Principles of Software Programming

Lecture 4
Object oriented programming

WU WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Anton Yeshchenko

SS 2018

Some slides and/or ideas were borrowed from:
MIT Introduction to Computer Science and Programming in Python
and Svitlana Vakulenko WS 2017 lecture slides

MARCH 2018

Recap

- **Control flow:**
  - if-else branches
  - loops
- **Lists:**
  - Arrays (lists)
  - create and fill Arrays

Recap more details.

SYNTACTIC SUGAR!!!!!!!!!!!!!

Today!

- Classes – Objects - Instances
- OOP and principles

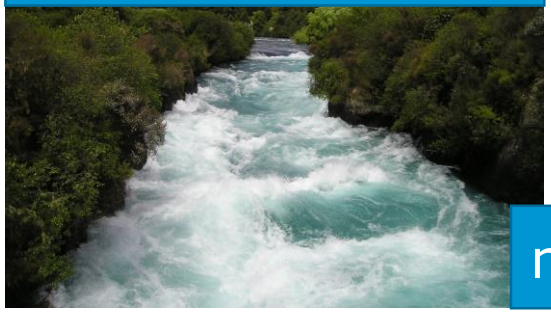
How are these related?

Not related!

HTTPS://CDN.MOS.CMS.FUTURECDN.NET/YCGAPULGOPZYWEMBUPMCNQ.JPG
HTTP://WWW.PETMD.COM/SITES/DEFAULT/FILES/SMALL-DOG-BREEDS.JPG
HTTPS://STATIC.PEXELS.COM/PHOTOS/2438/NATURE-FOREST-WAVES-TREES.JPG
HTTP://IMG1.COOKINGLIGHT.TIMEINC.NET/SITES/DEFAULT/FILES/STYLES/4_3_HORIZONTAL_-_1200X900/PUBLIC/IMAGE/2017/05/MAIN/PASTA-CHICKPEA-SAUCE-1707P46.JPG?ITOK=3MV9ZJMU
HTTPS://WWW.ECONOMIADIGITAL.ES/UPLOADS/S1/53/54/85/5/ATLAS_15_970X597.JPEG

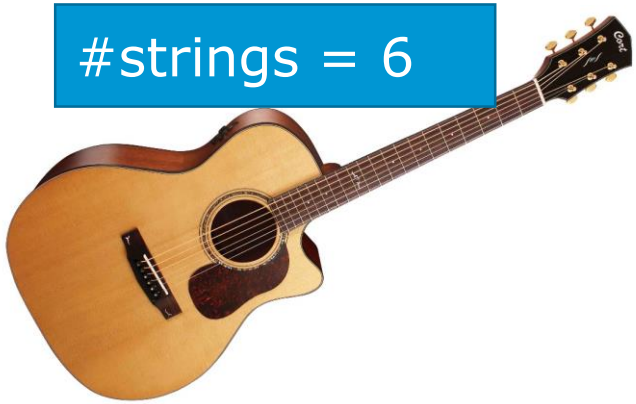
Simple characteristics

Length = 300km



name = “Tom”

#strings = 6



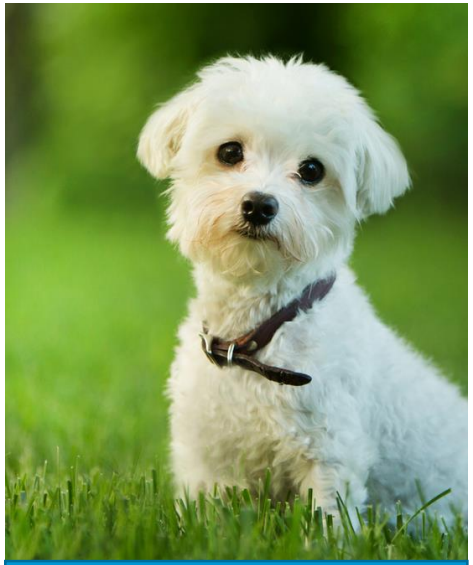
Calories=600

Color=“white”

Weight=200kg

HTTPS://CDN.MOS.CMS.FUTURECDN.NET/YCGAPULGOPZYWEMBUPMCNQ.JPG
HTTP://WWW.PETMD.COM/SITES/DEFAULT/FILES/SMALL-DOG-BREEDS.JPG
HTTPS://STATIC.PEXELS.COM/PHOTOS/2438/NATURE-FOREST-WAVES-TREES.JPG
HTTP://IMG1.COOKINGLIGHT.TIMEINC.NET/SITES/DEFAULT/FILES/STYLES/4_3_HORIZONTAL_-_1200X900/PUBLIC/IMAGE/2017/05/MAIN/PASTA-CHICKPEA-SAUCE-1707P46.JPG?ITOK=3MV9ZJMU
HTTPS://WWW.ECONOMIADIGITAL.ES/UPLOADS/S1/53/54/85/5/ATLAS_15_970X597.JPEG

Simple characteristics



name = “Tom”

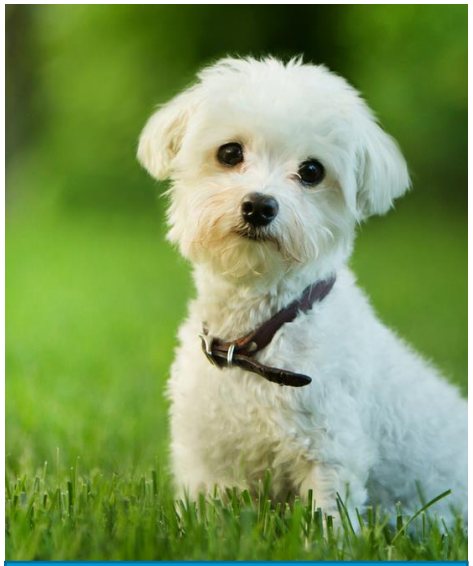
Is it enough to say that he is ‘Tom’ for him to be a dog?

Simple characteristics

Is it enough to say that he is 'Tom' for him to be a dog?

Define Tom better!

Our Tom can!
bark()
run()
carry_bone()
poo_in_the_park()
talk_to_a_dog(a_dog)



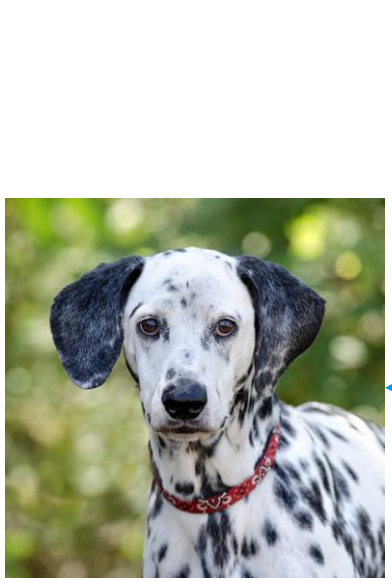
name = "Tom"

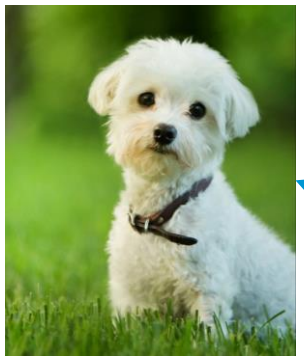
Is it enough to say that he is 'Tom' for him to be a dog?

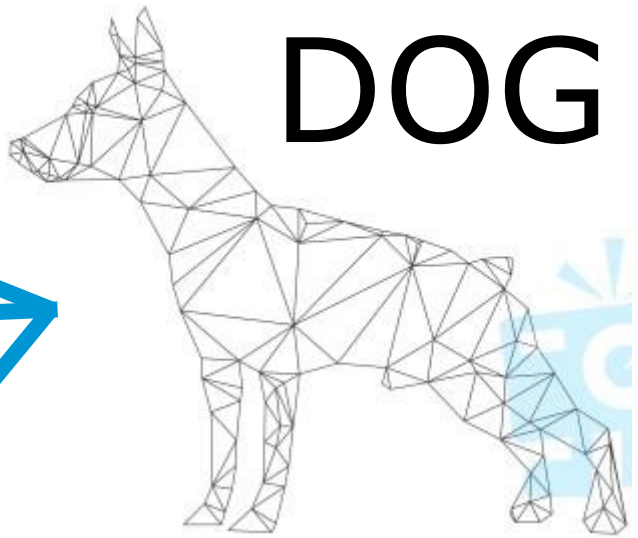
He is:
A dog
He has:
Weight=2kg
Height=20cm
Accessories = [belt, leash, gps tracker]

Classes, Objects, Instances!

- We can make a **class dog** (the blue print)
- "Tom" is an **instance** of the **class dog**







Can have weight, height, accessories,
Can bark, run, carry_done etc.

Plato(n) – Theory of Forms

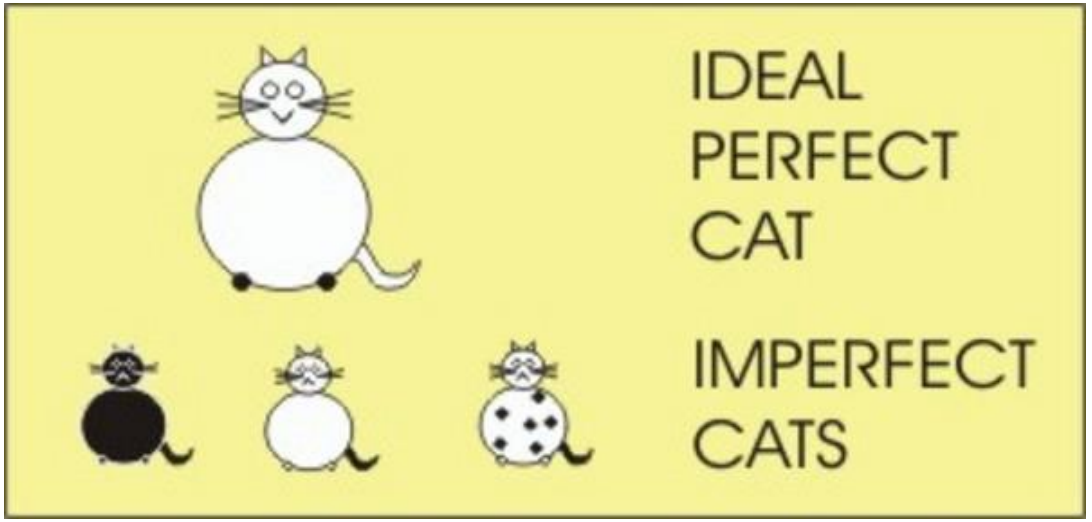
The theory of Forms (or theory of Ideas) typically refers to the belief that the material world as it seems to us is not the real world, but only an "image" or "copy" of the real world.

The forms, according to Socrates, are archetypes or abstract representatio ns of the many types of things, and properties we feel and see around us, that can only be perceived by reason (λογική). (That is, they are universals.)

In other words, Socrates was able to recognize two worlds: **the apparent world, which constantly changes**, and an **unchanging and unseen world of forms**, which may be the cause of what is apparent.

The standard meaning is that **an *object* is an instance of a *class*.**

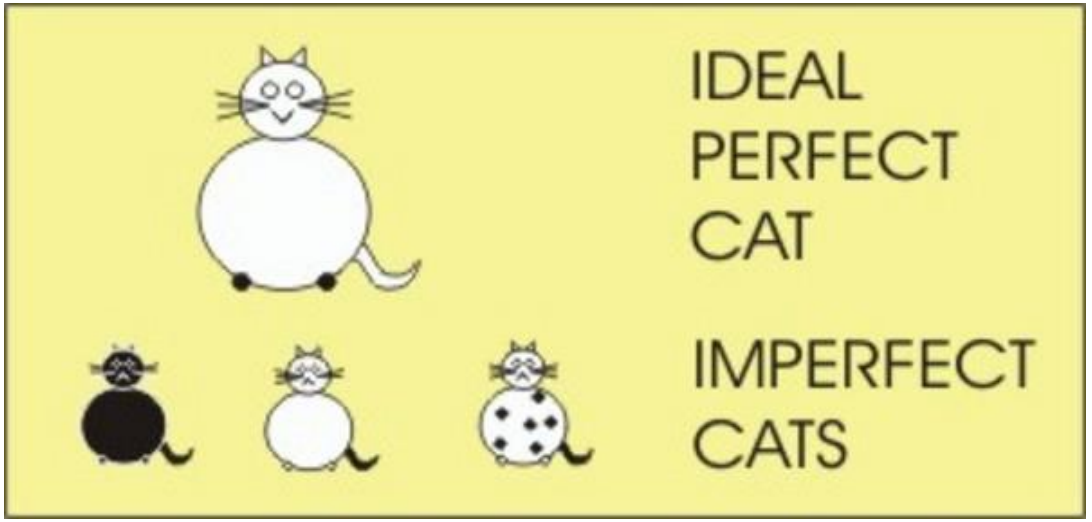
1. **Class** - is a blueprint which you use



HTTP://WWW.PROPROFS.COM/QUIZ-SCHOOL/STORY.PHP?TITLE=GREEKOLOGY

The standard meaning is that **an *object* is an instance of a *class*.**

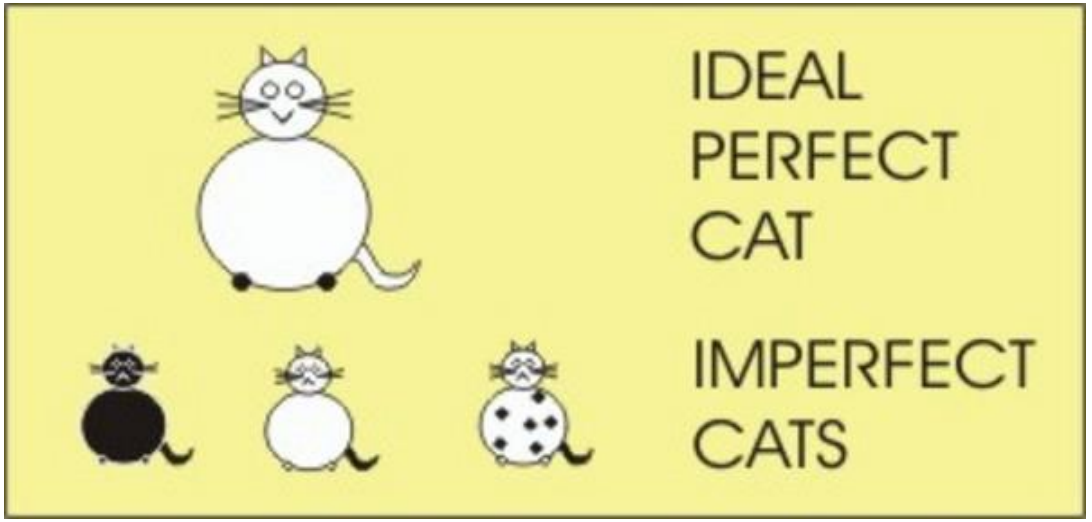
1. **Class** - is a blueprint which you use
2. to create **objects**



HTTP://WWW.PROPROFS.COM/QUIZ-SCHOOL/STORY.PHP?TITLE=GREEKOLOGY

The standard meaning is that **an *object* is an instance of a *class*.**

1. **Class** - is a blueprint which you use
2. to create **objects**
3. An object is an **instance** of a class



One can instantiate an object from a class.

HTTP://WWW.PROPROFS.COM/QUIZ-SCHOOL/STORY.PHP?TITLE=GREEKOLOGY

That's how we do it in python

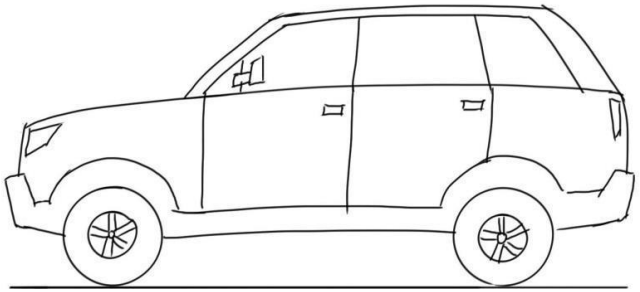
- use the `class` keyword to define a new type

name/type

```
class Coordinate:
    #define attributes here
```

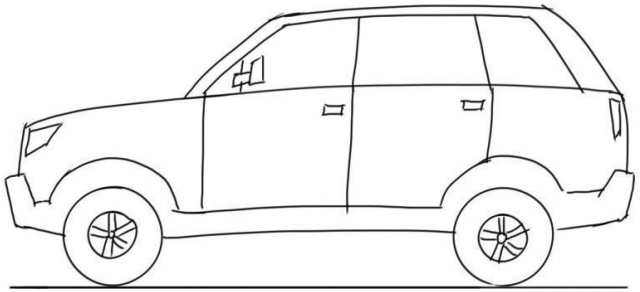
class definition

- similar to `def`, indent code to indicate which statements are part of the **class definition**



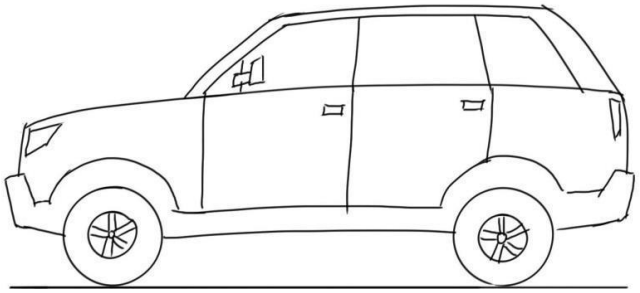
What are the attributes???

- data and procedures that "**belong**" to the class



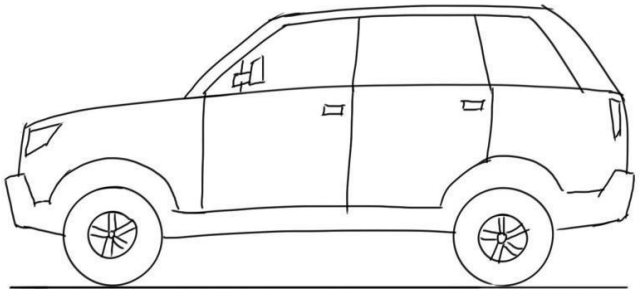
What are the attributes???

- data and procedures that "**belong**" to the class
- **data attributes**
  - think of data as other objects that make up the class
  - *for example, a coordinate is made up of two numbers*



What are the attributes???

- data and procedures that "**belong**" to the class
- **data attributes**
  - think of data as other objects that make up the class
  - *for example, a coordinate is made up of two numbers*
- **methods** (procedural attributes)
  - think of methods as functions that only work with this class
  - how to interact with the object
  - *for example you can define a distance between two coordinate objects but there is no meaning to a distance between two list objects*



Constructor! Create your variables here! **(Defining how to create an instance)**

- first have to define **how to create an instance** of object
- use a **special method called `__init__`** to initialize some data attributes

```
class Coordinate:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

special method to create an instance — is double underscore

what data initializes a `Coordinate` object

parameter to refer to an instance of the class

two data attributes for every `Coordinate` object

How to actually create an instance of the class

```
c = Coordinate(3,4)
origin = Coordinate(0,0)
print(c.x)
print(origin.x)
```

create a new object of type `Coordinate` and pass in 3 and 4 to the `__init__`

use the dot to access an attribute of instance `c`

- data attributes of an instance are called **instance variables**
- don't provide argument for `self`, Python does this automatically

What is a method of a class?

- procedural attribute, like a **function that works only with this class**
- Python always passes the object as the first argument
  - convention is to use `self` as the name of the first argument of all methods
- the "." **operator** is used to access any attribute
  - a data attribute of an object
  - a method of an object

Define a method of Coordinate class

```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def distance(self, other):
        x_diff_sq = (self.x-other.x)**2
        y_diff_sq = (self.y-other.y)**2
        return (x_diff_sq + y_diff_sq)**0.5
```

use it to refer to any instance

another parameter to method

dot notation to access data

- other than `self` and dot notation, methods behave just like functions (take params, do operations, return)

How to use a method

Using the class:

- conventional way

```
c = Coordinate(3,4)

zero = Coordinate(0,0)

print(c.distance(zero))
```

object to call method on

name of method

parameters not including `self` (`self` is implied to be `c`)

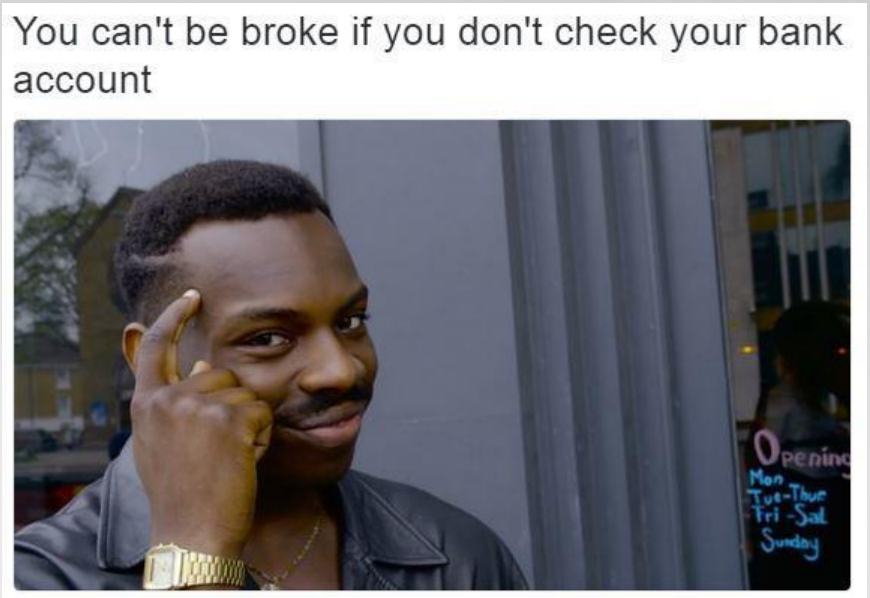
Naming convention recap

- Function names, variables -> **lowercase_with_underscores**
- Class names -> **CamelCase** (capital letters with no undescores)

Exercise 1! Classes

- Let's make some serious business!

1. Create a bank account class!
2. The account has some data(variables):
   1. the name of the account holder
   2. the account number
   3. the current balance.
3. Three things we can do with an account(methods):
   1. withdraw money
   2. put money into the account
   3. print out the data of the account.



4. Initialize one bank account with name, account number and current balance

Actually…

It turns out that - Everything is an object! (in python)

- Python supports many different kinds of data

```
1234      3.14159      "Hello"      [1, 5, 7, 11, 13]
{"CA": "California", "MA": "Massachusetts"}
```

- each is an **object**, and every object has:
  - a **type**
  - an internal **data representation** (primitive or composite)
  - a set of procedures for **interaction** with the object
- an object is an **instance** of a type
  - `1234` is an instance of an `int`
  - `"hello"` is an instance of a string

Method overloading

- Given a single method or function, we can specify the number of parameters ourself.
- Depending on the function definition, it can be called with zero, one, two or more parameters.

```
class Human:

    def sayHello(self, name=None):

        if name is not None:
            print 'Hello ' + name
        else:
            print 'Hello '

# Create instance
obj = Human()

# Call the method
obj.sayHello()

# Call the method with a parameter
obj.sayHello('Guido')
```

Method overloading!

1. You can withdraw money from bank account with foreign ATM (so there will be some fees)
   withdraw(self,amount, fees=None)

2. You can deposit not only Euro so conversion is required
   deposit(self,amount, currency="Euro")
3. Print data about account
   In German
   In English

Mind the order!!!!!!!

Different parameter list.

Quiz 1.

- Kahoot.it!


OBJECT ORIENTED
PROGRAMMING
memegenerator.net

What are the objects?

- objects are **a data abstraction** that captures…

(1) an **internal representation**

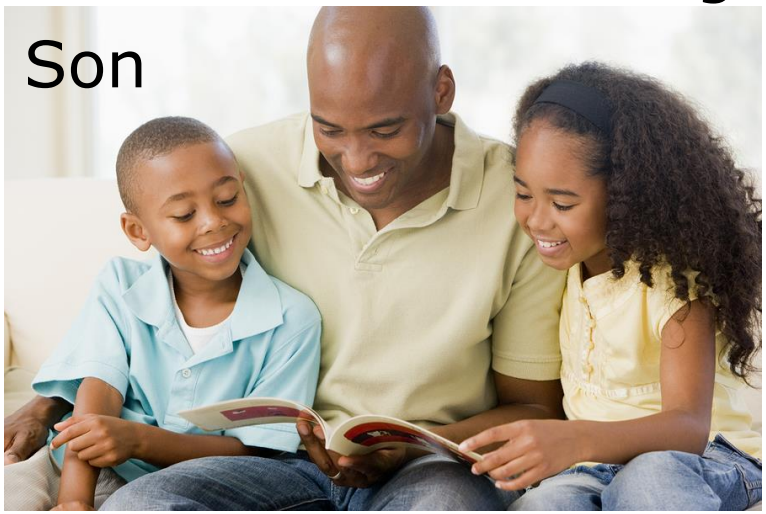
- through data attributes

(2) an **interface** for interacting with object

- through methods (aka procedures/functions)
- defines behaviors but hides implementation

Object oriented programming.
WHY?????? All objects

Father Daughter

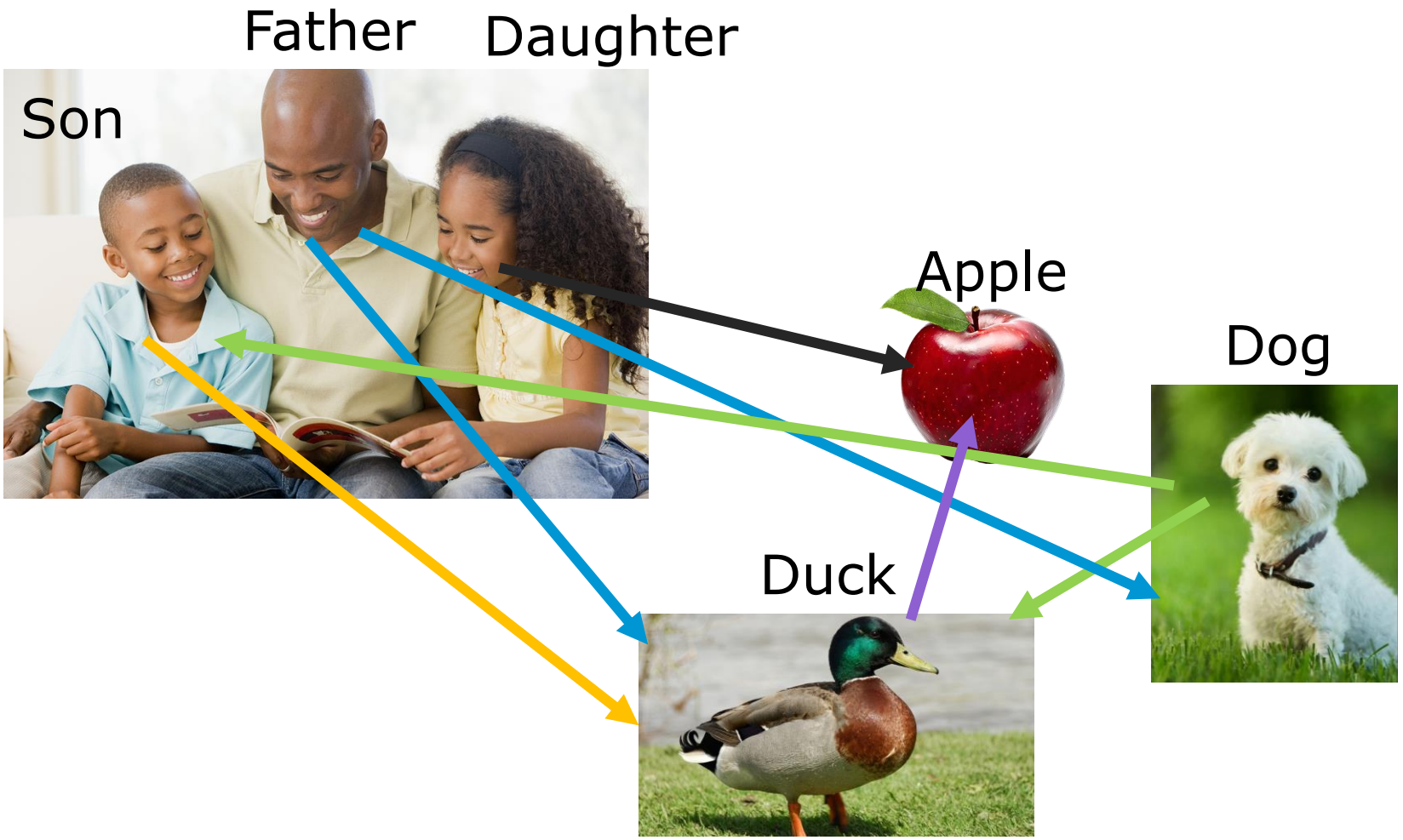


Apple

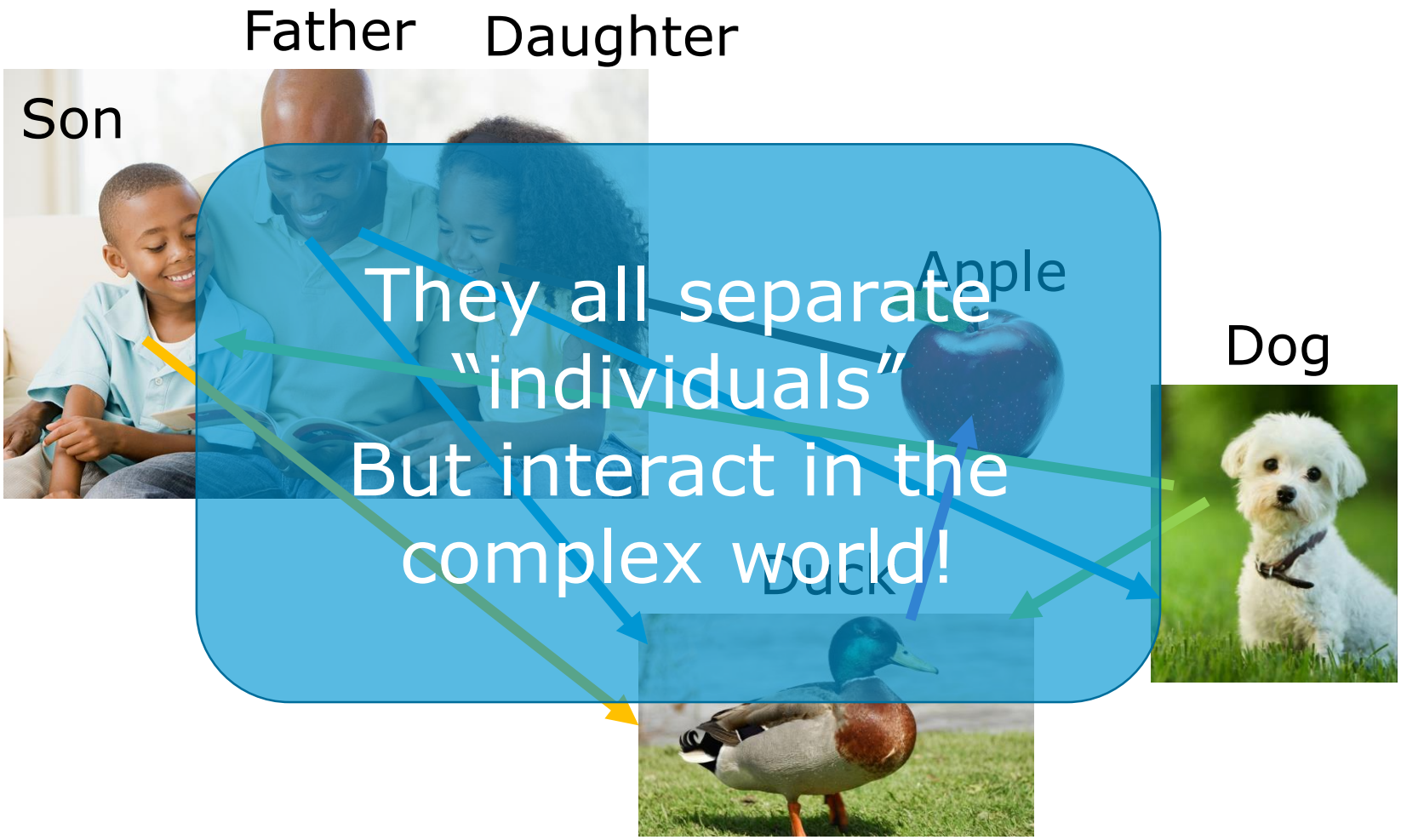
Dog

Duck

Object oriented programming. WHY?????? All objects

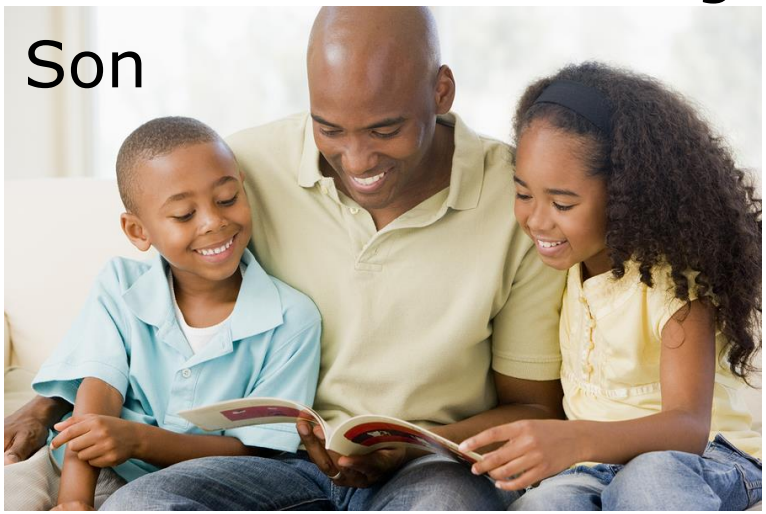


Object oriented programming. WHY?????? All objects



Object oriented programming. WHY?????? All objects

Father Daughter



Apple

Dog

divide-and-conquer (decomposition)
development
•implement and test behavior of each class separately
•increased modularity reduces complexity
Reuse!

Duck

4 principles of OOP

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism