Python and OOP : Class Objects

Overview • 11.14.2016

Review Topics for Python Class Objects

- Class objects and Data Science
- Basics: class, object, instance, self, init, main
- Basics cont'd: attributes, methods
- Object consistency and inheritance
- Private vs Public methods
- Bound vs Unbound
- DRY as a framework

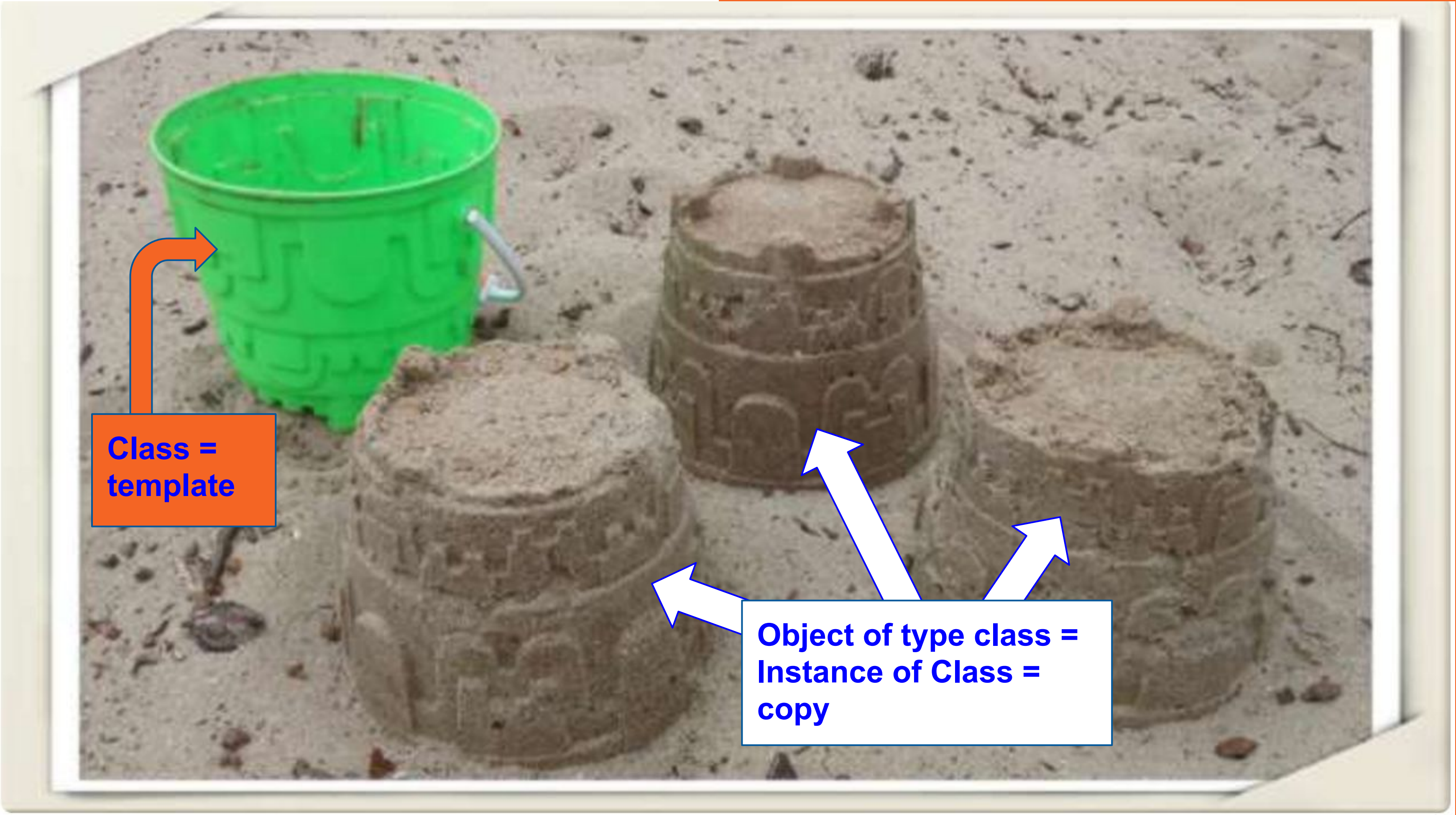
OOP and Data Science - Why?

It helps to use tools!

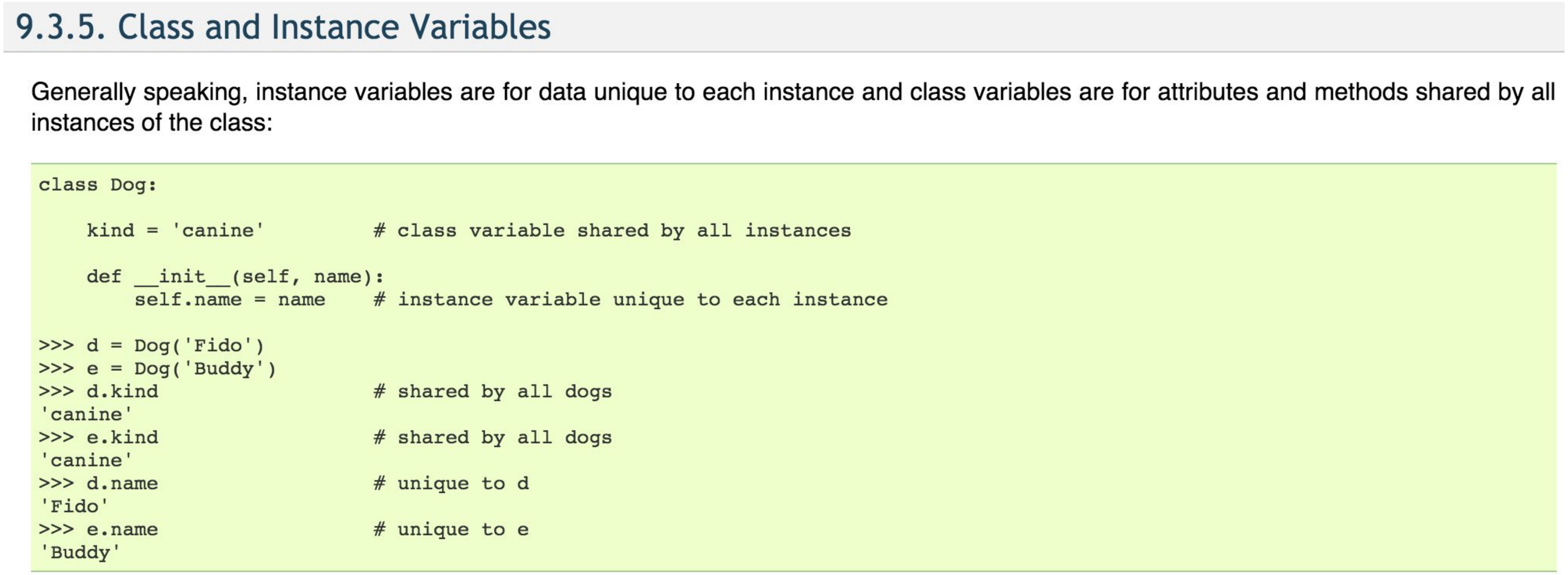
- Better, or rather more fundamental understanding, of how ML algorithms work
- Makes handling data and related tasks much easier: Object-oriented programming is all about grouping data and functionality together.

Check out these links:

- https://www.quora.com/Should-I-learn-object-oriented-programming-concepts-in-Python-as-a-data-scientist
- http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html


Class =
template
Object of type class =
Instance of Class =
copy

Source: https://docs.python.org/2/tutorial/classes.html
(Kelly cue: show jupyter notebook)



9.3.5. Class and Instance Variables

Generally speaking, instance variables are for data unique to each instance and class variables are for attributes and methods shared by all instances of the class:

```
class Dog:

    kind = 'canine'         # class variable shared by all instances

    def __init__(self, name):
        self.name = name    # instance variable unique to each instance

>>> d = Dog('Fido')
>>> e = Dog('Buddy')
>>> d.kind                  # shared by all dogs
'canine'
>>> e.kind                  # shared by all dogs
'canine'
>>> d.name                  # unique to d
'Fido'
>>> e.name                  # unique to e
'Buddy'
```

Class Objects - Basics

self

- ‘self’ is a parameter that refers to...the instance, of course!
- See accompanying jupyter notebook for further clarification

__init__

- Initializes the object and makes it ready to use
- Class needs to be designed to start-up object in a fully-initialized state

__main__

- The name of the module that your script is running in. When a python file is executed it is assigned a name. The initial file that is executed via "python file.py" is assigned the name "main" and it is stored under the variable __name__.
- if __name__ == "__main__"

⇒ is used to execute some code only if the file was run directly, and not imported.

Class Objects - Concept of Inheritance

- Inheritance is the process by which a "child" class derives the data and behavior of a "parent" class.
- the class inheritance mechanism allows multiple base classes
- a derived class can override any methods of its base class or classes
- a method can call the method of a base class with the same name

Class Objects - Basics

Attributes

- There are two kinds of valid attribute names, data attributes and methods.
- Data attributes need not be declared; like local variables, they spring into existence when they are first assigned to.

Methods

- Methods are attributes
- A function defined in a class is called a "method".
- Methods have access to all the data contained on the instance of the object; they can access and modify anything previously set on self.

Class Objects - Basics

Class-level vs Instance-level

- Class attributes are attributes that are set at the class-level, as opposed to the instance-level.
- Normal attributes are introduced in the __init__ method, but some attributes of a class hold for all instances in all cases.
    - Because they use self, they require an instance of the class in order to be used. For this reason, they're often referred to as "instance methods".
- See accompanying jupyter notebook for example.

DRY ⇒ Don’t Repeat Yourself

- One of the most important rules of programming (in general, not just when dealing with objects) is "DRY" or "Don't Repeat Yourself.
- If you are defining several classes and find that they differ only by a single character, then they probably share data and functionality in common.
- Find the underlying notion
    - Example: class Vehicles as opposed to class Trucks and class Cars.
    - Look up DRY in this article: https://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-programming/

Public vs Protected vs Private Methods

http://radek.io/2011/07/21/private-protected-and-public-in-python/

(Slide will be updated with highlights for this topic. But for now, check out the article! Does a great job of explaining...)

Things to think about as you complete Project 2

1. Inheritance
2. Public vs Private Methods
3. DRY

Resources that helped me:

- Overview: https://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-programming/
- Official Python Tutorial and Documentation over Class Objects: https://docs.python.org/2/tutorial/classes.html
- Private vs Public Methods: http://radek.io/2011/07/21/private-protected-and-public-in-python/