Bài 1

from collections import deque

def max\_kernel(num\_list, k):

    if not num\_list or k == 0:

        return []

    result = []

    dq = deque()

    for i in range(len(num\_list)):

        if dq and dq[0] < i - k + 1:

            dq.popleft()

        while dq and num\_list[dq[-1]] < num\_list[i]:

            dq.pop()

        dq.append(i)

        if i >= k - 1:

            result.append(num\_list[dq[0]])

    return result

num\_list = [3, 4, 5, 1, -44, 5, 10, 12, 33, 1]

k = 3

print(max\_kernel(num\_list, k))

Bài 2

def count\_chars(string):

    char\_count = {}

    for char in string:

        char\_count[char] = char\_count.get(char, 0) + 1

    return char\_count

print(count\_chars("Happiness"))

print(count\_chars("smiles"))

Bài 3

def word\_count(file\_path):

    word\_freq = {}

    with open(file\_path, 'r', encoding='utf-8') as file:

        for line in file:

            words = line.strip().lower().split()

            for word in words:

                word\_freq[word] = word\_freq.get(word, 0) + 1

    return word\_freq

file\_path = "P1\_data.txt"

print(word\_count(file\_path))

Bài 4

def levenshtein\_distance(s1, s2):

    m, n = len(s1), len(s2)

    dp = [[0] \* (n + 1) for \_ in range(m + 1)]

    for i in range(m + 1):

        for j in range(n + 1):

            if i == 0:

                dp[i][j] = j

            elif j == 0:

                dp[i][j] = i

            elif s1[i - 1] == s2[j - 1]:

                dp[i][j] = dp[i - 1][j - 1]

            else:

                dp[i][j] = 1 + min(dp[i - 1][j],

                                   dp[i][j - 1],

                                   dp[i - 1][j - 1])

    return dp[m][n]

print(levenshtein\_distance("kitten", "sitting"))

print(levenshtein\_distance("yu", "you"))