**Міністерство освіти і науки України**

**Національний Технічний Університет України**

**«Київський політехнічний інститут ім. Ігоря Сікорського»**

**Теплоенергетичний факультет**

**Кафедра автоматизації проектування енергетичних процесів і систем**

**Лабораторна робота №1**

з дисципліни

«Прикладні методи аналізу даних»

Виконав:

аспірант групи ТВ-71ф

Ляшенко М.М.

**Київ - 2018**

### Завдання 1: Обчислити поелементно

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

*# defining the tensor variables*

X = T.matrix(**"X"**)

W = T.matrix(**"W"**)

b\_sym = T.vector(**"b\_sym"**)

results, updates = theano.scan(**lambda** v: T.tanh(T.dot(v, W) + b\_sym), sequences=X)

compute\_elementwise = theano.function(inputs=[X, W, b\_sym], outputs=results)

*# test values*

x = np.eye(2, dtype=theano.config.floatX)

w = np.ones((2, 2), dtype=theano.config.floatX)

b = np.ones((2), dtype=theano.config.floatX)

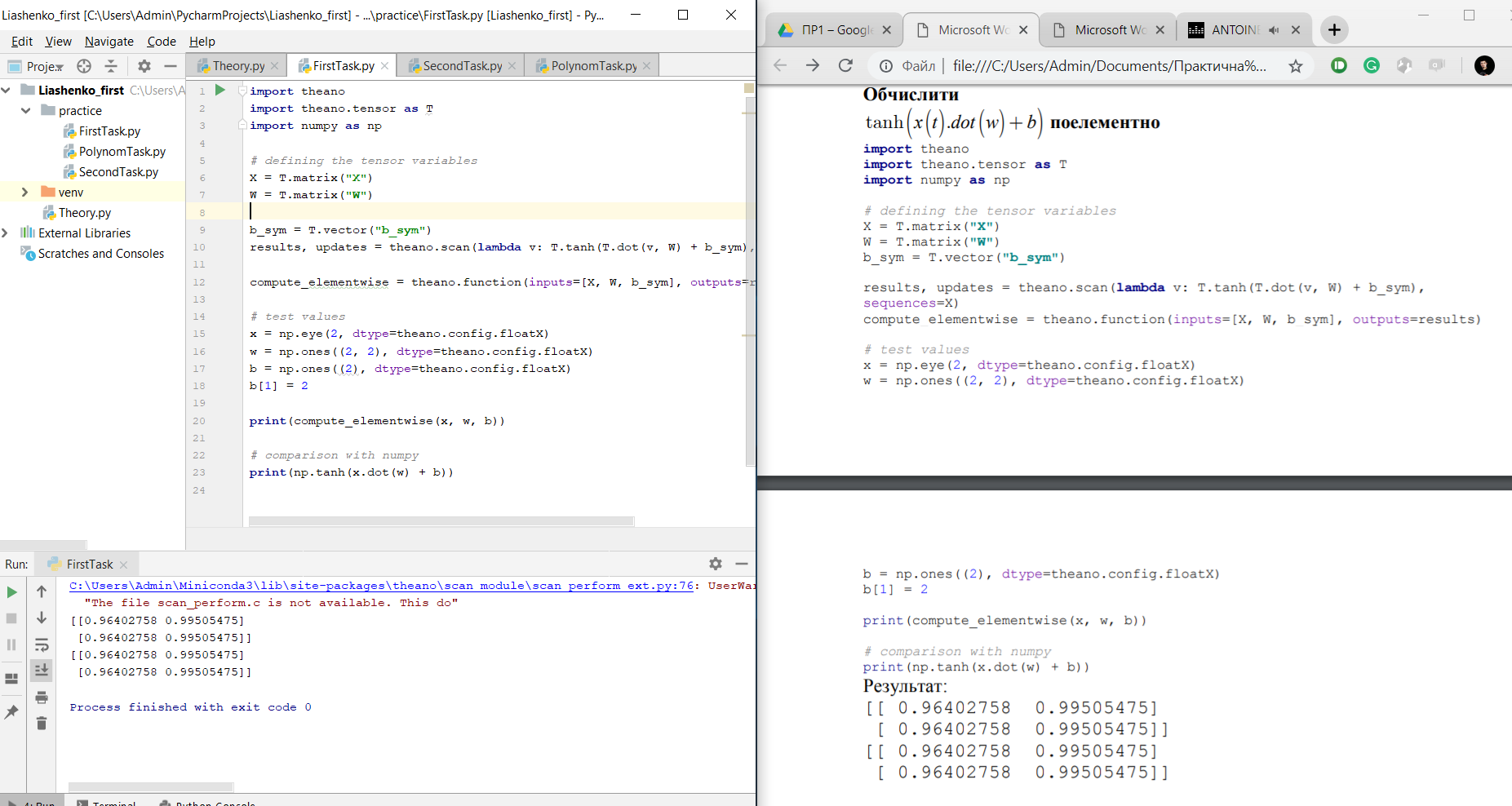
b[1] = 2

r = compute\_elementwise(x, w, b)

print(r)

*# comparison with numpy*

print(np.tanh(x.dot(w) + b))



Результати:

[

[0.96402758 0.99505475]

[0.96402758 0.99505475]

]

[

[0.96402758 0.99505475]

[0.96402758 0.99505475]

]

### Завдання 2: Обчислити послідовність

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

*# define tensor variables*

X = T.vector(**"X"**)

W = T.matrix(**"W"**)

b\_sym = T.vector(**"b\_sym"**)

U = T.matrix(**"U"**)

Y = T.matrix(**"Y"**)

V = T.matrix(**"V"**)

P = T.matrix(**"P"**)

results, updates = theano.scan(**lambda** y, p, x\_tm1: T.tanh(T.dot(x\_tm1, W) + T.dot(y, U) + T.dot(p, V)),

sequences=[Y, P[::-1]], outputs\_info=[X])

compute\_seq = theano.function(inputs=[X, W, Y, U, P, V], outputs=results)

*# test values*

x = np.zeros((2), dtype=theano.config.floatX)

x[1] = 1

w = np.ones((2, 2), dtype=theano.config.floatX)

y = np.ones((5, 2), dtype=theano.config.floatX)

y[0, :] = -3

u = np.ones((2, 2), dtype=theano.config.floatX)

p = np.ones((5, 2), dtype=theano.config.floatX)

p[0, :] = 3

v = np.ones((2, 2), dtype=theano.config.floatX)

r = compute\_seq(x, w, y, u, p, v)

print(r)

*# comparison with numpy*

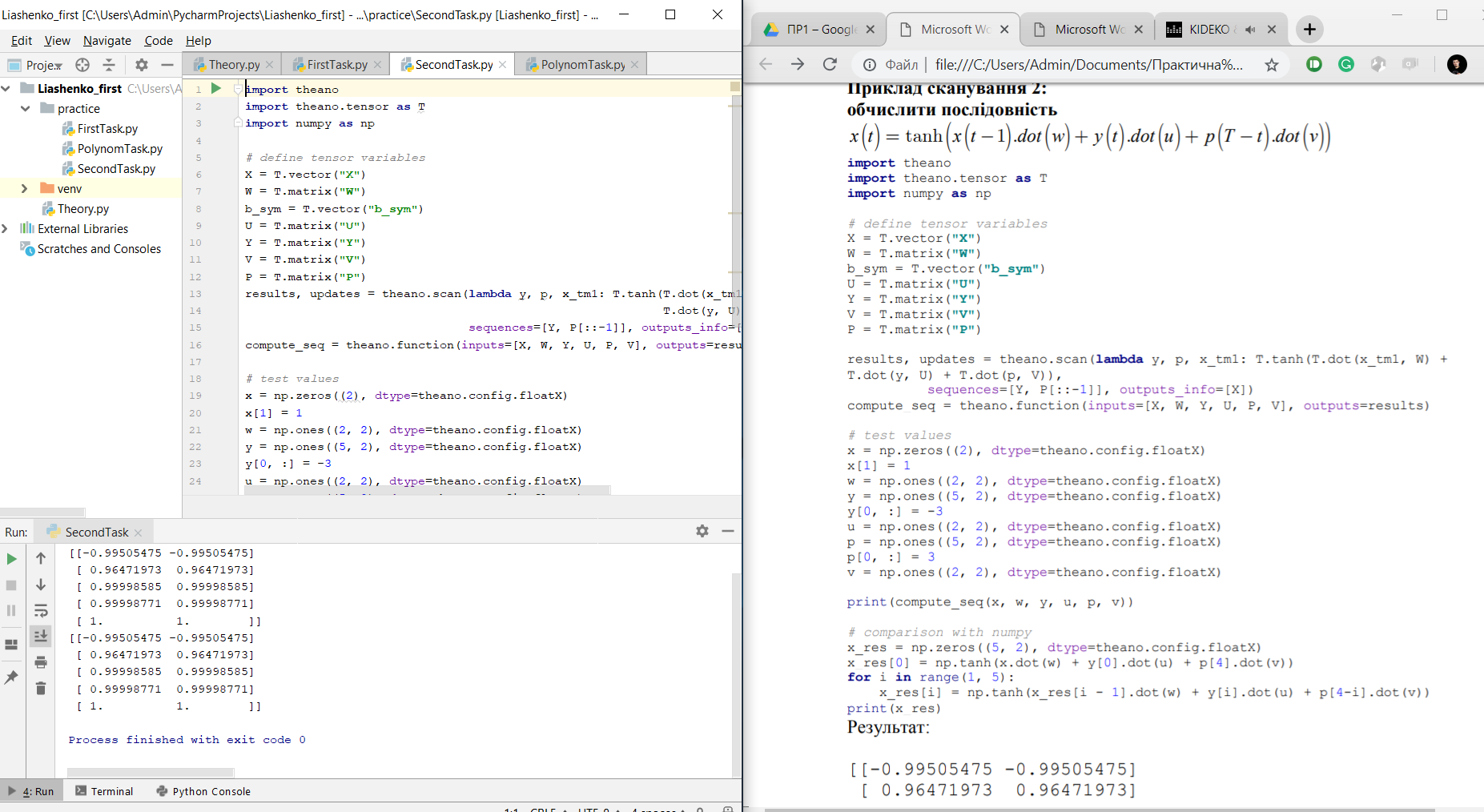
x\_res = np.zeros((5, 2), dtype=theano.config.floatX)

x\_res[0] = np.tanh(x.dot(w) + y[0].dot(u) + p[4].dot(v))

**for** i **in** range(1, 5):

x\_res[i] = np.tanh(x\_res[i - 1].dot(w) + y[i].dot(u) + p[4-i].dot(v))

print(x\_res)



Результати:

[

[-0.99505475 -0.99505475]

[ 0.96471973 0.96471973]

[ 0.99998585 0.99998585]

[ 0.99998771 0.99998771]

[ 1. 1. ]

]

[

[-0.99505475 -0.99505475]

[ 0.96471973 0.96471973]

[ 0.99998585 0.99998585]

[ 0.99998771 0.99998771]

[ 1. 1. ]

]

### Завдання 3: Обчислення норми X по рядках

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

*# define tensor variable*

X = T.matrix(**"X"**)

results, updates = theano.scan(**lambda** x\_i: T.sqrt((x\_i \*\* 2).sum()), sequences=[X])

compute\_norm\_lines = theano.function(inputs=[X], outputs=results)

*# test value*

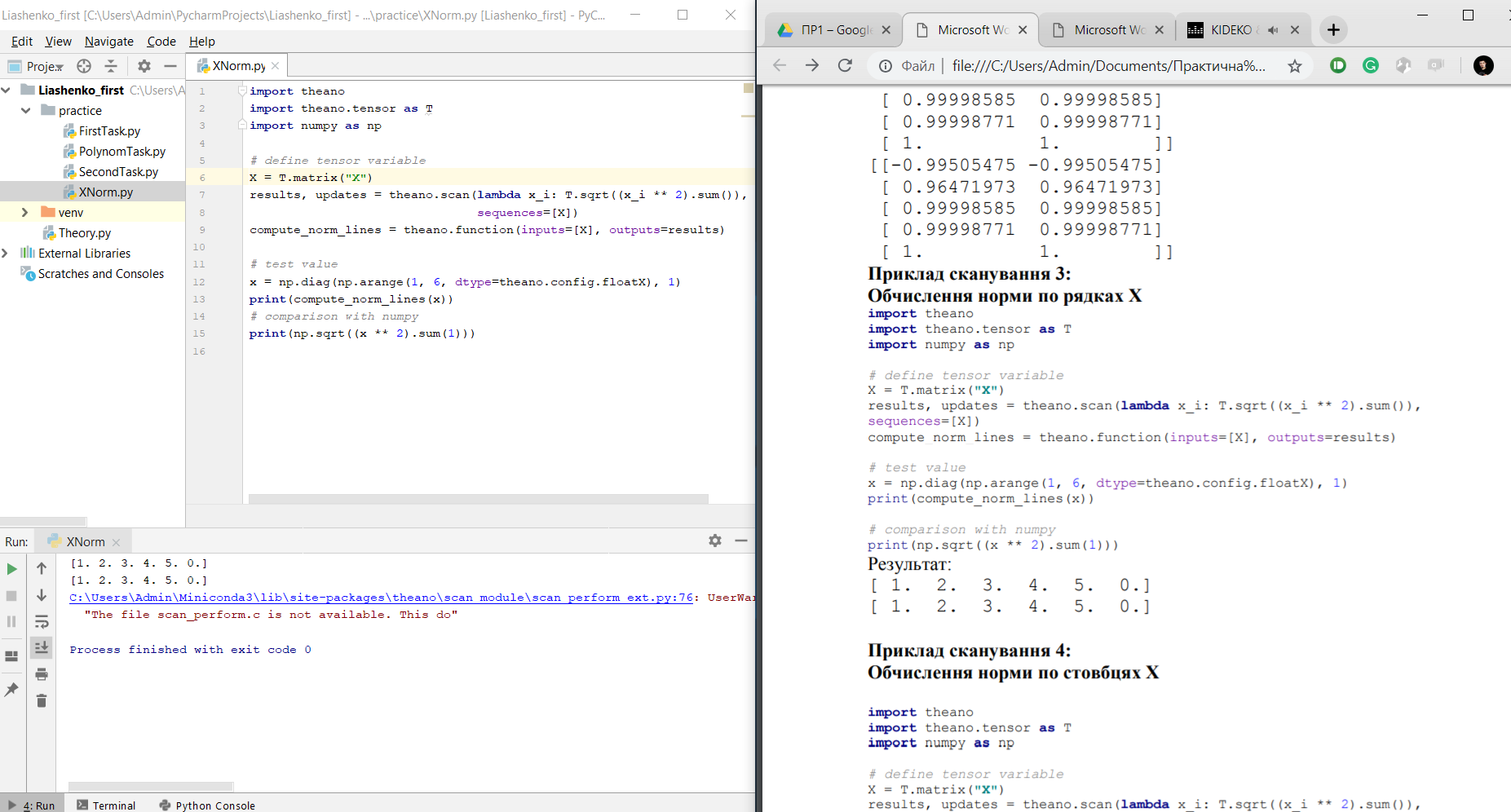
x = np.diag(np.arange(1, 6, dtype=theano.config.floatX), 1)

r = compute\_norm\_lines(x)

print(r)

*# comparison with numpy*

print(np.sqrt((x \*\* 2).sum(1)))



Результати:

[1. 2. 3. 4. 5. 0.]

[1. 2. 3. 4. 5. 0.]

### Завдання 4: Обчислення норми X по стовбцях

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

*# define tensor variable*

X = T.matrix(**"X"**)

results, updates = theano.scan(**lambda** x\_i: T.sqrt((x\_i \*\* 2).sum()), sequences=[X.T])

compute\_norm\_cols = theano.function(inputs=[X], outputs=results)

*# test value*

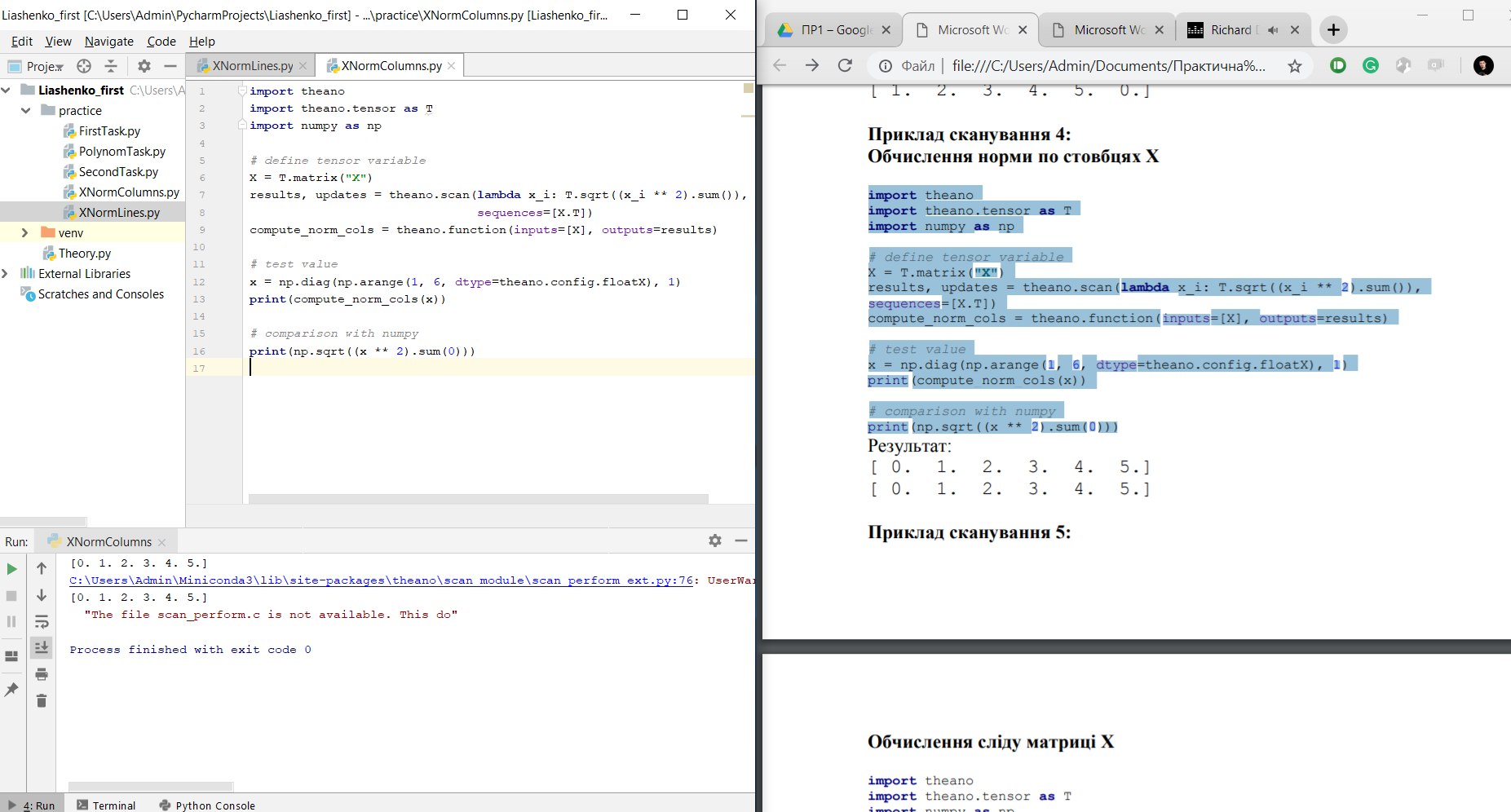
x = np.diag(np.arange(1, 6, dtype=theano.config.floatX), 1)

r = compute\_norm\_cols(x)

print(r)

*# comparison with numpy*

print(np.sqrt((x \*\* 2).sum(0)))



Результати:

[0. 1. 2. 3. 4. 5.]

[0. 1. 2. 3. 4. 5.]

### Завдання 5: Обчислення сліду матриці X

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

floatX = **"float32"**

*# define tensor variable*

X = T.matrix(**"X"**)

results, updates = theano.scan(**lambda** i, j, t\_f: T.cast(X[i, j] + t\_f, floatX),

sequences=[T.arange(X.shape[0]), T.arange(X.shape[1])],

outputs\_info=np.asarray(0., dtype=floatX))

result = results[-1]

compute\_trace = theano.function(inputs=[X], outputs=result)

*# test value*

x = np.eye(5, dtype=theano.config.floatX)

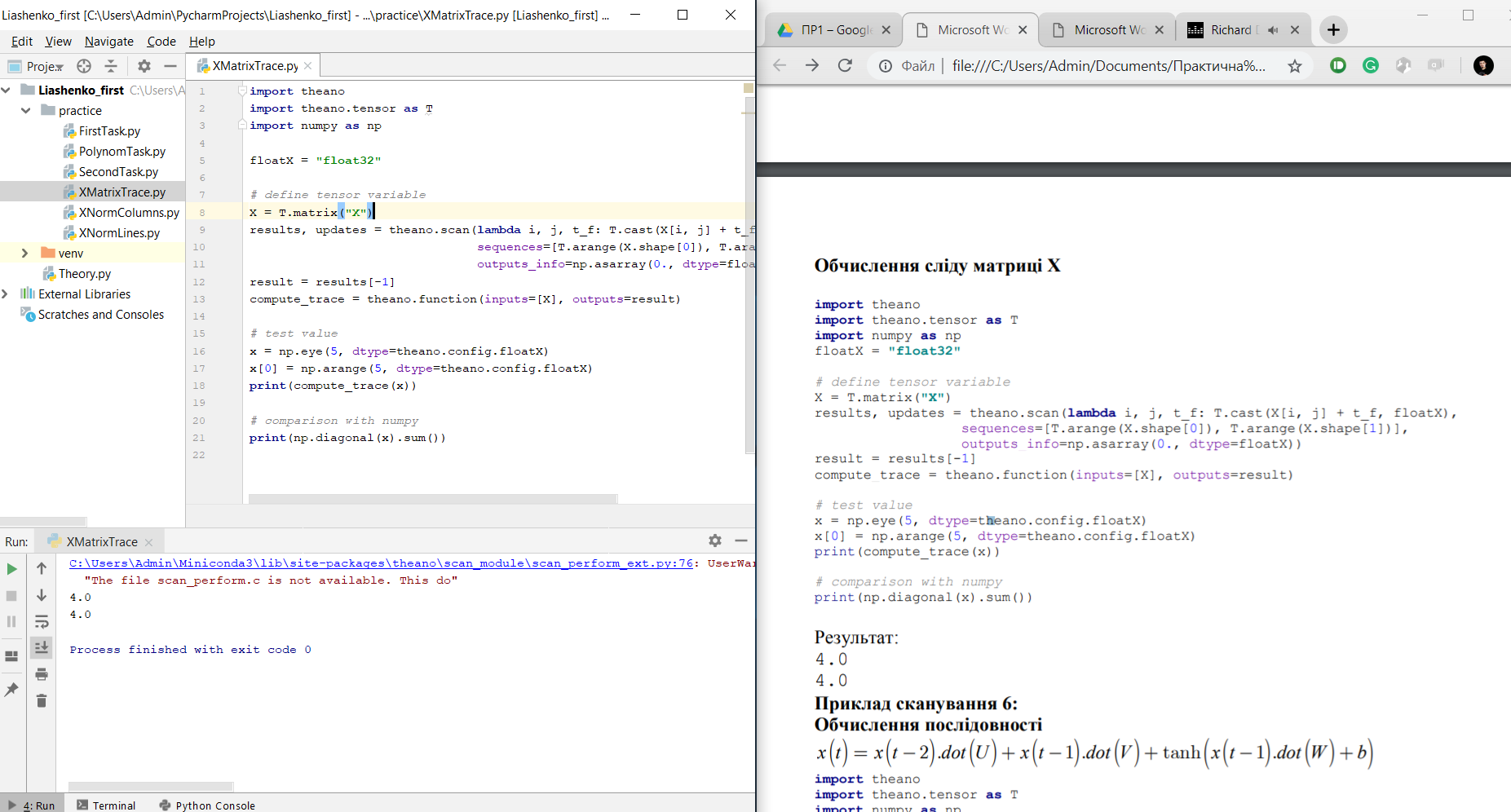
x[0] = np.arange(5, dtype=theano.config.floatX)

r = compute\_trace(x)

print(r)

*# comparison with numpy*

print(np.diagonal(x).sum())



Результати:

4.0

4.0

### Завдання 6: Обчислення послідовності

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

*# define tensor variables*

X = T.matrix(**"X"**)

W = T.matrix(**"W"**)

b\_sym = T.vector(**"b\_sym"**)

U = T.matrix(**"U"**)

V = T.matrix(**"V"**)

n\_sym = T.iscalar(**"n\_sym"**)

results, updates = theano.scan(**lambda** x\_tm2, x\_tm1: T.dot(x\_tm2, U) + T.dot(x\_tm1, V) + T.tanh(T.dot(x\_tm1, W) + b\_sym),

n\_steps=n\_sym, outputs\_info=[dict(initial=X, taps=[-2, -1])])

compute\_seq2 = theano.function(inputs=[X, U, V, W, b\_sym, n\_sym], outputs=results)

*# test values*

x = np.zeros((2, 2), dtype=theano.config.floatX) *# the initial value must be able to return x[-2]*

x[1, 1] = 1

w = 0.5 \* np.ones((2, 2), dtype=theano.config.floatX)

u = 0.5 \* (np.ones((2, 2), dtype=theano.config.floatX) - np.eye(2, dtype=theano.config.floatX))

v = 0.5 \* np.ones((2, 2), dtype=theano.config.floatX)

n = 10

b = np.ones((2), dtype=theano.config.floatX)

r = compute\_seq2(x, u, v, w, b, n)

print(r)

*# comparison with numpy*

x\_res = np.zeros((10, 2))

x\_res[0] = x[0].dot(u) + x[1].dot(v) + np.tanh(x[1].dot(w) + b)

x\_res[1] = x[1].dot(u) + x\_res[0].dot(v) + np.tanh(x\_res[0].dot(w) + b)

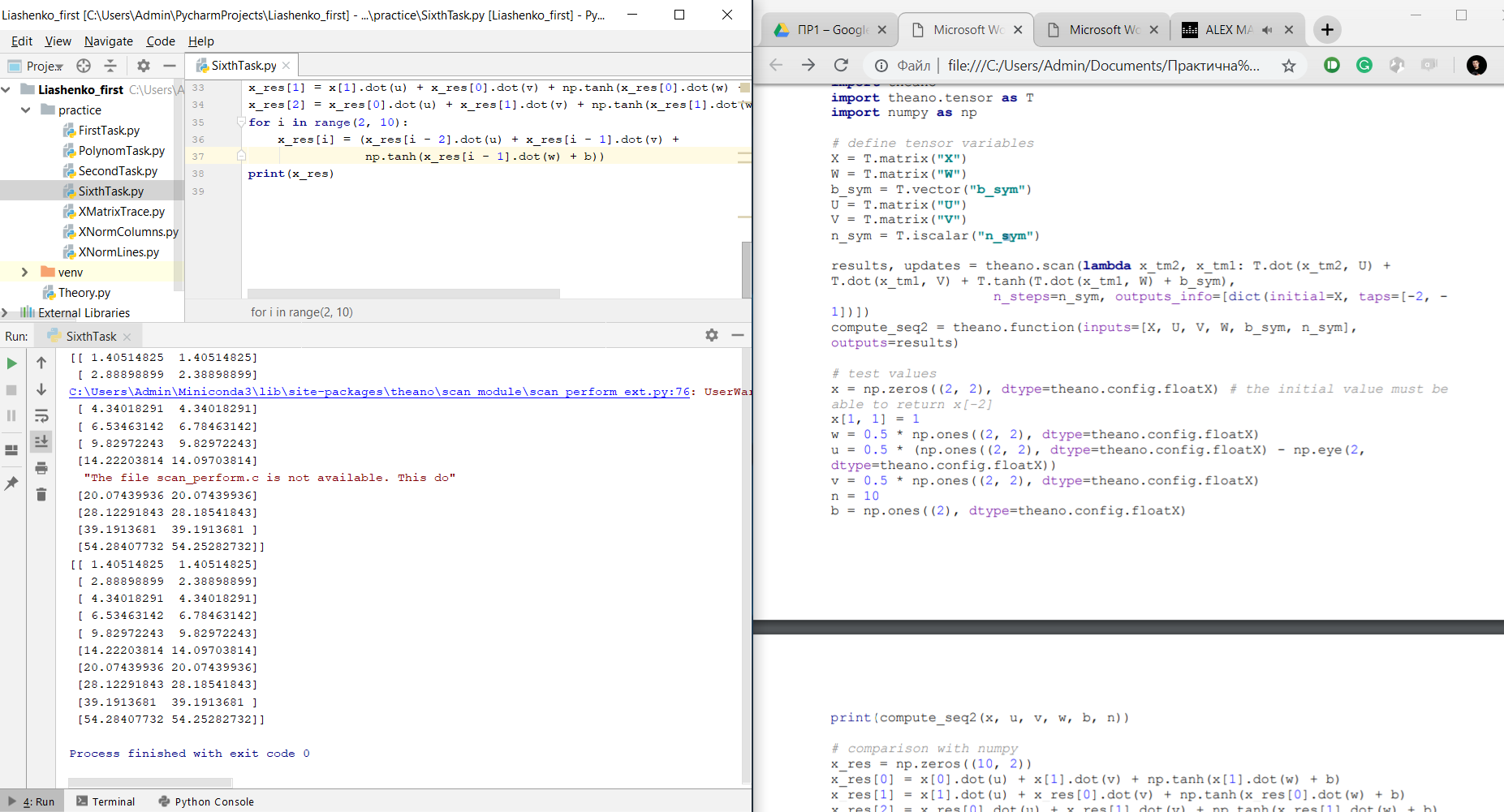
x\_res[2] = x\_res[0].dot(u) + x\_res[1].dot(v) + np.tanh(x\_res[1].dot(w) + b)

**for** i **in** range(2, 10):

x\_res[i] = (x\_res[i - 2].dot(u) + x\_res[i - 1].dot(v) +

np.tanh(x\_res[i - 1].dot(w) + b))

print(x\_res)



Результати:

[

[ 1.40514825 1.40514825]

[ 2.88898899 2.38898899]

[ 4.34018291 4.34018291]

[ 6.53463142 6.78463142]

[ 9.82972243 9.82972243]

[14.22203814 14.09703814]

[20.07439936 20.07439936]

[28.12291843 28.18541843]

[39.1913681 39.1913681 ]

[54.28407732 54.25282732]

]

[

[ 1.40514825 1.40514825]

[ 2.88898899 2.38898899]

[ 4.34018291 4.34018291]

[ 6.53463142 6.78463142]

[ 9.82972243 9.82972243]

[14.22203814 14.09703814]

[20.07439936 20.07439936]

[28.12291843 28.18541843]

[39.1913681 39.1913681 ]

[54.28407732 54.25282732]

]

### Завдання 7: Обчислення Якобіана

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

*# define tensor variables*

v = T.vector()

A = T.matrix()

y = T.tanh(T.dot(v, A))

results, updates = theano.scan(**lambda** i: T.grad(y[i], v), sequences=[T.arange(y.shape[0])])

compute\_jac\_t = theano.function([A, v], results, allow\_input\_downcast=**True**) *# shape (d\_out, d\_in)*

*# test values*

x = np.eye(5, dtype=theano.config.floatX)[0]

w = np.eye(5, 3, dtype=theano.config.floatX)

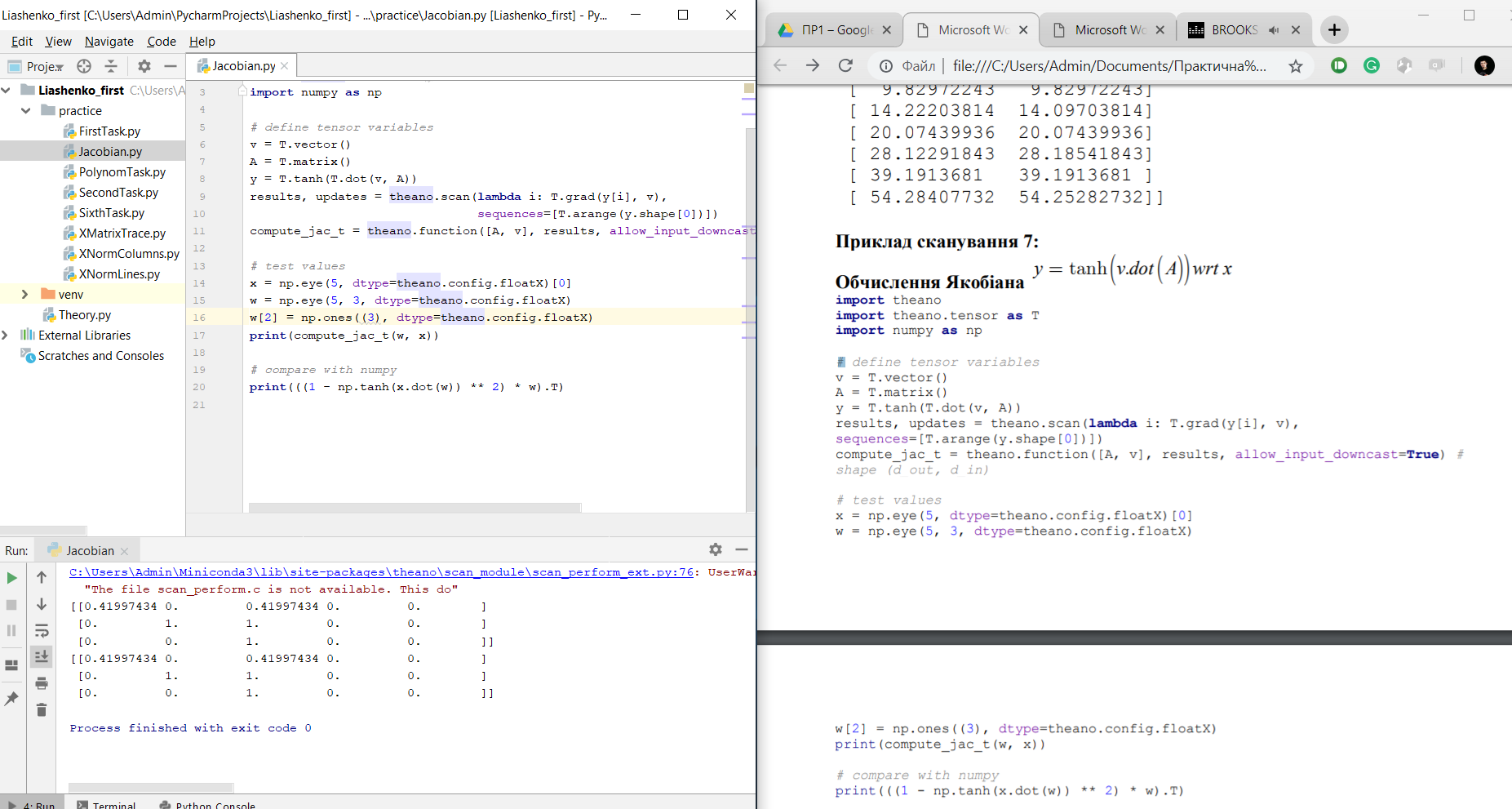
w[2] = np.ones((3), dtype=theano.config.floatX)

r = compute\_jac\_t(w, x)

print(r)

*# compare with numpy*

print(((1 - np.tanh(x.dot(w)) \*\* 2) \* w).T)



Результати:

[

[0.41997434 0. 0.41997434 0. 0. ]

[0. 1. 1. 0. 0. ]

[0. 0. 1. 0. 0. ]

]

[

[0.41997434 0. 0.41997434 0. 0. ]

[0. 1. 1. 0. 0. ]

[0. 0. 1. 0. 0. ]

]

### Завдання 8: Обчислення , де є біноміальним

**import** theano

**import** theano.tensor **as** T

**import** numpy **as** np

*# define tensor variables*

X = T.matrix(**"X"**)

W = T.matrix(**"W"**)

b\_sym = T.vector(**"b\_sym"**)

*# define shared random stream*

trng = T.shared\_randomstreams.RandomStreams(1234)

d=trng.binomial(size=W[1].shape)

results, updates = theano.scan(**lambda** v: T.tanh(T.dot(v, W) + b\_sym) \* d, sequences=X)

compute\_with\_bnoise = theano.function(inputs=[X, W, b\_sym], outputs=results,

updates=updates, allow\_input\_downcast=**True**)

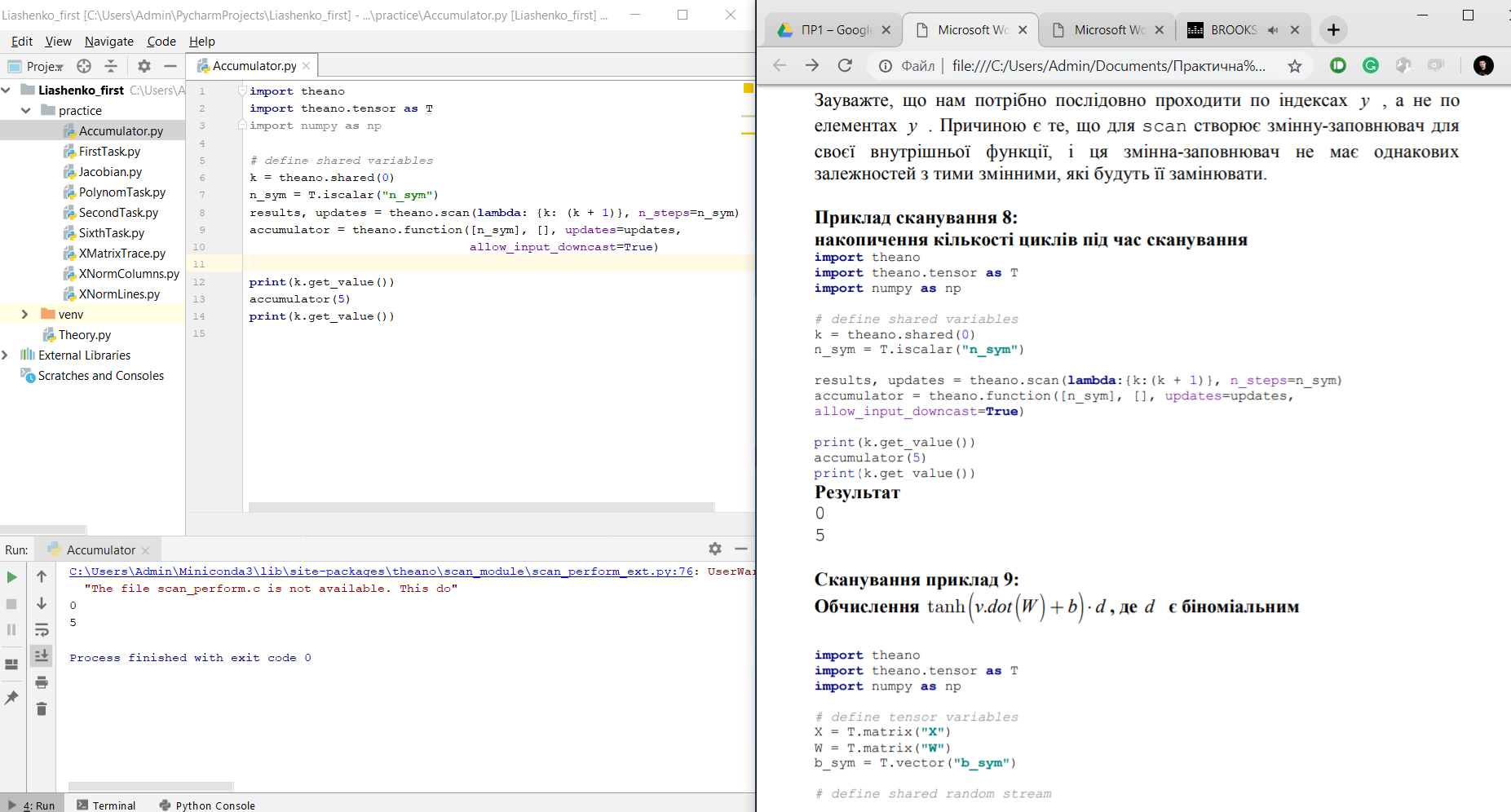
x = np.eye(10, 2, dtype=theano.config.floatX)

w = np.ones((2, 2), dtype=theano.config.floatX)

b = np.ones((2), dtype=theano.config.floatX)

r = compute\_with\_bnoise(x, w, b)

print(r)



Результати:

[

[0.96402758 0. ]

[0. 0.96402758]

[0. 0. ]

[0.76159416 0.76159416]

[0.76159416 0. ]

[0. 0.76159416]

[0. 0.76159416]

[0. 0.76159416]

[0. 0. ]

[0.76159416 0.76159416]

]

### Завдання 9: Обчислення

**import** theano

**import** theano.tensor **as** T

theano.config.warn.subtensor\_merge\_bug = **False**

k = T.iscalar(**"k"**)

A = T.vector(**"A"**)

**def** inner\_fct(prior\_result, B):

**return** prior\_result \* B

*# Symbolic description of the result*

result, updates = theano.scan(fn=inner\_fct,

outputs\_info=T.ones\_like(A),

non\_sequences=A, n\_steps=k)

*# Scan has provided us with A \*\* 1 through A \*\* k. Keep only the last*

*# value. Scan notices this and does not waste memory saving them.*

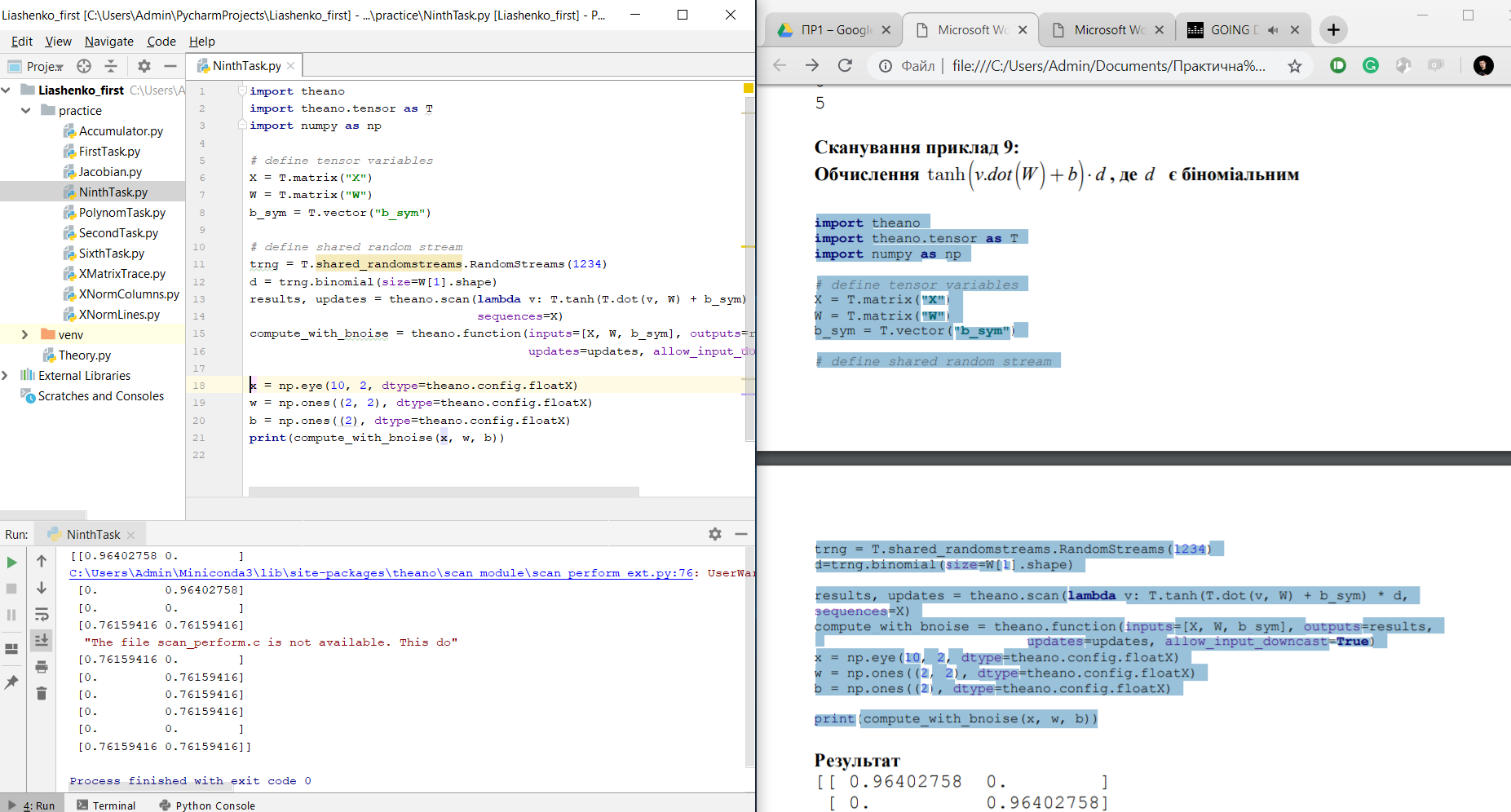
final\_result = result[-1]

power = theano.function(inputs=[A, k], outputs=final\_result,

updates=updates)

r = power(range(10), 2)

print(r)



Результати:

[ 0. 1. 4. 9. 16. 25. 36. 49. 64. 81.]

### Завдання 10: Обчислення полінома

**import** numpy

**import** theano

**import** theano.tensor **as** T

theano.config.warn.subtensor\_merge\_bug = **False**

coefficients = theano.tensor.vector(**"coefficients"**)

x = T.scalar(**"x"**)

max\_coefficients\_supported = 10000

*# Generate the components of the polynomial*

full\_range=theano.tensor.arange(max\_coefficients\_supported)

components, updates = theano.scan(fn=**lambda** coeff, power, free\_var:

coeff \* (free\_var \*\* power),

outputs\_info=**None**,

sequences=[coefficients, full\_range],

non\_sequences=x)

polynomial = components.sum()

calculate\_polynomial = theano.function(inputs=[coefficients, x],

outputs=polynomial)

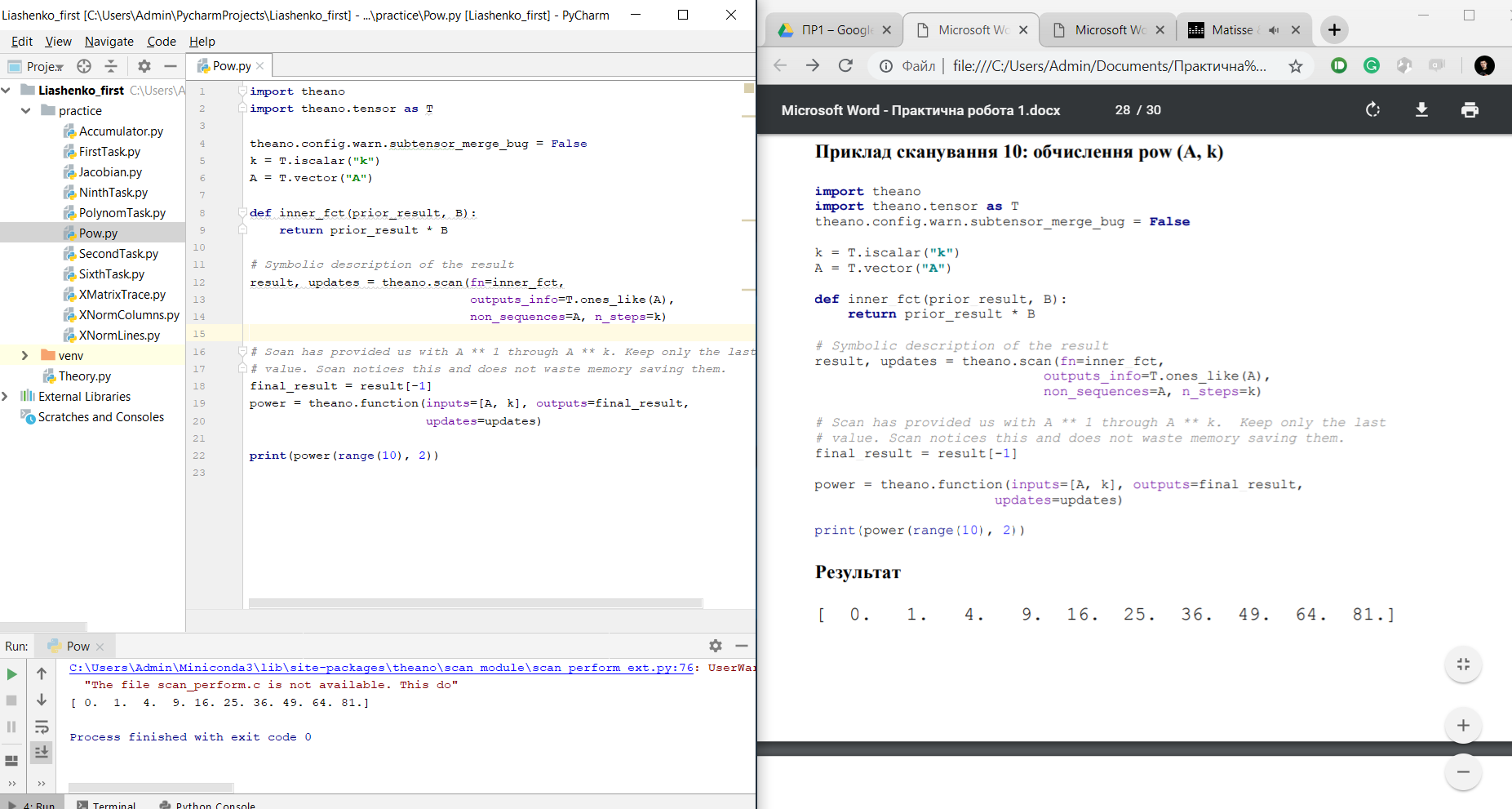
test\_coeff = numpy.asarray([1, 0, 2], dtype=numpy.float32)

r = calculate\_polynomial(test\_coeff, 3)

print(r)

Результати:

19.0



### Висновки

Всі матераіли можна подивитися тут - <https://github.com/maxml/MachineLearning>

Було налаштовано середовище розробки PyCharm. Були налаштовані, використані та розібрані математичні бібліотеки numpy, Theano, порівняні між собою результати деяких алгоритмів, які були релаізовані з допомогою цих бібліотек. В процесі роботи були були розібрені основні дії при вирішенні таких задач:

* додавання двох скалярів
* додавання двох матриць
* обчислення логістичної функції
* обчислення функцій з двома виходами
* використання розподілених змінних
* використання випадкових чисел
* обмін потоків між функціями

На прикладах були показані та вирішені прикладні задачі, які допомогли розібратися та навчитися виконувати кожний підвид вирішуємих задач.