Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №7**

По дисциплине: “Языки программирования”

Тема: “Изучение NumPy. Сравнение производительности с классическими библиотеками Python”

**Выполнил**:

студент 2 курса

группы ПО-7

Комиссаров А.Е.

**Проверила:** Дряпко А. В.

Брест 2021

**Цель:**Изучить NumPy, сравнить производительность с классическими библиотеками Python

**Задание:**

1. Для написания кода использовать библиотеки классического Python, NumPy и SciPy.
2. Код демонстрируется в Jupyter Notebook
3. По каждому заданию должно быть предоставлено не менее 3-х вариантов решения, среди которых:
4. чистый NumPy (максимально оптимизованный, векторизованный)
5. любой не векторизованный вариант
6. любой другой вариант, желательно конкурентноспособный
7. Все варианты решения должны быть протестированы на скорость выполнения при помощи %timeit
8. Полученные результаты отразить в отчете и сделать выводы о производительности и комфорте использования NumPy в различных задачах.name:

**Код программы:**

from collections import Counter

from math import sqrt

import numpy as np

import matplotlib.pyplot as plt

import scipy

from scipy.sparse.linalg import spsolve

from scipy.stats import multivariate\_normal

################################################################################################ Task 1

array1\_py = [[1, 0, 1], [2, 0, 2], [3, 0, 3], [4, 4, 4]]

array1\_np = np.array(array1\_py)

def diag\_matrix\_py(arr):

product = 1

for i in range(len(arr[0])):

el = arr[i][i]

if el:

product \*= el

return product

def diag\_matrix\_np(arr):

diagonal = np.diag(arr)

return np.prod(diagonal[diagonal != 0])

print("py1:", end=" ")

%timeit diag\_matrix\_py(array1\_py)

print("np1:", end=" ")

%timeit diag\_matrix\_np(array1\_np)

################################################################################################ Task 2

arr2\_py = [[9, 4, 2], [6, 0, 0], [9, 9, 3]]

i2\_py = [1, 2, 1]

j2\_py = [1, 0, 1]

arr2\_np = np.array(arr2\_py)

i2\_np = np.array(i2\_py)

j2\_np = np.array(j2\_py)

def vector\_py(arr, i, j):

vector = []

for k in range(len(i)):

vector.append(arr[i[k]][j[k]])

return vector

def vector\_np(arr, i, j):

return arr[i,j]

print("py2:", end=" ")

%timeit vector\_py(arr2\_py, i2\_py, j2\_py)

print("np2:", end=" ")

%timeit vector\_np(arr2\_np, i2\_np, j2\_np)

################################################################################################ Task 3

x3\_py = [1, 2, 2, 4]

y3\_py = [4, 2, 1, 2]

x3\_np = np.array(x3\_py)

y3\_np = np.array(y3\_py)

def equal\_py(arr1, arr2):

return sorted(arr1) == sorted(arr2)

def equal\_np(arr1, arr2):

return np.array\_equal(np.bincount(arr1), np.bincount(arr2))

print("py3:", end=" ")

%timeit equal\_py(x3\_py, y3\_py)

print("np3:", end=" ")

%timeit equal\_np(x3\_np, y3\_np)

################################################################################################ Task 4

x4\_py = [6, 2, 0, 3, 0, 0, 5, 7, 0]

x4\_np = np.array(x4\_py)

def null\_in\_arr\_py(arr):

count = 0

for i in range(1, len(arr)):

if not arr[i-1] and arr[i] > count:

count = arr[i]

return count

def null\_in\_arr\_np(arr):

zero\_indexes = arr == 0

return arr[1:][zero\_indexes[:-1]].max()

print("py4:", end=" ")

%timeit null\_in\_arr\_py(x4\_py)

print("np4:", end=" ")

%timeit null\_in\_arr\_np(x4\_np)

################################################################################################ Task 5

photo\_arr = plt.imread("https://upload.wikimedia.org/wikipedia/commons/4/47/PNG\_transparency\_demonstration\_1.png")

photo\_arr\_py = photo\_arr.tolist()

photo\_arr\_np = np.array(photo\_arr\_py)

rgb\_arr\_py = [0.2989, 0.587, 0.114]

rgb\_arr\_np = np.array(rgb\_arr\_py)

def gray\_py(arr, rgb\_arr):

result\_arr = []

for height in arr:

new\_width = []

for width in height:

color = 0

for i in range(3):

color += width[i] \* rgb\_arr[i]

new\_width.append(color)

result\_arr.append(new\_width)

return result\_arr

def gray\_np(arr, rgb\_arr):

return np.dot(arr[...,:3], rgb\_arr)

arr\_py = gray\_np(photo\_arr, rgb\_arr\_py)

plt.imshow(arr\_py, cmap='gray')

plt.show()

print("py5:", end=" ")

%timeit gray\_py(photo\_arr\_py, rgb\_arr\_py)

print("np5:", end=" ")

%timeit gray\_np(photo\_arr\_np, rgb\_arr\_np)

################################################################################################ Task 6

arr6\_py = [2, 2, 2, 3, 3, 3, 5]

arr6\_np = np.array(arr6\_py)

def counts\_py(arr):

return tuple(zip(\*Counter(arr).items()))

def counts\_np(arr):

bin\_count = np.bincount(arr)

non\_zero\_indexes = np.nonzero(bin\_count)[0]

return non\_zero\_indexes, bin\_count[non\_zero\_indexes]

print("py6:", end=" ")

%timeit counts\_py(arr6\_py)

print("np6:", end=" ")

%timeit counts\_np(arr6\_np)

################################################################################################ Task 7

x7\_py = [2, 7, 6, 6, 9, 6, 3, 4, 9]

y7\_py = [1, 0, 0, 7, 2, 2, 4, 3, 0]

x7\_np = np.array(x7\_py)

y7\_np = np.array(y7\_py)

def euclid\_py(arr\_x, arr\_y):

return sqrt(sum((x - y) \*\* 2.0 for x, y in zip(arr\_x, arr\_y)))

def euclid\_np(arr\_x, arr\_y):

return np.linalg.norm(arr\_x - arr\_y)

print("py7: ", end=" ")

%timeit euclid\_py(x7\_py, y7\_py)

print("scipy7:", end=" ")

%timeit scipy.spatial.distance.euclidean(x7\_py, y7\_py)

print("np7: ", end=" ")

%timeit euclid\_np(x7\_np, y7\_np)

################################################################################################ Task 8

sigma = np.array([

[2.3, 0, 0, 0],

[0, 1.5, 0, 0],

[0, 0, 1.7, 0],

[0, 0, 0, 2]

])

mu = np.array([2, 3, 8, 10])

x = np.array([2.1, 3.5, 8, 9.5])

def logarithm(x, mu, S):

norm\_coefficient = len(S) \* np.log(2 \* np.pi) + np.linalg.slogdet(S)[1]

err = x - mu

numerator = spsolve(S, err).T.dot(err)

return -0.5 \* (norm\_coefficient + numerator)

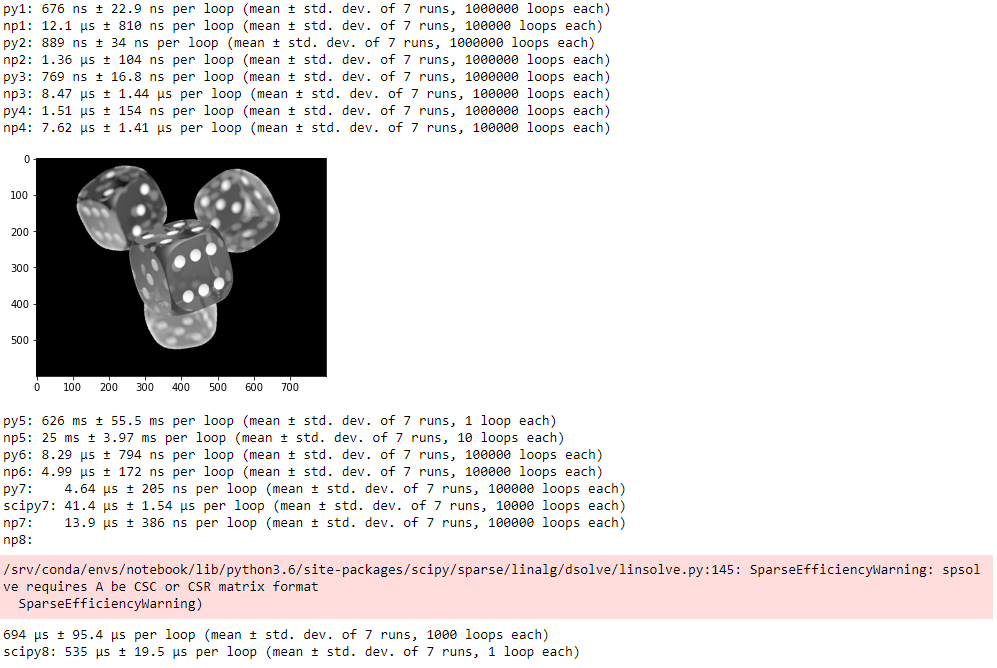
print("np8: ", end=" ")

%timeit logarithm(x, mu, sigma)

print("scipy8:", end=" ")

%timeit multivariate\_normal(mu, sigma).logpdf(x)

**Результат работы программы:**



**Вывод**: Я изучил NumPy, сравнил производительность с классическими библиотеками Python