Ejercicios semana 12 de marzo - Alejandro Santorum

March 17, 2018

Alejandro Santorum Varela - 17/03/2018
EJERCICIO 1
Apartado 1

Programamos una funcion que escoge un numero al azar entre 0 y 1. Después si x es menor que prob0 (la probabilidad que le hemos asignado al cero) la funcion devuelve cero, y uno en el caso contrario. Por lo tanto hay una probabilidad prob0 de que salga 0 y 1 - prob0 de que sea 1

```
In [25]: def moneda_trucada(prob0):
             x = random()
             if x <= prob0:
                 return 0
             else:
                 return 1
```

```
In [26]: ([moneda_trucada(1/3) for n in srange(10**5)]).count(1)
```

```
Out[26]: 66520
```

Se puede ver que de 10000 lanzamientos se han obtenido 66520 unos, que es prácticamente 2/3 de 10000, como cabía esperar porque le hemos asignado una probabilidad de 1/3 al cero.

A continuacion programamos una funcion que recibe el número de lanzamientos a realizar (número de veces que se llama a la función anterior) y la probabilidad de que el lanzamiento de como resultado 0 (y 1-prob0 como resultado 1).

```
In [27]: def lanzamientos(nLanza, prob0):
             count = 0;
             for i in xsrange(0, nLanza):
                 if moneda_trucada(prob0)==0:
                     count += 1
             return count
```

```
In [28]: L = [lanzamientos(1000, 1/10) for i in xsrange(1000)]
         media = sum(L)/len(L)
         print media.n()
```

```
99.8830000000000
```

Hemos realizado 1000 lanzamientos 1000 veces, con 1/10 de posibilidades de que saliera el cero (el cero y el uno lo podemos considerar cara o cruz independientemente). El resultado es una media de 99.88 veces que ha salido el cero, lo cual era lo esperado.

```
In [31]: def function(nVeces, nLanza):
             LT = list()
             for k in xsrange(1, 10):
                 L = [lanzamientos(nLanza, k*1/10) for i in xsrange(nVeces)]
                 media = sum(L)/len(L)
                 LT.append((k*1/10, media.n()))
             return LT

In [32]: function(1000, 1000)

Out[32]: [(1/10, 99.5020000000000),
          (1/5, 200.026000000000),
          (3/10, 300.350000000000),
          (2/5, 399.415000000000),
          (1/2, 500.380000000000),
          (3/5, 599.815000000000),
          (7/10, 699.763000000000),
          (4/5, 799.836000000000),
          (9/10, 899.930000000000)]
```

Apartado 2

A continuación se muestra una función que devuelve números aleatorios entre a y b, ambos inclusive.

```
In [35]: def aleatorio(a, b):
             return (a+(b-a)*random())

In [36]: Q = [aleatorio(5, 10) for i in xsrange(10)]
         print Q

[6.954032601787269, 9.890927197079826, 6.4521696544769815, 9.922604965607938, 9.139117386798496
```

Apartado 3

La continuación que sigue tiene el objetivo de contar el número de lanzamientos que se deben realizar para que un jugador con euros Euros se arruine, considerando que el casino tiene dinero infinito. Es obvio que el jugador tarde o temprano se arruina, ya que es el que tiene dinero finito.

```
In [37]: def cuantoTardaEnArruinarse(euros):
             count = 0
             while(1):
                 x = moneda_trucada(1/2)
                 count += 1
                 if x == 0:
                     euros -= 1
```

```
                    if euros==0:
                        return count
                else:
                    euros += 1
```

```
In [38]: W = [cuantoTardaEnArruinarse(100) for i in xsrange(10)]
         print W
```

```
[21262, 70270, 92276, 6648, 13708, 7960, 42740, 15806, 633432, 11420]
```

Suponiendo que el jugador decide retirarse cuando le quede la mitad o cuando alcance el doble de su dinero inicial, calcularemos la probabilidad de "ganar", que es duplicar su dinero.

La siguiente función simula un "Juego", es decir, recibe la cantidad inicial del jugador y realiza lanzamientos, dandole un euro si el jugador acierta o quitándoselo en caso contrario, hasta que el jugador gana (duplica su dinero) devolviendo 1 o pierde (alcanza la mitad de su dinero inicial) devolviendo 0.

```
In [40]: def juego(euros):
             ini = euros
             while(1):
                 x = moneda_trucada(1/2)
                 if x==0:
                     euros -= 1
                     if euros == (ini//2):
                         return 0
                 else:
                     euros += 1
                     if euros == 2*ini:
                         return 1
```

Por ultimo, la siguiente función realiza N juegos con un dinero inicial de euros y devuelve la suma de las veces que el jugador ganó, dividido entre el número de partidas.

```
In [41]: def probJuego(N, euros):
             count = 0
             for i in xsrange(N):
                 if juego(euros)==1:
                     count += 1
             return (count/N).n()
```

```
In [43]: %time probJuego(10000, 100)
```

```
CPU times: user 3min, sys: 908 ms, total: 3min 1s
Wall time: 2min 59s
```

```
Out[43]: 0.336400000000000
```

Se puede ver que gana al rededor de 1/3 de las partidas, lo cual es claramente NO rentable.