

Object deep and shallow copy

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 14, Fall 2020

Friday September 25

Finish Turtle object symbol table
example

Shallow and deep copies of objects

Shallow copy (1/2)

Scenario: You make and customize a `Turtle` object and want to duplicate (**copy**) the color, pen width, shape, etc to a second `Turtle` object, but don't want to re-type all the commands.

```
turt1 = turtle.Turtle()  
turt1.color('red')  
# etc
```

Shallow copy (2/2)

```
turt1 = turtle.Turtle()  
turt1.color('red')  
turt2 = turt1  # This does a shallow copy.  
# NOT what you want in above scenario
```

- A **shallow copy** creates a *nickname* — two names for the same data.
- Change turt2 color to red? turt1 also becomes red.
- Change turt1 shape to a turtle? turt2 also gets a turtle shape.
- Let's look at example code then show what happens in the symbol table.

Deep copy (1/2)

Often you want an *independent* (**deep copy**) copy of an object: all data are copied from object to another. Modifying one does not modify the other.

- Means we make a NEW symbol table for 2nd object. Copy all entries of one symbol table to the other.
- Also called **cloning** an object.

Deep copy (2/2)

```
turt1 = turtle.Turtle()  
turt1.color( 'red' )  
turt2 = turt1.clone()  # turt2 is a deep, independent copy
```

- `turtle` provides the `clone` method to do a deep copy (e.g. of `turt1`).
- Changes to `turt2` will not affect `turt1`.
- Let's look at example code then show what happens in the symbol table.