Classes

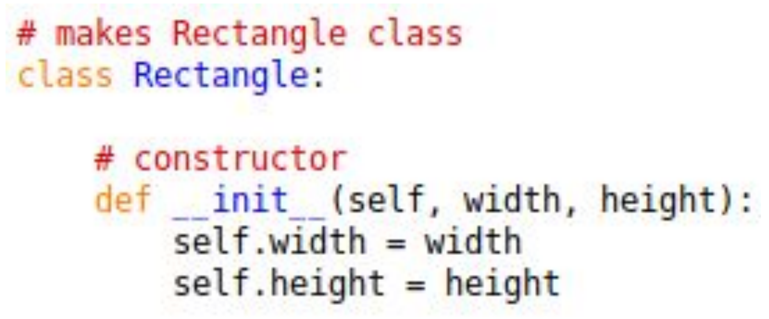
Fall 2019

Classes

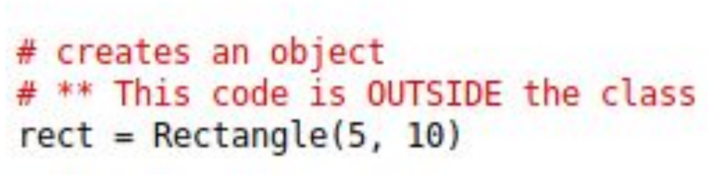
- Class - code that specifies data attributes and functions for a specific object
- Classes ***must*** have a constructor
    - What code outside of the class calls to create an object
    - Kind of like creating a string or int
        - Instead, you need to call a function (the class name)
    - Constructors are ***always*** named __init__ (underscores matter!!)
- Initialized by keyword “class” and whatever name you want to give your class
    - *Typically* class names are capitalized

Class Anatomy

- Basic class is made
- No functions other than a constructor yet
- ***EVERY*** function **in the class** has a parameter of self (as the *first* parameter)
    - When calling the function, self is not passed
    - It's merely an identifier for the class
    - It *must* be named "self"
- Any attributes of the class *must* be referenced via "self.variableName"



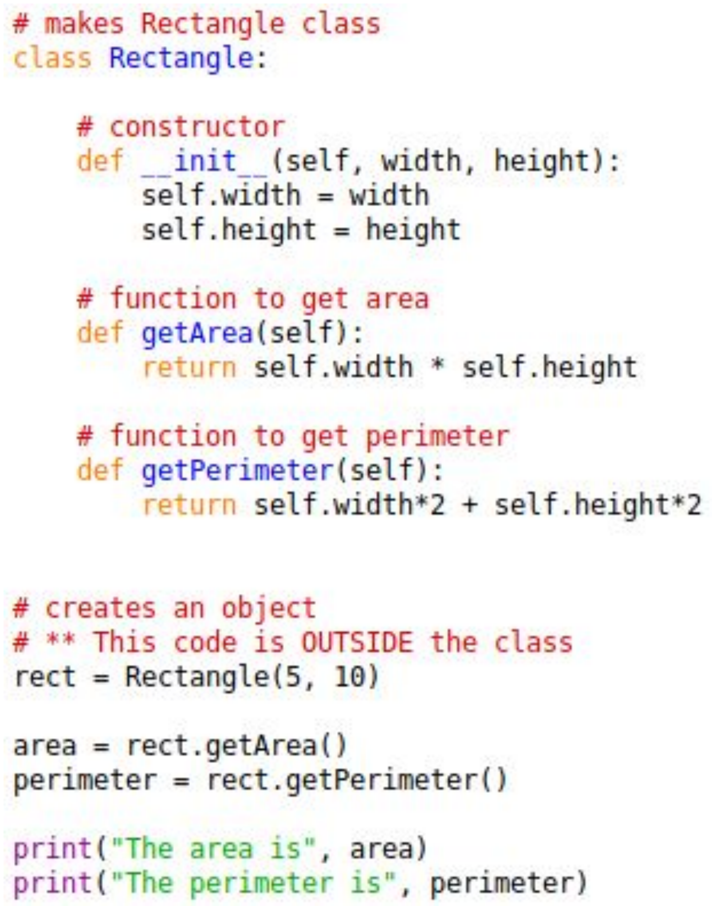
```
# makes Rectangle class
class Rectangle:

    # constructor
    def __init__(self, width, height):
        self.width = width
        self.height = height
```



```
# creates an object
# ** This code is OUTSIDE the class
rect = Rectangle(5, 10)
```

Class Functions

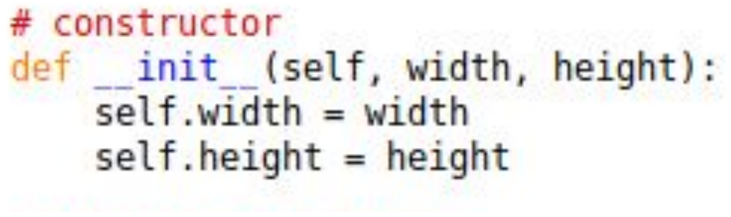
- Functions made in the class can be called once an object is created
- To call it, you just need to do “object.functionCall()”
    - Replace object with whatever you named your object
    - Replace functionCall with whatever function you want to call



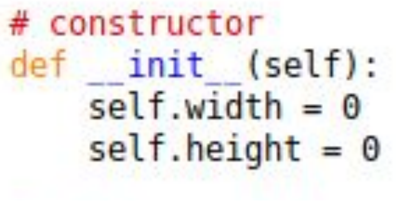
```python
# makes Rectangle class
class Rectangle:

    # constructor
    def __init__(self, width, height):
        self.width = width
        self.height = height

    # function to get area
    def getArea(self):
        return self.width * self.height

    # function to get perimeter
    def getPerimeter(self):
        return self.width*2 + self.height*2


# creates an object
# ** This code is OUTSIDE the class
rect = Rectangle(5, 10)

area = rect.getArea()
perimeter = rect.getPerimeter()

print("The area is", area)
print("The perimeter is", perimeter)
```

Constructors: Notes

- Constructors must initialize all of your data attributes
    - If you’re not passing any parameters, set the data attributes you’re going to use to an empty value
        - 0 for floats/ints
        - “” for strings
    - This makes them accessible throughout your *entire* class (which is the goal)



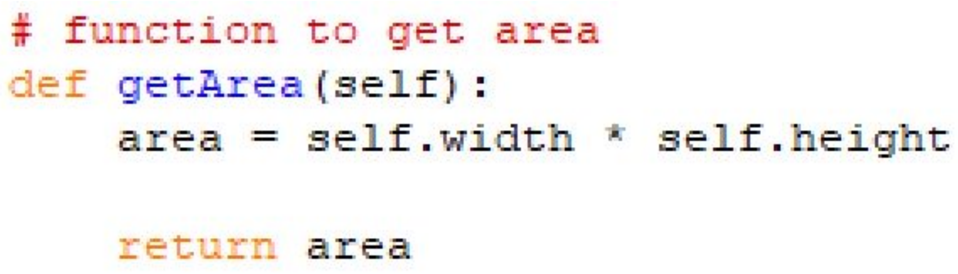
```python
# constructor
def __init__(self, width, height):
    self.width = width
    self.height = height
```



```python
# constructor
def __init__(self):
    self.width = 0
    self.height = 0
```

Variables: Notes

- *Only* data attributes are referenced via “self”
- Any function that creates a variable to be used inside of it doesn’t need to use self
- In this example:
  - area → variable the function created
  - width & height → data attributes the class created



```
# function to get area
def getArea(self):
    area = self.width * self.height

    return area
```

```python
# makes Rectangle class
class Rectangle:

    # constructor
    def __init__(self):
        self.width = 0
        self.height = 0

    # setter = width
    def setWidth(self, width):
        self.width = width

    # setter = height
    def setHeight(self, height):
        self.height = height

    # getter = width
    def getWidth(self):
        return self.width

    # getter = height
    def getHeight(self):
        return self.height

    # function to get area
    def getArea(self):
        return self.width * self.height

    # function to get perimeter
    def getPerimeter(self):
        return self.width*2 + self.height*2
```

```python
# creates an object
# ** This code is OUTSIDE the class
rect = Rectangle()

# sets width/height
rect.setWidth(10)
rect.setHeight(5)

area = rect.getArea()
perimeter = rect.getPerimeter()

print("The area is", area)
print("The perimeter is", perimeter)

# gets width/height
print("The width is", rect.getWidth())
print("The height is", rect.getHeight())
```