# CSC108H Lecture 8

Dan Zingaro

September 26, 2012

# What is True and False?

- We know that `True` is true and `False` is false
- The integer 0, the float 0.0, the empty string, the `None` object, and other empty data structures are considered `False`; everything else is `True`

```
if 0:
  print("hi")
if 4:
  print("bye")
```

# Short-circuit Evaluation

- When an expression contains `and` or `or`, Python evaluates from left to right
- It stops evaluating once the truth value of the expression is known
- For `x and y`, if `x` is `False`, there is no reason to evaluate `y`
- For `x or y`, if `x` is `True`, there is no reason to evaluate `y`
- The value of the expression ends up being the last condition actually evaluated by Python

```
0 and 3 # examples
3 and 0
3 and 5
1 or 0
0 or 1
True or 1 / 0
```

# ConcepTest

In the following expression, which parts are evaluated?

```
(7 > 2) or ( (9 < 2) and (8 > 3) )
```

- A. Only the red code
- B. Only the purple code
- C. Only the blue code
- D. The red code and the blue code
- E. All of the code is evaluated

# ConcepTest

In the following expression, which parts are evaluated?

```
( (7 > 2) and (9 < 2) ) or (8 > 3)
```

- A. Only the red code
- B. Only the purple code
- C. Only the blue code
- D. The red code and the blue code
- E. All of the code is evaluated

# ConcepTest

```
def is_odd(x):
  if x % 2 == 1:
    return True
  else:
    return False
```

Which of the following does exactly the same thing?

- A.

```
def is_odd(x):
  return x % 2
```

- B.

```
def is_odd(x):
  return x % 2 == 1
```

- C.

```
def is_odd(x):
  return x % 2 == 0
```

- D. None of the above

# Tips for Writing Code with Booleans

- If you are unsure of precedence, use parentheses
  - This can help you avoid mistakes and make code more clear
- Use the simplest expression possible
  - e.g. Avoid double negatives like `not not a`, avoid `== True` or `== False`
- Assume `below_zero` is a `Bool`. How can we simplify the following?

```
if below_zero == True:
  print("Brrrr!")
```

# ConcepTest

```
age = int(input("Enter your age: "))
if age < 18:
  print("minor")
elif age >= 18 and age < 30:
  print("adult")
elif age >= 30:
  print("older than Dan")
else:
  print("ageless")
```

What code can be removed without changing what the code does?

- A. The red can be removed
- B. The blue can be removed
- C. Both red and blue can be removed
- D. Nothing can be removed

# Docstrings: Good Descriptions

- We haven't talked much about step 4 of the design recipe
- In the description, you are to describe precisely what the function does
- Do **not** reveal how the function does it
- Make the purpose of every parameter clear
- Refer to every parameter by name
- Be clear about whether the function returns a value, and if so, what it returns
- Explain any conditions that the function assumes are true (e.g. `n` is an `int`)
- Write the description as a command (e.g. `Return the first ...`) rather than a statement (e.g. `Returns the first ...`)

# ConcepTest

```
def speed(distance, time):
  '''(float, float) -> float
  xxx description xxx

  >>> speed(20.0, 10.0)
  2.0
  '''
```

What is a good description for this docstring?

- A. Return a car's speed.
- B. Given an object's distance and time of travel, return its speed.
- C. Calculate the speed given the distance and time.
- D. Divide distance by time to return the speed.

# Boolean-Returning Functions

- "A iff B" means "(A if B) and (A only if B)"
- We assume that if a boolean function does not return `True`, it returns `False`

Here's a description for the `if_odd` function from earlier:

```
def is_odd(x):
  '''...

  Return true iff x is odd.
  ...
  '''
```