

CS1110

The kid spoke. Very squeakily.

"I charge you ... to ... to..." Get on with it! "T-t-tell me your n-name."

That's usually how they start, the young ones. Meaningless waffle. *He knew, and I knew that he knew, my name already; otherwise how could he have summoned me in the first place? You need the right words, the right actions, and most of all the right name.* I mean, it's not like hailing a cab – you don't get just anybody, when you call.

...

"I am Bartimaeus! I am Sakhr al-Jinni, N'gorso the Mighty, and the Serpent of Silver Plumes! I have rebuilt the walls of Uruk, Karnak, and Prague. I have spoken with Solomon....I am Bartimaeus! I recognize no master!"

– *The Amulet of Samarkand*, Jonathan Stroud

CS1110

Lecture 7: More on function calls; ~~if-then-else~~

Announcements

No office hours this week:

They've been replaced by the scheduled one-on-ones.

Readings:

Today: 3.9-3.10 (note that our notation differs slightly) ~~and~~
~~5.1-5.7~~

Next time: ~~5.1-5.7~~ and 10.0-10.2 and 10.4-10.6

Bring this handout to next lecture; we'll do conditionals then
(having decided to slow down the pace a little.)

A Piazza Parable

Starring:

"Student": Prof. Lee

"Professor": Prof. Marschner

Moral: When posting a question on Piazza, please paste in the **exact error messages you get.**

(Don't paste in your code.)

CS1110

The kid spoke. Very squeakily.

"I charge you ... to ... to..." Get on with it! "T-t-tell me your n-name."

That's usually how they start, the young ones. Meaningless waffle. *He knew, and I knew that he knew, my name already; otherwise how could he have summoned me in the first place? You need the right words, the right actions, and most of all the right name.* I mean, it's not like hailing a cab – you don't get just anybody, when you call.

...

"I am Bartimaeus! I am Sakhr al-Jinni, N'gorso the Mighty, and the Serpent of Silver Plumes! I have rebuilt the walls of Uruk, Karnak, and Prague. I have spoken with Solomon....I am Bartimaeus! I recognize no master!"

– *The Amulet of Samarkand*, Jonathan Stroud

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.

So, 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: ~~Erase~~ 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`?

code with function call

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

function "definition" (in `lec07.py`)

```
def violate_physics(...?):
    """Changes lt_speed"""
    1 ...?
```

`lt_speed` 3×10^8

`violate_physics:...`

What does your code for `violate_physics` look like?

- (A) 0 params, 0 local vars
- (B) 1 param, 0 local vars
- (C) 0 params, 1 local var
- (D) ≥ 1 of each
- (E) There can't be such a function

*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.*

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

code with function call

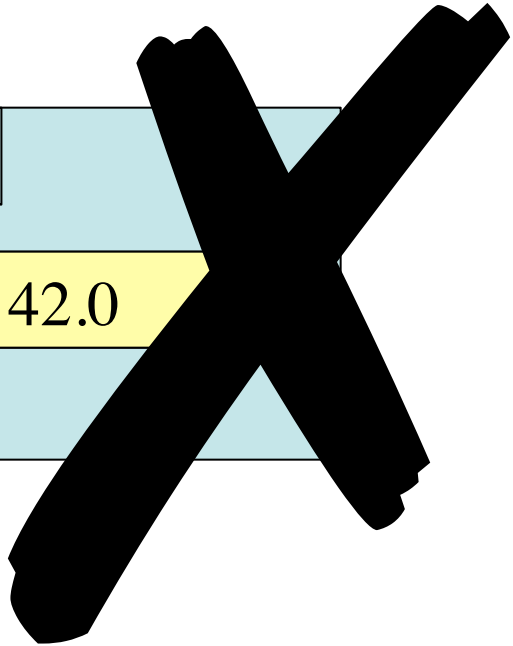
```
import lec07
lt_speed = 3e8
lec07.v_p_try1()
```

function definition

```
def v_p_try1():
    """Changes lt_speed"""
    1 lt_speed = 42.0
```

`lt_speed` 3×10^8

`v_p_try1:1`
`lt_speed` 42.0



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.*

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

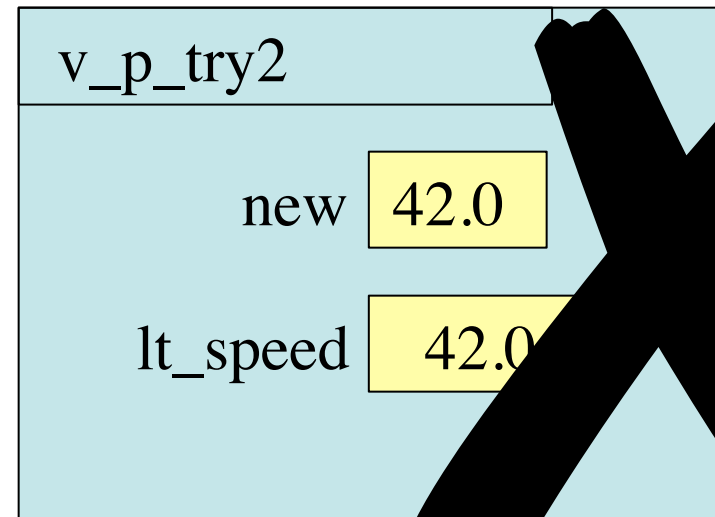
code with function call

```
import lec07
lt_speed = 3e8
lec07.v_p_try2(42.0)
```

function definition

```
def v_p_try2(new):
    """Changes lt_speed to new"""
    1 lt_speed = new
```

`lt_speed` 3×10^8



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.*

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

code with function call

```
import lec07
lt_speed = 3e8
lec07.v_p_try3(lt_speed)
```

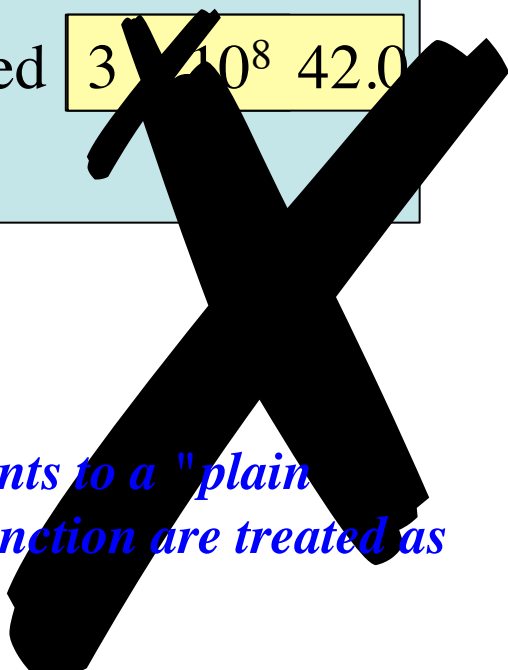
function definition

```
def v_p_try3(lt_speed):
    """Changes lt_speed to 42.0"""
    1 lt_speed = 42.0
```

`lt_speed` 3×10^8

note: only one `lt_speed` in the frame

<code>v_p_try3:1</code>	
<code>lt_speed</code>	3×10^8 42.0



*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.*

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

code with function call

```
import lec07  
lt_speed = 3e8  
lt_speed = lec07.boring(-3e8)
```

function definition

```
def boring(new):  
    """Returns new"""  
    1 return new
```

`lt_speed` ~~3×10^8~~ -3×10^8

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.

How many things are wrong with this picture?

(A) 0-1 (B) 1-2 (C) 3-5 (D) more than 5

(E) You mean besides the fact that you think I can answer this?

[note: the ...distanceFrom... value will be 5.0]

p id1

id1
Point
x 0.0
y 3.0
z 4.0

p_new
Point
x 0.0
y 0.6
z 0.8

rescale: 5

pt p norm 5

pt.x 0.0 pt.y 0.6 pt.z 0.8

FRAME CROSSED OUT