A1 iscurrency spec change (posted online): for iscurrency:

"Returns: True if <currency> is a valid 3-letter code for a currency,
False otherwise"

Precondition: <currency> is a string."

Want to review last lecture? Additional materials on the course
homepage are: solution slides with animations, and code you can
paste into the Online Python Tutor

Readings for next time: 10.0-10.2, 10.4-10.6, 10.8-10.13
Frames and objects are real

Q: What do these drawings on paper have to do with real programming?

A: Frames, objects, and variables are *exactly* what's being created in Python.

- The Online Python Tutor shows them to you
- The very curious can look at the Python module `traceback`; this is used by our `unittest` module to print frame information.

So it's good to have a notation to talk about them.
The power of the "true name"

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

\[ \text{const\_list} \]

\begin{align*}
\text{lt\_speed} & \quad 3 \times 10^8 \\
\pi & \quad 3.14159 \\
h & \quad 6 \times 10^{-34}
\end{align*}

\text{violate\_physics: 3}

\text{new 42}

\text{constants id3}

"lt\_speed = new" wouldn't change the global \text{lt\_speed}.

But I have the \textit{true name} \text{id3} stored in \texttt{constants}. "\texttt{constants.lt\_speed = new}" \textit{does} change the value in \texttt{id3}.
function definition (in lec07.py)

```python
def new_rescale(pt):
    """Demo.
    Precond: pt is a Point object""

    norm  = 5.0
    pt.y = pt.y / norm
```

code with function call

```python
import point
import lec07
p = point.Point(0,3,4)
lec07.new_rescale(p)  # what does this line do?
```
Stack of frames: When functions call functions

function definitions

```python
def g(m):
    """Returns: energy equivalent of mass m""
    1  E = f(m, lt_speed)
    2  return E

def f(x, y):
    """Returns: x times square of y""
    1  return x * (y**2)
```

lt_speed = 3e8
print g(3)

code with function call

Two "live" frames (stack shown growing downwards, here)
def f(x, y):
    """Returns: x times square of y"""
    return x * (speed ** 2)

def g(m):
    """Returns: energy equivalent of mass m"""
    E = f(m, lt_speed)
    return E

Traceback (most recent call last):
  ... line 2, in <module>: print g(3)
  ... line 1, in g: E = f(m, lt_speed)
  ... line 1, in f: return x * (speed ** 2)
NameError: global name 'speed' is not defined
Conditionals
Example for Conditionals (Valentine's Day Special)

- **id7**: Flower
  - **num_petals**: 3
    - The number of petals. $\geq 0$, initialized to random number.
  - **he_loves_me**: True
    - None if there are no petals. Otherwise, True if the next petal corresponds to "he loves me", False if it corresponds to "he loves me not".