05-Exercises

November 13, 2024

1 05 - Exercises: object-oriented programming

This week we saw: - Python's imperative paradigm in object-oriented programming - Classes, modules and packages

Here are some exercises to help you get comfortable with these concepts :)

1.1 New format

Let's try a different format for the exercises: this set contains 3 exercises of increasing difficulty levels. We'll start with an exercise to build your strength in OOP from the ground up, then we'll add some details and complexity in the following one, and lastly, we'll practice advanced concepts. Let's start!

1.1.1 Constructing a library

Your final objective will be to construct a library class that is able to handle different media. We will start with constructing a simple class for a single media type, a book, then build on that class to construct a more general abstraction of a medium, and lastly, we will construct a whole library concept that is able to deal with all these different media types. Feel free to adapt the exercises according to your creativity! You can implement different media types (e.g. CDs, Notebooks, etc.) and you can add other methods accordingly (e.g. `get_song_number()` for the CD item, etc.)

1. Practice basic class structure - Difficulty level: 1 (5 points) Create a class called `Book` that represents a book in a library.

Let's break down the task: 1. Each book should have the following attributes: - `title` (string) - `author` (string) - `pages` (integer) - `year` (integer) 2. Add a method `book_info()` that returns a string containing the book's title, author, and year in a readable format like `"Title: book title, Author: author name, Year: year"`. 3. Add a method `is_long_book()` that returns `True` if the book has more than 300 pages, otherwise returns `False`. 4. Create a few instances of the `Book` class to test the `book_info()` and `is_long_book()` methods.

```
[ ]:
```

2. Practice inheritance and method overriding - Difficulty level: 2 (5 points) Extend the library concept by creating a hierarchy of media items.

Let's break down the task: 1. Create a base class called `LibraryItem` with the following attributes: - `title` (string) - `author` (string) - `year` (integer) 2. Add a method `get_description()`

to `LibraryItem` that returns a string in the format `"Title: medium title, Author: author name, Year: year"`. 3. Create two subclasses that inherit from `LibraryItem`: `Book` and `Magazine`. 4. For the `Book` class, add an additional attribute `pages` (integer). 5. For the `Magazine` class, add an additional attribute `issue_number` (integer). 6. Override the `get_description()` method in both `Book` and `Magazine` classes to include the specific attributes. For example, the `Book` class should return `"Title: title, Author: author name, Year: year, Pages: pages"`, and the `Magazine` class should return `"Title: title, Author: author name, Year: year, Issue Number: issue_number"`. 7. Create instances of both `Book` and `Magazine` and test the `get_description()` method.

[ ]:

```

```

3. Practice polymorphism, encapsulation, and collection management - Difficulty level: 3 (5 points) Build a small system for managing a library with different types of items.

Let's break down the task: 1. Create a base class `LibraryItem` as before, but this time add an attribute `_checked_out` (boolean, default is `False`). Add methods `check_out()` and `return_item()` to change the `_checked_out` status. 2. Inherit `LibraryItem` into `Book` and `Magazine`, and add the relevant attributes as in Exercise 2. 3. Create a class `Library` that manages a collection of `LibraryItem` objects. The Library class should have: - A private list attribute `_collection` to store items. - A method `add_item(item)` to add a `LibraryItem` to the collection. - A method `list_items()` to print details of each item in the collection, using polymorphism to call `get_description()` on each item. - A method `check_out_item(title)` to find an item by title, mark it as checked out, and return a message indicating whether the checkout was successful (if the item exists and isn't already checked out). - A method `return_item(title)` to mark a checked-out item as returned. 4. Implement error handling for cases like trying to check out an item that's already checked out or returning an item that isn't checked out. 5. Test the system by adding various `Book` and `Magazine` instances to the library, checking items in and out, and listing the collection's current state.

[ ]:

```

```

1.2 NOTE:

These exercises might seem difficult at the beginning, but don't worry! Here are some reasons why you should not be scared: 1. You have double the time to solve them (2 weeks instead of the usual 1) 2. You can practice the lowest difficulty exercises until you feel comfortable 3. You can proceed step by step to understand every request better 4. They leave space for creativity, so go online and snoop around to discover new concepts! 5. You can always use BIKI or ChatGPT (see down below for tips) 6. You can ask me for help if everything else doesn't work :)

Tips on how to use LLMs (large language models): instead of copy-pasting the whole exercise into the prompt, try asking the chatbot **single questions**: e.g. "Can you tell me what a private attribute of a class is in Python?", or "Can you tell me how to override an inherited method in a child class in Python?"

[ ]:

```

```