PHOT 110: Introduction to programming

Exercises 14: Objects, Methods, and Class Hierarchy

Michaël Barbier, Spring semester (2023-2024)

1. Library system

Use the code below to define a new class called `Book`.

```python
class Book():
    """Book class defines books in a library"""

    # Attributes of the class
    title = "Nasreddin Hoca Fıkraları"
    author = "Memet Fuat"
    category = "Philosophy"
    language = "TR"

    # Methods of the class
    def print_info(self):
        print(f"'{self.title}' by {self.author}")


if __name__ == "__main__":
    library = []
    book = Book()
    print("The first default book is:")
    book.print_info()
    library.append(book)
```

```
The first default book is:
'Nasreddin Hoca Fıkraları' by Memet Fuat
```

Afterwards, make the following adaptations/extensions:

- Add a constructor method to the `Book` class using `__init__(self, title, etc.)`, and your favorite book to the `library`.
- Add a for-loop to print the information of each book.
- Adapt the constructor to include two extra attributes:
  - The id of the book as a string, example: `id = "AB-14"`,
  - The total number of books with that id: `n_total = 1`,
  - The number of available books with that id: `n_available = 1`
- Add an extra method `borrow(book_id)` which:
  - Returns with a confirmation message, and reduces the number of available books by one if there are still books with that id.
  - Returns with a message: "The book: [the_book_title] is not available at the moment." if there are no books available anymore.

2. Subclasses

First define a class: `Ticket`

```
class Ticket():
    """The Ticket class is defined for general transport vehicles"""

    # Attributes of the class
    customer = "your_name"
    price = 20

    # Methods of the class
    # ...
```

- Add a constructor (`__init__(self, customer, price, etc.)`) for the class,
- Then add a method which
- Then create a derived class `Metro` (subclass), for which you add the following attributes:

```
valid_from_time = [get_the_current_date_time]
valid_until_time = [valid_from_time + 2 hours]
```

- Afterwards create an instance of your new Metro class and check whether the `valid_until_time` is correct.

For the date and time in Python you can use the built-in `datetime` package. See the example below:

```python
from datetime import datetime, timedelta

current_time = datetime.now()
current_time_str = current_time.strftime("%Y-%m-%d %H:%M:%S")
print("It is now: " + current_time_str)

later_time = current_time + timedelta(hours=3)
later_time_str = later_time.strftime("%H:%M:%S")
print("Three hours later the time will be: " + later_time_str)
```

```
It is now: 2025-05-19 22:37:11
Three hours later the time will be: 01:37:11
```