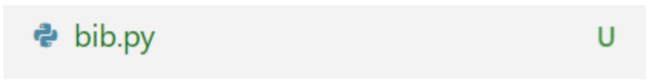
Exercise Courses

In this exercise we are going to go over step-by-step how to create classes and how to use them.

Step 1: create a python file

Create a python file called **book.py**.



bib.py U

Step 2: create a Book class

A class is a code template for creating objects. Objects have variables and have functions associated with them. In python a class is created by the **keyword class**. An object is created using the constructor of the class. This object will then be called the instance of the class.

In **book.py**, create the a class named Book.

```
class Book:
```

Step 3: create constructor

Constructors are generally used for instantiating an object. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. In Python the **init**() method is called the constructor and is always called when an object is created.

In **book.py**, create the constructor function given 2 parameters (name,pages). The constructor is used to create **Book objects**.

```
class Book:

    def __init__(self, fname, fpages fisbnummer):
        self.name = fname
        self.pages = fpages
        self.isbnummer = fisbnummer
```

The term **self** represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

Step 4: __eq__ function

We want to be able to compare books based on isbnummer. For this we can create an __**eq**__ function. Python automatically calls the __**eq**__ method of a class when you use the == operator to compare the instances of the class. By default, Python uses the is operator if you don't provide a specific implementation for the __**eq**__ method.

```
class Book:

    def __init__(self, name, pages, isbnummer):
        self.name = name
        self.pages = pages
        self.isbnummer = isbnummer

    def has_more_pages(self, book):
        return self.pages > book.pages

    def __eq__(self,isbnummer):
        return self.isbnummer == isbnummer
```

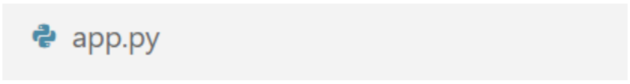
Step 5: __str__ function

We want to be able to print out a text with all the information about the object. The **__str__** method in Python represents the class objects as a string – it can be used for classes. The **__str__** method should be defined in a way that is easy to read and outputs all the members of the class.

```
class Book:

    def __init__(self, name, pages, isbnummer):
        self.name = name
        self.pages = pages
        self.isbnummer = isbnummer

    def has_more_pages(self, book):
        return self.pages > book.pages

    def __eq__(self,isbnummer):
        return self.isbnummer == isbnummer

    def __str__(self):
        return f'The name of the book is {self.name}, this book has {self.pages} pages and has as
isbnummer {self.isbnummer}.'
```

Step 6: create app.py file

Create a python file called **book.py**.


app.py

Step 7: create book object

Create a book object in the app.py file. First you need to import the book.py file, then your can create an object and print out the object. The print functions will look at the **str** function in the book.py file and based of that will determine what would be printed out.

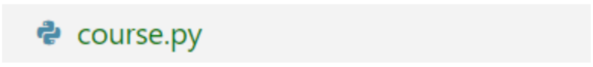
```
from book import Book


book1 = Book("Java",203,382983)
book2 = Book("Python",124,453234)

print(book1)
print(book2)

if book1 == book2:# the compare function __eq__ in the book class is used to compare these objects
    print(True)
else:
    print(False)
```

The output of the above code is:

```
The name of the book is Java, this book has 203 pages and has as isbnummer 453234.
The name of the book is Python, this book has 124 pages and has as isbnummer 453234.
False
```

Step 8: create course.py file

Create a python file called **course.py**.


course.py

Step 9: create a course class

In **course.py**, create the a class named Course.

```
class Course:
```

Step 10: create constructor

In **course.py**, create the constructor function given 2 parameters (fname,fcourse_book). The constructor is used to create **Course objects**.

```
class Course:

    def __init__(self, fname, fcourse_book):
        self.name = fname
        self.course_book = fcourse_book
```

Step 11: functions

Same as in the class Book, we want to be able to compare courses and print out the information of the course object. We compare courses based on name and course book. This ofcourse can be changed depending what you need.

```
class Course:

    def __init__(self, fname, fcourse_book):
        self.name = fname
        self.course_book = fcourse_book

    def __eq__(self, fname, fcourse_book):
        return self.name == fname and self.course_book == fcourse_book

    def __str__(self):
        return f'The course {self.name} follows the following book: \n {self.course_book} '
```

Step 12: create course object

Create a course object in the app.py file. First you need to import the course.py file, then your can create an object. We can use to previous made books to create the course objects.

```
from book import Book
from course import Course

book1 = Book("Impossible",203,453234)
book2 = Book("Python",124,453234)

course1 = Course("java programming", book1)
course2 = Course("programming1", book2)

print(course1)
print(course2)

if course1.__eq__(course2.name, course2.course_book):
    print(True)
else:
    print(False)
```