ChatGPT

Saved memory full ⓘ

explain np.random.seed()

`np.random.seed()` is used in **NumPy** to control the randomness, so that your random results can be **reproduced**.

Here's the breakdown:

🔹 What happens normally:

When you use functions like `np.random.randint()`, `np.random.randn()`, or `np.random.normal()`, NumPy generates random numbers using a **random number generator**.
By default, it uses the **current system time** as the seed, so every time you run the code, you usually get **different random numbers**.

🔹 What `np.random.seed()` does:

- It **sets the seed value** for NumPy's random number generator.
- If you give the **same seed number**, NumPy will generate the **exact same random numbers** every time you run the code.

Example:

```python
import numpy as np

# Without setting seed
print(np.random.randint(1, 10, 5))
print(np.random.randint(1, 10, 5))
```

👉 Each run gives different results because the generator is not fixed.

```python
import numpy as np

np.random.seed(42)   # Fix the randomness

print(np.random.randint(1, 10, 5))
print(np.random.randint(1, 10, 5))
```

👉 Every time you run this, the numbers will be the same:

```csharp
[7 4 8 5 7]
[3 7 4 8 8]
```

🔹 Why it is useful:

- **Reproducibility**: If you are writing research code, training a machine learning model, or debugging, you want the same "random" results every time you rerun the script.
- **Sharing with others**: Others can reproduce your exact results if you set the same seed.

⚡ In short:

`np.random.seed()` **controls randomness** → same seed = same results, different seed (or no seed) = different results.

Do you want me to also explain the difference between `np.random.seed()` and the newer way `np.random.default_rng()` (which is recommended now)?

↪ np.random.seed(42)

can we use any value in seed

Yes 👍 — you can use **any integer value** as the seed in `np.random.seed()`.

🔹 Details:

- The seed must be a **non-negative integer**.
- Commonly people use small numbers like `0`, `1`, `42` (famous in programming culture), etc.
- Technically, NumPy accepts values from **0 up to $2^{32} - 1$** (about 4.29 billion).
- If you pass the same seed, you'll always get the same random sequence.
- If you don't set a seed (or set `None`), NumPy will use the current system time, so results will differ on each run.

Example:

python Copy code

```python
import numpy as np

np.random.seed(0)
print(np.random.randint(1, 10, 5))  # Always the same output

np.random.seed(123)
print(np.random.randint(1, 10, 5))  # Different, but still reproducible

np.random.seed(42)
print(np.random.randint(1, 10, 5))  # Another reproducible set
```

So yes — **you can use any integer seed you like**, the choice doesn't matter, it just fixes the random sequence.

👉 Do you want me to also show what happens if you use the **same seed multiple times inside one script**?