**A Brief Introduction**

This lesson welcomes you to the world of object-oriented programming.

**We'll cover the following**

* [Procedural programming](https://www.educative.io/module/lesson/learn-oop-in-python/RMrEDZO7QvE#Procedural-programming)
* [Object-oriented programming](https://www.educative.io/module/lesson/learn-oop-in-python/RMrEDZO7QvE#Object-oriented-programming)
* [Anatomy of objects and classes](https://www.educative.io/module/lesson/learn-oop-in-python/RMrEDZO7QvE#Anatomy-of-objects-and-classes)
* [User-defined data types](https://www.educative.io/module/lesson/learn-oop-in-python/RMrEDZO7QvE#User-defined-data-types)

**Procedural programming**

If you are here, you are probably already familiar with the basics of programming and have used *methods* in your programs at some point.

Procedural programming is one programming paradigm among many others.

In procedural programming, a program is divided into smaller parts called methods. These **methods** are the **basic entities** used to construct a program. One of the main advantages of procedural programming is code reusability. However, the implementation of a complex real-world scenario becomes a difficult and unwieldy task.

**Object-oriented programming**

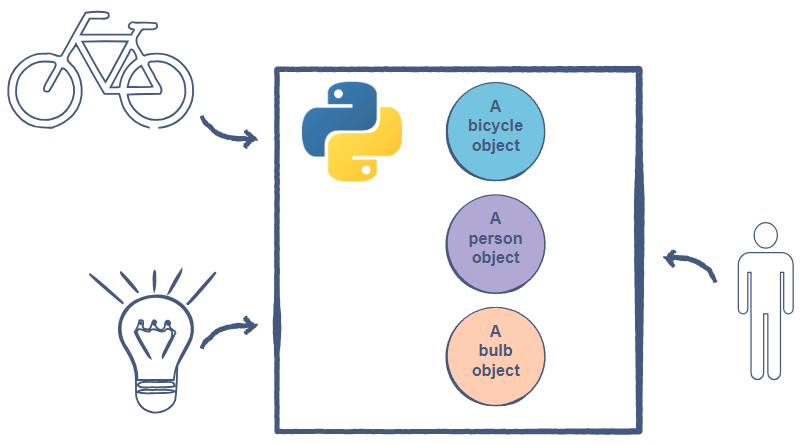
*Object-oriented programming*, also referred to as **OOP**, is a programming paradigm that includes, or relies, on the concept of **classes and objects**.

The basic entities in object-oriented programming are **classes and objects.**

Programming isn’t much use if you can’t model real-world scenarios using code, right? This is where object-oriented programming comes.

The basic idea of OOP is to divide a sophisticated program into a number of **objects** that talk to each other.

Objects in a program frequently represent real-world objects.

’

It is also possible for objects to serve application logic and have no direct, real-world parallels. They manage things like authentication, templating, request handling, or any of the other myriad features needed for a practical application.

**Anatomy of objects and classes**

Objects may contain data in the form of *fields* (variables) and methods to operate on that data.

Think about real-world objects around you. *What are the characteristics of these objects*? Take the example of a *light bulb*. It has a **state**, which means that it is either *on* or *off*. It also has a **behavior**, which means that when it is turned on it lights up, and when it is turned off, it does not produce any light. To conclude this, one can say:

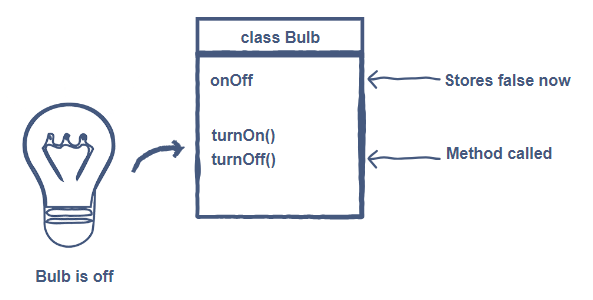
Objects are a collection of **data** and their **behaviors**.

But *where do these objects come from?*

The answer to the above question is **classes**.

A **class** can be thought of as a *blueprint* for creating objects.

The illustration below shows what a LightBulb class should look like:

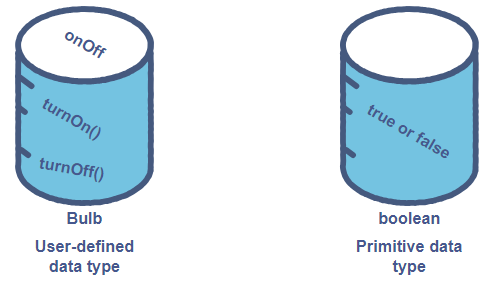


From the above illustration, you can see that the state of the object is generally modeled using *variables* in a class, and the behavior is modeled using *methods*.

There can be many different objects of the same class. Each can be in an independent state, but all of them will share the same behavior and characteristics.

**User-defined data types**

It can be inferred from the discussion above that classes are user-defined data types implemented using primitive data types, e.g., boolean, int, char, etc. While primitive data types only focus on modeling the state of the object, **user-defined data types** can encapsulate the state and its behaviors into a unit.



**Introduction to Objects and Classes**

**We'll cover the following**

* [A brief encounter](https://www.educative.io/module/lesson/learn-oop-in-python/7nOGX66VkmO#A-brief-encounter)
* [Objects and classes](https://www.educative.io/module/lesson/learn-oop-in-python/7nOGX66VkmO#Objects-and-classes)
  + [Properties](https://www.educative.io/module/lesson/learn-oop-in-python/7nOGX66VkmO#Properties)
  + [Methods](https://www.educative.io/module/lesson/learn-oop-in-python/7nOGX66VkmO#Methods)
* [Benefits of objects and classes](https://www.educative.io/module/lesson/learn-oop-in-python/7nOGX66VkmO#Benefits-of-objects-and-classes)

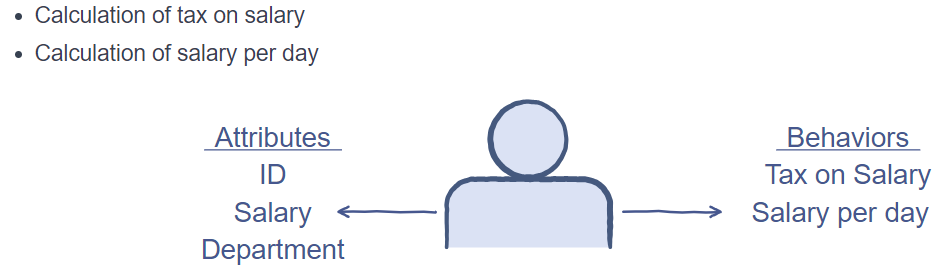
**A brief encounter**

We see *objects* everywhere in our surroundings. These objects have certain properties that define them. There are certain behaviors that these objects perform on their own, and there are actions that can be performed on them.

Let’s take the example of a company employee. An employee has the following properties or **attributes**:

* ID
* Salary
* Department

The following actions or **behaviors** can be performed on an employee:

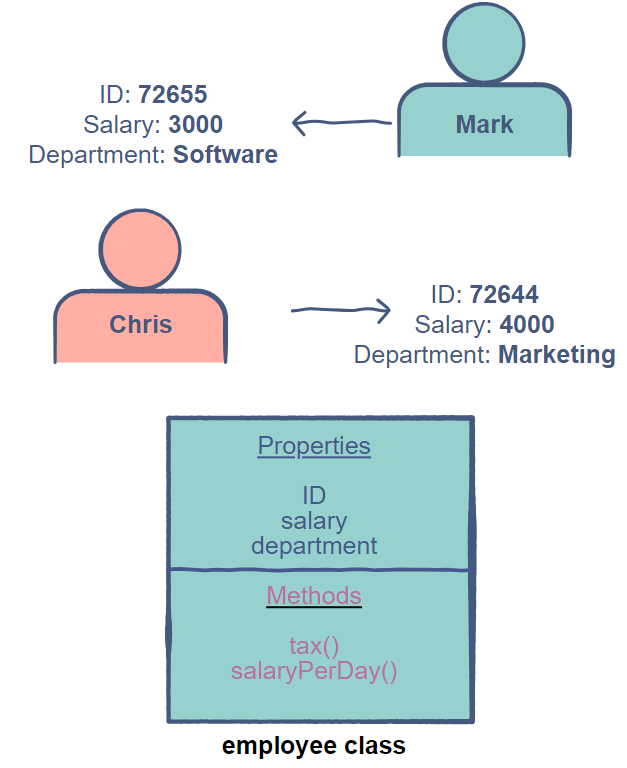


In a company, each worker has a different name, salary, and department, but the type of each worker is employee. So, there is a generic blueprint for each worker working in the company, but each of them has different attributes.

A class has a singular blueprint, and objects are part of a class and are differentiated by their distinct properties.

Objects and classes

Suppose, there are two employees at Educative, Mark and Chris. The properties of Mark and Chris are given in the image below:



**Properties**

Properties are variables that contain information regarding the object of a class. An employee object will have an ID, a salary, and the department as its *properties*. New properties can be added to become a part of an object of the employee class.

Attributes are also referred to as properties or members. For consistency, we will be using properties.

**Methods**

Methods are like functions that have access to properties (and other methods) of a class. Methods can accept parameters and return values. They are used to perform an action on an object of a class. In the example above, we have tax() and salaryPerDay() as class methods.

Behaviors are also referred to as member functions or methods. For consistency, we will be using methods.

**Benefits of objects and classes**

Objects and classes allow us to create complex applications in Python. This is why they are considered the building blocks of OOP principles.

* Objects and classes are also instrumental for compartmentalizing code. Different components can become separate classes that would interact through interfaces. These ready-made components will also be available for use in future applications.
* The use of classes makes it easier to maintain different parts of an application since it is easier to make changes in classes.

**Declaring a Class in Python**

Learn how to declare classes in Python.

**We'll cover the following**

* [Declaration](https://www.educative.io/module/lesson/learn-oop-in-python/g78R6q3B5kY#Declaration)
  + [Naming rules](https://www.educative.io/module/lesson/learn-oop-in-python/g78R6q3B5kY#Naming-rules)
* [Creating a class object](https://www.educative.io/module/lesson/learn-oop-in-python/g78R6q3B5kY#Creating-a-class-object)

**Declaration**

In Python, classes are defined as follows:

class ClassName:

pass

The class keyword tells the compiler that we are creating a custom class, which is followed by the class name and the : sign.

All the properties and methods of the class will be defined within the class scope.

**Naming rules**

The following rules must be adhered to when naming classes:

1. Must start with a *letter* or *underscore*
2. Should only be comprised of *numbers*, *letters*, or *underscores*

**Creating a class object**

The name of the class, MyClass, will be used to instantiate an object of the class in our main program. We can create an object of a class by simply using the name of the class followed by a pair of parenthesis. It looks similar to calling a function, but Python can distinguish between the two and creates a new object of the corresponding class. An example of this is given below:

class MyClass:

pass

obj = MyClass() # creating a MyClass Object

print(obj)

Well done! You’ve created your first object-oriented program in Python. Printing this object, obj, will show the memory address at which this object is stored.

This is just a basic implementation of a Python class and doesn’t serve any particular purpose as it does not contain any properties or methods.