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They've been replaced by the scheduled one-on-ones.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

Slides by D. Gries, L. Lee, S. Marschner, W. White
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.)
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 \(\text{violate\_physics}(...)\) that changes the value of \(lt\_speed\)?

**code with function call**

```python
import lec07
lt_speed = 3e8
lec07.violate_Physics(...)
```

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

```python
def violate_Physics(...):
    """Changes lt\_speed""
    1 ...?
```

*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.

\[ \text{lt\_speed} \quad 3 \times 10^8 \]

What does your code for \(\text{violate\_physics}\) look like?

(A) 0 params, 0 local vars
(B) 1 param, 0 local vars
(C) 0 params, 1 local var
(D) \(\geq 1\) of each
(E) There can't be such a function
That is, if \texttt{lt\_speed} is a variable, can you write a function \texttt{violate\_physics(...)} that changes the value of \texttt{lt\_speed}?

*Given the Python we know at this point, where \textit{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`?

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

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

*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 \texttt{lt\_speed} is a variable, can you write a function \texttt{violate\_physics(...)} that changes the value of \texttt{lt\_speed}?

*code with function call*

\begin{verbatim}
import lec07
lt_speed = 3e8
lec07.v_p_try3(lt_speed)
\end{verbatim}

*function definition*

\begin{verbatim}
def v_p_try3(lt_speed):
    """Changes lt\_speed to 42.0""
    lt_speed = 42.0
\end{verbatim}

*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

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

function definition

```python
def boring(new):
    '''"""Returns new"""''
    return new
```
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]