10. Random Module

Importing the Random Module

Before we can use the functions in the `random` module, we need to import it into our Python script. This can be done using the following line of code:

```
import random
```

Alternatively, you can import specific functions from the `random` module using the following syntax:

```
from random import randint, choice
```

This imports the `randint` and `choice` functions from the `random` module, allowing you to use them directly without having to prefix them with `random.`.

Generating Random Numbers

One of the most common use cases of the `random` module is generating random numbers. The `random` module provides several functions for this purpose.

`random()`

The `random()` function generates a random floating-point number between 0 and 1. Here's an example:

```
import random

# Generate a random float between 0 and 1
random_float = random.random()
print(random_float)
```

This will output a random float, such as `0.7183492537948495`.

`randrange()`

The `randrange()` function generates a random integer within a specified range. It takes two or three arguments:

- `start` (optional): The start of the range (inclusive). If omitted, the default value is 0.
- `stop`: The end of the range (exclusive).
- `step` (optional): The step size between numbers in the range. If omitted, the default value is 1.

Here are some examples:

```
import random

# Generate a random integer between 0 and 10
random_int = random.randrange(10)
print(random_int)

# Generate a random even integer between 2 and 10
random_even = random.randrange(2, 10, 2)
print(random_even)

# Generate a random integer between -5 and 5
random_negative_positive = random.randrange(-5, 5)
print(random_negative_positive)
```

`randint()`

The `randint()` function is similar to `randrange()`, but it generates a random integer within a closed interval (both ends are inclusive). It takes two arguments:

- `a`: The lower bound of the interval (inclusive).
- `b`: The upper bound of the interval (inclusive).

Here's an example:

```
import random

# Generate a random integer between 1 and 6 (inclusive)
random_dice_roll = random.randint(1, 6)
print(random_dice_roll)
```

Making Random Choices

The `random` module also provides functions for making random choices from sequences (such as lists or strings).

`choice()`

The `choice()` function returns a random element from a non-empty sequence. Here's an example:

```python
import random

# Define a list of fruits
fruits = ['apple', 'banana', 'cherry', 'date', 'elderberry']

# Choose a random fruit from the list
random_fruit = random.choice(fruits)
print(random_fruit)
```

This will output a random fruit from the `fruits` list, such as `'banana'`.

`choices()`

The `choices()` function returns a list of random elements from a non-empty sequence. It takes two arguments:

- `population`: The sequence to choose elements from.
- `weights` (optional): A sequence of relative weights for each element in the `population` sequence.
- `k` (optional): The number of elements to choose. If not provided, a single element is returned.

Here's an example:

```python
import random

# Define a list of colors
colors = ['red', 'green', 'blue', 'yellow', 'purple']

# Choose 3 random colors from the list
random_colors = random.choices(colors, k=3)
print(random_colors)

# Define weights for each color
```

```
weights = [0.1, 0.3, 0.2, 0.2, 0.2]

# Choose a random color, considering the weights
weighted_color = random.choices(colors, weights=weights)
print(weighted_color)
```

In this example, `random_colors` will contain a list of three randomly chosen colors from the `colors` list, while `weighted_color` will contain a list with one randomly chosen color, where the probability of each color is determined by the `weights` list.

`sample()`

The `sample()` function returns a list of unique random elements from a sequence. It takes two arguments:

- `population`: The sequence to choose elements from.
- `k`: The number of elements to choose.

Here's an example:

```
import random

# Define a list of numbers
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Choose 5 unique random numbers from the list
random_numbers = random.sample(numbers, k=5)
print(random_numbers)
```

This will output a list of 5 unique random numbers from the `numbers` list, such as `[8, 4, 2, 1, 9]`.

Seeding the Random Number Generator

By default, the `random` module uses the current system time as a seed for generating random numbers. However, you can set your own seed using the `seed()` function. This is useful for testing and debugging purposes, as it ensures that the same sequence of random numbers is generated every time the program is run.

```
import random

# Set the seed
```

```python
random.seed(42)

# Generate random numbers
print(random.random())    # Output: 0.6767492789859154
print(random.randint(1, 10))  # Output: 6
```

If you run the code above multiple times, you will get the same random numbers each time because the seed is fixed.

See Also
11. Loops