МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота №1

з дисципліни: «Чисельні методи»

16 Варіант

СПЕЦІАЛЬНОСТІ 121 – Інженерія програмного забезпечення

Виконав: Левак О. О.

Група: ІТ-91

Перевірила: Тимофєєва Ю.С.

Київ 2020

**Лабораторна робота №1. Розв’язування нелінійних рівнянь**

**Мета:** навчитися розв’язувати нелінійні рівняння, що мають вигляд f(x) = 0, трьома методами, а саме методи половинного ділення, метод Ньютона та метод простої ітерації і реалізувати їх мовою програмування Python, застосовуючи необхідні бібліотеки.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Чисельне розв’язування нелінійних (алгебраїчних або трансцендентних) рівнянь, що мають вигляд f(x)=0 полягає в тому, щоб знайти значення, що задовольняють (із вказаною точністю) дане рівняння та складається з наступних основних етапів:

1.Виділення(ізоляція, локалізація) коренів рівняння.

2.Уточнення за допомогою деякого обчислювального алгоритму конкретного виділеного кореня із вказаною точністю.

На першому етапі необхідно знайти відрізки з області визначення функції f(x), всередині яких міститься тільки один корінь розв'язуваного рівняння. Іноді обмежуються розглядом лише певної частини області визначення, що викликає інтерес. На цьому етапі використовуються графічні або аналітичні способи.

**Метод половинного ділення:**

Нехай задано рівняння , яке на відрізку [a; b] має єдиний розв'язок, при чому, функція на заданому відрізку є неперервною. Для знаходження шуканого розв'язку розділимо відрізок [a; b] навпіл точкою. Якщо значення функції в даній точці відмінне від нуля (), то можливі два випадки:

1. Функція змінює знак на відрізку [a; c].

2. Функція змінює знак на відрізку [c; b].

Вибираючи той відрізок, на якому функція змінює знак і продовжуючи процес половинного ділення дальше, отримаємо як завгодно малий відрізок, який буде містити корінь рівняння.

**Метод простої ітерації:** Метод простої ітерації (також відомий як метод послідовних наближень) є одним з найбільш важливих способів чисельного розв'язання рівняння. Основна ідея даного методу полягає в тому, що ми замінюємо рівняння рівносильним йому рівнянням. При цьому вважаємо, що функція є неперервною на проміжку. Оберімо, довільним чином, наближене значення кореня і підставимо його в праву частину рівняння.

Підставивши, тепер в праву частину рівняння замість число, отримаємо нове число і так далі продовжуємо даний процес. В результаті отримаємо послідовність чисел. Якщо отримана послідовність збіжна, тобто існує, то переходимо до границі в рівнянні. Тобто границя є коренем рівняння з довільним степенем точності. Також, слід зазначити, що ітераційний процес збігається до єдиного кореня рівняння, якщо на відрізку, який містить корінь, виконується умова, де і збіжність процесу ітерації буде тим швидше, чим менше число , яке задовольняє нерівність. Якщо ж умова не виконується, то потрібно рівняння перетворювати до рівняння таким чином, щоб дана умова виконувалась. Цього можна досягнути, наприклад, шукаючи функцію із співвідношення. Процес ітерації слід продовжувати до тих пір, поки не буде виконуватись умова.

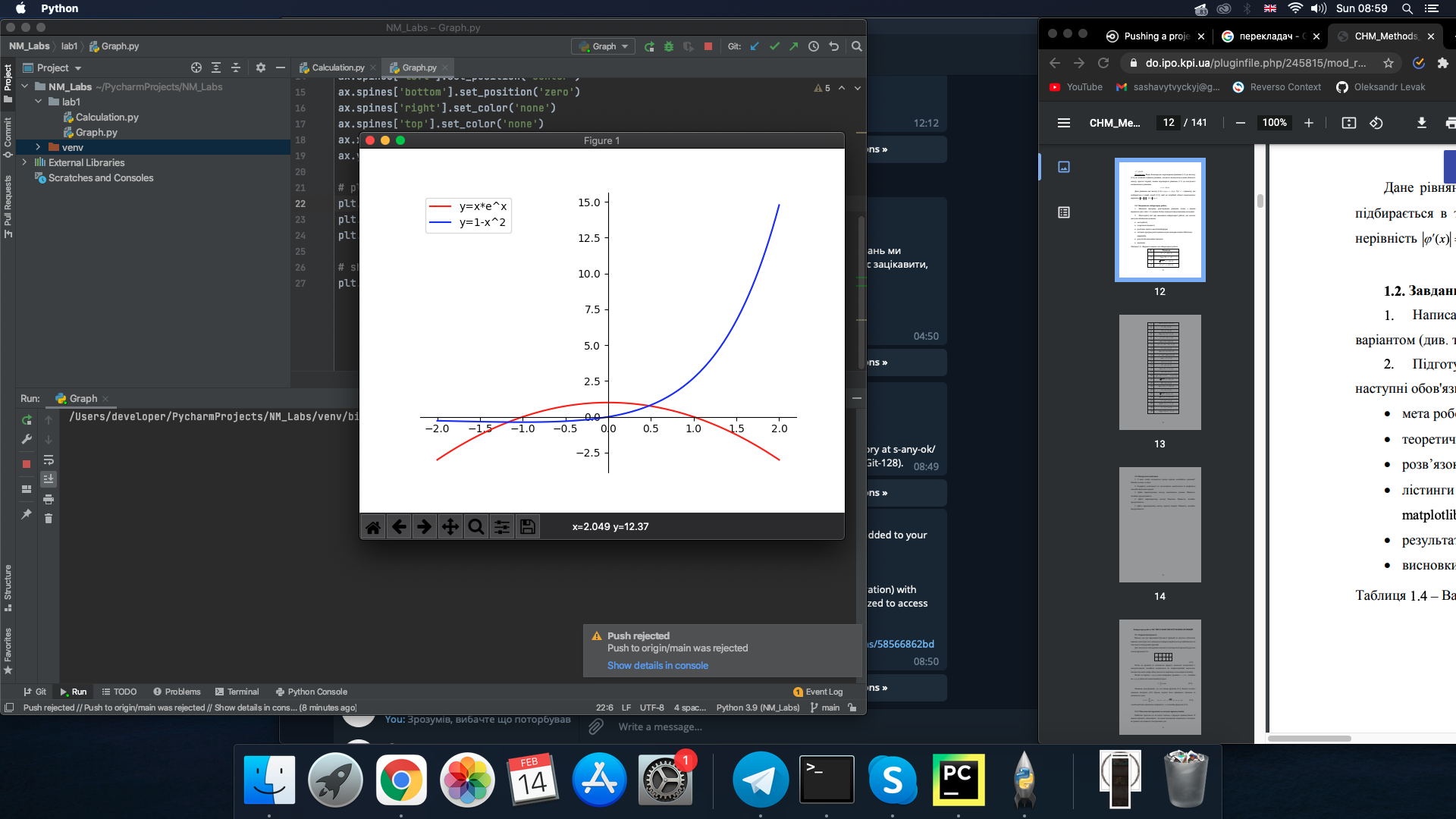
**Метод Ньютона**: припустимо, що розглядається нелінійне рівняння виду, де — функція неперервна на відрізку і має на даному відрізку, відмінні від нуля, похідні першого і другого порядків. Тоді, ідея методу Ньютона полягає в тому, що на кожній ітерації графік функції замінюється дотичною (звідки інша назва цього методу — метод дотичних) і точку перетину кожної з цих дотичних з віссю абсцис приймають за чергове наближення до шуканого кореня. Зазначимо, що перша дотична проводиться через точку — кінець відрізка, для якого виконується умова. В результаті вона перетинає вісь в деякій точці. Далі, обчислюється значення функції і в знайденій точці знову виконується побудова дотичної. Продовжуючи даний процес далі, отримують послідовність значень, яка збігається до точного розв'язку рівняння. Виведемо розрахункові формули методу для рішення нелінійного рівняння. Для цього, на першому кроці, запишемо рівняння дотичної до графіка функції, яку ми будували в точці: вона перетнула вісь в деякій точці. Координати отриманої точки будуть задовольняти рівняння даної дотичної.

Звідси, знаходимо перше наближення до шуканого кореня. На наступному кроці, запишемо рівняння дотичної до графіка функції в точці. Зазначимо, що дана дотична також перетинає вісь в деякій точці. Підставляємо координати отриманої точки в рівняння.

Зауваження: ітераційний процес методу дотичних необхідно продовжувати до тих пір поки модуль різниці між наступним і попереднім наближенням не стане меншим як завгодно малого наперед заданого числа.

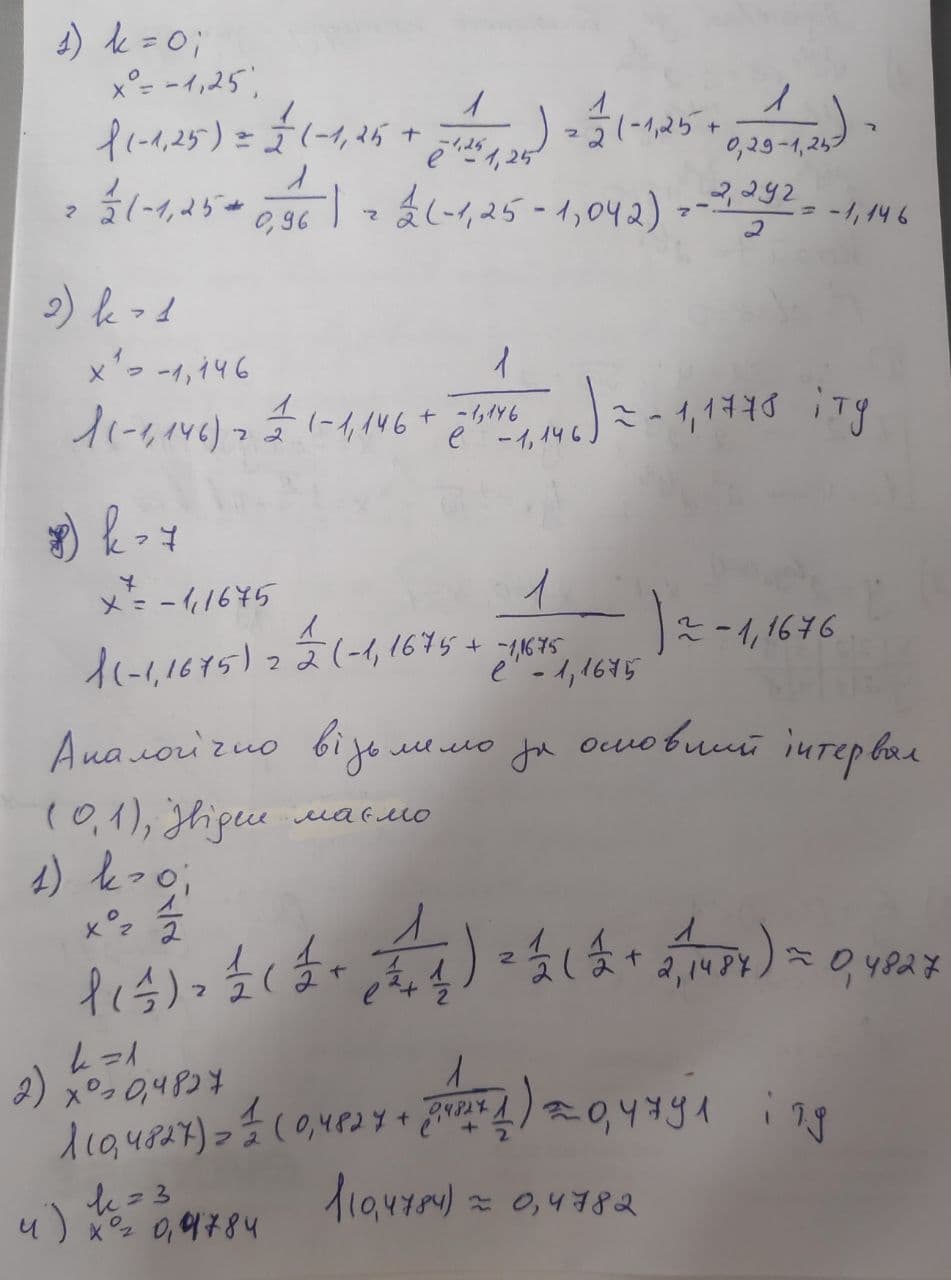
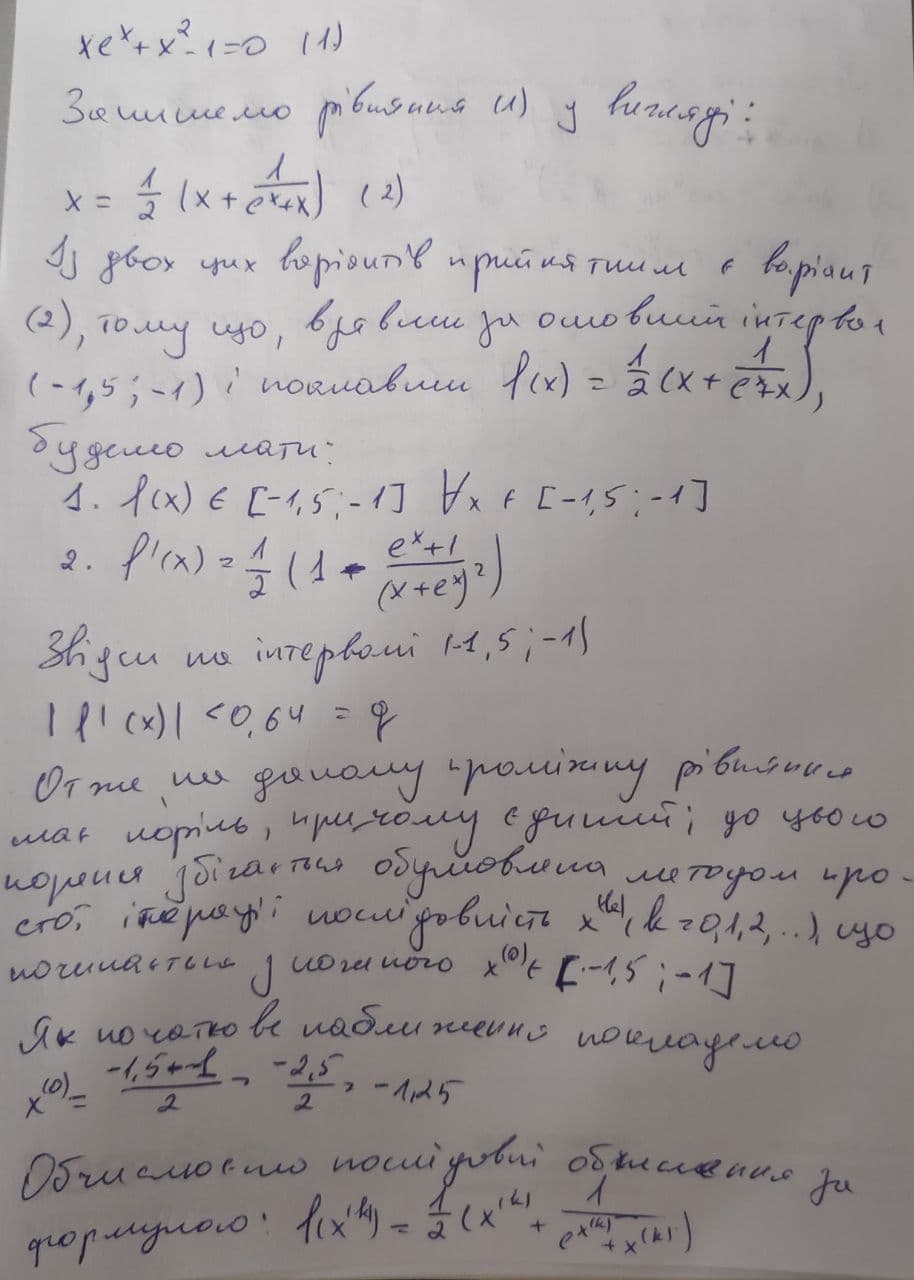
**РОЗВ’ЯЗОК ЗАДАЧІ В АНАЛІТИЧНІЙ ФОРМІ**

Побудувавши графік функції f1(x) = xex та f2(x) = 1-x2 рис(1.1). Визначаємо, що рівняння має 2 корені, що лежать в інтервалі від -1.5 до -1, і від 0 до 1.



**Рисунок 1.1**

Використаємо метод простої ітерації.



**ЛІСТИНГ ПРОГРАМИ РОЗВ’ЯЗУВАННЯ ЗАДАЧІ**

**Github:** <https://github.com/s-any-ok/Numerical-Methods>

**Graph**

**import matplotlib.pyplot as plt**

**import numpy as np**

**# 100 linearly spaced numbers**

**x = np.linspace(-2,2,100)**

**e = 2.718281828459045**

**# the functions**

**y = x\*e\*\*x + x\*\*2 - 1**

**y1 = 1 - x\*\*2**

**y2 = x\*e\*\*x**

**# setting the axes at the centre**

**fig = plt.figure()**

**ax = fig.add\_subplot(1, 1, 1)**

**ax.spines['left'].set\_position('center')**

**ax.spines['bottom'].set\_position('zero')**

**ax.spines['right'].set\_color('none')**

**ax.spines['top'].set\_color('none')**

**ax.xaxis.set\_ticks\_position('bottom')**

**ax.yaxis.set\_ticks\_position('left')**

**# plot the function**

**plt.plot(x,y1, 'r', label='y=x\*e^x')**

**plt.plot(x,y2, 'b', label='y=1-x^2')**

**plt.legend(loc='upper left')**

**# show the plot**

**plt.show()**

**Calculation**

**from sympy import \***

**import numpy as np**

**x = symbols('x')**

**init\_printing(use\_unicode=True)**

**# const**

**e = 2.718281828459045**

**def halfDivisionMethodFn(func, seg):**

**arr = seg.copy()**

**result = 1**

**i = 0;**

**while round(result, 4) != 0.0:**

**arr.append([arr[0], arr[1]])**

**d = (arr[0] + arr[1]) / 2**

**result = func(d)**

**print('k', i)**

**print('a', round(arr[0], 4), 'b', round(arr[1], 4))**

**print('f(a)', round(func(arr[0]), 4), 'f(b)', round(func(arr[1]), 4))**

**print('(a+b)/2', round(d, 4))**

**print('f((a+b)/2)', round(result, 4))**

**print()**

**if func(arr[0]) \* result >= 0:**

**arr[0] = d**

**else:**

**arr[1] = d**

**i = i + 1**

**print('result', round(result, 4))**

**print()**

**def newtonMethodFn(fnSign, func, seg):**

**f1 = diff(fnSign)**

**f2 = diff(f1)**

**fx1 = lambdify(x, f1, 'numpy')**

**fx2 = lambdify(x, f2, 'numpy')**

**if func(seg[1]) \* fx2(seg[1]) > 0:**

**x0 = seg[1]**

**else:**

**x0 = seg[0]**

**print('f``(x1)', round(fx2(x0), 4))**

**print('f(x1)\*f``(x1)', round(func(x0) \* fx2(x0), 4))**

**print('f1', f1)**

**print('f2', f2)**

**print()**

**i = 0;**

**while abs(round(func(x0)/fx1(x0), 4)) != 0.0:**

**print('k', i)**

**print('x1', round(x0, 4))**

**print('f(x1)', round(func(x0), 4))**

**print('f`(x1)', round(fx1(x0), 4))**

**print('-f(x1)/f`(x1)', round(-(func(x0)/fx1(x0)), 4))**

**print()**

**x0 = x0 - (func(x0)/fx1(x0))**

**i = i + 1**

**print('result', round(x0 - (func(x0) / fx1(x0)), 4))**

**print()**

**def simpleIterationMethodFn(fn, seg):**

**x0 = (seg[0]+seg[1])/2**

**i = 0**

**while round(x0, 4) != round(fn(x0), 4):**

**print('k', i)**

**print('x0', round(x0, 4))**

**print('f(x0)', round(fn(x0), 4))**

**print()**

**x0 = fn(x0)**

**i = i + 1**

**print('result', round(fn(x0), 4))**

**print()**

**# def funcTest(x):**

**# return e\*\*(2\*x) + 3\*x - 4**

**#**

**# def funcTest2(x):**

**# return np.log(4-3\*x)/2**

**#**

**# seg3 = [0.4, 0.6]**

**funcSign = x\*exp(x) + x\*\*2 - 1**

**funcSign1 = 1 - x\*\*2**

**funcSign2 = x\*exp(x)**

**def startFunc(x):**

**return x\*e\*\*x + x\*\*2 - 1**

**def startFuncFirstPart(x):**

**return (1/2)\*(x + 1/(e\*\*x + x))**

**# segment1**

**seg1 = [0, 1]**

**seg2 = [-1.5, -1]**

**halfDivisionMethodFn(startFunc, seg2)**

**print('==================================================== \n')**

**newtonMethodFn(funcSign, startFunc, seg2)**

**print('==================================================== \n')**

**simpleIterationMethodFn(startFuncFirstPart, seg2)**

**print()**

**halfDivisionMethodFn(startFunc, seg1)**

**print('==================================================== \n')**

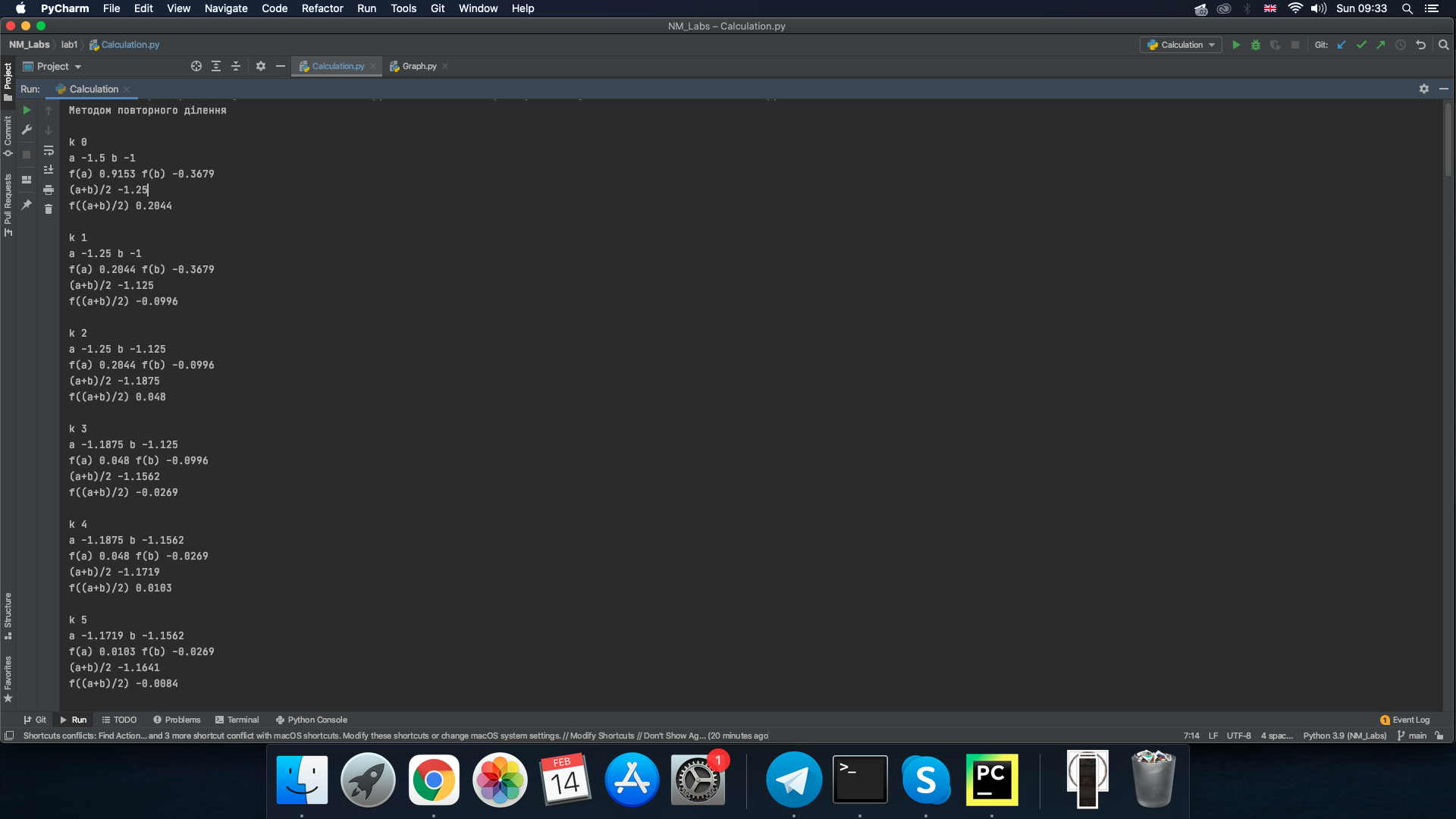
**newtonMethodFn(funcSign, startFunc, seg1)**

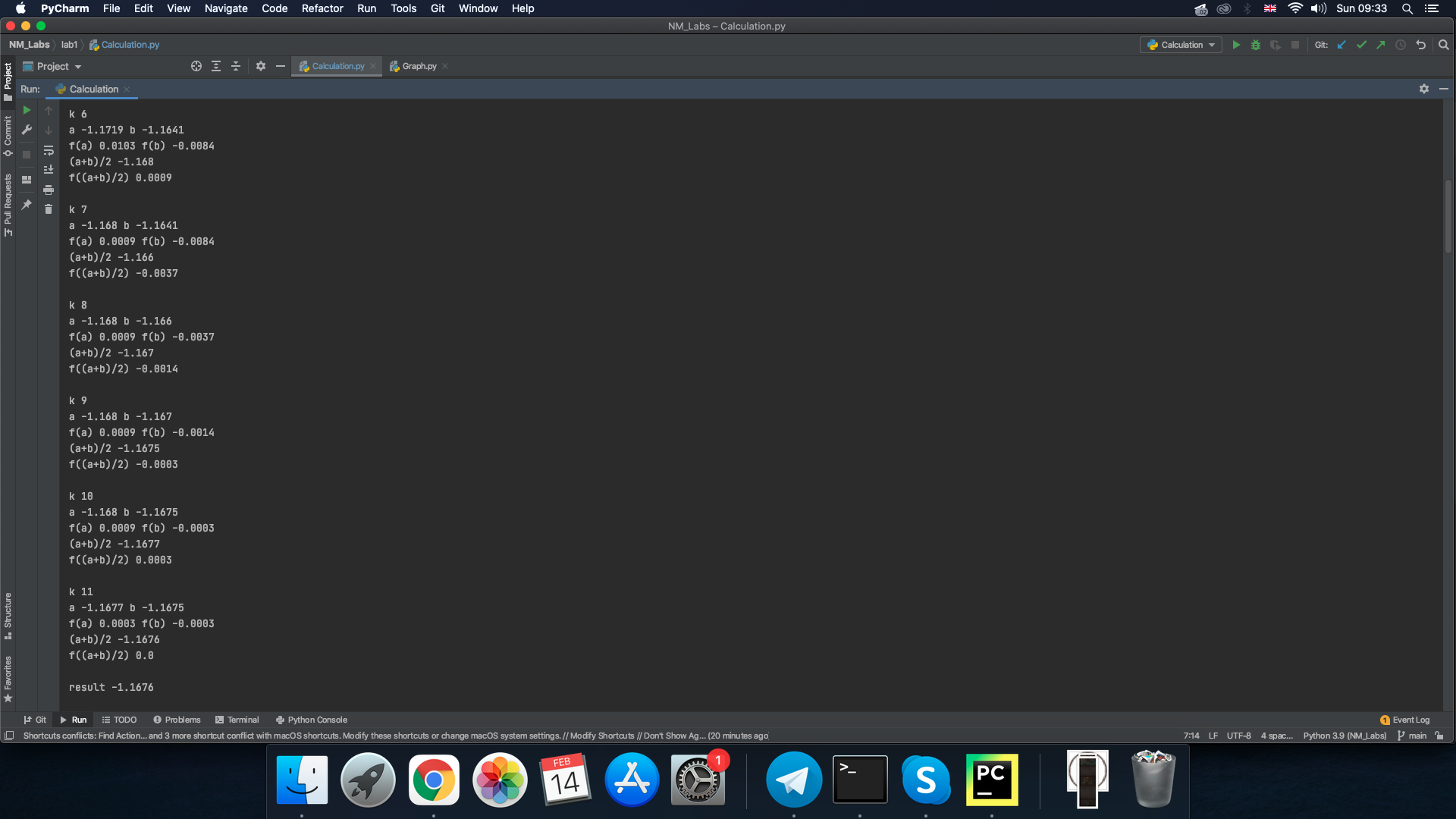
**print('==================================================== \n')**

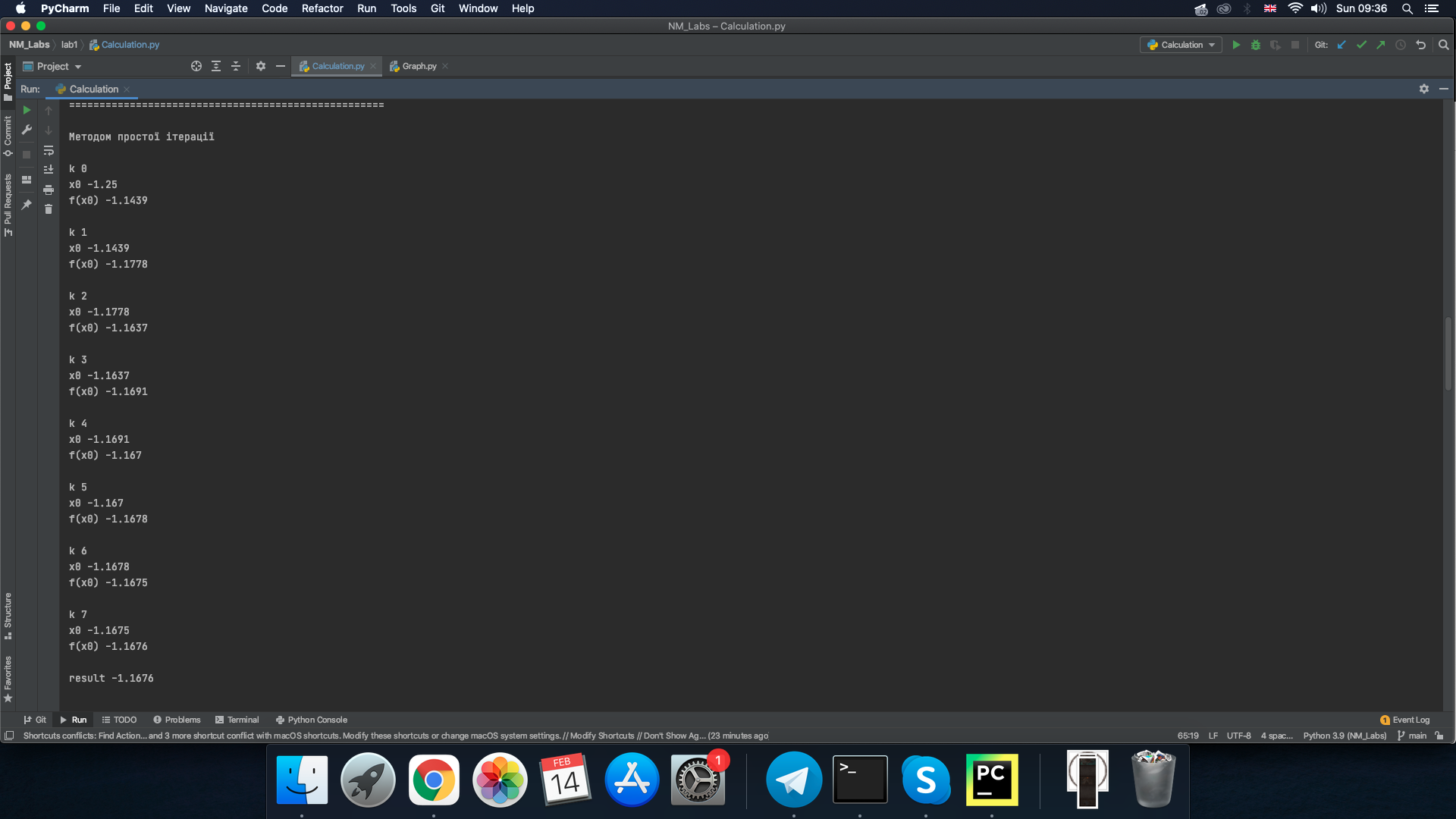
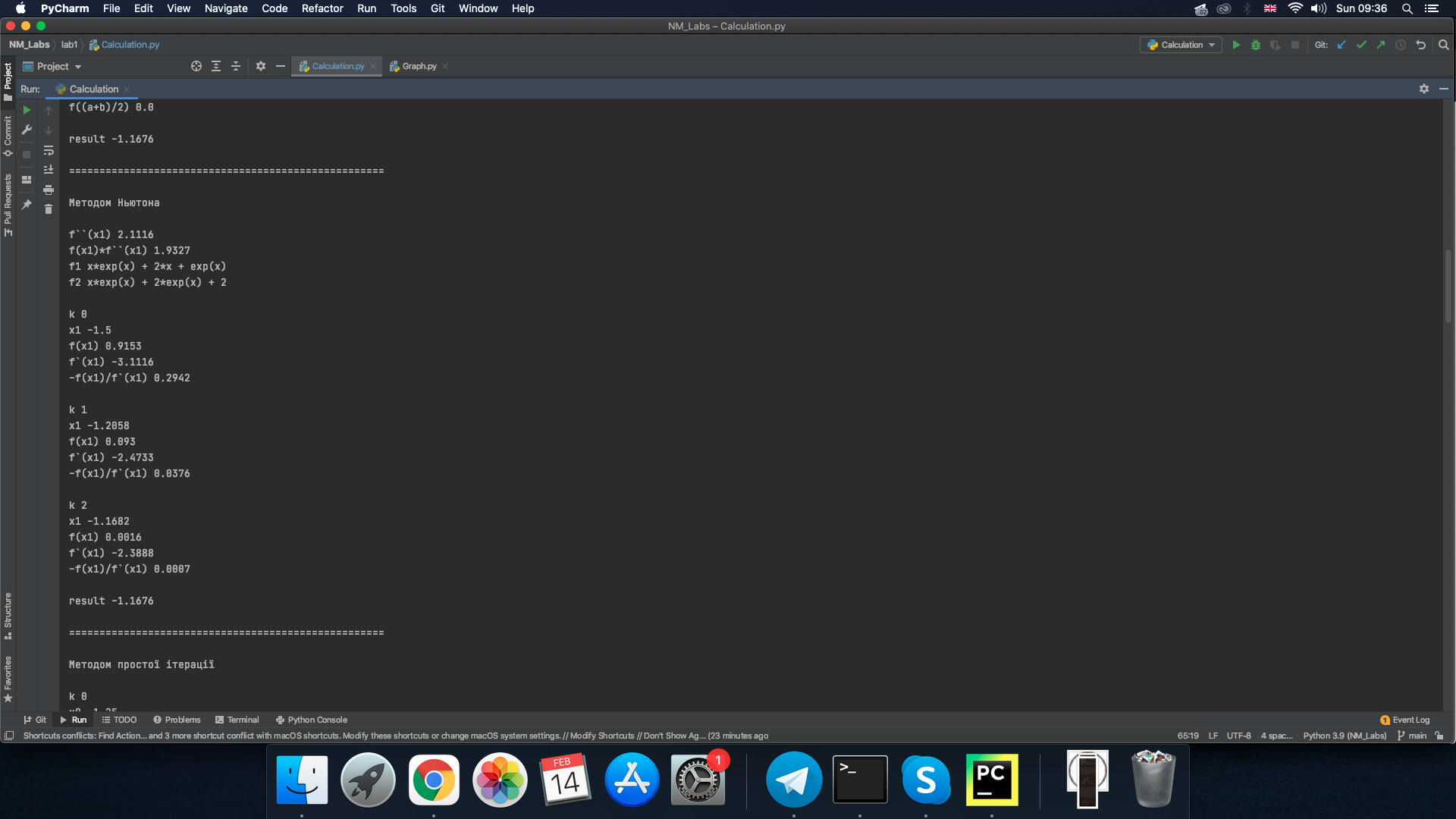
**simpleIterationMethodFn(startFuncFirstPart, seg1)**

**РЕЗУЛЬТАТИ ВИКОНАННЯ ПРОГРАМИ**

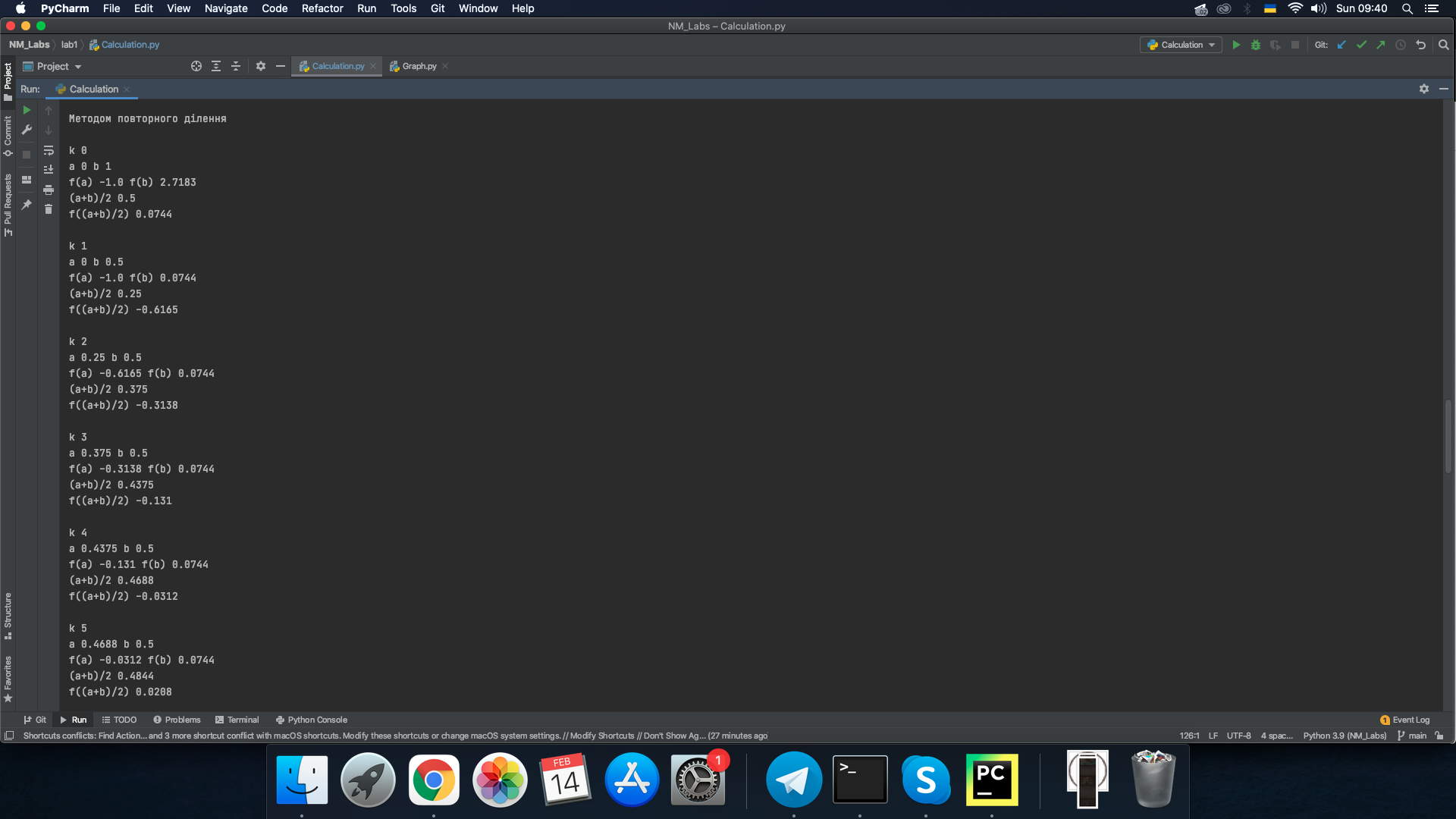
Результати знаходження першого кореня, що лежить в інтервалі від -1.5 до -1

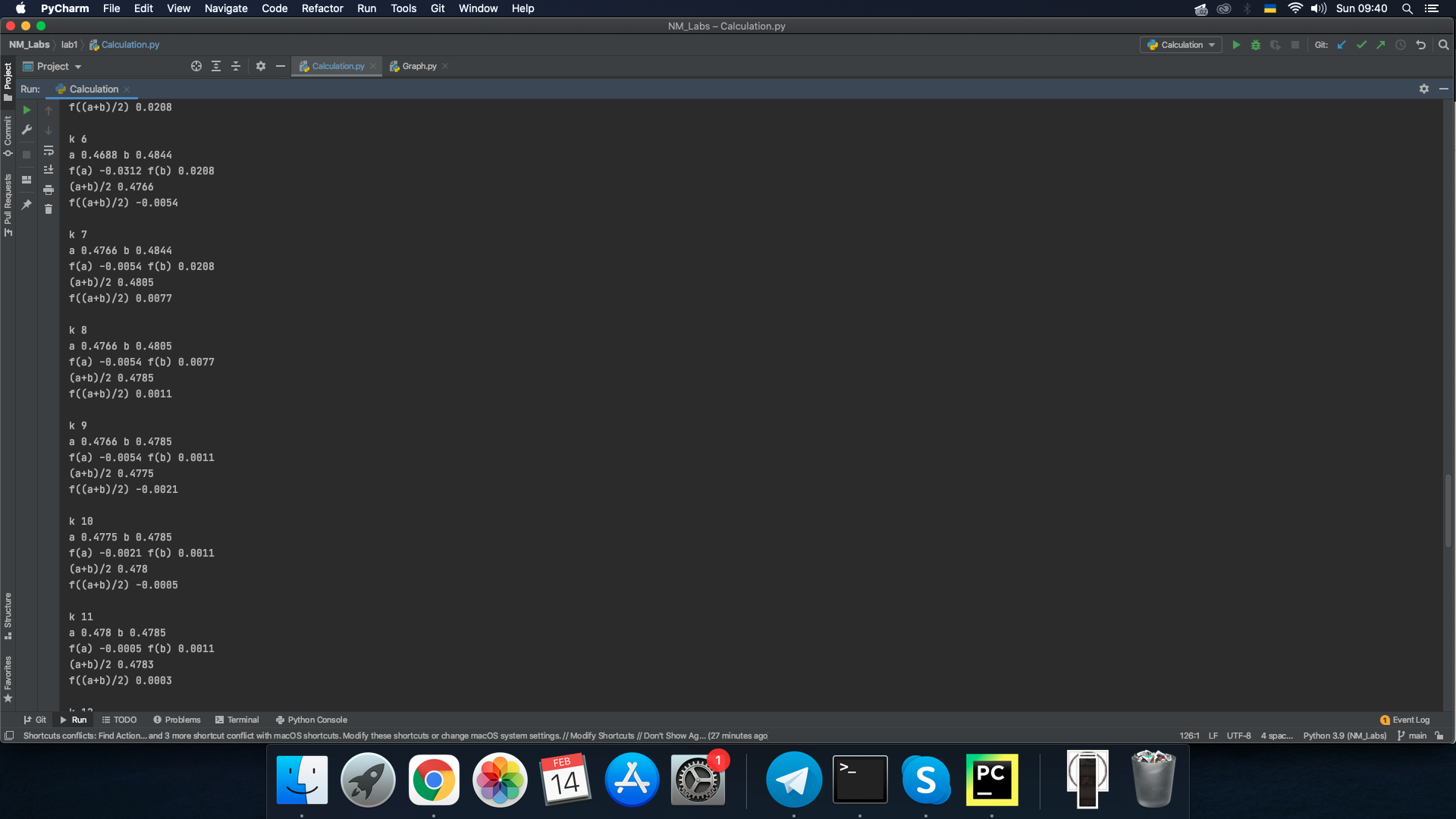


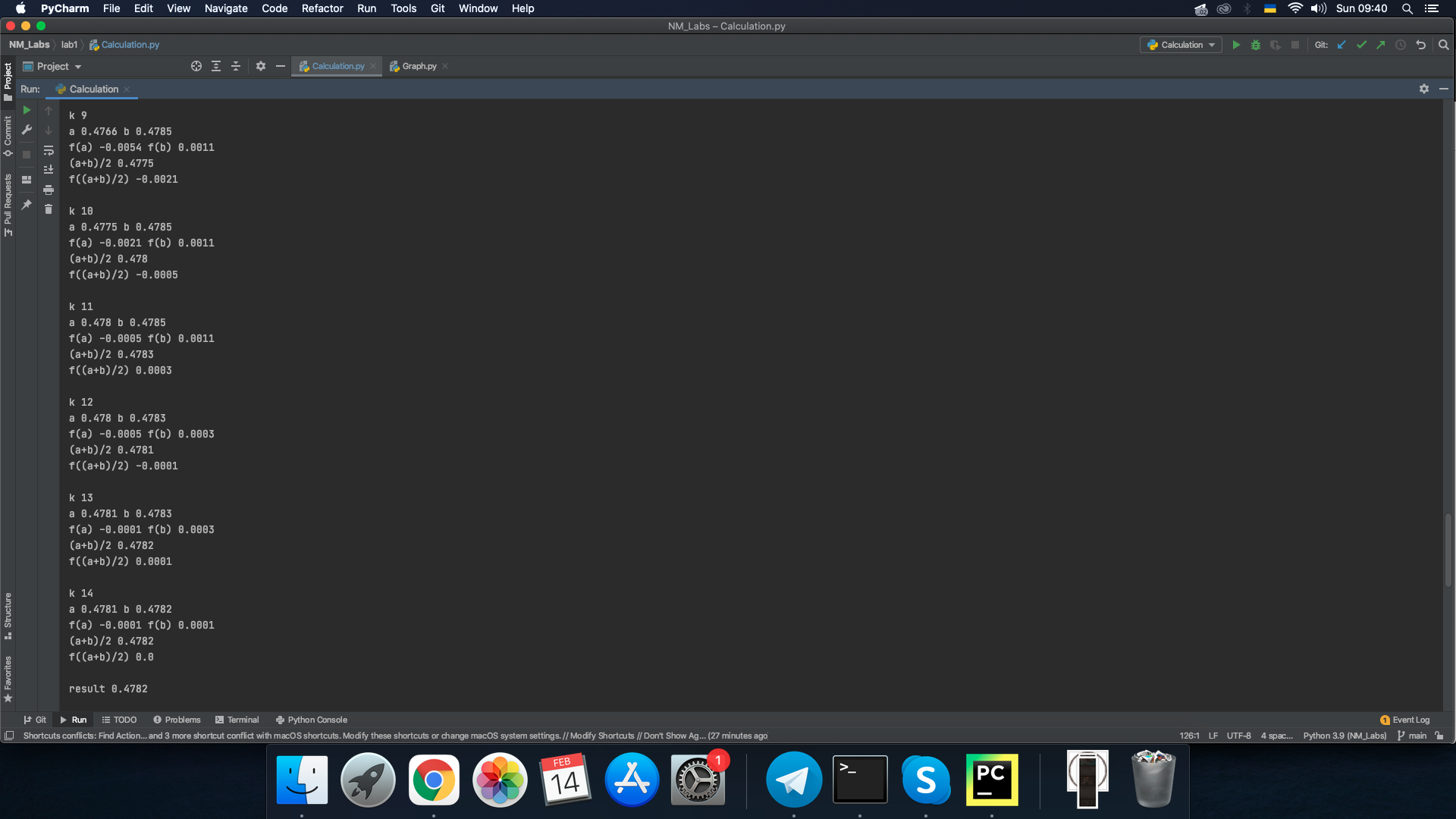
****

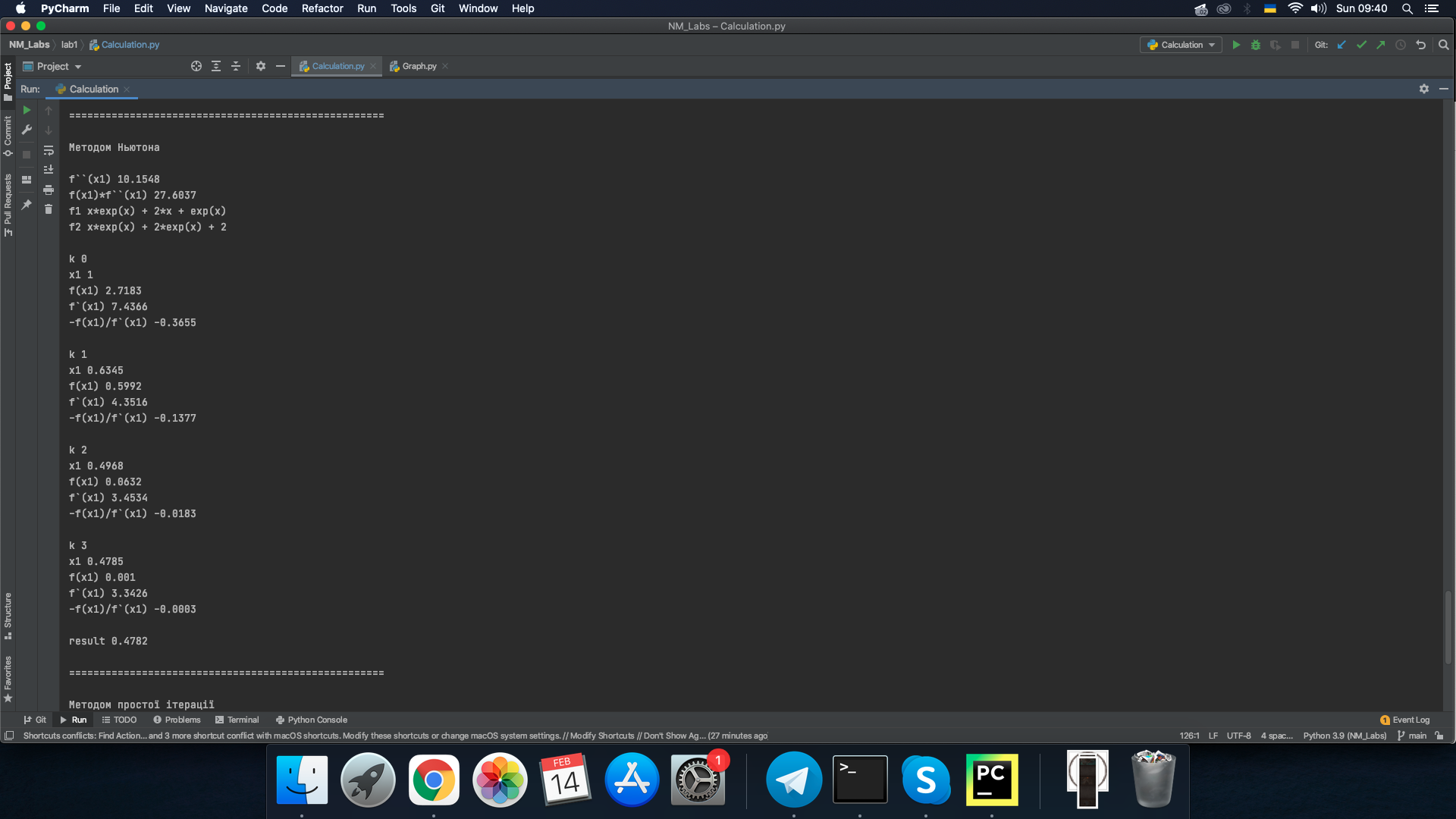
****

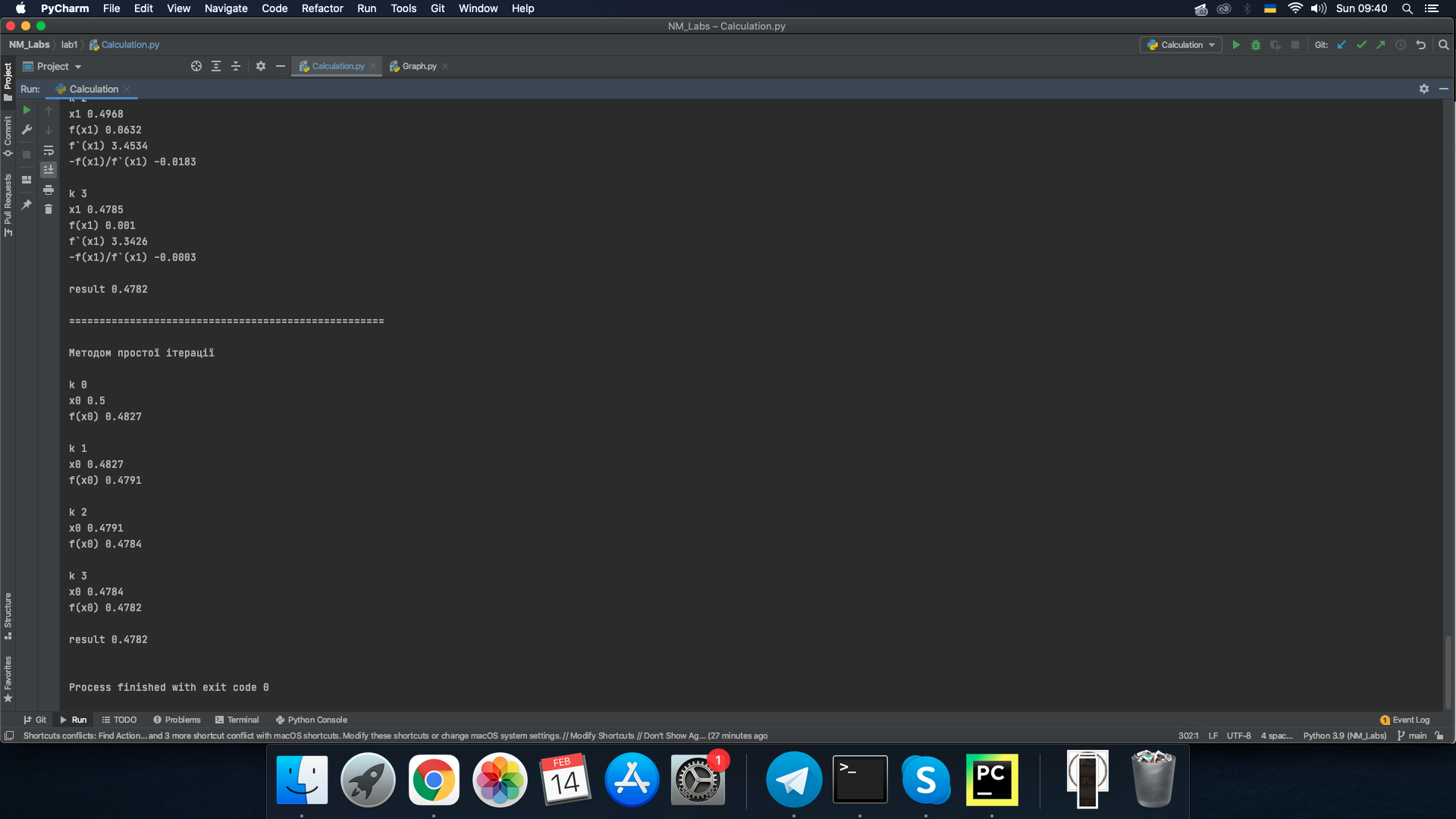
Результати знаходження другого кореня, що лежить в інтервалі від 0 до 1











**Висновки :**

Після виконання лабораторної роботи я навчився розв’язувати нелінійні рівняння трьома методами. 1) Методом повторного ділення, 2) методом Ньютона та 3) методом простої ітерації, а також реалізовувати дані методи за допомогою мови програмування Python.