Advanced Python Concepts & Exercises

Part 1: Object-Oriented Programming (OOP)

1.1 Dunder Methods (Magic Methods)

Dunder (double underscore) methods allow customization of object behavior.

Commonly Used Dunder Methods

- __init__(self, ...) - Constructor, initializes an object.
  - 🧸 **Toddler Example:** Imagine you get a new toy. Before you can play, you have to unwrap it. __init__ is like unwrapping and setting up your toy before using it.
- __str__(self) - String representation for print(obj).
  - 🧸 **Toddler Example:** If you have a teddy bear and someone asks what it looks like, you describe it as "a fluffy brown bear." __str__ is how an object describes itself in words.
- __repr__(self) - Official string representation, used in debugging.
  - 🧸 **Toddler Example:** If a store needs to stock your teddy bear, they need an exact description like "TeddyBear(color='brown', size='medium')." __repr__ is the official way an object explains itself to computers.
- __len__(self) - Returns the length of an object.
  - 🧸 **Toddler Example:** If you have a toy box, __len__ tells you how many toys are inside when you count them.
- __getitem__(self, key) - Enables indexing (obj[key]).
  - 🧸 **Toddler Example:** If your toy box has different sections, __getitem__ is like picking out a toy by saying, "Give me the third one."
- __setitem__(self, key, value) - Enables item assignment (obj[key] = value).
  - 🧸 **Toddler Example:** If you want to replace the third toy in your box, __setitem__ lets you put a new one in its place.
- __call__(self, ...) - Allows an object to be called as a function.
  - 🧸 **Toddler Example:** If you press a toy car's button and it starts moving, that's like calling an object as a function.
- __eq__(self, other) - Customizes equality checks (obj1 == obj2).
  - 🧸 **Toddler Example:** If you have two identical teddy bears, __eq__ lets you check if they are the same instead of just looking at them one by one.

Real-life Use Case Exercise:

Create a ShoppingCart class that:

1. Uses __getitem__ and __setitem__ to manage products by their names.

2. Implements __len__ to return the number of unique items.
3. Uses __call__ to calculate the total price.

1.2 Dataclasses

Python’s @dataclass simplifies class creation.

Real-life Use Case Exercise:

Create a BankAccount dataclass that:

1. Stores account_number, holder_name, and balance.
2. Uses __post_init__ to validate that balance is not negative.

1.3 Pydantic for Data Validation

Pydantic validates structured data.

Real-life Use Case Exercise:

Create a UserRegistration model that:

1. Ensures email is valid.
2. Checks password has at least 8 characters.
3. Ensures age is 18 or older.

Part 2: Asynchronous Programming (asyncio)

2.1 Basic async and await

Real-life Use Case Exercise:

Build an async function download_file(url: str) that:

1. Simulates downloading a file using asyncio.sleep(2).
2. Returns the file name when done.

2.2 Running Tasks in Parallel (asyncio.gather)

Real-life Use Case Exercise:

Extend download_file(url: str) to:

1. Download multiple files in parallel.
2. Use asyncio.gather() to wait for all downloads to finish.

2.3 Async Iterators (async for)

Real-life Use Case Exercise:

Create an async generator sensor_data_stream() that:

1. Yields fake temperature readings every second.
2. Uses async for to process readings in real-time.

Final Thoughts

This document fully covers: ✅ **OOP:** Dunder methods, Dataclasses, Pydantic. ✅ **Async Programming:** asyncio, queues, semaphores, aiohttp. ✅ **Real-world applications.** ✅ **Toddler-friendly examples for better understanding.**

Would you like solutions for any of these exercises? 😊