Práctica 0: Python

Nuria Bango Iglesias (nubango@ucm.es)

Álvar Julián de Diego López (alvarded@ucm.es)

```python
import numpy as np
import matplotlib.pyplot as plt
import random
# calcula la integral de fun entre a y b por el metodo montecarlo
# generando para ello num_puntos aleatoriamente

# comparar tiempos de ejecucion entre ambas versiones

#calcular f(x) en muchos puntos para calcular el rectangulo que tenga dentro la funcion
#calcular que parte queda debajo
#calcular muchos puntos x,y aleatorios y mirar si estan por debajo o por encima de la curva mirando si y es
#repetir 1M o 2M de veces y sacar % del area que corresponde a la integral


def f(x):
    return x**x+1

# version iterativa que realiza num_puntos iteraciones para calcular el resultado
def integra_mc(fun, a, b, num_puntos = 10000):
    div = (b-a)/1000
    M = 0

    plt.figure()

    for i in range(1000):
        plt.plot(a+div*i , fun(a+div*i), '.', c='blue', zorder=2)
        if fun(a+div*i) > M:
            M = fun(a+div*i)

    debajo = 0
    for i in range(num_puntos):
        x = np.random.rand(1) * (b - a) + a
        y = np.random.rand(1) * M
        plt.plot(x, y, 'x', c = 'red', zorder=1)
        if(y < fun(x)):
            debajo +=1

    plt.savefig('mcbucle.pdf')
    plt.close

    areaTotal = abs(b-a) * abs(M - 0)
    return areaTotal * (debajo / num_puntos)
```

```python
def integra_mc_vec(fun, a, b, num_puntos = 10000):
    """Calcula la integral de f entre a y b por el metodo de
    Monte Carlo sin usar bucles"""
    eje_x = np.linspace(a, b, num_puntos)
    eje_y = fun(eje_x)
    max_f = max(eje_y)
    min_f = 0
    random_x = np.random.uniform(a, b, num_puntos)
    random_y = np.random.uniform(min_f, max_f, num_puntos)
    f_random_x = fun(random_x)

    plt.figure()
    plt.plot(random_x, random_y, 'x', c = 'red')
    plt.plot(eje_x, eje_y, '-')
    plt.savefig('mc.pdf')
    plt.close
    debajo = sum(random_y < f_random_x)
    area_total = abs(b-a) * abs(max_f - min_f)
    return area_total * (debajo / num_puntos)

print(integra_mc(f,0, 4))
print(integra_mc_vec(f,0, 4))
```