morpheus = payments.Person(1000000.0, service1)
neo = payments.Person(5.0, service1)
trinity = payments.Person(50.0, service2)
num_transactions = 0

num_transactions += transfer(morpheus, trinity, 3000.0)
num_transactions += transfer(trinity, neo, 1.0)
```