CS1110
Lecture 7: More on function calls

Q: Why is it important to understand the notation for and mechanics of variables, objects, and frames?
A: You get a clear model of what names are accessible and what objects they refer to. Bonus: you’ll understand error messages better.

No, to review: what is a variable (in Python)? A name for referring to a value/ object. Two names can refer to the same thing; example: “that person talking in front of the room” and “the CS1110 prof with black hair”.
What a name refers to can change (hence the name “variable”): “that person talking in front of the room” could refer to the person Prof. Lee at one time, and the person Prof. Marschner at another.

What is an object? An actual thing that can be referred to.

What is an ID? The unique identifier --- “one true name” --- for an object. Each object has a distinct id.

What is a frame? The function’s “local view of the world”: the names it defines and uses locally. These names disappear when the function call finishes.

How evaluate a function call expression, reformatted slightly:

 Uno: Create a frame for the call
 Dos: Assign arguments to parameters
   (a) For each parameter (“the names in parentheses in the function header”), put a variable with that name in the frame
   (b) Evaluate the arguments (“the values of the stuff in parentheses in the function call”)
   (c) Put the argument values in the corresponding parameter variables in the frame. [The potentially hard/new concept embedded here: again, it’s important to distinguish names for things from the things that are named.]
 Tres: Execute function body, updating the frame’s program counter (line number) as you go
 Quatro: Cross out the frame

The value of the function call expression is the returned value (if there is one)

Ex: Can a Python function* change the speed of light?

That is, if lt_speed is a variable, can you write a function violate_physics(...) that changes the value of lt_speed?

*Given the Python we know at this point, where all assignments to a “plain variable” (not expressions with a “dot” in them) within a function are treated as referring to a local variable.

import lec07
lt_speed = 3e8
lec07.violate_physics(...)

import lec07
lt_speed = 3e8
lec07.v_p_try1()

import lec07
lt_speed = 3e8
lec07.v_p_try2(42.0)
If functions are passed the IDs of objects as arguments, then they can "reach out" beyond the frame because they have a "handle" on the object: they can "summon" the object by its "true name".

With that in mind, now let's do the exercise.