# RANDOM FUNCTIONS

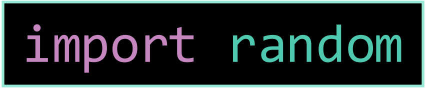
## PYTHON RANDOM MODULE

Jaume Boguñá

**Dive into Python**

# Random module in Python

- Provides a suite of **functions** to generate **random numbers**.



```
import random
```

- Includes tools to manipulate **sequences randomly.**

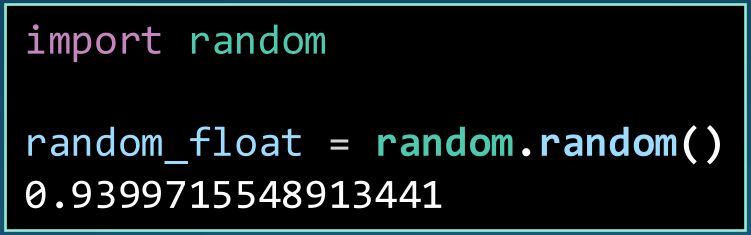
- Used in **simulations**, **games**, **data analysis**, and more.

# Functions for Floats

## 1. random()

**Purpose:**

Returns a **random floating**-point number **between 0.0** and **1.0**.



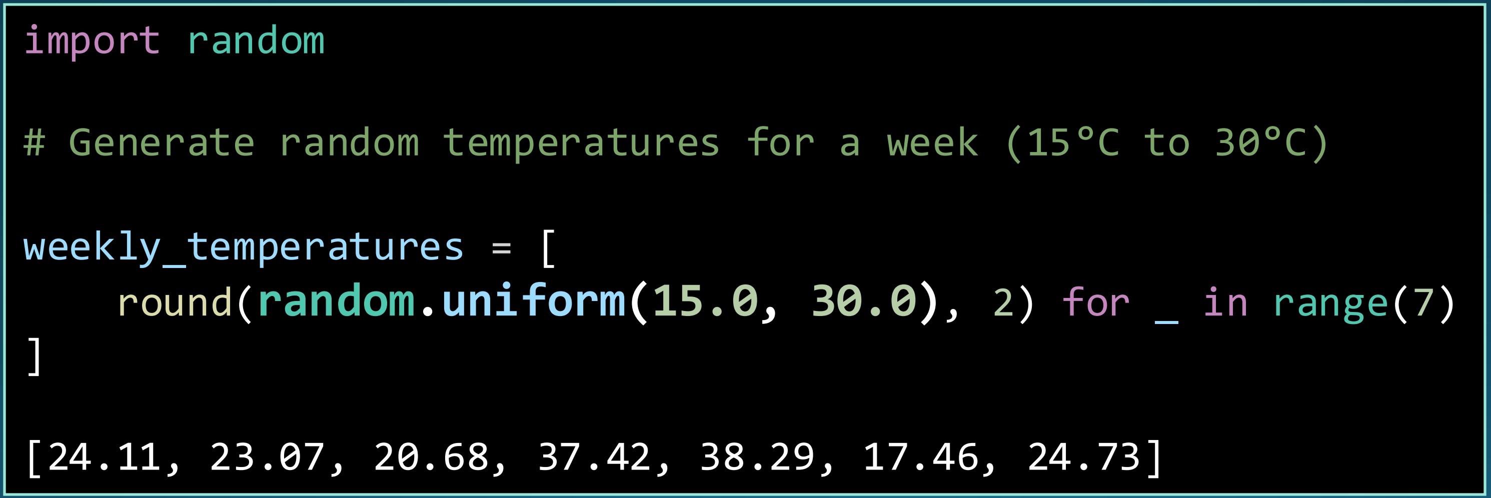
```
import random

random_float = random.random()
0.9399715548913441
```

# Functions for Floats

## 2. uniform(*a*, *b*)

**Purpose:**

Returns a **random floating**-point number **between a** and **b**.



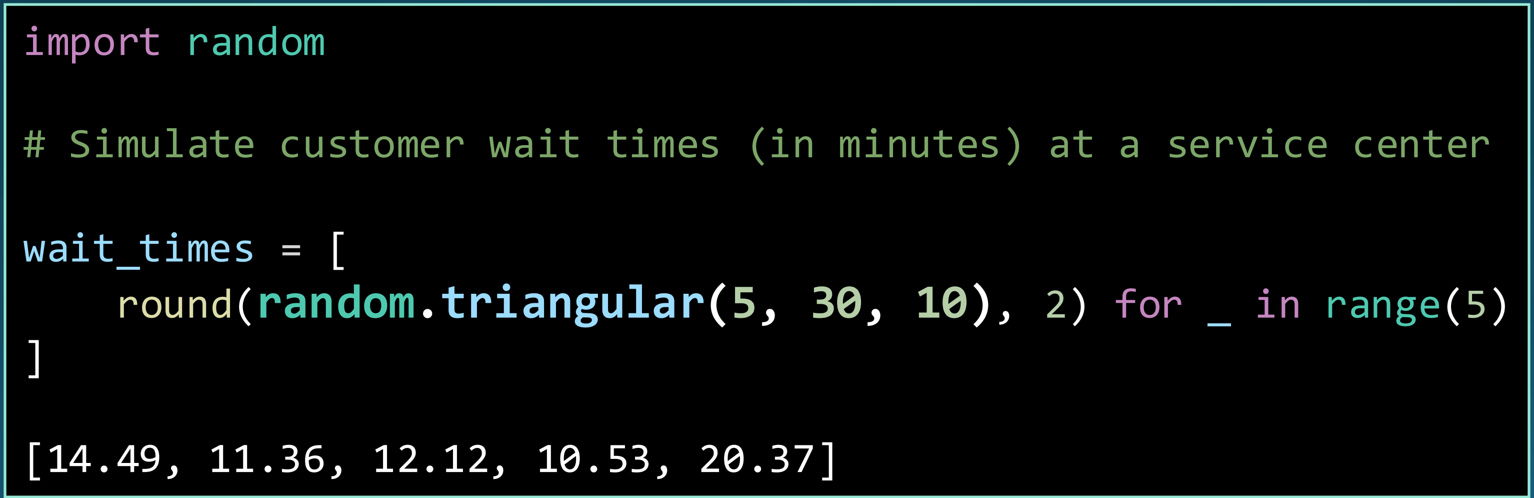
```
import random

# Generate random temperatures for a week (15°C to 30°C)

weekly_temperatures = [
    round(random.uniform(15.0, 30.0), 2) for _ in range(7)
]

[24.11, 23.07, 20.68, 37.42, 38.29, 17.46, 24.73]
```

# Functions for Floats

## 3. triangular(*low, high, mode*)

**Purpose:**

Returns a **random floating-point** number **between low** and **high**, with an **optional mode** parameter for a **non-uniform distribution**.



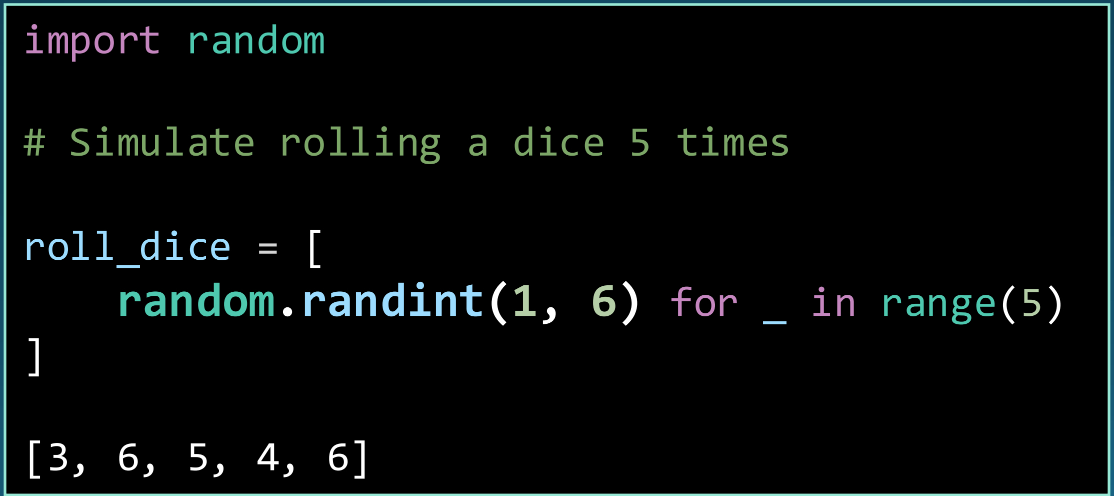
```
import random

# Simulate customer wait times (in minutes) at a service center

wait_times = [
    round(random.triangular(5, 30, 10), 2) for _ in range(5)
]

[14.49, 11.36, 12.12, 10.53, 20.37]
```

# Functions for Integers

## 1. `randint(a, b)`

**Purpose:**

Returns a **random integer** between **two** specified **integers**, inclusive.



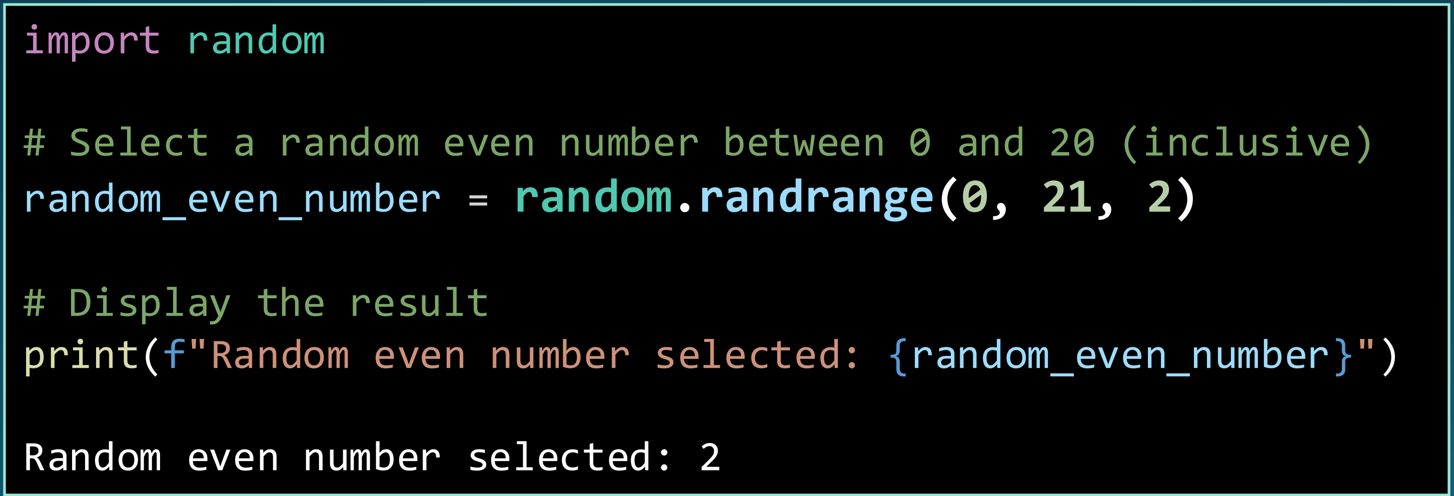
```
import random

# Simulate rolling a dice 5 times

roll_dice = [
    random.randint(1, 6) for _ in range(5)
]

[3, 6, 5, 4, 6]
```

# Functions for Integers

## 2. randrange(*start*, *stop*, *step*)

**Purpose:**

Selects a **random element** from the specified **range**.



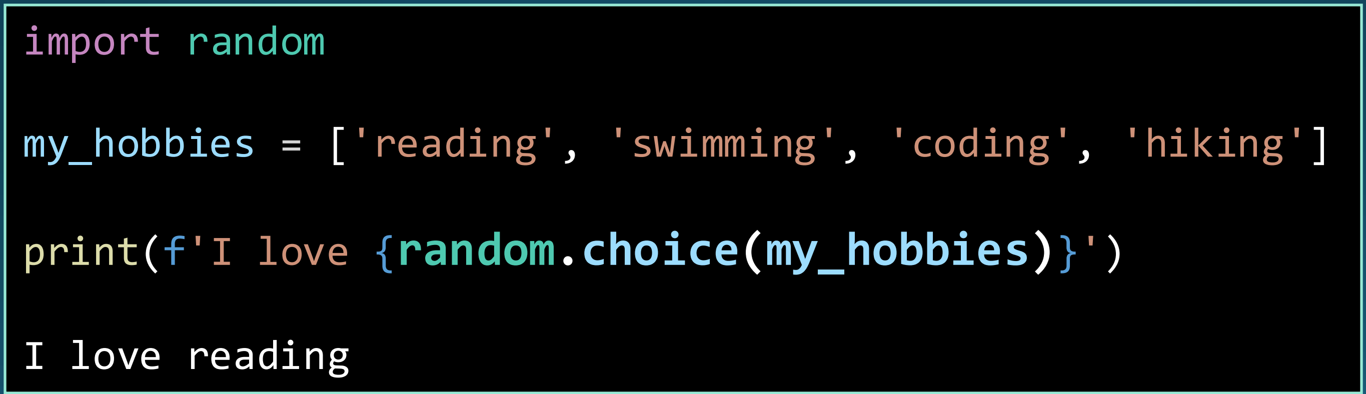
```
import random

# Select a random even number between 0 and 20 (inclusive)
random_even_number = random.randrange(0, 21, 2)

# Display the result
print(f"Random even number selected: {random_even_number}")

Random even number selected: 2
```

# Functions for Sequences

## 1. choice(*seq*)

**Purpose:**

Returns a **randomly** selected **element** from a non-empty **sequence**.



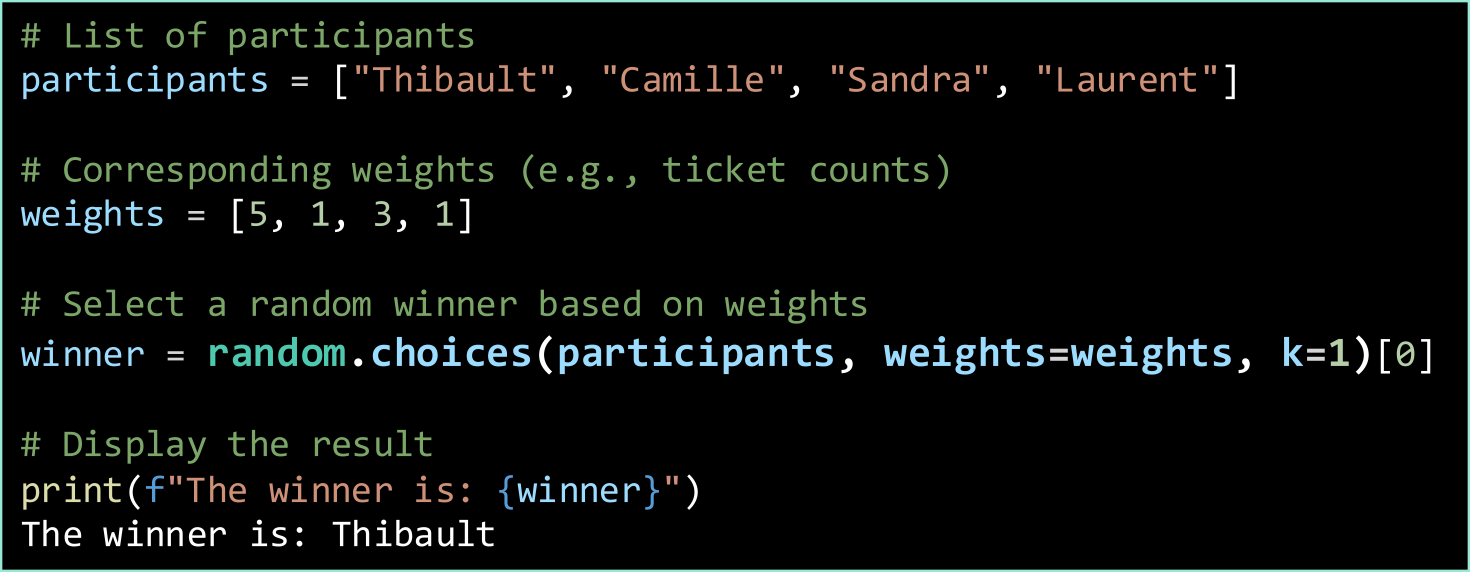
```
import random

my_hobbies = ['reading', 'swimming', 'coding', 'hiking']

print(f'I love {random.choice(my_hobbies)}')

I love reading
```

# Functions for Sequences

## 2. choices(*seq, weights=None, k=1*)

**Purpose:**

Returns a list with **k elements** from the **sequence**, with **optional weights** to influence selection **probability**.



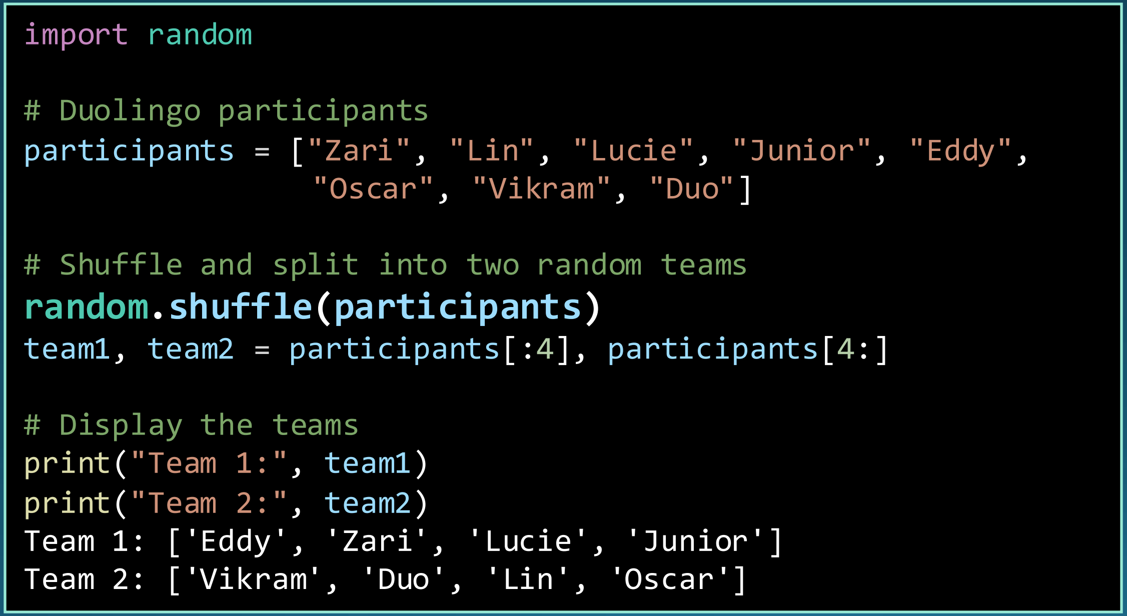
```
# List of participants
participants = ["Thibault", "Camille", "Sandra", "Laurent"]

# Corresponding weights (e.g., ticket counts)
weights = [5, 1, 3, 1]

# Select a random winner based on weights
winner = random.choices(participants, weights=weights, k=1)[0]

# Display the result
print(f"The winner is: {winner}")
The winner is: Thibault
```

# Functions for Sequences

## 3. shuffle(*seq, random=None*)

**Purpose:**

**Shuffles** the **sequence** x in place.



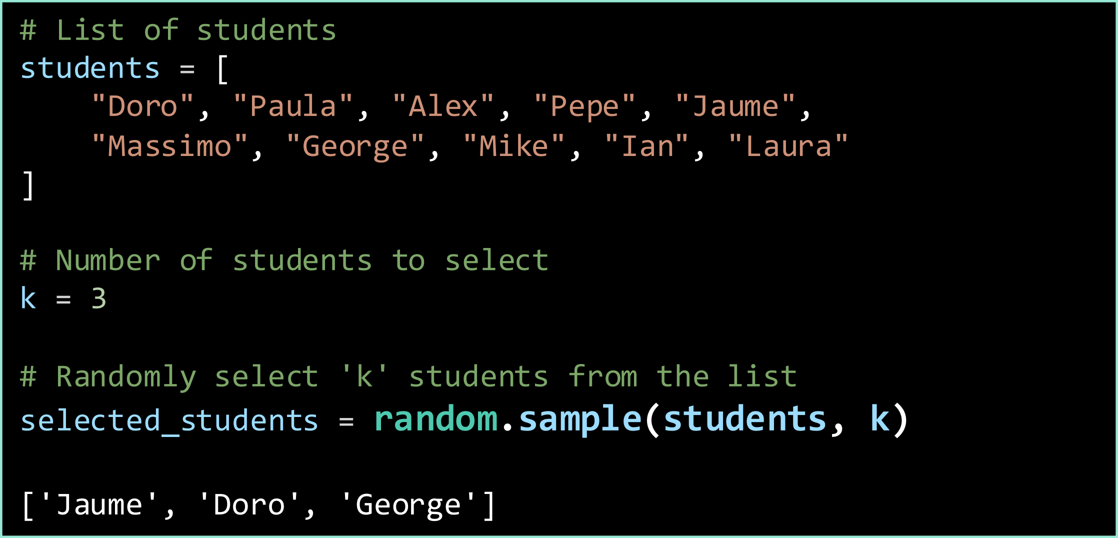
```
import random

# Duolingo participants
participants = ["Zari", "Lin", "Lucie", "Junior", "Eddy",
                "Oscar", "Vikram", "Duo"]

# Shuffle and split into two random teams
random.shuffle(participants)
team1, team2 = participants[:4], participants[4:]

# Display the teams
print("Team 1:", team1)
print("Team 2:", team2)
Team 1: ['Eddy', 'Zari', 'Lucie', 'Junior']
Team 2: ['Vikram', 'Duo', 'Lin', 'Oscar']
```

# Functions for Sequences

## 4. sample(*seq, k, counts=None*)

**Purpose:**

Returns a specified number of **unique elements** chosen from the **sequence** (without replacement).



```
# List of students
students = [
    "Doro", "Paula", "Alex", "Pepe", "Jaume",
    "Massimo", "George", "Mike", "Ian", "Laura"
]

# Number of students to select
k = 3

# Randomly select 'k' students from the list
selected_students = random.sample(students, k)

['Jaume', 'Doro', 'George']
```

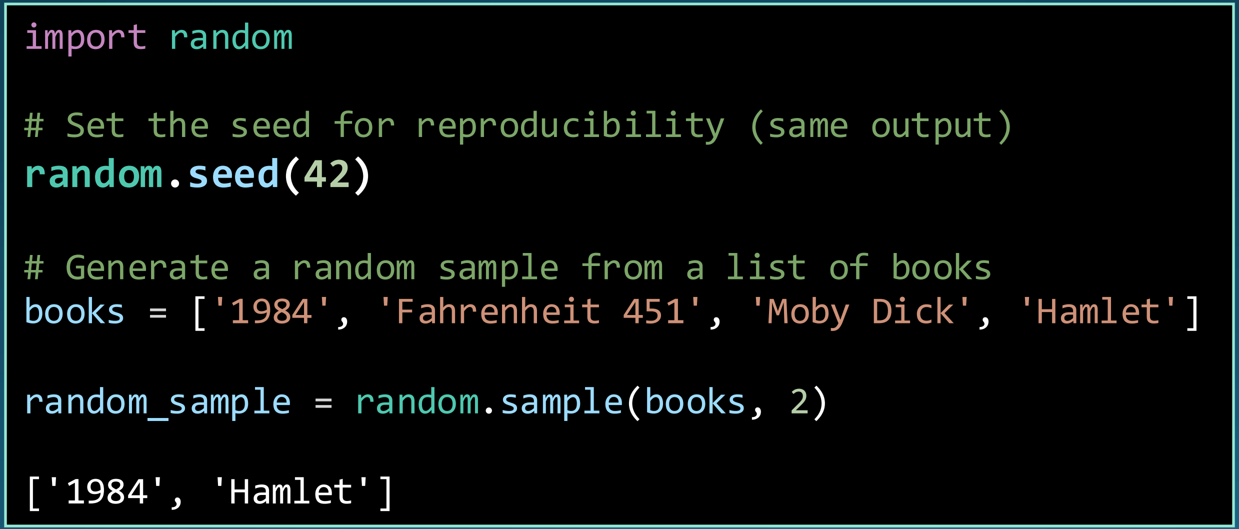
# Bookkeeping Function

## 1. seed()

**Purpose:**

Initializes the **random number generator** for **predictable** sequences.

Ensures **reproducibility**.



```python
import random

# Set the seed for reproducibility (same output)
random.seed(42)

# Generate a random sample from a list of books
books = ['1984', 'Fahrenheit 451', 'Moby Dick', 'Hamlet']

random_sample = random.sample(books, 2)

['1984', 'Hamlet']
```

# Summary

| | |
|---|---|
| `random()` | Returns a random float between 0.0 and 1.0. |
| `uniform(a, b)` | Returns a random float between a and b. |
| `triangular(low,high,mode)` | Random float between low and high with an optional mode. |
| `randint(a,b)` | Random integer between two specified integers, inclusive. |
| `randrange(start,stop,step)` | Selects a random element from the specified range. |
| `choice(seq)` | Returns a randomly selected element from a non-empty sequence. |
| `choices(seq,weights=None,k=1)` | Randomly selects k elements from a sequence, with optional weights. |
| `shuffle(seq,random=None)` | Shuffles the sequence x in place. |
| `sample(seq,k,counts=None)` | Specified number of unique elements chosen from the sequence |
| `seed()` | Initializes the random number generator for predictable sequences. |

Like

Comment

Share

Jaume Boguñá

Aerospace Engineer | Data Scientist