# LECTURE 23

## OBJECT-ORIENTED PROGRAMMING

MCS 260 Fall 2020

Emily Dumas

# REMINDERS

- Worksheet 8, Quiz 8 available
- Project 3 description release today
- Project 2 grades Saturday

# 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".

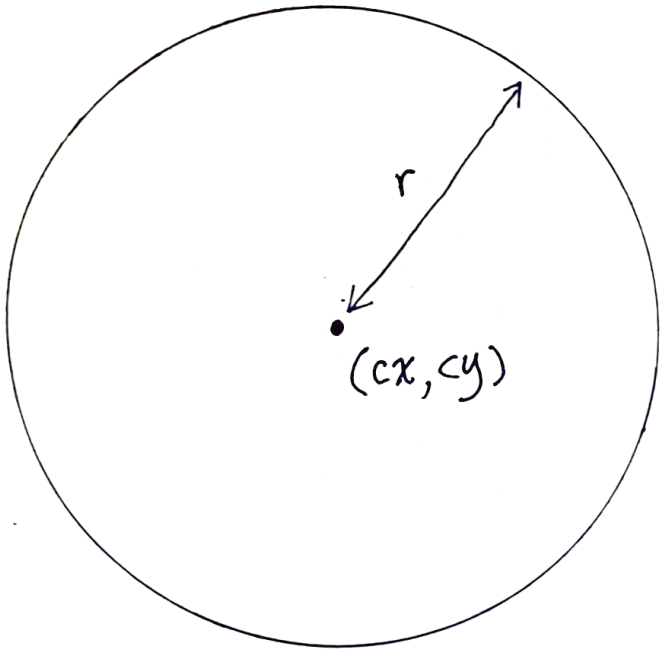
Objects bundle together **data** and **behavior** (things you can do with a specific sort of data).

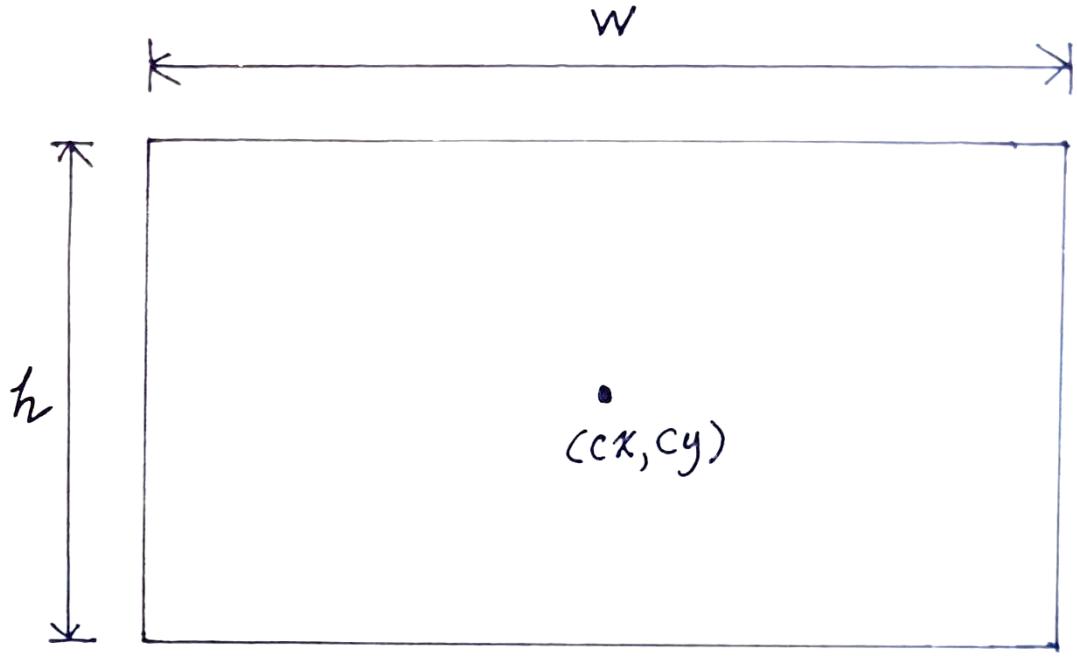
# SAMPLE PROBLEM

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 mutable and can contain named **attributes**, which are like variables.

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

# ACTING ON OBJECTS

Now imagine that moving geometric objects right and left (x-translation) is important in our program. How would we do it?

We could create functions that modify the objects:

```
circle_translate_x(circle,delta_x)
rectangle_translate_x(rectangle,delta_x)
```

# 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 then exists in each object of that type using syntax like `C.translate_x(dx)` if `C` is an object of type `Circle`.

# IMPORTANT NOTE

A method is called like this: `C.translate_x(dx)`

But it receives argument list: `(C, dx)`

**Methods always receive the object as their first argument!**

# __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.

If we define a method `__init__(self,...)` in a class, it becomes the constructor.

# `__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

- 2020-10-15 Initial publication