Business Programming (using Python)

Zhaohu (Jonathan) Fan

Information Technology Management

Scheller College of Business

Georgia Institute of Technology

October 12, 2023

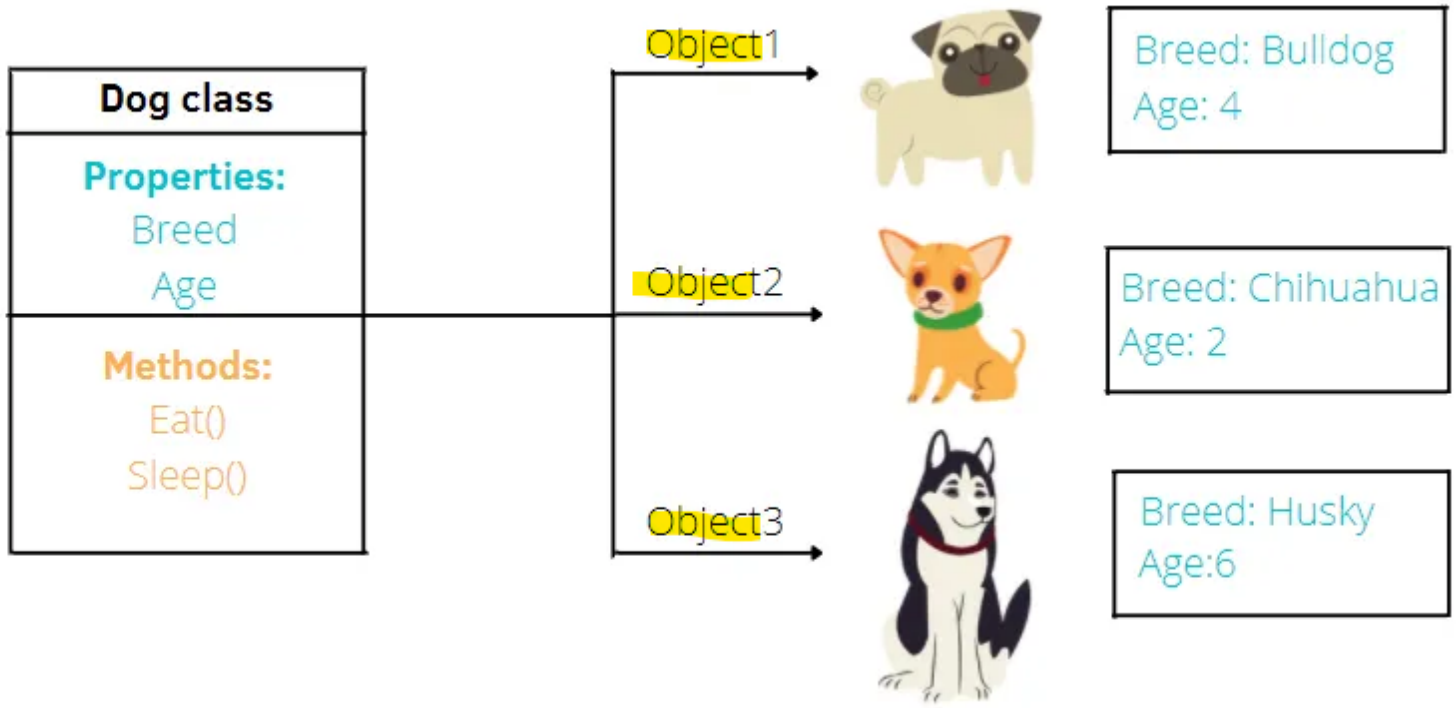
Main topics

- Prepare for Midterm
  - HW #6
  - Explaining Python Classes in a simple way
  - Exercises

Prepare for Midterm

- The midterm consists of 25-28 multiple choice questions (similar to the practice exam). It covers M2:Python Fundamentals excluding *class*.
  - **Studnets will be taking exams while in the classroom on Thursday Oct. 19th.**
  - Closed book/closed notes
  - You are allowed 3 double-sided 8.5" x 11" sheets as cheat-sheet. Additionally, you are allowed 1 piece of blank paper for scratch paper. You must show sheets to the TA. You may use an on screen calculator or a physical calculator. (Physical scientific calculator recommended)

Dog Class


Dog class
Properties:
Breed
Age
Methods:
Eat()
Sleep()
Object1
Object2
Object3
Breed: Bulldog
Age: 4
Breed: Chihuahua
Age: 2
Breed: Husky
Age:6

Class

- **Class**: Template (blueprint) for creating objects!
  - An **object** is an instance of a **class**; More than one object instance can be created from a single class.
  - **Class** defines all *properties* and *methods* for an object.
  - **Objects** of the same class have a same set of properties (such as color, year, engine, etc.).
  - **Objects** of the same class may have different characteristics for each property (such as red, blue colors), and different event responses.

Class

- **Initializer method** `__init__`, called in this way because it's the method that initializes the attributes of the object. Moreover, it is automatically called once the object is created.
  - **Magical methods** are special methods with double underscores on both sides. For example, `__add__` and `__mul__` are used to respectively sum and multiplicate objects of the same class, while `__repr__` returns a representation of the object as a string.
  - **Instance methods** are methods that belong to the object created. For example, l.append(4) adds an element at the end of the list.

Dog Class: Create a Class

- We will create an empty class and we'll progressively add parts of code within the tutorial.
    - We created a class with the name Dog, where `pass` is used to indicate that nothing is defined.

Python

```
>>> class Dog:
>>>     pass
```

Dog Class: Create a Class

- Once we defined the Dog class, we can create an object of the class, which is assigned to the variable jack. It's built using a similar notation we use to call a function: Dog().
    - The object can also be called instance. Don't be confused if you find the words "instance" or "object" written somewhere, they always refer to the same thing.

Python

```
>>> jack = Dog()
>>>print(type(jack))
```

Dog Class: Initializer method

- the initializer method `__init__` is used to initialize the attributes of the Dog class.
  - `self` is a standard notation used as the first argument and refers to the object, which will be created later. It's also useful to access the attributes that belong to the class.
  - `name`, `breed` and `age` are the remaining arguments. Each argument is used to store the specific attribute's value of the object.

Python

```
class Dog:
  def __init__(self,name,breed,age):
    self.Name = name
    self.Breed = breed
    self.Age = age
    print("Name: {}, Breed: {}, Age: {}".format(self.Name,
                                               self.Breed,self.Age))
```

Dog Class: Initializer method

- Name, Breed and Age are the defined attributes. **Note the attributes are typically not capitalized**.

Python

```
jack = Dog('Jack','Husky',5)
#Name: Jack, Breed: Husky, Age: 5
print(jack)
#<__main__.Dog object at 0×000002551DCEFFD0>
print(jack.Age)
#5
```

Dog Class: Magical Method

- There is also the possibility to print the same information in a more sophisticated way. For this purpose, we use the magical method `__repr__:`

Python

```
class Dog:
    def __init__(self,name,breed,age):
        self.Name = name
        self.Breed = breed
        self.Age = age
    def __repr__(self):
        return "Name: {}, Breed: {}, Age: {}".format(self.Name,
                                                     self.Breed,self.Age)
```

Dog Class: Magical Method

- The method `__repr__` takes a unique parameter selffrom which it can access the attributes of the object.
    - If we display the new instance created, we can look at the attributes and their respective values.

Python

```
jack = Dog('Jack','Husky',5)
print(jack)
#Name: Jack, Breed: Husky, Age: 5
```

Dog Class: Instance Method

- The instance methods are methods that belong to the class. As the magical methods, they take an input the parameter `self` to access the attributes of the class. Let's see an example:

Python

```
class Dog:
    def __init__(self,name,breed,age,tired):
        self.Name = name
        self.Breed = breed
        self.Age = age
        self.Tired = tired
    def __repr__(self):
        return "Name: {}, Breed: {}, Age: {}".format(self.Name,
                                            self.Breed,self.Age)
    def Sleep(self):
        if self.Tired==True:
            return 'I will sleep'
        else:
            return "I don't want to sleep"
```

Dog Class: Instance Method

- In the initializer method, we added a new argument `tired` and, consequently, a new attribute `Tired`. After, we define a new method called Sleep: if the attribute's value is equal to True, the dog will sleep, otherwise, it won't.

Python

```
jack = Dog('Jack','Husky',5,tired=False)
print(jack.Sleep())
#I don't want to sleep
```

Exercises

- Please click on the link provided below.
  - In-Class Exercise