1. Python classes

Python classes provides to build data and functionality together. For example handling customer information is easier when all the customer's data is encapsulated inside a single class. That would contain customer's name, age etc.

Almost every data type you have used so far in Python is a wrapped inside of a certain class (`int`, `str`, `list`, etc.). Python is object oriented programming language after all.

Let's create an example class `Dog` which stores name and age of a dog:

1.1: Example program of Python class: `example1.py`

```
# example1.py

class Dog:
    name = None
    age = None


if __name__ == "__main__":
    dog = Dog()
    dog.name = "Brian"
    dog.age = 10
    print(f"name: {dog.name}, age: {dog.age}")
```

1.2: Output of `example1.py`

```
$ python3 example1.py
name: Brian, age: 10
```

2: Methods and `__init__`

To add functionality to a class you need to create methods. In Python methos can be created like you would create a function. The difference is that in the definition the method takes `self` as the first input parameter.

```
def method(self, ...):
```

wish.

Now when you call a new `Dog` object it creates an "empty" object (all variables are `None`). To create an objects with instances customized to a specific initial state we can use `__init__`:

```
def __init__(self, ...):
```

Necessary arguments needed to initialize an object are passed to `__init__`. For our `Dog` class those would be name and age.

2.1: Example program with methods and `__init__`: `example2.py`

```
# example2.py

class Dog:
    # Initializer:
    def __init__(self, new_name, new_age):
        self.name = new_name
        self.age = new_age

    # Other methods:
    def info(self):
        print(f"Name: {self.name}, Age: {self.age}")

    def woof(self, phrase):
        print(f"{self.name}: {phrase}")


if __name__ == "__main__":
    dog1 = Dog("Max", 2)
    dog2 = Dog("Lucy", 12)
    dog1.info()
    dog2.info()
    dog1.woof("Woof woof!")
    dog2.woof("Im a dog!")
```

2.2: Output of `example2.py`

```
$ python3 example2.py
Name: Max, Age: 2
Name: Lucy, Age: 12
Max: Woof woof!
Lucy: Im a dog!
```

Note that `self` is passed only in the definition of a method.

Last modified: Monday, 3 July 2023, 9:43 AM