**Módulo 5.4**

**Manejo de datos aleatorios**

La biblioteca random es una biblioteca estándar de Python que proporciona funciones para generar números aleatorios. Es útil en una variedad de aplicaciones, como la simulación, el cifrado y la generación de contraseñas.

****random()**:**

El número aleatorio generado está distribuido uniformemente en el rango.

|  |
| --- |
| **Nota:**  El inicio se incluye siempre en el rango.  El fin depende del método si se incluye o excluye del rango |

****random.****

* ﻿﻿**random():**

**Este método devuelve un número aleatorio de punto flotante en el rango (0,0,1,0). **fin** se excluye**

* ****uniform(inicio, fin)****:

**Este método devuelve un número aleatorio de punto flotante en el rango específico (inicio, fin). inicio, fin se incluyen en el rango.**

* ****r**andint(inicio, fin):**

Este método devuelve un número entero aleatorio en el rango(inicio, fin). **inicio, fin** se incluyen en el rango.

* **randrange(inicio, fin,step):**

Este método devuelve un número entero aleatorio en el rango(inicio, fin, step o paso). ****fin** se excluye**

* ****choice(seq)****:

Este método devuelve un elemento aleatorio de una secuencia seq. La secuencia puede ser una lista, tupla o string.

* ****shuffle(seq)****:

Este método mezcla los elementos de una secuencia seq en orden aleatorio. La secuencia puede ser una lista

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | # salida flotante  import random  for x in range(5):  print (f"Numero aleatorio entre 0 y 1 en {x} :{random.random()=}") |   ·Salida esperada por consola   |  | | --- | | Numero aleatorio entre 0 y 1 en 0 :random.random()=**0.047085427130461044**  Numero aleatorio entre 0 y 1 en 1 :random.random()=**0.11902979805180813**  Numero aleatorio entre 0 y 1 en 2 :random.random()=**0.5717714253197419**  Numero aleatorio entre 0 y 1 en 3 :random.random()=**0.838727677943541**  Numero aleatorio entre 0 y 1 en 4 :random.random()=**0.14433511978974312** |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | # salida flotante  inicio = 10.0  fin = 50.0  for x in range(5):  print (f"Numero aleatorio entre {inicio} y {fin} en {x} :{random.uniform(inicio,fin)=}") |   ·Salida esperada por consola   |  | | --- | | Numero aleatorio entre 10.0 y 50.0 en 0 :random.uniform(inicio,fin)=**34.05440444969503**  Numero aleatorio entre 10.0 y 50.0 en 1 :random.uniform(inicio,fin)=**14.973531497915529**  Numero aleatorio entre 10.0 y 50.0 en 2 :random.uniform(inicio,fin)=**10.692426973915769**  Numero aleatorio entre 10.0 y 50.0 en 3 :random.uniform(inicio,fin)=**16.254231000595972**  Numero aleatorio entre 10.0 y 50.0 en 4 :random.uniform(inicio,fin)=**49.49714573075685** |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | # salida entero  inicio = 1  fin = 1000  for x in range(5):  print (f"Numero aleatorio entre {inicio} y {fin} en {x} :{random.randint(inicio,fin)=}") |   ·Salida esperada por consola   |  | | --- | | Numero aleatorio entre 1 y 1000 en 0 :random.randint(inicio,fin)**=339**  Numero aleatorio entre 1 y 1000 en 1 :random.randint(inicio,fin)**=615**  Numero aleatorio entre 1 y 1000 en 2 :random.randint(inicio,fin)=**757**  Numero aleatorio entre 1 y 1000 en 3 :random.randint(inicio,fin)=**130**  Numero aleatorio entre 1 y 1000 en 4 :random.randint(inicio,fin)=**352** |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | # salida entero  inicio = 1  fin = 1000  step = 5  for x in range(5):  print (f"Numero aleatorio entre {inicio} y {fin} con un paso de {step} en {x} :{random.randrange(inicio,fin,step)=}") |   ·Salida esperada por consola   |  | | --- | | Numero aleatorio entre 1 y 1000 con un paso de 5 en 4 :random.randrange(inicio,fin,step)**=601**  Numero aleatorio entre 1 y 1000 con un paso de 5 en 4 :random.randrange(inicio,fin,step)**=546**  Numero aleatorio entre 1 y 1000 con un paso de 5 en 4 :random.randrange(inicio,fin,step)**=551**  Numero aleatorio entre 1 y 1000 con un paso de 5 en 4 :random.randrange(inicio,fin,step)**=786**  Numero aleatorio entre 1 y 1000 con un paso de 5 en 4 :random.randrange(inicio,fin,step)**=866** |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | import random  lista = ['lunes','martes','miercoles','jueves','viernes','sabado','domingo']  print (f"lista original:{lista}")  print (f"random.choice(lista):{random.choice(lista)}")  random.shuffle(lista)  print (f"random.shuffle(lista):{lista}") |   ·Salida esperada por consola   |  | | --- | | lista original:['lunes', 'martes', 'miercoles', 'jueves', 'viernes', 'sabado', 'domingo']  random.choice(lista):**miercoles**  random.shuffle(lista):**['jueves','sabado','lunes', 'miercoles', 'domingo', 'viernes', 'martes']** |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | numeros = list(range(1, 7))  numero = random.choice(numeros)  print(f'Se ha seleccionado de forma aleatoria el valor:{numero}') |   ·Salida esperada por consola   |  | | --- | | Se ha seleccionado de forma aleatoria el valor:**6** |   · |

﻿**Manejos de grandes cantidades de datos - Bibliotecas Externas**

Hay muchas bibliotecas externas que se pueden utilizar para el manejo de datos en Python. Algunas de las más populares son:

* NumPy: una biblioteca para trabajar con matrices y vectores de gran tamaño y alta dimensionalidad, ideal para cálculos científicos y matemáticos.
* Pandas: una biblioteca de análisis de datos que proporciona estructuras de datos flexibles y herramientas de manipulación de datos para hacer análisis de datos más fácil y rápido.
* Matplotlib: una biblioteca de visualización de datos para crear gráficos en 2D y 3D en Python.
* SciPy: una biblioteca para matemáticas, ciencias e ingeniería que proporciona herramientas para la optimización, álgebra lineal, integración y otras tareas.
* Scikit-learn: una biblioteca para aprendizaje automático en Python que proporciona herramientas para la clasificación, regresión, clustering y otras tareas.
* TensorFlow: una biblioteca de aprendizaje automático de Google para construir y entrenar redes neuronales.
* Keras: una biblioteca de aprendizaje automático de alto nivel que se ejecuta en TensorFlow.
* PyTorch: una biblioteca de aprendizaje profundo de código abierto para Python, que se utiliza para desarrollar y entrenar modelos de aprendizaje profundo.

Estas son solo algunas de las bibliotecas de manejo de datos en Python más populares, pero hay muchas otras que también son útiles dependiendo de las necesidades específicas del proyecto.