Generación de números aleatorios con lenguaje Python

En el campo de la ingeniería muchas veces necesitamos números al azar ya sea dentro de un conjunto acotado o libremente. Son muy útiles para comprobar funciones matemáticas que hemos "creado" a partir de datos recabados ("dos puntos definen una linea recta, tres puntos definen un plano", etcétera). Veamos cómo lograr eso con **Python**.

Que exactamente números aleatorios no serán, como bien dijo el físico John Von Neumann: con métodos aritméticos esto no será posible, y los ordenadores pues funcionan es en base a esto.

El primer intento serio de generar números aleatorios con hardware recuerdo que sucedió en 1999 con los procesadores marca Intel Pentium 3 (mejor dicho *Pentium !!!* https://www.wired.com/1999/01/hardwiring-security/) el cual trajo un un chip especialmente dedicado para ello. Si bien dimos un paso adelante, esa empresa también dio un paso hacia atrás al incluir un número identificador único incrustado, una especie de serial pero digital con el cual se podía rastrear cada uno de estos procesadores. **Es mi tesis que el gobierno estadounidense, muy receloso en cuanto a técnicas de cifrado** (y los números aleatorios son pilar para ello) **ordenó *de seguro que dicho identificador estuviera relacionado de alguna manera con los números "aleatorios" generados*.**

Con esta introducción lo que quiero dejar claro es que en realidad, sin entrar en el campo de la física y la mecánica cuántica, lo que generamos son **números pseudoaleatorios** *(o seudoaleatorios) es decir, números al zar bien creados artificialmente pero no exactamente aleatorios.*

Situación actual desde el año 2017

Captura de pantalla de la página web de la empresa Intel que explica el funcionamiento del chip generador de números aleatorios *derechos totalmente reservados de Intel, imagen presentada con propósito noticioso y educativo.* https://software.intel.com/content/www/us/en/develop/articles/intel-digital-random-number-generator-drng-software-implementation-guide.html

En la imagen anterior, con derechos totalmente reservados a la empresa Intel, agrego una captura de pantalla de la página web donde explica cómo funciona el hardware generador de números aleatorios que puede ser utilizado junto con el lenguaje C (incluye ejemplos de código) para aplicaciones de alta seguridad como cifrado y transacciones bancarias. Es bueno saber que ya contamos con dicha tecnología al día de hoy pero para implementarla se deben de seguir muchos pasos muy rigurosos.

Pueden encontrar mayor información en este enlace: https://www.welivesecurity.com/la-es/2014/05/23/importancia-generar-numeros-aleatorios-seguridad-informatica/

En nuestro día a día necesitamos números pseudoaleatorios para comprobar valores que pudiera introducir un usuario o usuaria, para identificar ciertas transacciones, etc.

Función *gauss()*

Comienzo con la función que más usamos en ingeniería, la distribución gaussiana, la cual podemos llamar con desviación estándar de 1 y media 0 (primero debemos llamar a la librería *random*):

```
[cc lang="python" width="90%" escaped="true" theme="blackboard"]
import random
for _ in range(10):
print((random.gauss(0,1)))
[/cc]
```

Si deseamos representar gráficamente dichos valores, les invito a leer en este enlace un artículo que explica cómo hacerlo con valores de temperatura de un ordenador (*monitorización*) y que puede ser adaptado para esta tarea gaussiana.

Función *shuffle()*

Exactamente no es para generar un número pseudoaleatorio sino para que con una lista dada - que podemos generar con la función anterior *gauss()* - cambiar su orden. Utilizo una lista de ocho elementos numerados de cero a siete para ilustrar:

[cc lang="python" width="90%" escaped="true" theme="blackboard"]
import random
orden = [0, 1, 2, 3, 4, 5, 6 , 7]
print(random.shuflle(orden))
[/cc]

Lenguaje Python librería random función shuffle()

¿Qué utilidad podríamos darle a la función *shuffle()*? Se me ocurre que tal vez si tenemos varios servidores web, numerarlos y realizar un balance de carga (repartir las solicitudes web) cada hora, por ejemplo.

También podemos usar texto como elemento de la lista, pero esa mezcla de números y letras generalmente no se estila; creo mucho en la homogeneidad de los datos para no estar creando rutinas de excepción de manera frecuente.

Función *choice()*

En realidad son dos funciones que las veo como singular y plural, y las considero como una sola. Tal como vimos con la función anterior, también trabajamos con una lista pero podemos escoger de manera pseudoaleatoria un elemento de la lista con *choice()* o dos o más elementos con *choices()*. También funciona con cadenas de texto.

[cc lang="python" width="90%" escaped="true" theme="blackboard"]
import random
orden = [0, 1, 2, 3, 4, 5, 6 , 7]
print(random.choice(orden))
print(random.choice(orden))
print(random.choices(orden,k=3))
[/cc]

Nótese que si se escoge dos o más, la función puede devolver elementos repetidos. Volviendo al ejemplo de nuestro enjambre de servidores web, tal vez alguno de ellos tiene mayores recursos de hardware (mayor cantidad de memoria RAM, procesador más rápido y/o con más núcleos, etc.) dado ese panorama especial podremos instruir a la función *choices()* con el parámetro *weights*. Para eso debemos especificar una *lista de pesos* de los elementos, pongamos que el servidor 7 tiene el doble de potencia:

```
print(random.choices(orden, weights=[1,1,1,1,1,1,1,2], k=3))
```

Función *randint()*

Esta función es mucho más sencilla, le especificamos el rango de valores para el número deseado y devuelve uno aleatorio, incluso puede ser uno de los dos valores acotados. Si queremos un número entre 7 y 700:

```
print(random.randint(7, 700))
```

Función *randrange()*

De manera muy parecida a la anterior, pero podemos especificar que sea divisible en un valor específico, por ejemplo que sea divisible entre 7:

```
print(random.randrange(7, 700, 7))
```