What is OOP? (Object-Oriented Programming)

Objects have two parts

- Attributes
- Methods (Behavior)

Example: Cow is an object,

Attributes of a cow: name, breed, size

Behavior of a cow: eating, mooing, sleeping

What is OOP? (another viewpoint)

Objects have two parts ← NOUNS

- Attributes ← ADJECTIVES
- Methods (Behavior) ← VERBS

Example: Cow is an object,

Attributes of a cow: name, breed, size

Behavior of a cow: eating, mooing, sleeping

What is OOP? (Object-Oriented Programming)

Classes are used to define the object and create a template for the object you're creating

Class definition is like a cookie cutter

The object is the cookie

Each cookie cutter can generate many distinct cookies

Methods

Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

These are like functions that belong to (or “are bound to”) a class

Attributes

Attributes are values (data) that are defined inside the body of a class.

Attributes are bound to each individual object.

The __init__() statement defines required attributes

The structure of a class definition

```
class Cow:
    species = "moo cow"
    def __init__(self,name):
        self.name = name
    def get_species( self):
        return self.species
```

The class definition

Class level attribute

Initialization statement

Instance level attribute

Class method

This is returned when the method is run

The structure of a class definition

```
class Cow:
    species = "moo cow"
    def __init__(self,name):
        self.name = name
    def get_species( self):
        return self.species
```

"self" brings the current object into the method

A required argument

Instantiation and use of the class

We make the cow object “Betsy” and get her species

```
10 Betsy = Cow("Betsy")
11 #note that the name argument is now required when creating a Cow()
12 print("{} is a {}".format(Betsy.name, Betsy.get_species()))
```

```
Betsy is a moo cow
```

OOP Principles

The basic principles of OOP in python

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Encapsulation

Encapsulation is grouping similar data and functions into a group to make them easier to access and use.

Abstraction

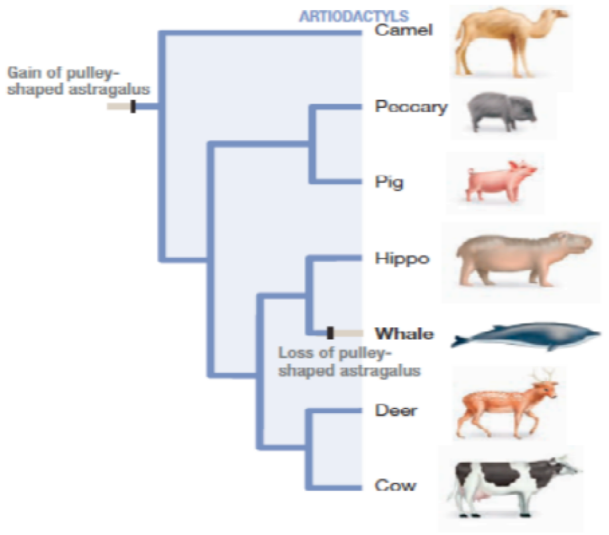
Abstraction is the concept of hiding irrelevant details. In other words, make complex system simple by hiding the unnecessary detail from the user.

In Python, you can denoted a private attribute or method by using “_” or “__”.

Inheritance

Inheritance is the ability to create a new class using the existing details of an existing class without changing it.

Think of this as a (base class)parent and (derived class)child class.



Polymorphism

This is an ability in OOP to use a common interface for multiple data types by defining how data types will interact with each other you can use a combination of this and inheritance to extract away the interaction.