Exercises for Course 262:

Advanced Python Programming: Hands-On

Contents

Exercise 1: Object-Oriented Programming

In this exercise, you will learn how to

- create a class
- create objects
- create attributes and access them
- create methods and call them

1. Create *products.py*. Create a class called `Product`, with the following attributes:
   - `id`
   - `description`
   - `price`

2. Create a constructor for `Product`, which accepts initial values for the 3 attributes.
   - `id` and `description` are required
   - `price` is optional with a default value of 0.

3. Create another file called *products-test.py*. Create 3 instances of `Product` as follows:

ID	Description	Price
101	Coke Can	25.00
208	Lays Chips	105.00
560	Mott's Apple Juice	200.00

4. Create a list named `products`, and add the 3 products you just created to the list.

5. Iterate through the `products` list, and print the attributes of your products.

   Sample Output:

```
ID: 101
Description: Coke Can
Price: 25.0
```

```
ID: 208
Description: Lays Chips
Price: 105.0

ID: 560
Description: Mott's Apple Juice
Price: 200.0
```

This is the end of the exercise.

Exercise 2: Decorators

In this exercise, you will learn:

- How to create class methods
- How to invoke class methods

1. In *products.py*, create a class called `ProductsService`. This class will be responsible for retrieving `Product` objects from a data source such as a database. But for now, we will use a list in place of an actual database.

2. In `ProductsService`, create a *static* attribute called `products`. It should be a list containing the following 3 Product objects:

ID	Description	Price
101	Coke Can	25.00
208	Lays Chips	105.00
560	Mott's Apple Juice	200.00

3. In `ProductsService`, create a *class* method called `get_products()`, which should return the `products` list.

4. In `ProductsService`, create a *class* method called `find()`, which should accept an `id` as a parameter. `find()` should return a `Product` object that matches the `id` parameter. It should return `None` if `id` was not found.

5. Create another file called *products-service-test.py*. Use `ProductsService`'s `get_products()` method to retrieve the list of products.

6. Iterate through the list of products, and print the attributes of your products.

   Sample Output:

```
ID: 101
Description: Coke Can
Price: 25.00
```

```
ID: 208
Description: Lays Chips
Price: 105.00

ID: 560
Description: Mott's Apple Juice
Price: 200.00
```

7. Still in *products-service-test.py*, ask the user for a product ID. Use `ProductsService`'s `find()` method to retrieve the product and display it's attributes. If the product ID entered does not exist, display the message *"That product doesn't exist."*

   Sample Output:

```
Enter a product ID: 560
Description: Mott's Apple Juice
Price: 200.00

Enter a product ID: 100
That product doesn't exist.
```

This is the end of the exercise.

Exercise 3: Inheritance and Polymorphism

In this exercise, you will learn how to:

- override methods
- make use of polymorphism
- make attributes private1

1. In the `Product` class, override the `__str__()` method to return the string representation of the product. Here's an example of the expected string that's returned:

   ```
   ID: 208
   Description: Lays Chips
   Price: 105.00
   ```

2. Modify *products-test.py* so that the product instances are printed, instead of the individual attributes of the instances. The output should still be the same, but this time, you should be printing `Product` objects instead of attributes of `Product` objects.

3. In `ProductsService`, make the `products` attribute private. In *products-service-test.py*, try accessing the `products` attribute directly using the class name. You should not be able to do so.

✋ *This is the end of the exercise.*

Exercise 4: Exceptions

In this exercise, you will learn how to:

- raise exceptions
- handle exceptions using `try-except` block
- create your own exception type

1. Modify `ProductsService`'s `find()` method so that instead of returning `None` when a product is not found, raise a `LookupError` instead.

2. Modify *products-service-test.py* so that when a `LookupError` occurs, the message "That product doesn't exist" is displayed.

   ```
   Enter a product ID: 100
   That product doesn't exist.
   ```

3. In *products.py*, create your own exception type called `ProductNotFoundException` whose default error message is "That product doesn't exist".

4. Modify `ProductsService`'s `find()` method so that your `ProductNotFoundException` is raised instead of `LookupError`.

5. Modify *products-service-test.py* to handle `ProductNotFoundException` instead of `LookupError`.

✋ *This is the end of the exercise.*