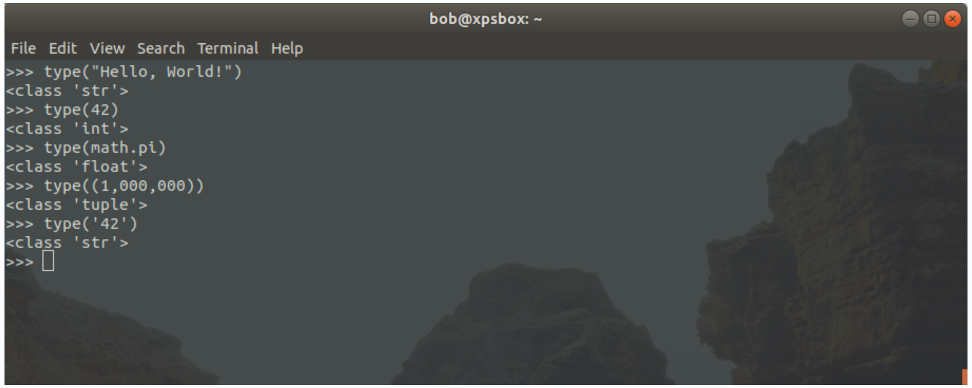
EGM722 – Programming for GIS and Remote Sensing

Week 2, Part 4: Classes and Objects

The world is filled with objects

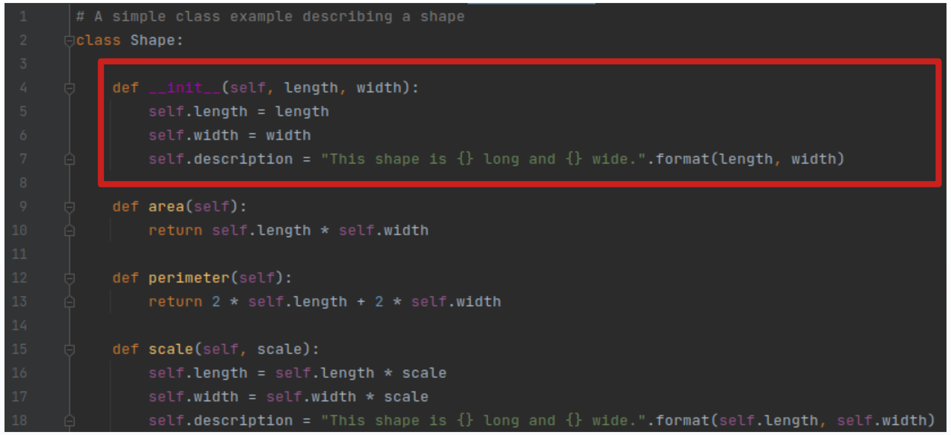
- Python is an object-oriented programming language
- Object: the basic “thing” that python works with
- Objects have:
  - type
  - properties
  - methods



```
>>> type("Hello, World!")
<class 'str'>
>>> type(42)
<class 'int'>
>>> type(math.pi)
<class 'float'>
>>> type((1,000,000))
<class 'tuple'>
>>> type('42')
<class 'str'>
>>> 
```

Classes

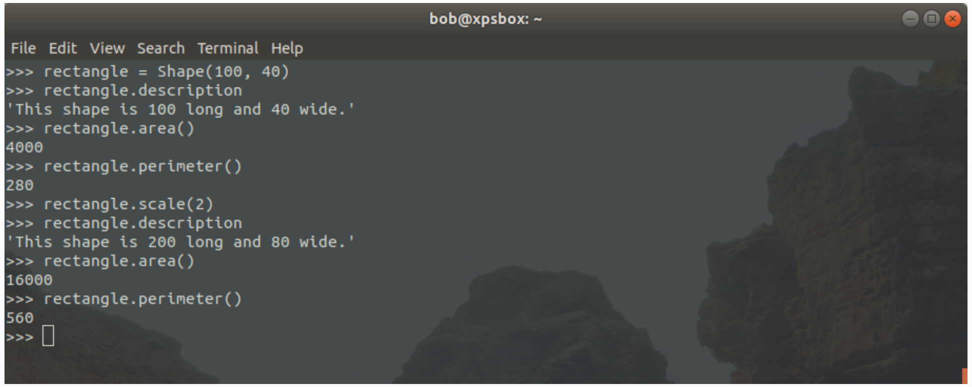
- Class: a blueprint that tells python how to create an object
  - Defines methods
  - Sets/describes attributes
- The __init__() method tells python how to “build” the new object



```python
# A simple class example describing a shape
class Shape:

    def __init__(self, length, width):
        self.length = length
        self.width = width
        self.description = "This shape is {} long and {} wide.".format(length, width)

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * self.length + 2 * self.width

    def scale(self, scale):
        self.length = self.length * scale
        self.width = self.width * scale
        self.description = "This shape is {} long and {} wide.".format(self.length, self.width)
```

Attributes and methods

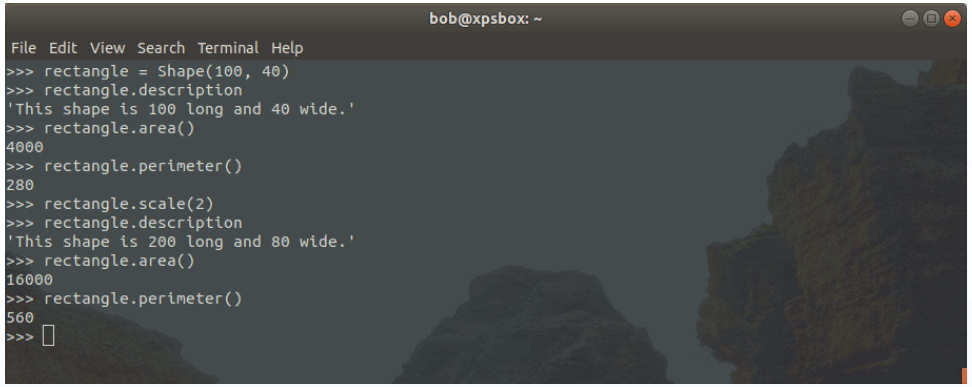
- Recall: a method is a function that operates on an object
  - e.g., str.upper()
- Attributes are variables that belong to an object
  - Instance attributes: belong only to that instance
  - Class attributes: shared by all instances of a class



```
>>> rectangle = Shape(100, 40)
>>> rectangle.description
'This shape is 100 long and 40 wide.'
>>> rectangle.area()
4000
>>> rectangle.perimeter()
280
>>> rectangle.scale(2)
>>> rectangle.description
'This shape is 200 long and 80 wide.'
>>> rectangle.area()
16000
>>> rectangle.perimeter()
560
>>> 
```

Using the class

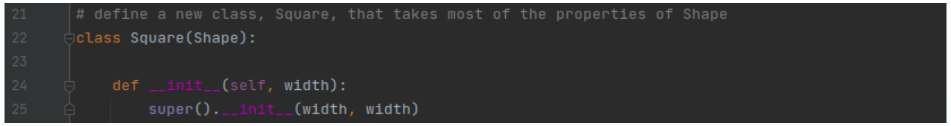
- Note: this only provides the blueprint
- To use the class:
  - Pass parameters x, y
- Question: is this class mutable?



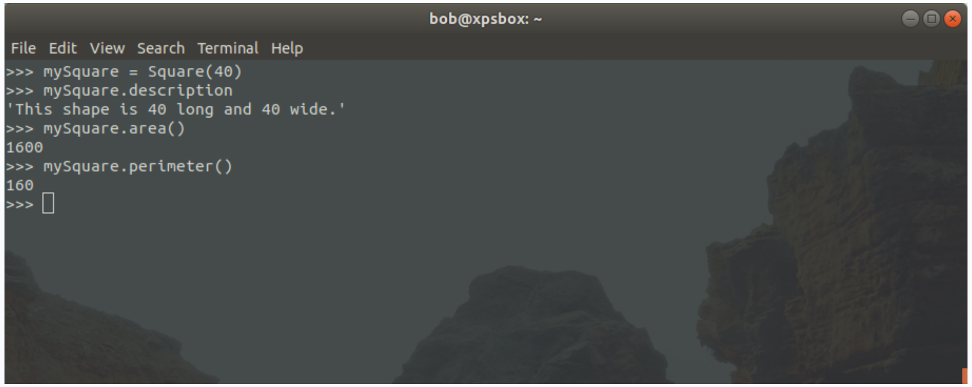
```
>>> rectangle = Shape(100, 40)
>>> rectangle.description
'This shape is 100 long and 40 wide.'
>>> rectangle.area()
4000
>>> rectangle.perimeter()
280
>>> rectangle.scale(2)
>>> rectangle.description
'This shape is 200 long and 80 wide.'
>>> rectangle.area()
16000
>>> rectangle.perimeter()
560
>>> 
```

Inheritance

- Let’s define a new class, Square
- Could write everything from scratch
  - But there’s a lot of overlap
- Inheritance: define new class by modifying existing class
- Child class: a class that inherits from parent class
  - Child class has attributes, methods of parent class
  - Only write methods that need to be different



```python
# define a new class, Square, that takes most of the properties of Shape
class Square(Shape):

    def __init__(self, width):
        super().__init__(width, width)
```



```
>>> mySquare = Square(40)
>>> mySquare.description
'This shape is 40 long and 40 wide.'
>>> mySquare.area()
1600
>>> mySquare.perimeter()
160
>>> 
```

Summary

- Python is an object-oriented programming language
- A class provides a blueprint for python to create objects
- Class defines attributes and methods for the object
- Inheritance allows us to create new classes without starting from scratch