LECTURE 25

OBJECT-ORIENTED PROGRAMMING

MCS 260 Fall 2021

David Dumas

REMINDERS

- Thurs lab students attempt worksheet problem 1.
- Project 3 description release soon.
- Project 2 grades posted.
- Please read Project 2 feedback and solution
- Aggregate columns added to gradebook: Hwk avg, Lab avg, Proj avg, midterm grade. Contact me if you have questions or believe there is an error.

CUSTOM TYPES IN PYTHON

In Python, **classes** are the way to define your own types. A value of that type is an **object** or **instance**.

Analogy: class "Cat", instance "Mr. Mittens".

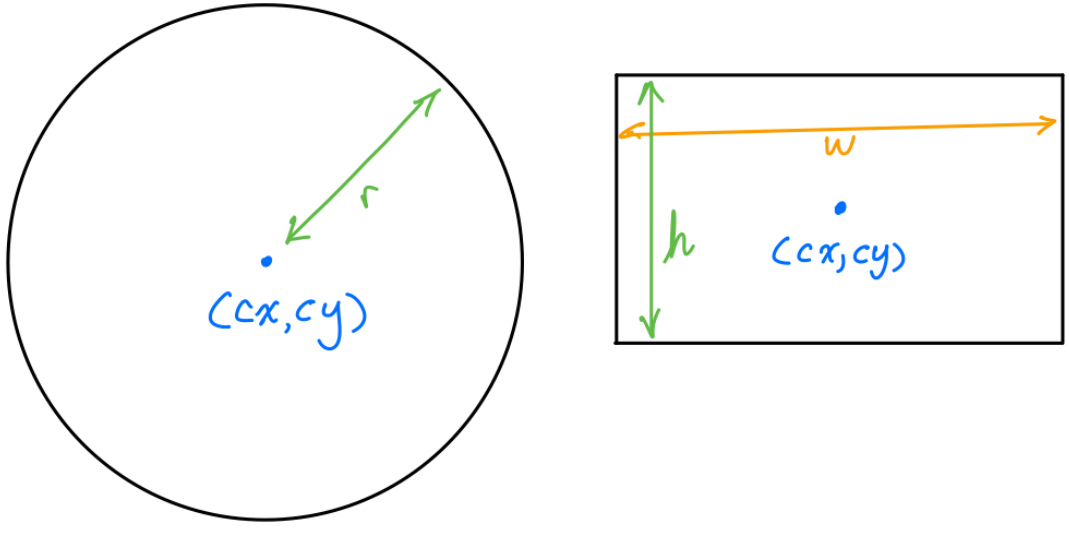
Objects bundle together **data** and **behavior** (things you can do with a specific sort of data).

SAMPLE PROBLEM

Suppose we are writing programs that will work with geometric objects in the plane, such as circles and rectangles.

How should we represent these objects as numeric data?

REPRESENTATION



But what type should we use? `list`, `tuple`, `dict`?

CLASSES

We can create our own type called **Circle**, using a **class** definition.

By convention class names LookLikeThis (capitalized words with no separator).

Classes are can contain internal variables, called **attributes**.

Classes can contain their own functions, called **methods**.

`Circle()` will create a new object of type `Circle`.

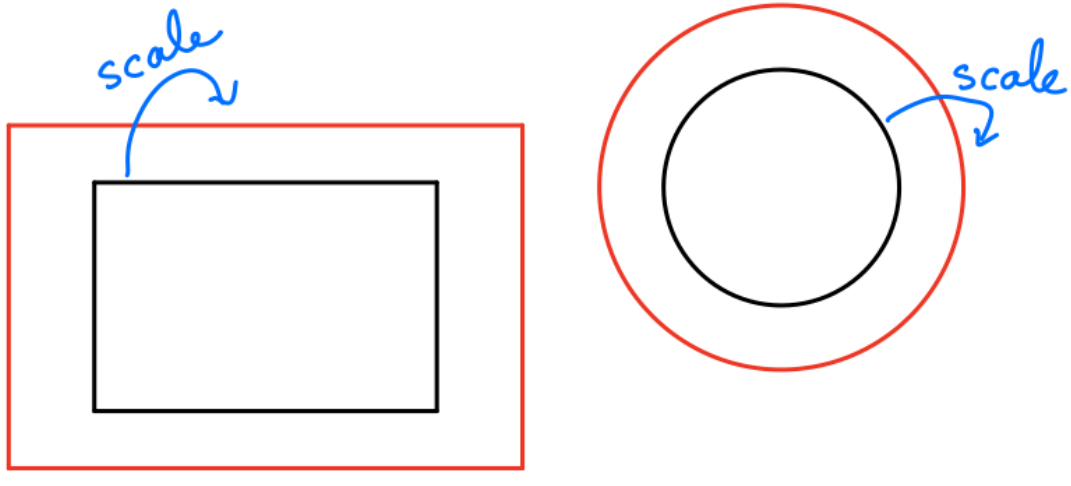
CLASS DOCSTRINGS

The first statement inside a class definition can be a string literal.

If so, that string is the class's docstring.

ACTING ON OBJECTS

Now imagine that our program needs to change the size of the objects, while keeping them in the same position (e.g. increase all sizes by 25%.)



How might we do that?

FUNCTION APPROACH

We could create functions that modify the objects:

```
circle_scale(circle,factor)
rectangle_scale(rectangle,factor)
```

METHODS

Notice the functions we just defined take an object as the first argument and modify it in some way?

This is so common that there is a language feature just for this purpose.

A **method** is a function that is defined inside a class, and which is then attached to every instance of it.

We could e.g. define a `scale` method so that we can call `C.scale(1.25)` to scale an object `C` of type `Circle`.

IMPORTANT NOTE

Method calls look like this: `C.scale(1.5)`

What happens is: `Circle.scale(C, 1.5)`

Python adds the object to the beginning of the argument list!

__INIT__

For a class `Circle`, when we call `Circle()` we are actually running a special method called the **constructor**. It sets up a new object for us.

There is a default constructor that doesn't do very much.

We can define our own constructor by naming a method `__init__(self,...)`.

__STR__

When Python needs to convert an object to a string, it calls the `__str__(self)` method, if it exists.

Define this and return a string that is a human-readable representation of what the object is.

REFERENCES

- In *Downey*:
  - Chapter 17 discusses classes, objects, and methods

REVISION HISTORY

- 2021-10-20 Initial publication
- 2021-10-20 Correct project 3 release date