Implementing Properties in a Class

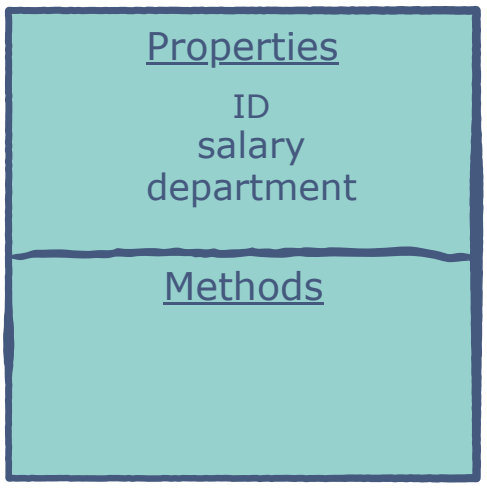
In this lesson, you will learn how to create properties, both in classes and outside of classes, and how to access them.

We'll cover the following

Building on the concepts of the previous lesson, we will implement a class, `Employee`, in Python.

Implementation of the Employee Class

Let's implement the `Employee` class illustrated below. We'll start with just adding the properties of the class and will later extend it by adding methods in it.



Employee class

With_None | Without_None

```
# this code will compile
class Employee:
    # defining the properties and assigning them none
```

```
    ID = None
    salary = None
    department = None

```

We've defined three properties as class variables; `ID`, `salary` and `department`, for the class `Employee`. We will discuss the concept of class variables later. For now, focus on the syntax.

> Note that if you do not initialize the values of properties, the Python code will **not compile**. Initializing the values of properties inside the class is necessary.

The code in the second tab will not compile since the properties in the class have not been initialized.

Accessing Properties and Assigning Values

To access properties of an object, the **dot** notation is used:

```
object.property
```

There are two ways to assign values to properties of a class.

1. Assign values when defining the class.
2. Assign values in the main code.

initializing | assigning

```
class Employee:
    # defining the properties and assigning values to them
    ID = 3789
    salary = 2500
    department = "Human Resources"


# cerating an object of the Employee class
Steve = Employee()

# printing properties of Steve - an object of the Employee class
print("ID =", Steve.ID)
print("Salary", Steve.salary)
```

```
print("Department:", Steve.department)
```

Creating Properties Outside a Class

Python, being a particularly user-friendly language, provides the user with a feature that most languages usually do not. That is, creating properties of an object **outside** the class. Let's see an example of this by extending the example of the `Employee` class we discussed above:

1 of 3

> **Note:** The property, `title`, will only be added to Steve and all other future objects will only have the properties which are declared in the class.

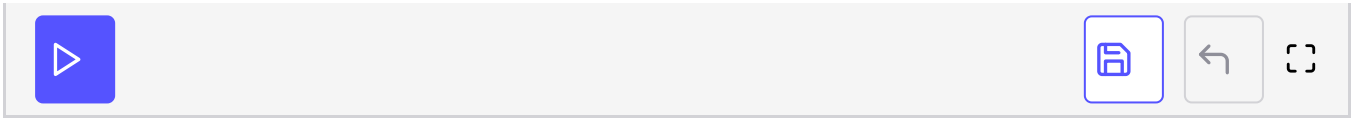
```
class Employee:
    # defining the properties and assigning them None
    ID = None
    salary = None
    department = None


# cerating an object of the Employee class
Steve = Employee()

# assigning values to properties of Steve - an object of the Employee class
```

```
# assigning values to properties of Steve - an object of the Employee class
Steve.ID = 3789
Steve.salary = 2500
Steve.department = "Human Resources"
# creating a new attribute for Steve
Steve.title = "Manager"

# Printing properties of Steve
print("ID =", Steve.ID)
print("Salary", Steve.salary)
print("Department:", Steve.department)
print("Title:", Steve.title)
```



This is a basic implementation of the Python class with only properties. In the next lesson, we will learn how to initialize objects in Python.