Week 1 - Lesson 4

OOP Basics and Dataclasses (Python)

1) Class vs Object

A **class** is a blueprint (template). An **object** is a real instance created from that blueprint, holding actual values.

```
# class = blueprint
class Task:
pass

# object = instance
t1 = Task()
```

2) __init__ and self (Java 'this')

__**init**__ is the constructor in Python. It runs when you create an object. **self** refers to the current object (like **this** in Java).

```
class Task:
def __init__(self, text, done=False):
# text, done are inputs (temporary/local)
# self.text, self.done are stored on the object (fields)
self.text = text
self.done = done

t1 = Task("banana") # done defaults to False
t2 = Task("toy car", done=True) # override default
```

Key idea

Parameters like **text** and **done** are temporary inside __**init**__. Using **self.** stores them on the object so they stay after the constructor finishes (so you can access **t1.text** and **t1.done**).

3) What is a dataclass?

A **dataclass** auto-generates boilerplate for data-holding classes (like __init__ and a nice print/repr). It helps you write clean, short classes for simple models.

```
from dataclasses import dataclass

@dataclass
class Task:
text: str
done: bool = False

t = Task("study")
print(t.text, t.done) # study False
t.done = True
print(t) # Task(text='study', done=True)
```

4) Import styles: import vs from

Goal	Example	Notes
Import the whole module	import dataclasses @dataclasses.dataclass class Task: ...	Clear where it comes from
Import only what you need	from dataclasses import dataclass @dataclass class Task: ...	Shorter to write

5) Dataclasses + JSON (why asdict is useful)

JSON can't store custom objects directly. A common approach is converting a dataclass object to a dict first.

```
from dataclasses import asdict
import json

tasks = [Task("study", True), Task("gym", False)]
data = [asdict(t) for t in tasks] # Task -> dict

with open("data.json", "w", encoding="utf-8") as f:
json.dump(data, f, indent=2)
```

Quick recap

- Class = blueprint; Object = instance with real values.
- __init__ is the constructor; self is the current object (Java this).
- self.field stores data on the object; plain variables inside __init__ are temporary.
- @dataclass generates __init__ and a readable repr for data models.
- Use asdict() when you need to save dataclass objects to JSON.

Mini-check questions (with answers)

A) What prints?

```
t = Task("study")
print(t.text, t.done)
# Answer: study False
```

B) After toggling done, what prints?

```
t.done = True
print(t)
# Answer: Task(text='study', done=True)
```