Material complementario
El módulo random de Python

Fundamentos de Programación

Febrero 2019

El módulo random de Python
Descripción general

- El módulo *random* contiene un conjunto de funciones que permiten generar secuencias de números pseudo-aleatorios de acuerdo a determinadas distribuciones probabilísticas.
- Una secuencia pseudo-aleatoria no es aleatoria (sólo lo parece).
- La pseudo-aleatoriedad es una propiedad estadística que normalmente se refiere a que una distribución de números así generados es indistinguible de una distribución uniforme y a que, dado un número de la secuencia, el siguiente es impredecible.
- Esta propiedad es suficiente en muchas aplicaciones prácticas, pero no en otras (como la protección criptográfica de datos) que suelen estar expuestas a ataques muy sofisticados y requieren, por tanto, números realmente aleatorios, basados en medidas precisas de procesos físicos no deterministas (ruido térmico, ruido barométrico, efecto fotoeléctrico, etc.).

El módulo random de Python
Descripción general

- La generación de números pseudo-aleatorios parte, en todo caso, de una inicialización aleatoria (o casi aleatoria) y utiliza un algoritmo determinista para generar el resto de la secuencia.
- Sin inicialización aleatoria, el algoritmo siempre generaría la misma secuencia de números (que seguiría cumpliendo las condiciones estadísticas de pseudo-aleatoriedad).
- La inicialización permite que distintas ejecuciones del código generen distintas secuencias de valores.
- Podríamos utilizar como semilla de inicialización los microsegundos de la hora, pero la cantidad de entropía (la incertidumbre) de ese número podría ser insuficiente.

El módulo random de Python
Descripción general

- En Linux se ha desarrollado un repositorio de bits aleatorios que se va rellenando periódicamente con información tomada de la actividad de los dispositivos físicos conectados al ordenador: ficheros /dev/random (bloqueante) y /dev/urandom (no bloqueante, pero potencialmente inseguro).
- En Windows existe algo parecido: *CryptGenRandom*.
- En general, este repositorio no dispone de suficiente volumen de datos como para generar una larga secuencia de números aleatorios de alta calidad, pero sí para producir la semilla.
- Las funciones del módulo random dependen de la función *random()*, que genera números reales uniformemente distribuidos en el intervalo [0.0,1.0) utilizando un generador tipo Mersenne Twister (véase Mersenne Twister - Wikipedia), con periodo = 2**19937 - 1.

El módulo random de Python
Funciones más importantes

- **import random**
- random.**seed([x])**

  Genera la semilla de inicialización del algoritmo a partir de un entero o un objeto *x*; en este último caso, la semilla es *hash(x)*; si no se suministra *x*, se utiliza el repositorio de aleatoriedad; finalmente, si éste no existiera, se utiliza la hora del sistema.

  Al importar el módulo *random* por primera vez, se aplica *seed()* sin semilla, utilizando el repositorio de aleatoriedad si estuviera disponible.
- random.**random()**

  Retorna un número de coma flotante pseudo-aleatorio de 53 bits de precisión en el rango [0.0,1.0), aplicando un generador Mersenne Twister con periodo 2**19937 - 1.

El módulo random de Python
Funciones más importantes

- random.**randint(a, b)**
  Retorna un entero pseudo-aleatorio en el rango [a,b].
- random.**randrange(start, stop[, step])**
  Retorna un número escogido pseudo-aleatoriamente entre los generados por *range(start,stop,step).*

- random.**choice(seq)**
  Retorna un elemento escogido pseudo-aleatoriamente de la secuencia (no vacía) *seq.*
- random.**choices(seq[, weights, cum_weights, k])**
  Retorna una lista de k elementos (por defecto, k=1) tomados uno a uno de forma aleatoria de la secuencia seq (de modo que podrían aparecer repetidos). Opcionalmente, se puede pasar una lista (del mismo tamaño que seq) de pesos relativos (weights) o de pesos acumulados (cum_weights), indicando el peso de cada elemento en la elección (por defecto, se usan pesos iguales).

El módulo random de Python
Funciones más importantes

- random.**shuffle(x)**
  Desordena pseudo-aleatoriamente los elementos de una lista *x*. Nótese que a partir de un cierto tamaño de *x*, el número de posibles permutaciones es mayor que el periodo del generador, lo que significa que la mayor parte de ellas nunca son generadas.
- random.**sample(s,k)**
  Retorna una nueva lista con *k* elementos escogidos pseudo-aleatoriamente de la secuencia o conjunto *s* (que permanece inalterado). Si se trata de generar una lista pseudo-aleatoria de enteros tomados de un rango conocido, se recomienda usar la función *range()* como primer argumento, lo que ahorra tiempo y memoria. Por ejemplo:

```
lista = random.sample(range(10000000),100)
```

- random.**uniform(a,b)**
  Retorna un número real pseudo-aleatorio de acuerdo a una distribución uniforme en el intervalo [a,b).

El módulo random de Python
Funciones más importantes

- random.**normalvariate(mu,sigma)**
  Retorna un número real pseudo-aleatorio de acuerdo a una distribución normal de media *mu* y desviación típica *sigma*.
- random.**gauss(mu,sigma)**
  Hace lo mismo que *normalvariate()* pero es ligeramente más rápida.
- random.**lognormvariate(mu,sigma)**
  Retorna un número real pseudo-aleatorio de acuerdo a una distribución log-normal: si se le aplica el logaritmo natural, se obtiene una distribución normal de media *mu* y desviación típica *sigma>0.0*.
- random.**expovariate(l_param)**
  Retorna un número real pseudo-aleatorio, de acuerdo a una distribución exponencial con λ = *l_param* ≠ *0*.
- random.**gammavariate(alpha,beta)**
  Retorna un número real pseudo-aleatorio, de acuerdo a una distribución gamma con *alpha>0* y *beta>0*.

El módulo random de Python
Ejemplos de uso

Función que genera una cadena pseudo-aleatoria de longitud n:

```
import random
import string
def rnd_str(n):
    result = ''
    s = string.ascii_letters + string.digits
    for i in range(n):
        result = result + random.choice(s)
    return result
for i in range(10):
    print(rnd_str(8))
```

Función que genera un real pseudo-aleatorio en el rango [x,y):

```
import random
def randfloat(x,y):
    return random.random()*(y-x)+x
for i in range(10):
    print(randfloat(23.1217,56.345))
```

El módulo random de Python
Ejemplos de uso

Función que genera una lista de longitud n, con listas de longitud k generadas pseudo-aleatoriamente a partir de la lista x:

```
import random

def randlist(x,n,k):
    result = list()
    for i in range(n):
        result.append(random.sample(x,k))
    return result

x = ["Kepa",56,77.87,"Gernika",12,4,0.003,"Joana","Bilbao"]
r = randlist(x,5,3)
for i in range(5):
    print(r[i])
```

El módulo random de Python
Ejercicios

1. Escribir una función que retorne una palabra de `n` letras minúsculas generada al azar, de modo que al generar cada letra vocales y consonantes tengan la misma probabilidad: las palabras generadas de esta forma tendrán en promedio el mismo número de vocales que de consonantes.
2. Escribir una función que tome como entrada un texto (lo más fácil sería tomar ese texto de un archivo) y un entero n>0. La función deberá retornar un texto de salida con n palabras, eligiendo cada vez una palabra al azar del texto de entrada.
3. Escribir una función que simule el lanzamiento de dos dados, es decir, que cada vez que se le llame retorne un par de números entre 1 y 6.
4. Escribir una función `g(x,v,n)` que retorne una lista con `n` valores que sigan una distribución gaussiana alrededor de `x`, con varianza `v`.
5. Escribir una función que retorne una lista con `k` permutaciones distintas (aleatorias) de los `n` primeros números naturales. La función tendrá como argumentos los valores de `k` y `n` (`k<=n!`).