Lecture 19: Object-Oriented Programming

CS 51P

November 15, 2023

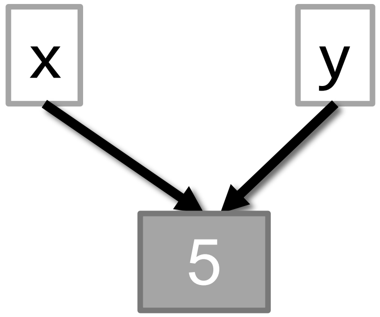
Types in Python

Primitive Types

- int
- float
- bool

```
x = 5
y = 5
```

```
>>> x == y
True
>>> x is y
True
```



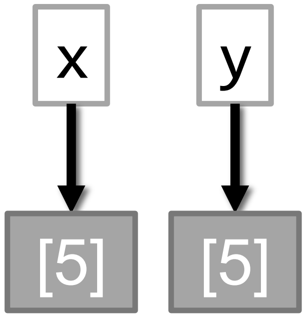
Objects

- list
- dictionary

- Create your own…

```
x = [5]
y = [5]
```

```
>>> x == y
True
>>> x is y
False
```



class: programmer-defined type

- Defining a type:
  - **Step 1:** how would you describe it? what distinguishes one object of this type from another?

- Example: Classroom type
  - attributes: building, room number, capacity, accessible

Syntax: Defining a Class

name of your new type

```
class Classroom:
    def __init__(self):
        self.building = None
        self.room_number  = None
        self.capacity = None
```

class attributes

Creating and using a class

```
room1 = Classroom()

room1.building = "Edmunds"
room1.room_number = "114"
room1.capacity = 40

print(room1.bulding, room1.room_number)
print(room1.capacity)
```

Exercise 1

- Define a class `Rectangle` with attributes width and height and method `__init__`

- Define a function `create_rect(w,h)` that takes two arguments `w` and `h`, creates a rectangle with width `w` and height `h`, and returns that rectangle.

Special methods

special methods have double underscores in name

- __init__
  - constructor
  - called when you create an object

self refers to this instance. always the first parameter.

```
def __init__(self, building, room, capacity):
        self.building = building
        self.room_number = room
        self.capacity = capacity
```

self.variable_name refers to instance attributes (i.e., variables)

- __str__
  - called when you print an object

all methods have self as the first parameter even if they have no other parameters

```
def __str__(self):
    return self.building + self.room_number
        + ", capacity " + str(self.capacity)
```

Exercise 2

- Add a second constructor to your class `Rectangle` that takes three parameters (self, width, and height).
- Add a __str__ method to your class Rectangle so that the following code:

```
my_rectangle = Rectangle(47, 4)
print(my_rectangle)
```

prints

```
47x4
```

class: programmer-defined type

- Defining a type:
  - **Step 1:** how would you describe it? what distinguishes one object of this type from another?
  - **Step 2:** what can an object of this type do?

- Example: Classroom type
  - attributes: building, room number, capacity, accessible
  - methods: find current attribute values, change capacity, check capacity

Additional Methods

Functions defined in a class are called **methods**

```
class Classroom:
    def __init__(self, building, room, capacity):
        self.building = building
        self.room_number = room
        self.capacity = capacity

    def __str__(self):
        return(self.building + self.room_number +
              ", capacity " + str(self.capacity))

    def get_building(self):
         return self.building

    def get_room_number(self):
         return self.room_number

    def set_capacity(self, capacity):
         self.capacity = capacity

    def check_capacity(self, num):
        return num <= self.capacity
```

methods that return the current value in an attribute are called **getter** or **accessor** methods

methods that modify the current value in an attribute are called **setter** or **mutator** methods

Example

Write a function `enough_space` that takes two parameters: `rooms` (a `list` of `Classrooms`) and `num_people` (`int`). The function should return a list of rooms that have capacity greater than or equal to `num_people`.

Write a main function that creates a list of two classrooms and then calls `enough_space` with that list and prints the results.

Exercise 3

- Modify your class `Rectangle` to add an additional method area that returns the area of the rectangle

- Write a `main` function that creates two rectangles, uses the area method to compute the area of each and then prints which one is bigger