Class Basics

quiz

Exercise 1: Class vs instance (i.e. object) attributes

class definition

```
class Exercise1:
  a = 1
  def seta(self, value):
    self.a = value
```

create some objects

```
e1 = Exercise1

e2 = Exercise1()
e2.a = 2

e3 = Exercise1()
e3.seta(3)

e4 = Exercise1()
```

what will be printed?

```
print(f'{e1.a = }')
print(f'{e2.a = }')
print(f'{e3.a = }')
print(f'{e4.a = }')
```

Exercise 1: Class vs instance (i.e. object) attributes

answers

`e1.a = 1`

*e1 refers to **the class** not an object, so e1.a accesses the **class attribute** a*

`e2.a = 2`

*e2 is an object, so `e2.a = 2` **creates an instance attribute** a and assigns it the value 2*

`e3.a = 3`

the self keyword in the method refers to the object, so is an object, so `self.a = 3` would be equivalent to `e3.a = 3` (like we saw previously)

`e4.a = 1`

*e4 is an object, but it **does not have an instance attribute** a, thus `e3.a` **resolves to the class attribute** a, which has the value 1*

Exercise 1: Class vs instance (i.e. object) attributes

class definition

```
class Exercise1:
  a = 1
  def seta(self, value):
    self.a = value
```

create some objects

```
e1 = Exercise1

e2 = Exercise1()
e2.a = 2

e3 = Exercise1()
e3.seta(3)

e4 = Exercise1()
```

change an attribute

```
e1.a = 4
```

what will be printed now?

```
print(f'{e1.a = }')
print(f'{e2.a = }')
print(f'{e3.a = }')
print(f'{e4.a = }')
```

Exercise 1: Class vs instance (i.e. object) attributes

answer

`e1.a = 4`	*This line changed the class attribute a to the value 4. All examples referring to the class attribute will now print 4 instead of 1*

`e1.a = 4`	*class attribute*
`e2.a = 2`	*instance attribute*
`e3.a = 3`	*instance attribute*
`e4.a = 4`	*class attribute*

Exercise 2: Class vs instance vs static methods

class definition

```
class Exercise2:
    def seta_1(self, value):
        self.a = value
    @classmethod
    def seta_2(cls, value):
        cls.a = value
    @staticmethod
    def seta_3(value):
        a = value
```

call the methods on a new object

```
e1 = Exercise2()


e1.seta_1(1)
e1.seta_2(2)
e1.seta_3(3)
```

what will be printed?

```
print(f'{e1.a = }')
```

Exercise 2: Class vs instance vs static methods

`e1.seta_1(1)`	*This line will create a new instance attribute 'a' and assign it the value 1*
`e1.seta_2(2)`	*This line will create a new class attribute 'a' and assign it the value 2*
`e1.seta_3(3)`	*This line will create a new local variable 'a' (inside the scope of the static method) and assign it the value 3. This has no effect anywhere in the code.*

answer

`e1.a = 1`	*Since there is an instance attribute 'a', it will return this value*

Key takeaways

We can store data on a **class level** (shared across all objects of that class) or on an **instance level** (each instance has its own)

There are 3 types of methods in classes:

- **instance methods**: first argument passed to the method is the object that calls the method
- **class methods**: first argument is the class of the object that calls the method. Can also be called from the class itself (not an object)
- **static methods**: no extra arguments. They are simple functions defined in the namespace of a class