**Python 09: Everything is an Object.**

**Introduction**: This document presents learning activities for Python 09. In Python 09, you will get introduced with classes and objects in Python. This will be the fist step in Object Oriented Programming in Python.

**Note**: In this phase, it is expected the learner can divide the program into smaller learning steps. The goal and direction of the topics will be provided. The learning must take smaller steps towards the goals such that can implement solutions to the given problems and product(s).

**Materials:**

The activities are designed based on these following references:

* **BRef-01**: Book, Bill Lubanovic; "Introducing Python: Modern Computing in Simple Packages"; [Available here](https://www.oreilly.com/library/view/introducing-python-2nd/9781492051374/)
* **ORef-01**: Online Tutorial; Charles Severance; "Python for Everybody"; [Available here](https://books.trinket.io/pfe/index.html)

**Path:**

**Step: Class and Object.**

**Goals:**

After taking this step, you will be able to:

1. interpret and implement Python programs with Python objects and classes: defining a class, instantiating an object, attributes and methods, initializing and object.

2. understand the concepts of namespace and scope in Python programs.

**What to Learn?**

1. Using **BRef-01: Chapter 10** answer and experiment the following questions:
   1. What is an Object in Python?

Everything in Python is essentially a Object. Variables, Tuples …  
A object is a instance of a class.

* 1. What is a class and how can you define a class in Python? Make an example.

A class is basically a blue print for creating objects.

* 1. How can you create and use an object from a class? Experiment with examples.

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

* 1. What are attributes? How they help programmers to encapsulate their data values?
  2. What is a method? Focus only on *instance methods*.
  3. How can you initialize an instance (object)? Implement a class with **init** method.

**Exercises:**

1. Design at least ten different exercises of your own. They should improve understanding topics of this step. Share your exercises with your learning community and practice.

**Problems:**

1. We're going to create a small application for a car dealer. The problem is split up into three steps to help you set up a proper structure for the application.
   * Create a class named Car. Create four fields (brand, model, color, price) and implement these using the init function. Also create a fifth field called sold. The default value for sold is False. Create a method called sell that changes the value of sold to True. Create another method called print that prints all fields in a for humans readable way. Create 4 objects of the Car class. Sell a few of the cars. Call the print function on all of them.
   * Continue on your code from the previous exercise with a new class named Customer. Give it one field (name) and properly implement it. Also create a method print that returns all fields in a for humans readable way. Modify the Car class with a field sold\_to, set this field to a object of Customer within the sell method (which now needs a parameter with a Customer object). Edit the print method of the Car to also print the information about the customer if the car has been sold. Adjust other code were needed to get everything working properly.
   * The car dealer wants to extend his business by selling motorcycles as well. Write all code to properly introduce this into the existing application.
2. Write a class called Product. The class should have attributes called name, amount, and price, holding the product’s name, the number of items of that product in stock, and the regular price of the product. There should be a method get\_price that receives the number of items to be bought and returns a the cost of buying that many items, where the regular price is charged for orders of less than 10 items, a 10% discount is applied for orders of between 10 and 99 items, and a 20% discount is applied for orders of 100 or more items. There should also be a method called make\_purchase that receives the number of items to be bought and decreases amount by that much.
3. Write a class called Password\_manager. The class should have a list called old\_passwords that holds all of the user’s past passwords. The last item of the list is the user’s current password. There should be a method called get\_password that returns the current password and a method called set\_password that sets the user’s password. The set\_password method should only change the password if the attempted password is different from all the user’s past passwords. Finally, create a method called is\_correct that receives a string and returns a boolean True or False depending on whether the string is equal to the current password or not.
4. Write a class called Converter. The user will pass a length and a unit when declaring an object from the class—for example, c = Converter(9,'inches'). The possible units are inches, feet, yards, miles, kilometers, meters, centimeters, and millimeters. For each of these units there should be a method that returns the length converted into those units. For example, using the Converter object created above, the user could call c.feet() and should get 0.75 as the result.
5. **(Optional)** Implement an object oriented version of tic-tac-toe game.
   * For two players and in each round the program asks the players to specify the position.
   * After giving the position by each player, the board is printed in the terminal.
   * The program determines the winner at the end.

**Assignment:**

Create a car parking machine application. The main functionality of this terminal application is to check-in and check-out cars from a day car park while it keeps track on the current capacity and calculates the owed parking fee when a car checks out.

The classes that need to be programmed for this assignment are described below.

* CarParkingMachine class with the following attributes:
  + capacity (int, default 10) - how many cars can be checked-in at the same time.
  + hourly\_rate (float, default 2.50) - used to calculate the parking fee.
  + parked\_cars (dict => key: license\_plate, value: ParkedCar object) - keeps track of the cars present in the car parking
* This class has the following methods:
  + init (construtor) that receives the capacity, hourly\_rate and sets the parked\_cars dict as empty.
  + check\_in that receives the license\_plate (str), the time (datetime) that the car is parked. If the maximum capacity is reached it should return False and not check-in the car.
  + check-out that receives the license\_plate (str) and returns the owed parking fee total (by calling the get\_parking\_fee method).
  + get\_parking\_fee that receives the license\_plate (str) and calculates/returns the parking fee (hourly\_rate \* whole parking hours rounded-up, with max of 24 hours).
* ParkedCar class storing information of a parked car with the following attributes:
  + license\_plate (str) - license plate of the car.
  + check-in (datetime) - datetime object of the time checked-in.
* This class has the following methods:
  + init (construtor) that receives the license\_plate and check-in

Additional research is required on how to handle datetime objects to calculate the difference between the check-in and check-out time and how to round-up in hours. *Hint*: import the datetime module (datetime and timedelta objects).

To test your class, use the provided unit test file (see ./assignment\_data/car\_parking\_test.py) and complete the test functions with your own code.

For the main of this program use a CarparkingMachine with default values (capacity: 10, rate: 2.50).

Build menu structure as following. The input can be case-insensitive (so E and e are valid inputs)

* [I] Check-in car by license plate
  + output: License registered (only if checked-in)
  + or: Capacity reached!
* [O] Check-out car by license plate
  + output: Parking fee: {parking\_fee} (2 decimals!) EUR (if license is found)
  + or: License {license\_plate} not found!
* [Q] Quit program

**Extra Steps:**

The follwoing concepts are not part of the main learning path, but can be considered as *optional* learning activities for those who seek more challenges:

* Inheritance
* Polymorphysm
* Class methods
* Static methods
* Duck Typing
* Magic methods
* Aggregation and Composition
* Dataclasses