CS1110 Summer 2021 — Assignment 3*

Due Saturday 10/3 at 11:59pm

Read this before you begin

We anticipate that this assignment should take a maximum of 2–3 hours so that students are able to successfully complete this assignment and study for Test 1 during overlapping time periods.

Learning Goals The purpose of this assignment is to help you to solidify your understanding of objects and control flow using memory diagramming. Relevant material can be found in Canvas: Visualizing variables (Unit 2); Visualizing function calls (Unit 4), Python Objects (Unit 8), and Conditional Statements (Units 9 and 10).

Python is an object-oriented, procedural programming language. Properly diagramming memory in Python will solidify your understanding of the following central concepts related to the object-oriented and procedural nature of the Python programming language. Specifically, this assignment will let you practice the following:

1. Which statements create variables or change the values of what variables — including global variables, local variables, and object attributes in heap space.
2. That the result of a constructor expression is the ID of the new object that is created.
3. How frames summarize the state of the process of executing a function call, including:
   (a) *variables* that contain store information local to the corresponding function;
   (b) the *program counter*, which records the line number to be executed next;
   (c) *parameters*, i.e., local variables whose purpose is to hold the input values that the function is supplied when called.
   (d) *arguments*, which 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 branches of an `if`-statement are executed in a given setting.

Submission For this assignment, you will submit only one file: the PDF version of your drawing.

Submission within 24 hours after the deadline will be accepted with a 10% late penalty. Even if only one file (of several) is submitted late, the assignment as a whole will be marked late. Assignment submission on Canvas closes 24 hours after the deadline and no further submission will be allowed.

Do not put your name in the files that you will submit, but *do* include your NetID. The reason is to help promote fairness in grading. Thank you for heeding our request, even if it sounds a little strange.

Group work and academic integrity You must work either on your own or with one partner. *If you work with a partner, you must first register as a group in Canvas and then submit your work as a group.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others and seek help from the course staff, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is *certainly* not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, please seek help from the course staff.

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

1 Your Task

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.

Files and Classes The file `a3.py` contains the code you are to work with. This file imports the module `payments` from `payments.py`, which provides two classes: `Person` and `Service`. *You do not need to look at `payments.py`* — we will talk about class definitions later. For this assignment, all you need to know is:

- The class `Service` defines the following attributes:
  - `name`, the name of the service;
  - `acct`, the amount of money that the service provider has earned in fees;
  - `rate`, the percentage of each transaction claimed by the service; and
  - `min`, the minimum service charge.

  Calling the constructor sets these attributes, in this order. So, the constructor expression `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`.
- The class `Person` defines the following attributes:
  - `acct`, the amount of money that the person has;
  - `service`, the service that the person uses.

  Calling the constructor sets these attributes, in this order. So, `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`.

Your submission should consist of a diagram, compliant with the notational conventions given in Section 2, the lectures, and the worked examples in Section 3, showing what happens during the execution of it.

For this assignment, you should not make any changes to the files `payments.py` and `a3.py`.

Submission format You may use the PowerPoint template provided with the assignment, or any other drawing tool to create digital drawings, as long as you can turn your drawing into a PDF. If you prefer, you can also draw your solution by hand, but please make sure that the scan is of good quality. If you do not have a scanner at home, there are several apps available for your phone that can take a picture of a piece of paper and enchance it.

Your submission must be less than 100MB in size, and ideally less than 10MB. If you scanned your work and the file is too large, try lowering the DPI (dots-per-inch) setting on your scanner.

Submissions that are not legible will not be graded.

2 Diagram Conventions

1. Do not draw multiple versions of the same thing. 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. *This should happen each time a variable is reassigned.* Similarly, frames should be crossed out rather than erased, and objects should never be removed. Some of you drew a new box for each line in Assignment 1, which was okay, but for this assignment we want to save you from having to do this each time.

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 value 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.

5. 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.

6. Don't draw the objects (folders) for imported modules or for function definitions.

   Don't draw call frames for built-in functions, such as `int()`.

7. 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. 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.

8. Bare else statements should *not* be treated as executable lines.

9. Choose consecutive `id`s for the objects you create. This is different than what Python Tutor does!

3 Worked Examples

To help solidify the concepts that are exercised in this assignment, we provide below 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. For instance, we will not grade based on 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.

- **Example assignments.** An example of this assignment for a previous semester: assignment and solution.

4 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 `a3.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.