Random Module in Python - Complete Guide

The `rando` module in Python is used to generate pseudo-random numbers. It is commonly used in simulations, games, data sampling, testing, and many other areas that require randomness.

How to Import

```
import random
```

---

1. random.random()

- **Description:** Returns a float number between `0.0` and `1.0`.
- **Use Case:** Useful for simulating probabilities or percentages.

Examples:

```
random.random()# Example 1: 0.8234
random.random()# Example 2: 0.1729
random.random()# Example 3: 0.5312
```

---

2. random.randint(a, b)

- **Description:** Returns a random integer between `a` and `b` (inclusive).
- **Use Case:** Picking a number in a defined integer range.

Examples:

```
random.randint(1,  10) # Example 1: 7
random.randint(100,  200)   # Example 106
random.randint(-10, 0)# Example 3: -6
```

---

3. random.uniform(a, b)

- **Description:** Returns a float number between `a` and `b`.
- **Use Case:** Useful for simulating continuous values like temperature, distance.

Examples:

```
random.uniform(1, 5) # Example 1: 3.789
random.uniform(0, 1) # Example 2: 0.481
random.uniform(-5, 5) # Example 3: -2.56
```

4. random.choice(sequence)

- **Description:** Returns a random element from a non-empty sequence (list, tuple, string).
- **Use Case:** Selecting a random item, like a quiz question or lottery draw.

Examples:

```
random.choice([1, 2, 3])   # Example 1:
2 random.choice("Python")  # Example 2:
'h' random.choice((10, 20, 30))
```

5. random.choices(sequence, k=n)

- **Description:** Returns a list of `k` random elements (with replacement).
- **Use Case:** Useful when multiple selections with duplicates are allowed.

Examples:

```
random.choices([1, 2, 3],          # Example 1: [2, 2]
k=2)                              # Example 2: ['B', 'A',
random.choices(["red", "blue"], k=5) # Example 3: ['blue', 'red',
'blue',
```

6. random.sample(sequence, k)

- **Description:** Returns `k` unique random elements (without replacement).
- **Use Case:** Useful when sampling without duplicates, e.g., lottery winners.

Examples:

```
random.sample([1, 2, 3, 4], 2)   # Example 1: [1, 3]
random.sample(range(10), 5)      # Example 2: [0,
2, 5, 7, 9] random.sample("ABCDE", 3) # Example 3:
```

7. random.shuffle(sequence)

- **Description:** Shuffles the sequence in place (works on lists only).
- **Use Case:** Randomizing the order of elements (e.g., shuffling a deck of cards).

Examples:

```
items = [1, 2, 3,
                                  # Example 1: [3, 2,
4]
random.shuffle(items)            # Example 2: ["blue",
                                 "red"]
colors = ["red",
```

8. random.seed(value)

- **Description:** Initializes the random number generator for reproducibility.
- **Use Case:** Ensures same random results for testing and debugging.

Examples:

```
random.seed(5)
print(random.random()) # Example 1: Always same
output random.seed(10)
print(random.randint(1, 10))      # Example 2:
Reproducible random.seed(7)
```

9. random.randrange(start, stop[, step])

- **Description:** Returns a random integer from a given range.
- **Use Case:** Selecting a value from a custom step range.

Examples:

```
random.randrange(1, 10)      # Example 1: 6
random.randrange(0, 100, 10)      # Example 2: 30
random.randrange(5, 50, 5) # Example 3: 25
```

10. random.getrandbits(k)

- **Description:** Returns an integer with `k` random bits.
- **Use Case:** Useful in cryptography and generating binary flags.

Examples:

```
random.getrandbits(4) # Example 1: 11 (binary 1011)
random.getrandbits(8) # Example 2: 243
random.getrandbits(1) # Example 3: 0 or 1
```