Practica 0: Python

Alberto Muñoz Fernández

Óscar García Castro

```
import imp
from random import Random
import numpy as np
import scipy.integrate
import matplotlib.pyplot as plt
import time

def cuadrado(x):    #Función cuadrática
    return x*x

def calcularMax(fun, a, b):     #Cálculo para obtener el máximo valor y
de fun en el tramo a-b
    max = 0
    inter = (b-a)/1000
    numAux = a
    for i in range (1000) :
        aux = fun(numAux)
        if (aux > max)  :
            max = aux
        numAux += inter
    return max

def integra_it(fun, a, b, max, num_puntos=10000) :  #Método iterativo
resolución problema
    tic = time.process_time()
    ptos = 0
    for i in range (num_puntos) :
        x = np.random.uniform(a, b)
        y = np.random.uniform(0, max)
        if (fun(x) > y):
            ptos += 1
    toc = time.process_time()
    return 1000* (toc - tic)


def integra_vec(fun, a, b, max, num_puntos=10000) :     #Método vectorial
resolución problema
    tic = time.process_time()
    x = np.random.uniform(a, b, num_puntos)
    y = np.random.uniform(0, max, num_puntos)
```

```
    ar = fun(x)
    np.sum(y < ar)
    toc = time.process_time()
    return 1000* (toc - tic)


def main() :
    max = calcularMax(cuadrado, 0, 100)

    sizes = np.linspace(1000, 1000000, 20)          #Cantidades de puntos
para las pruebas de tiempo
    times_it = []
    times_vec = []
    for size in sizes:          #Rellenado de los vectores con el tiempo
de ejecución con variación del número de puntos
        max = calcularMax(cuadrado, 0, 1000)
        times_it += [integra_it(cuadrado, 0, 1000, max, int(size))]
        times_vec += [integra_vec(cuadrado, 0, 1000, max, int(size))]

    plt.figure()        #Dibujado de los vectores
    plt.scatter(sizes, times_it, c = 'red', label = 'iterativo')
    plt.scatter(sizes, times_vec, c = 'blue', label = 'vector')
    plt.legend()
    plt.savefig('times.png')
    plt.show()

main()
```