Web Application Development using Python

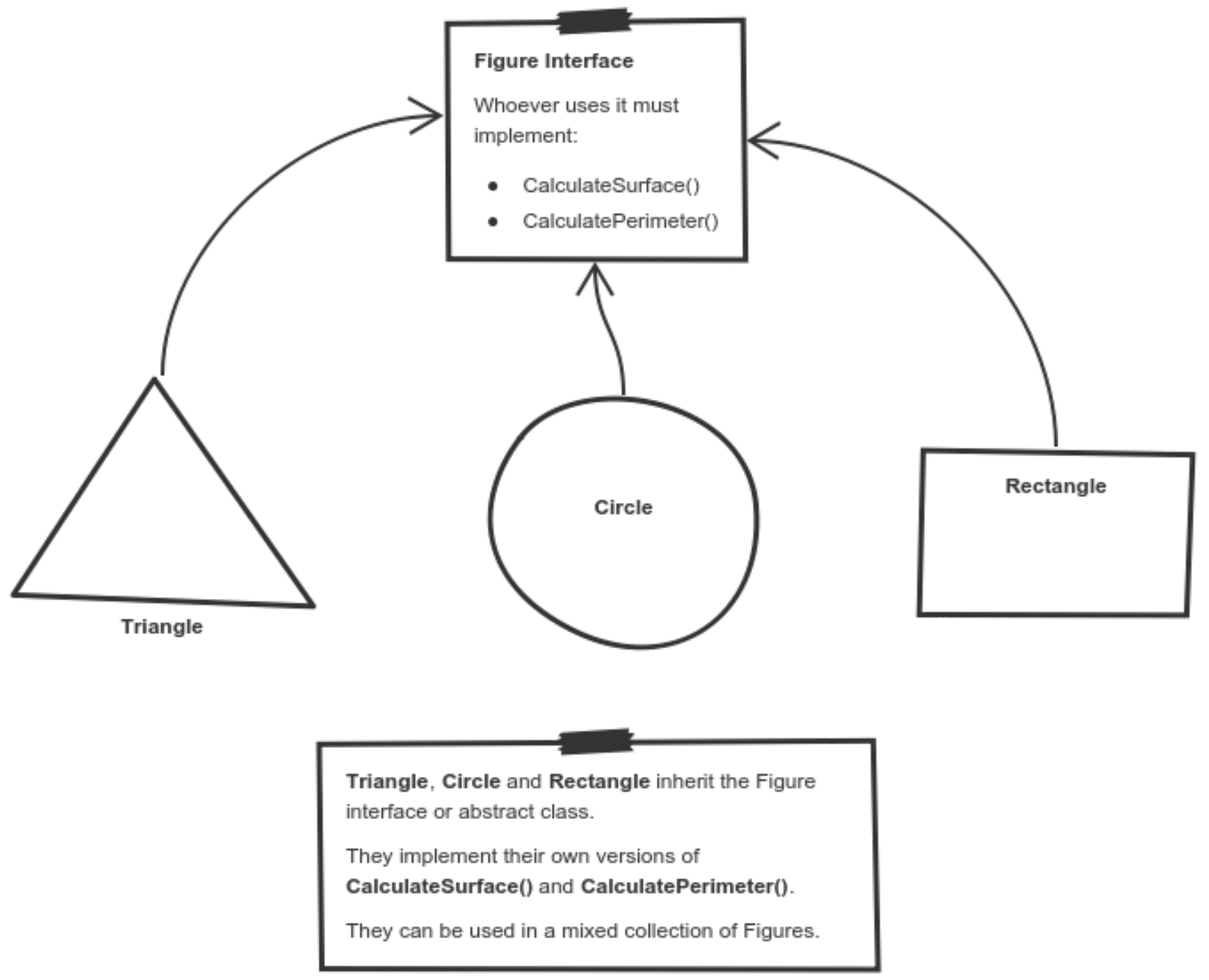
Introduction to Object Oriented Programming

Prepared by George Khoury

Outline

- What is Object Oriented Programming?
- OOP vs. “functional programming”
- **Classes**
  - The `__init__()` method
  - Instantiation
  - Accessing Attributes
  - Calling Methods
- Instance / class object attribute
- How to instantiate objects?

What is Object Oriented Programming?



Triangle, **Circle** and **Rectangle** inherit the Figure interface or abstract class.

They implement their own versions of **CalculateSurface()** and **CalculatePerimeter()**.

They can be used in a mixed collection of Figures.

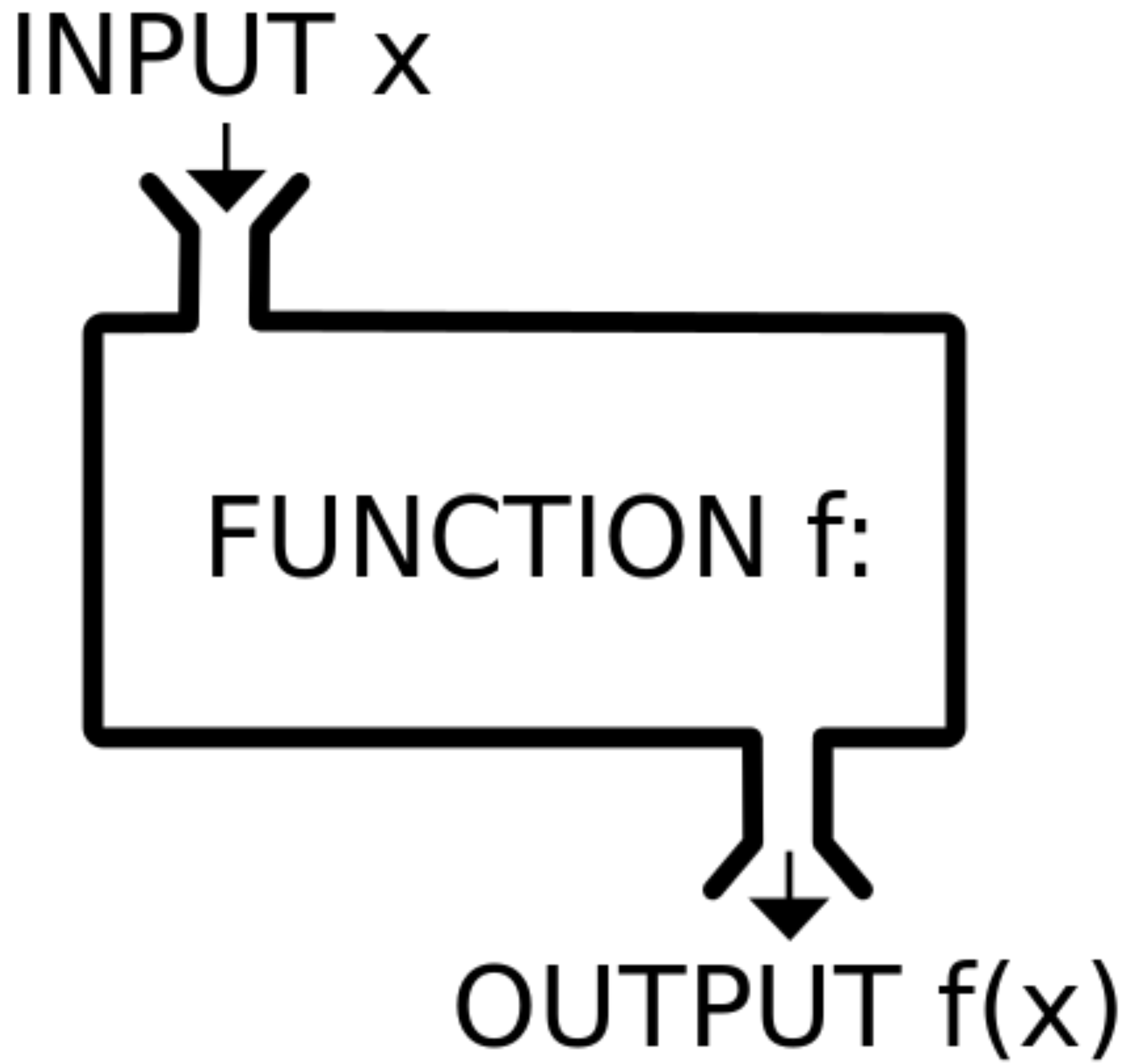
What is Object Oriented Programming?

- Object-oriented programming is one of the most effective approaches to writing software.
- In object-oriented programming you write classes that represent real-world things and situations, and you create objects based on these classes.
- When you write a class, you define the general behavior that a whole category of objects can have.
- When you create individual objects from the class, each object is automatically equipped with the general behavior; you can then give each object whatever unique traits you desire.

What is Object Oriented Programming?

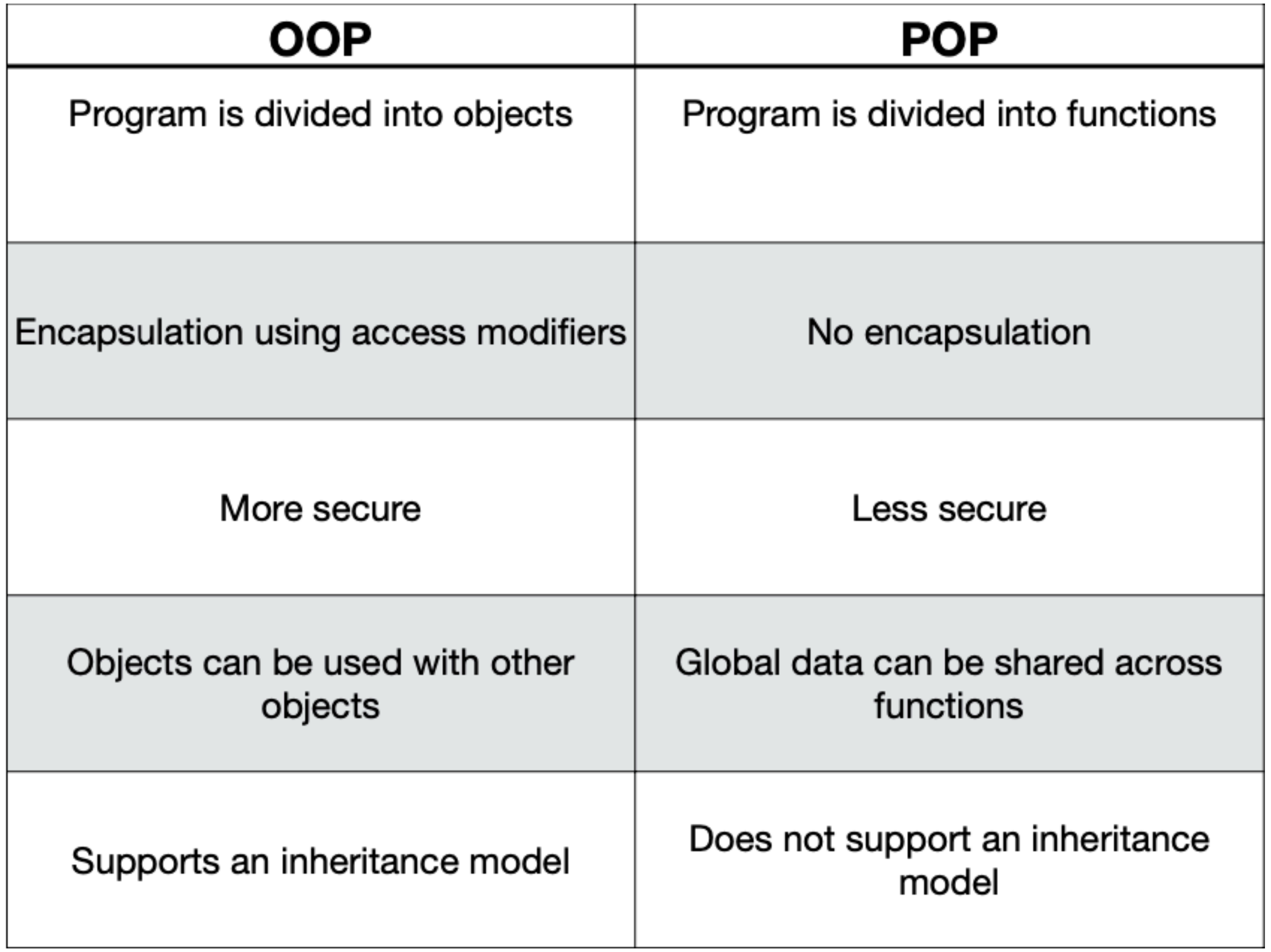
- In Python, we use classes to model objects.
  - **Attributes** define properties of an object.
  - **Methods** define actions an object can perform.
- OOP allows us to write neat and modular reusable code.
- Applications usually define more than one object type.
- Different objects work together to achieve a task.

OOP vs. “functional programming”



OOP vs. “functional programming”

- The four major principles of object orientation are:
    - **Encapsulation** - a mechanism for restricting the access to some of an object's components, this means that the internal representation of an object can't be seen from outside of the objects definition.
    - Data Abstraction
    - Polymorphism
    - Inheritance



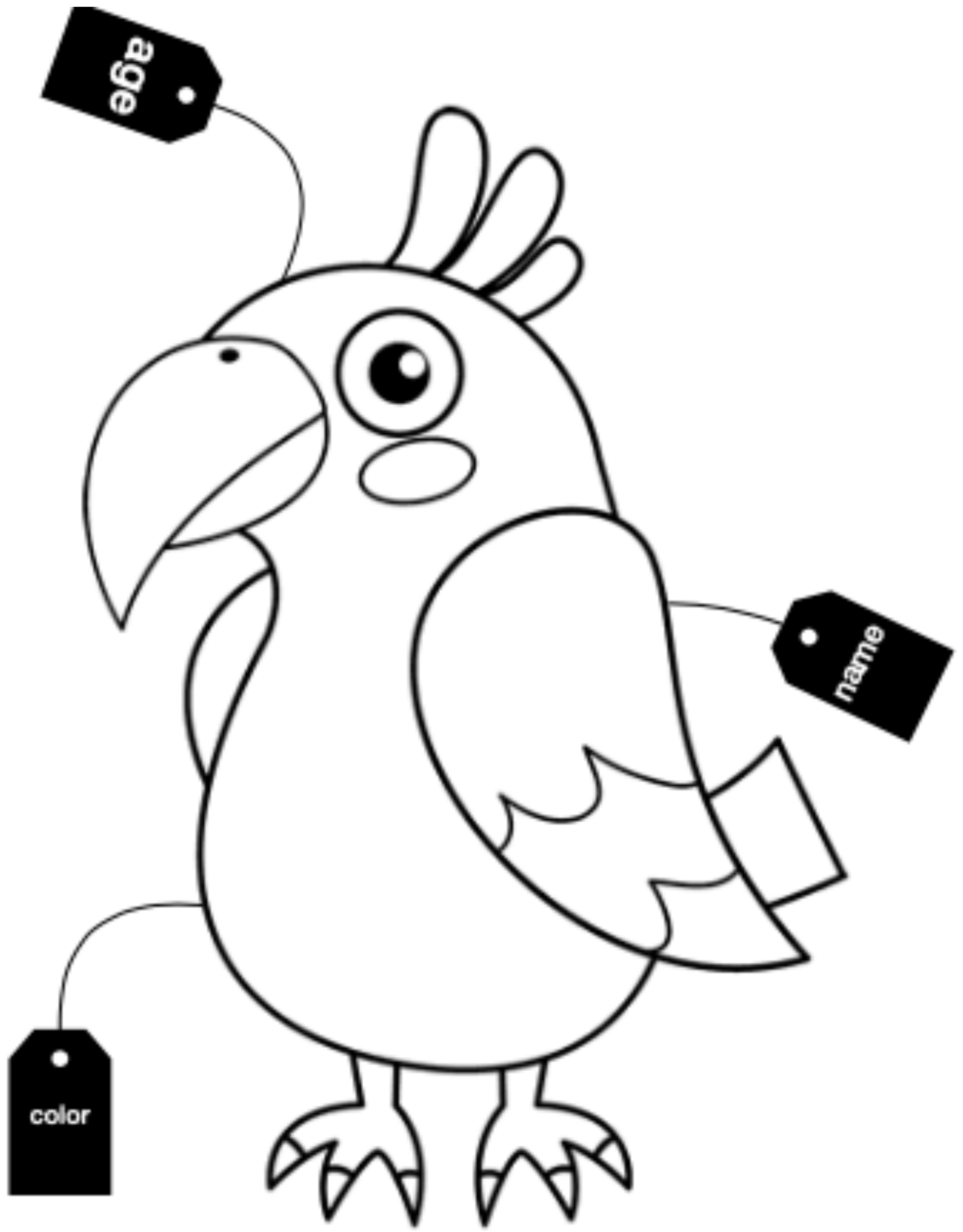
OOP	POP
Program is divided into objects	Program is divided into functions
Encapsulation using access modifiers	No encapsulation
More secure	Less secure
Objects can be used with other objects	Global data can be shared across functions
Supports an inheritance model	Does not support an inheritance model

Classes

A “blueprint” or a definition of your object.

Classes

- A class is a blueprint for the object.
- We can think of class as a sketch of a parrot with labels.
- It contains some details about the parrot.
  - Some attributes like **name**, **age**, **color**, etc.
  - Some actions this class can perform like **sing**, **dance**, **eat**, etc.



Instances

- Think of a class as a set of instructions for how to make an instance.
- An object or instance represents a specific copy of that class.
- When a class is defined, only the definition of the object is created.
- To use an instance of class, first we have to create it.

name = "Blu"
age = 10

Learning Resources

- https://docs.python.org/3.8/tutorial/classes.html
- https://docs.python.org/3.8/tutorial/classes.html#class-definition- syntax
- https://www.python-course.eu/object_oriented_programming.php
- https://www.tutorialspoint.com/python/python_classes_objects.htm