Docstrings

Lecture 7, Week 3
January 24, 2011
CSC108H1S
Velian Pandeliev

Docstrings and `help()`

A docstring (short for documentation string) is a description of what a function does.

Python looks for a free-floating string immediately after the function definition and makes that the output of `help()`.

Docstrings are typically specified using triple-nested single (or double) quotes:

```
'''This is a docstring'''
```

Every function you write should have a docstring.

Rules for Writing Docstrings

Adapted from notes by Diane Horton

Why docstrings matter

Reason 1: If you write a docstring before you write the function, you will know what the function is supposed to do!

Many bugs are due to failure to think this through.

Reason 2: If you write a good docstring, another programmer who calls your function knows everything they need to use it properly.

Important since (a) old code lives a long time and (b) new code rarely written from scratch.

1. Describe precisely what the function does

```
def vowels(word):

'''Returns whether the word has
vowels.'''
```

True if there are vowels or False?

```
def remove_duplicates(s):

'''Removes duplicate characters from
s.'''
```

Does it remove extra occurrences or all?

1. Describe precisely what the function does

```
def vowels(word):

'''Return True iff the string word
has vowels. Do not treat Y as a
vowel.'''


def remove_duplicates(s):

'''Return the string s, except only
the first occurrence of each character
in s is kept.'''
```

2. Do not reveal *how* the function does it.

```
def add_border(pic, c):
    '''Add a border to pic ~~by
    adding 4 overlapping
    rectangles~~.'''
```

3. Make the purpose of every parameter clear.

```
~~def add_border(pic, c):~~

~~    '''Add a border to pic.'''~~
```

```
def add_border(pic, c):

    '''Add a 20-pixel wide border
of colour c to picture pic.'''
```

4. Refer to every parameter by name.

```
~~def average_red(pic):~~
    ~~'''Compute the average amount of red in a picture.'''~~
```

```
def average_red(pic):
    '''Compute the average amount of red in picture pic.'''
```

5. Be clear about whether the function returns a value (and if so, what)

```
~~def average_red(pic):~~

    ~~'''Compute the average amount~~
~~of red in picture pic'''~~

def average_red(pic):

    '''Return the average amount of
red (a float) in the pixels of
picture pic.'''
```

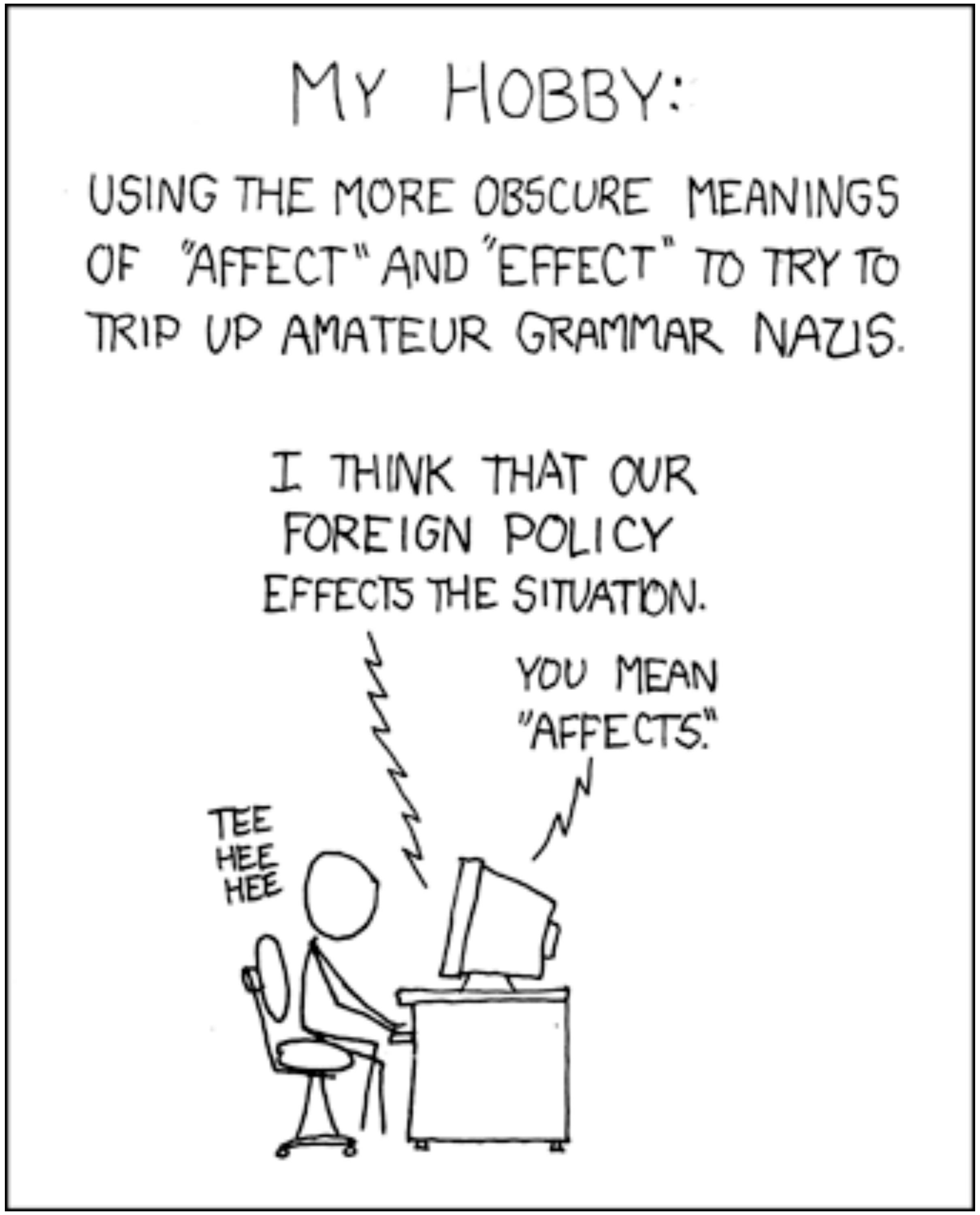
6. Explain any conditions that the function assumes are true.

```
def speed(d, t):
    '''Return a car's speed.'''
```

Does the function assume that time must not be zero? Negative?

```
def speed(d, t):
    '''Return the speed (a float) 
of an object that travels distance 
d in time t. d and t are ints. t is 
non-zero. '''
```

7. Be concise and grammatically correct.


MY HOBBY:
USING THE MORE OBSCURE MEANINGS OF "AFFECT" AND "EFFECT" TO TRY TO TRIP UP AMATEUR GRAMMAR NAZIS.
I THINK THAT OUR FOREIGN POLICY EFFECTS THE SITUATION.
YOU MEAN "AFFECTS."
TEE HEE HEE

8. Write as a command rather than a statement.

```
~~def cube(x):~~
    ~~'''Returns the cube of x'''~~
```

```
def cube(x):
    '''Return the cube of x'''
```

Docstrings for Boolean functions

Boolean functions are ones that return True or False.

Example:

```
def is_odd(n):
    '''Return True if integer n is
odd, and False otherwise.'''
```

We can shorten docstrings by describing the True condition more precisely.

Docstrings for Boolean functions

`odd()` should return True if an integer is odd

`odd()` should NOT return True in any other case

So, we can say:

```
'''Return True if and only if
integer is odd'''
```

This implies that `odd()` returns False in all other cases, saving us the trouble of saying so.

There's a conventional shorthand for this condition: 'if and only if' shortens to 'iff'