Daily Coding Problem #174

Problem

This problem was asked by Microsoft.

Describe and give an example of each of the following types of polymorphism:

- Ad-hoc polymorphism
- Parametric polymorphism
- Subtype polymorphism

Solution

Ad-hoc polymorphism is a type of polymorphism where you can implement and overload functions based on different types.

For example, the + operator in Python behaves differently when it's given a number vs. when it's given two strings. That's because it's been overloaded:

- For numbers, we need to do addition
    - For integers, we can just do integer addition
    - For floats, we must use floating-point
- For strings, we need to do string concatenation
- For lists, we need to do list concatenation

```
> 3 + 3

5
```

```
> "foo" + "bar"

"foobar"
```

Parametric polymorphism, on the other hand, has a generic implementation independent of types. That's why it is called parametric -- the type is parameterized.

For example, the append function of an array works generically on integers or strings.

```
[1, 2, 3].append(4)

["foo", "bar"].append("baz")
```

Subtype polymorphism is when we subclass a class and use each subclass' methods to implement a specific interface (or use the default interface). Because the interface is consistent it should still work regardless of whichever type of object we use.

For example, say we have a generic `Shape` class that might draw a shape to the screen. Then we might also have `Circle` and `Square` subclasses, each with their own `draw` method implemented differently, since they're different objects. It shouldn't matter -- a `render_scene` function should still be able to call `draw()` on any kind of shape and have it still work.

```
class Shape:
    def draw(self, coords):
        print("Drawing shape at", coords)

class Circle(Shape):
    def draw(self, coords):
        print("Drawing circle at", coords)

class Square(Shape):
    def draw(self, coords):
        print("Drawing square at", coords)

def render_scene(objs, coords):
    for shape, coord in zip(objs, coords):
        shape.draw(coord)
```