



INGENIERIA EN COMPUTACION



Materia: Programación Python

Alumno: Ayon Dimas Andres

Matricula: 366845

Profesor: Pedro Nunes Yepiz

Actividad: #8

Tema: Listas, range y random

Ensenada B.C 4 de octubre del 2023

INTRODUCCION

Aquí podremos ver la manera en la que funcionan estas tres sentencias en Python, como lo son el range, random y lo principal de este tema que son las listas. Donde podremos ver para que nos sirven y a su vez en qué momento se pueden utilizar las listas de Python, y podremos ver que estas son más diferentes a C, ya que en Python no se requiere de mucha sintaxis a la hora de aplicar esta sentencia, lo cual facilita el flujo de programa, esto debido a que Python hace cosas de forma autónoma en base a lo que nosotros escribimos dentro del programa.

COMPETENCIA

Aprender a aplicar estas sentencias a mencionadas para su implementación dentro del programa en Python cuando es adecuado utilizarlas. Es de suma importancia dominar estas sentencias ya que son base fundamental de Python y nos serán útiles a la hora de querer guardar información, ya que esto nos permite hacerlo.

FUNDAMENTOS

RANGE

Range es un tipo de dato que representa una secuencia de números inmutable. Para crear un objeto de tipo range, se pueden usar dos constructores:

- `range(fin)`: Crea una secuencia numérica que va desde 0 hasta `fin - 1`.
- `range(inicio, fin, [paso])`: Crea una secuencia numérica que va desde `inicio` hasta `fin - 1`. Si además se indica el parámetro `paso`, la secuencia genera los números de `paso` en `paso`.

Veámoslo con un ejemplo:

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5, 10))
[5, 6, 7, 8, 9]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(5, -5, -2))
[5, 3, 1, -1, -3]
```

RANDOM

Este módulo implementa generadores de números **pseudoaleatorios** para varias distribuciones. Para los enteros, existe una selección uniforme dentro de un rango. Para las secuencias, existe una selección uniforme de un elemento aleatorio, una función para generar una permutación aleatoria de una lista in-situ y una función para el muestreo aleatorio sin reemplazo.

Para números reales, existen funciones para calcular distribuciones uniformes, normales (Gaussianas), log-normales, exponenciales negativas, gamma y beta. Para generar distribuciones angulares está disponible la distribución de von Mises.

Las funciones proporcionadas por este módulo en realidad son métodos enlazados a instancias ocultas a la clase `random.Random`. Puedes instanciar tus propias instancias de `Random` para obtener generadores que no compartan estado.

La clase `Random` puede ser también subclaseada si quieres usar un generador básico diferente para tu propio diseño: en este caso, invalida los métodos `random()`, `seed()`, `getstate()` y `setstate()`.

Opcionalmente, se puede substituir un método `getrandbits()` por un nuevo generador — esto permite a `randrange()` producir selecciones sobre un rango arbitrariamente amplio. El módulo `random` también proporciona la clase `SystemRandom`, la cual usa la función del sistema `os.urandom()` para generar números aleatorios a partir de fuentes proporcionadas por el sistema operativo.

LISTAS

Una lista en Python es un tipo de datos nativos construido dentro del lenguaje de programación Python. Estas listas son similares a matrices (o arrays) que se encuentran en otros lenguajes. Sin embargo, en Python se manejan como variables con muchos elementos. Aun así, las listas se consideran un tipo de datos para guardar colecciones de información.

Una lista en Python es:

1. Ordenada: esto quiere decir que los elementos dentro de ella están indexados y se accede a ellos a través de una locación indexada.
2. Editable: los elementos dentro de una lista pueden editarse, añadir nuevos o eliminar los que ya tiene.
3. Dinámica: las listas pueden contener diferentes tipos de datos y hasta de objetos. Esto significa que pueden soportar paquetes multidimensionales de datos, como un array o muchos objetos.

4. No única: esencialmente, esto quiere decir que la lista puede contener elementos duplicados sin que arroje un error.

PROCEDIMIENTO

Aquí tenemos un ejemplo de un programa en Python que incluye **listas, range y random**:

```
import random
```

```
# Definir el rango y la cantidad de números aleatorios a
```

```
generar inicio = 1 fin = 100 cantidad_numeros = 10
```

```
# Generar una lista de números aleatorios en el rango especificado
```

```
numeros_aleatorios = [random.randint(inicio, fin) for _ in range(cantidad_numeros)]
```

```
# Imprimir la lista de números aleatorios
```

```
print("Lista de números aleatorios:")
```

```
print(numeros_aleatorios)
```

```
# Calcular la suma de los números en la lista
```

```
suma = sum(numeros_aleatorios) print(f"Suma
```

```
de los números: {suma}")
```

```
# Calcular el valor promedio de los números en la lista
```

```
promedio = suma / cantidad_numeros print(f"Valor
```

```
promedio: {promedio}")
```

Encontrar el número máximo y mínimo en la lista

```
maximo = max(numeros_aleatorios) minimo =  
min(numeros_aleatorios) print(f"Número máximo:  
{maximo}") print(f"Número mínimo: {minimo}")
```

RESULTADOS Y CONCLUSIONES

En esta práctica en Python centrado en listas, la librería **random** y la función **range**, he adquirido una comprensión más profunda de cómo utilizar estas herramientas esenciales para la programación. Las listas son estructuras de datos fundamentales que permiten almacenar y manipular colecciones de elementos de manera eficiente. Al estudiar la librería **random**, además se pudo explorar cómo generar números aleatorios, lo que es valioso en numerosos contextos, desde juegos hasta simulaciones.

Además, la función **range** muy útil ya que sirve para crear secuencias numéricas y controlar bucles de manera más efectiva. En conjunto, estas habilidades me han proporcionado una base sólida en Python que será útil en una variedad de proyectos futuros. Al comprender cómo trabajar con listas y cómo utilizar la librería **random** y la función **range**, estoy mejor equipado para abordar desafíos de programación más complejos y realizar tareas comunes de manera más eficiente. Este informe de práctica ha sido un paso crucial en mi desarrollo como programador de Python.