

# Compound boolean expressions and practice

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 08, Fall 2020

Friday September 11

# Project 3 overview (1/2)

- Treat one of your space scenes from Project 2 as a "compound shape": add ability to scale and position it.
- Make museum scene: includes mini versions of your space and Mondrian scenes (picture frames, view outside window, sculptures, patterns on carpet, etc.)
- Possible due to **encapsulation**: functions enable you to draw simple shapes, compound shapes, and your scenes *independent* of one another (**and** at any scale and position).

# Project 3 overview (2/2)

- Use for loops for all shapes, no repetitive code.
- Use best practices for code organization (e.g. headers, docstrings, inline comments, etc.)
- Using command-line arguments is **optional**. If you do, please describe in docstrings and report what command-line arguments are and how to run your code to produce your scene.
- Add randomness the change aspect of compound shapes (e.g. color).

Let's find some bugs related to  
positional and keyword arguments

Let's work on Question 1 of the  
conditional worksheet

# More on boolean operators

# Compound boolean operations

Allow us to check if multiple conditions are true

| Operation | Result                                                                               |
|-----------|--------------------------------------------------------------------------------------|
| not x     | True if x is false, False if x is true                                               |
| x and y   | True only if <i>both</i> x and y is true, otherwise False                            |
| x or y    | True if <i>at least one of</i> x or y is true<br>False if they are <i>both</i> false |

```
if x > 10 and y < 5:  
    # do stuff
```

# Precedence Rules: Higher in the table get evaluated earlier

```
if x > 10 and y < 5 or z == 2:  
    # do stuff
```

| Operation            | Result                                                                               |
|----------------------|--------------------------------------------------------------------------------------|
| math operators       | * / + - etc                                                                          |
| relational operators | < > == != etc                                                                        |
| not x                | True if x is false, False if x is true                                               |
| x and y              | True only if <i>both</i> x and y is true, otherwise False                            |
| x or y               | True if <i>at least one of</i> x or y is true<br>False if they are <i>both</i> false |

# Lazy evaluation (1/2)

If there is more than one of the same compound boolean operator in a conditional statement, **Python evaluates left-to-right**.

In a long string of `and` conditions, Python stops checking and considers the entire expression `False` if it encounters one `False`:

```
if a and b and c and d and e and f and g:  
    print('yes')
```

If `a` is `False`, Python does not even bother checking `b`, `c`, etc.

# Lazy evaluation (2/2)

In a long string of `or` conditions, Python stops checking and considers the entire expression `True` if it encounters one `True`:

```
if a or b or c or d or e or f or g:  
    print('yes')
```

If `a` is `True`, Python does not even bother checking `b`, `c`, etc.