Das random-Modul in Python

Definition:
Das random-Modul gehört zur Standardbibliothek von Python und ermöglicht die Erzeugung von Pseudo-Zufallszahlen. Es wird verwendet, um Zufallssimulationen, Zahlengenerierung und zufällige Auswahl von Elementen umzusetzen.

Wenn man Python installiert, werden automatisch etwa 200 Module wie math, string, random, datetime oder os mitgeliefert. Diese gehören zur sogenannten Standardbibliothek und können direkt verwendet werden – ganz ohne zusätzliche Installation oder Download.

Zweck:
Das random-Modul unterstützt bei:

- Erzeugung von Zufallszahlen (z. B. random(), randint(), uniform())
- Steuerung der Zufälligkeit mit seed()
- Auswahl zufälliger Elemente (z. B. choice(), sample())
- Zufälliges Mischen von Listen (z. B. shuffle())
- Simulationen (z. B. Würfelwurf, Zufallslisten, kleine Spiele)

Hinweis: Die erzeugten Zufallszahlen sind nicht wirklich zufällig, sondern pseudo-zufällig – sie basieren auf einem Algorithmus und können mit seed() reproduzierbar gemacht werden.

Das random-Modul in Python

Wichtige Funktionen aus dem Modul random:

```
import random

print(random.random())      # Zufallszahl zwischen 0.0 und 1.0, (0.0 inklusive, 1.0 exklusive)

print(random.randint(1, 6)) # Ganze Zufallszahl zwischen 1 und 6, ganzzahlig, inkl. 1 und 6

print(random.uniform(1.5, 3.5)) # Zufallszahl mit Komma (float), inkl. 1.5 und 3.5

print(random.choice(["rot", "grün", "blau"]))  # Zufälliges Element , z. B. "blau"
```

Das random-Modul in Python

seed() und Wiederholbarkeit:

Python verwendet im random-Modul einen Pseudo-Zufallszahlengenerator. Das bedeutet:

- Die Zahlen sehen zufällig aus,
- aber sie werden durch einen Algorithmus erzeugt
- … basierend auf einem Startwert: dem sogenannten Seed.

Wenn du den gleichen Seed vorgibst, bekommst du immer die gleichen Zufallszahlen zurück – in derselben Reihenfolge!

BeispielCode

```
import random

random.seed(42)                 # Schritt 1: Setzt den Startwert (Seed)
print(random.randint(1, 100))   # Schritt 2: Gibt eine Zufallszahl zwischen 1 und 100 aus
                                # Ausgabe: z. B. 82(Das ist immer gleich, solange der Seed 42 ist.)

random.seed(42)                 # Schritt 3: Wieder der gleiche Startwert
print(random.randint(1, 100))   # Schritt 4: Gleiche Zahl wie zuvor  # Ausgabe: wieder 82
```

Das random-Modul in Python

seed() und Wiederholbarkeit:
Wenn du keinen seed() setzt, dann verwendet Python beim Starten deines Programms einen zufälligen Startwert (**einen anderen Seed**), der in der Regel auf der aktuellen Uhrzeit basiert (z. B. Systemzeit in Nanosekunden), automatisch. Dadurch ist der „Zufall“ wirklich nicht vorhersagbar – du bekommst bei jedem Ausführen andere Ergebnisse. Beispiel: Ein Würfelspiel wäre langweilig, wenn immer dieselbe Zahl kommt !

BeispielCode ohne seed()

```
import random

print(random.randint(1, 100))  # z. B. 27
                               # Beim nächsten Ausführen: vielleicht 17, 59, 3 usw.
                               # Die Ergebnisse ändern sich jedes Mal.
```

BeispielCode mit seed()

```
random.seed(42)
print(random.randint(1, 100))  # Immer: 82
                               # Bei jedem Ausführen bleibt das Ergebnis gleich.
```

Das random-Modul in Python

seed() und Wiederholbarkeit: Warum ist das wichtig?

Vergleichbarkeit sicherstellen: Mit seed() erzeugt Python jedes Mal die gleiche Folge an Zufallszahlen – das ist entscheidend, um Ergebnisse reproduzieren und vergleichen zu können.

Automatisierte Tests: Konstante Zufallswerte ermöglichen das Schreiben von zuverlässigen Tests, erleichtern das Debugging und sorgen für nachvollziehbare Codekontrolle.

Fehlersuche & Analyse: Wenn ein Programm unerwartet reagiert, lässt sich das Verhalten mit reproduzierbaren Zufallswerten gezielt rekonstruieren und analysieren.

Simulation & Wissenschaft: Reproduzierbare Ergebnisse sind zentral für Experimente und Analysen – gezielt gesteuerter Zufall macht das möglich.

Das random-Modul in Python

Weitere nützliche Funktionen im Modul random : sample()

```
import random

liste = [10, 20, 30, 40]

zufallsauswahl = random.sample(liste, 2)      # 2 zufällige, verschiedene Elemente aus der Liste
print(zufallsauswahl)                         # z. B. [20, 10]


                                              # Gibt k verschiedene Elemente aus der Liste zurück
                                              # Die Original-Liste bleibt unverändert
                                              # Ergebnis ist eine neue Liste mit zufälliger Auswahl
                                              # Keine Wiederholung möglich – jedes Element ist einzigartig
```

Das random-Modul in Python

Weitere nützliche Funktionen im Modul random: shuffle()

```
import random

zahlen = [10, 20, 30, 40]

random.shuffle(zahlen)

                                # Mischt die Reihenfolge der Elemente in der übergebenen Liste
                                # die Original-Liste wird direkt verändert

print(zahlen)                   # z. B. [30, 10, 40, 20]

print(random.shuffle(zahlen))  # ergibt None, da shuffle() nichts zurückgibt!
```

Das random-Modul in Python

Überblick: Meistbenutzte Funktionen aus dem Modul random

Funktion	Beschreibung	Grenzen: inkl./exkl.	Bemerkung
random.random()	Zufallszahl(float) zwischen 0.0 und 1.0	0.0 inkl., 1.0 exkl.	Ideal für Wahrscheinlichkeiten
random.randint(a, b)	Ganze Zahl (int) zwischen a und b	beide inkl.	Würfel, Zufallsindex
random.uniform(a, b)	Kommazahl (float) zwischen a und b	beide inkl.	Für float-Werte mit Bereich
random.choice(liste)	Ein zufälliges Element aus der Liste	–	Ein Wert. Bei mehreren Aufrufen kann derselbe Wert mehrfach erscheinen.
random.sample(liste, k)	k zufällige verschiedene Elemente aus der Liste	–	Gibt neue Liste zurück. Original-Liste bleibt unverändert.
random.shuffle(liste)	Mischt Reihenfolge der Listenelemente	–	Verändert die Original-Liste direkt. Nur für veränderbare Sequenzen (z. B. Listen). Gibt None zurück.
random.seed(x)	Initialisiert den Zufallsgenerator.	–	Macht Zufallswerte reproduzierbar – bei gleichem Seed immer gleiche Ergebnisse

Das random-Modul in Python

Zufall ist nicht gleich Zufall: random vs. secrets

Wann ist random nicht genug?

Das Modul random erzeugt Pseudo-Zufallszahlen: Sie sehen zufällig aus, aber sind vorhersehbar, wenn man den Seed kennt. Für Spiele, Simulationen und Übungen ist das völlig ausreichend. Aber für Sicherheitszwecke (z. B. Passwörter, Tokens, Bestätigungs-Codes) reicht das nicht aus.

Lösung: secrets-Modul für kryptographisch sicheren Zufall

https://docs.python.org/3/library/secrets.html

https://www.geeksforgeeks.org/python/secrets-python-module-generate-secure-random-numbers/

Verwende random für Spiel und Spaß – aber secrets für Sicherheit und Passwörter.

Das random-Modul in Python

Mehr erkunden

https://docs.python.org/3/library/random.html

https://www.w3schools.com/python/module_random.asp

https://www.geeksforgeeks.org/python/python-random-module/