ASSIGNMENT NO 3

Random Module: Python's random module helps generate random numbers for simulations, games, and testing.

```
import random as rnd

print("Random float between 0 and 1:", rnd.random())

print("Random integer between 1 and 50:", rnd.randint(1, 50))

fruits = ["Apple", "Banana", "Mango", "Orange", "Grapes"]
print("Randomly selected fruit:", rnd.choice(fruits))

print("3 Random fruits for a game bonus:", rnd.sample(fruits, 3))
```

```
Random float between 0 and 1: 0.9945016280508552
Random integer between 1 and 50: 29
Randomly selected fruit: Grapes
3 Random fruits for a game bonus: ['Banana', 'Mango', 'Apple']
```

True vs. Pseudorandom: Computers use pseudorandom numbers that appear random but are generated by algorithms.

```
import random as rnd

rnd.seed(10)
print("First Run with seed = 10")
print("Random numbers:", rnd.random(), rnd.randint(1, 50), rnd.random())

rnd.seed(10)
print("\nSecond Run with same seed = 10")
print("Random numbers:", rnd.random(), rnd.randint(1, 50), rnd.random())

rnd.seed()
print("\nThird Run with system time (True Random-like)")
print("Random numbers:", rnd.random(), rnd.randint(1, 50), rnd.random())
```

```
First Run with seed = 10
Random numbers: 0.5714025946899135 28 0.48256167455085586

Second Run with same seed = 10
Random numbers: 0.5714025946899135 28 0.48256167455085586

Third Run with system time (True Random-like)
Random numbers: 0.7089126332197854 15 0.8590662006978449
```

Uniformity & Unpredictability: Random numbers should be evenly distributed and hard to predict.

```
import random as rnd

numbers = [rnd.randint(1, 10) for _ in range(20)]

print("Generated random numbers (1 to 10):")
print(numbers)

count_dict = {}
for n in numbers:
    count_dict[n] = count_dict.get(n, 0) + 1

print("\nFrequency of each number:")
for key, value in sorted(count_dict.items()):
    print(f"Number {key}: {value} times")
```

```
Generated random numbers (1 to 10):
[8, 5, 3, 1, 9, 8, 6, 2, 4, 6, 1, 7, 3, 10, 6, 7, 7, 5, 5, 8]

Frequency of each number:
Number 1: 2 times
Number 2: 1 times
Number 3: 2 times
Number 4: 1 times
Number 5: 3 times
Number 6: 3 times
Number 7: 3 times
Number 8: 3 times
Number 9: 1 times
Number 10: 1 times
```

Random Functions: Use random () for floats, randrange () or randint() for integers, and choice () to pick from a list.

```
import random as rnd

print("Random float between 0 and 1:", rnd.random())

print("Random integer between 5 and 15:", rnd.randint(5, 15))

print("Random even number between 2 and 20:", rnd.randrange(2, 21, 2))

colors = ["Red", "Blue", "Green", "Yellow", "Purple", "Orange"]
print("Randomly selected color:", rnd.choice(colors))
```

```
Random float between 0 and 1: 0.48256167455085586
Random integer between 5 and 15: 5
Random even number between 2 and 20: 8
Randomly selected color: Yellow
```

Seeding: seed () sets the RNG starting point for reproducibility—useful in simulations or testing.

```
import random as rnd
```

```
rnd.seed(25)
print("First Run with seed = 25")
print("Random numbers:", rnd.randint(1, 50), rnd.randint(1, 50), rnd.randint(1, 50))

rnd.seed(25)
print("\nSecond Run with same seed = 25")
print("Random numbers:", rnd.randint(1, 50), rnd.randint(1, 50), rnd.randint(1, 50))

rnd.seed(10)
print("\nThird Run with different seed = 10")
print("Random numbers:", rnd.randint(1, 50), rnd.randint(1, 50), rnd.randint(1, 50))
```

```
First Run with seed = 25
Random numbers: 25 50 1

Second Run with same seed = 25
Random numbers: 25 50 1

Third Run with different seed = 10
Random numbers: 37 3 28
```

Random Use Cases: Common in gaming, cryptography, and modeling real-world randomness.

```
import random as rnd
import secrets as sec

# Gaming Example – Rolling a dice
dice_roll = rnd.randint(1, 6)
print("Gaming Example - Dice Roll:", dice_roll)

# Cryptography Example – Secure random token
secure_token = sec.token_hex(8)
print("Cryptography Example - Secure Token:", secure_token)

# Simulation Example – Modeling daily temperature variation
temperatures = [round(rnd.uniform(25.0, 40.0), 2) for _ in range(7)]
print("Simulation Example - Weekly Temperatures (°C):", temperatures)
```

```
Gaming Example - Dice Roll: 3
Cryptography Example - Secure Token: 37e91f04a017a292
Simulation Example - Weekly Temperatures (°C): [28.88, 30.82, 29.96, 33.56, 25.47, 36.34, 29
```

Caution: Python's RNG isn't secure for sensitive apps like banking—use cryptographic libraries instead.

```
import random as rnd
import secrets as sec

insecure_pin = rnd.randint(1000, 9999)
print("Insecure PIN (not for banking):", insecure_pin)

secure_pin = sec.randbelow(10000)
print("Secure PIN (safe for security use):", secure_pin)
```

```
password = sec.token_hex(8)
print("Secure Random Password:", password)
```

```
Insecure PIN (not for banking): 2678
Secure PIN (safe for security use): 4393
Secure Random Password: 6cab17d3e2952789
```