Introducción a Python. Ejercicio U09\_E01

MasterD

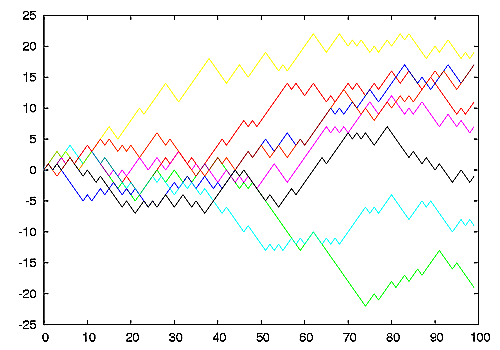
# El marinero borracho

Imagina un marinero que después de seis meses en alta mar, arriba a puerto con su barco y pasa su primera noche en tierra firme festejándolo de cantina en cantina. Cuando empieza a despuntar el alba, el último bar cierra y el marinero tiene que volver caminando a su barco.

El problema es que no se encuentra en muy buen estado y es incapaz de caminar recto. Cada vez que avanza un paso, lo mismo se tuerce hacia la izquierda que se vence hacia la derecha.

¿Cómo simulamos el recorrido que puede hacer nuestro marinero?

Este pequeño juego es conocido en ciencia como el modelo del camino aleatorio (*random walk* en inglés), y sirve para simular multitud de sistemas en los que la trayectoria o posición siguen un proceso aleatorio que solo depende de la posición anterior (<https://es.wikipedia.org/wiki/Camino_aleatorio>).



Ejemplo de Random Walks

## Una primera versión

A continuación te mostramos una versión rudimentaria del algoritmo para simular un recorrido aleatorio de nuestro marinero.

Vamos a representar la posición del marinero en un momento dado con un valor entero. Partimos de la posición x = 0 y a cada paso podemos ir aleatoriamente a la izquierda (-1) o a la derecha (+1). Nuestra primera versión queda así.

import random  
  
def marinero\_borracho\_v1(n):  
 """  
 Simular una trayectoria del marinero borracho  
 después de n pasos  
 """  
 # inicializar variables  
 # posición inicial  
 posicion = 0  
 # trayecto vacío  
 trayecto = []  
   
 # para cada paso a dar  
 for i in range(n):  
 # sacar un número aleatorio entre 0 y 1  
 if random.random() < 0.5:  
 # si es menor que 0.5, al lado negativo  
 paso = -1  
 else:  
 # si no, paso al lado positivo  
 paso = +1  
 # actualizamos la posición  
 posicion = posicion + paso  
 # y la trayectoria  
 trayecto.append(posicion)  
 # devolvemos el trayecto completo  
 return(trayecto)

Comprobemos que funciona

random.seed(123)  
marinero\_borracho\_v1(10)

[-1, -2, -3, -4, -3, -4, -3, -4, -3, -4]

marinero\_borracho\_v1(10)

[-1, -2, -3, -4, -5, -6, -5, -6, -7, -8]

Como corresponde al etílico y aleatorio estado de nuestro marinero, cada vez que intentemos repetir el trayecto obtendremos un nuevo camino.

Pero podemos aprovechar mejor el potencial de Python para construir una función más elegante, ¿verdad?

## Una versión más *pythonica*

Ya sabes que cuando tenemos que procesar y devolver secuencias, en Python tenemos alternativas mejores que un largo bucle.

En esta nueva versión, vamos a aprovechar las características de Python y vamos a dividir el algoritmo en dos partes

* Obtener una secuencia de pasos aleatorios (-1 ó +1) que da el marinero
* Calcular la trayectoria a partir de la secuencia de pasos

Es decir, quedará algo así

def marinero\_borracho\_v2(n):  
 pasos = ... # expresión que produce la secuencia de pasos aleatorios  
 trayectoria = ... # expresión que calcula la trayectoria a partir de los pasos  
 return(trayectoria)

**Pistas**

En las definiciones por comprensión, recuerda que la expresión que genera cada nuevo valor puede ser *cualquier tipo de expresión que devuelva un valor*. En particular, puede ser también una expresión condicional. De hecho, existe una variante de definiciones por comprensión que no hemos visto, adaptadas a este caso. Fíjate en este ejemplo:

[ "par" if i % 2 == 0 else "impar" for i in range(10) ]

['par',  
 'impar',  
 'par',  
 'impar',  
 'par',  
 'impar',  
 'par',  
 'impar',  
 'par',  
 'impar']

La expresión generadora utiliza una construcción valor1 if condicion else valor2 para obtener cada elemento. Usa esto para calcular los pasos.

El elemento *i-ésimo* del trayecto representa la posición tras los i primeros pasos. Recuerda que puedes calcular la suma de una serie de elementos con la función sum().

def marinero\_borracho\_v2(n):  
 """  
 Versión más pythonica  
 para simular la trayectoria  
 del marinero borracho  
 """  
 # construir una lista con los n pasos aleatorios (-1, +1)  
 pasos = [ -1 if random.random() < 0.5 else 1 for i in range(n) ]  
   
 # calcular el trayecto con las n posiciones parciales  
 trayecto = [ sum(pasos[0:(i+1)]) for i in range(n) ]  
   
 return(trayecto)

Probamos

random.seed(123)  
marinero\_borracho\_v2(10)

[-1, -2, -3, -4, -3, -4, -3, -4, -3, -4]

marinero\_borracho\_v2(10)

[-1, -2, -3, -4, -5, -6, -5, -6, -7, -8]

## Más *pythonico* todavía

Aún podemos hacer nuestro código más *pythonico*. ¿Cómo? Tratando de mejorar la eficiencia y aprovechando las funcionalidades de la biblioteca estándar.

**Pistas**

Al calcular los pasos estamos utilizando valores aleatorios para decidir si vamos a un lado (-1) o a otro (+1). Cuando todas las alternativas tienen la misma probabilidad de ocurrir, existe una función en el módulo de procesos aleatorios que permite elegir al azar entre varios elementos.

Recuerda también que hay alternativas más eficientes a construir una lista por comprensión.

Para iterar una secuencia y acumular los valores parciales también disponemos de una función específica en el módulo de herramientas para iteración y bucles. Piensa cómo utilizarla para calcular la trayectoria.

import itertools as it  
  
def marinero\_borracho\_v3(n):  
 """  
 Versión aún más pythonica  
 para simular la trayectoria  
 del marinero borracho  
 """  
 # generador de la secuencia aleatoria de n pasos  
 pasos = ( random.choice([-1,1]) for i in range(n) )  
   
 # generar la trayectoria acumulando los pasos  
 trayecto = it.accumulate(pasos)  
   
 return(trayecto)

Probamos

random.seed(123)  
list(marinero\_borracho\_v3(10))

[-1, 0, -1, 0, 1, 0, -1, 0, 1, 2]

list(marinero\_borracho\_v3(10))

[-1, -2, -3, -2, -1, -2, -3, -4, -3, -4]