Lab 2

Object Oriented Programming

Task 1: Class Concepts:

1. Class vs. Object:

- Explain the difference between a class and an object in Python

Class:

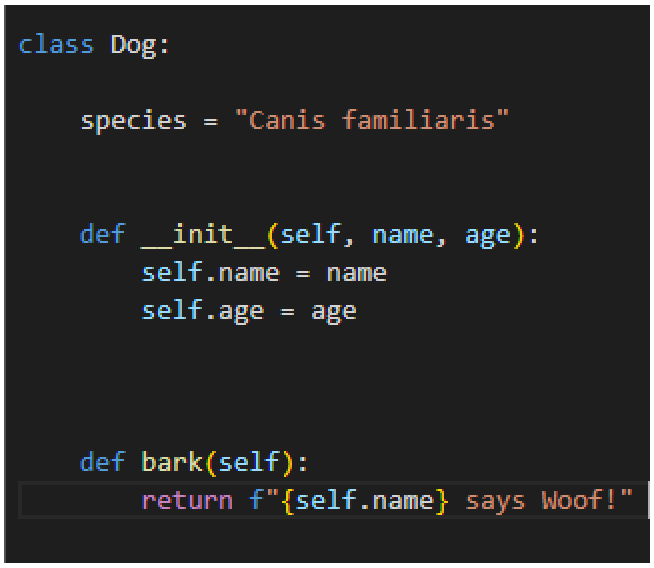
- A class is a blueprint or a template for creating objects. It defines a set of attributes (data) and methods (functions) that the objects created from the class will have.
- Classes encapsulate data for the object and provide methods to manipulate that data.
- A class does not occupy memory until an object is created from it.

Object:

- An object is an instance of a class. It is created based on the class template and contains real values instead of variables.
- Each object has its own state and can use the methods defined in the class to perform actions.
- Objects occupy memory, and multiple objects can be created from the same class with different attribute values.

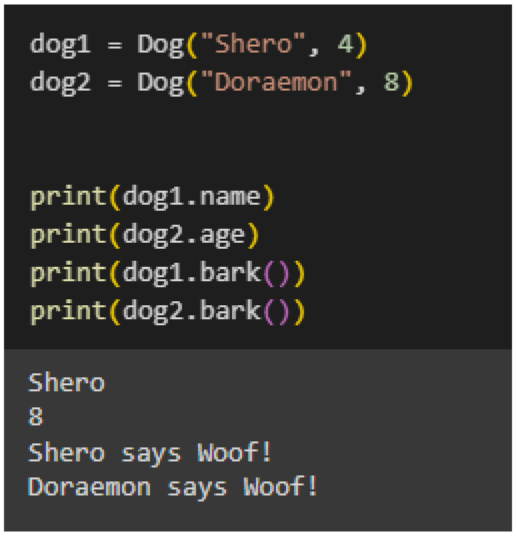
Example:

Class



```python
class Dog:

    species = "Canis familiaris"


    def __init__(self, name, age):
        self.name = name
        self.age = age



    def bark(self):
        return f"{self.name} says Woof!"
```

Object



```python
dog1 = Dog("Shero", 4)
dog2 = Dog("Doraemon", 8)


print(dog1.name)
print(dog2.age)
print(dog1.bark())
print(dog2.bark())
```

```
Shero
8
Shero says Woof!
Doraemon says Woof!
```

Constructor Method (__init__) vs __str__() Function:

- Explain the difference between them in Python.

➢ Constructor Method (__init__):

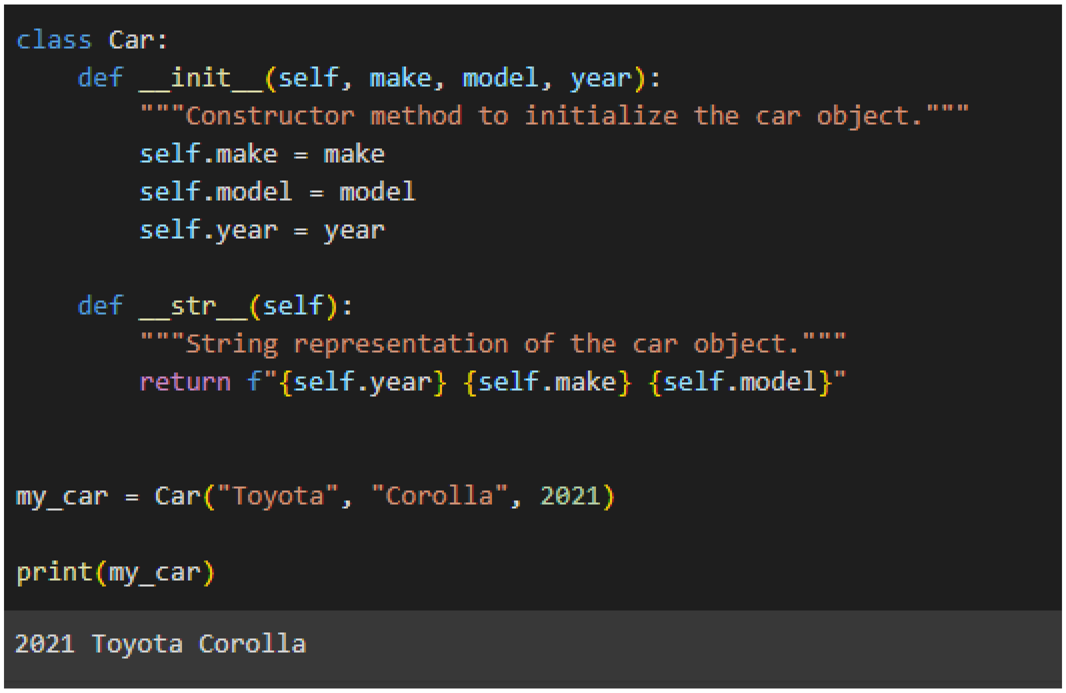
- The __init__ method is a special method called when an object is created from a class. It is used to initialize the object's attributes with specific values.
- This method is often referred to as the initializer or constructor.
- The __init__ method can take parameters to set the initial state of the object.

➢ __str__() Function:

- The __str__() function is a special method that defines how an object is represented as a string. This method is called when you use the print() function or str() on an object.
- It is used to provide a human-readable string representation of the object, making it easier to understand the object's state at a glance.
- If the __str__() method is not defined, Python will use the default implementation, which typically returns a string that includes the object's type and memory address.

- *Example.*

➢ Constructor Method (__init__) vs __str__() Function:



```python
class Car:
    def __init__(self, make, model, year):
        """Constructor method to initialize the car object."""
        self.make = make
        self.model = model
        self.year = year

    def __str__(self):
        """String representation of the car object."""
        return f"{self.year} {self.make} {self.model}"


my_car = Car("Toyota", "Corolla", 2021)

print(my_car)
```

```
2021 Toyota Corolla
```