**Vidzeme University of Applied Sciences**

**Faculty of Engineering**

Introduction to Python and Data Exploration

**group 06**

**practical work #4**

Valmiera, 2024

**Table of Contents**

[1.0 Content 4](#_Toc164695555)

[2.0 Tasks and their solutions 5](#_Toc164695556)

[2.1 The Tasks 5](#_Toc164695557)

[3.0 References 8](#_Toc164695558)

[4.0 Appendix (Code) 9](#_Toc164695559)

[4.1 Task\_4.1\_Gr.06.py 9](#_Toc164695560)

[4.2 Task\_4.2\_Gr.06.py 9](#_Toc164695561)

[4.3 Task\_4.3\_Gr.06.py 10](#_Toc164695562)

|  |  |  |  |
| --- | --- | --- | --- |
| Document versions | | | |
| Version | Status / Changes | Date | Author |
| 0.1 | First version implementation | 18.04.2024 | Kristers Kalniņš |
| 0.2 | Added images of the codes | 22.04.2024 | Kristers Kalniņš |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Contacts and responsible (-s) | | | |
| Name Surname | Department | Position | Contact information (e-mail) |
| Emīls Konrāds | Group #06 | Coordinator | emils.konrads@va.lv |
| Gatis Jurisons | Group #06 | Member | gatis.jurisons@va.lv |
| Kristers Kalniņš | Group #06 | Member | kristers.kalnins@va.lv |
|  |  |  |  |
|  |  |  |  |

# Content

**Goal:** The team (further referenced as team 06) must complete the given task by the client (university professor) described in tasks 1-3 of unit 4.

**Content**: 574 words, three images, three appendixes.

# Tasks and their solutions

Team 06 had organized a meeting after being given the task. The team had started organizing a workflow and work management chain on the next Friday. The team 06 decided to split the workload in 2 different categories:

* Documentation
* Coding

## The Tasks

The practical work consisted of 3 tasks:

* **Task 4.1:** Write a Python program to read mbox-short.txt. For each line starting with "From", split the line and print the second word in alphabetical order. Count and print the total number of "From" lines.
* **Task 4.2:** Write a Python program to read mbox-short.txt. Create a histogram using a dictionary to count messages from each domain name. Sort and print the histogram with domain names and counts.
* **Task 4.3:** Write a Python program to read a file and print the letters in decreasing order of frequency. Convert all input to lowercase and count only the letters a-z. Compare letter frequency between two different languages using tuples. (Osis, Moodle.va.lv, 2024)

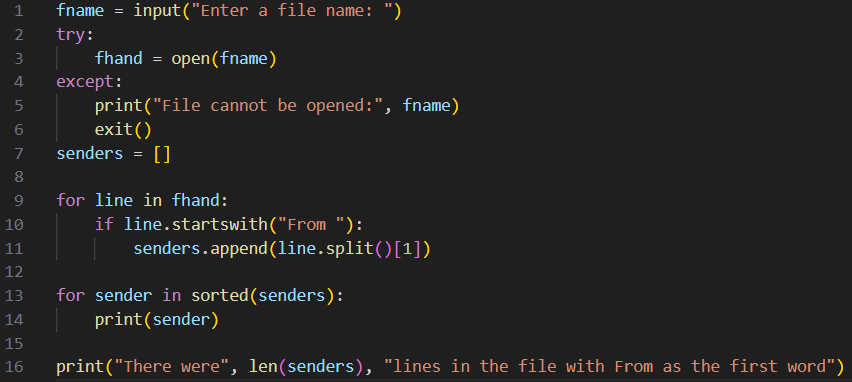


Image: 1.1. Code of the first task (Task\_4.1\_Gr.06.py)

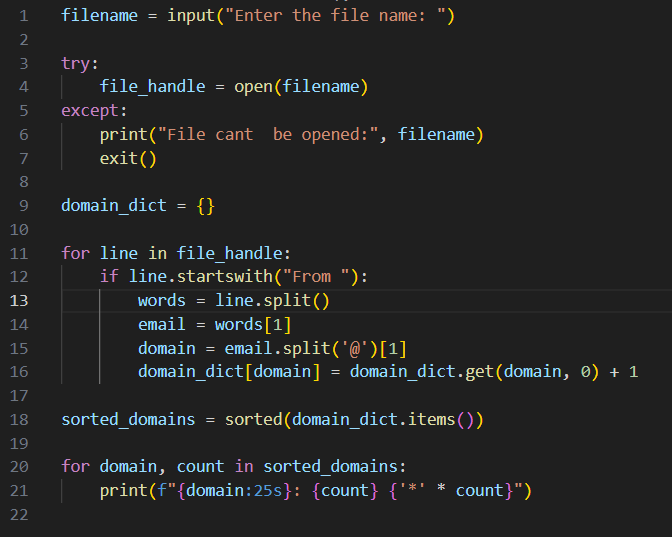


Image: 1.2. Code of the second task (Task\_4.2\_Gr.06.py)

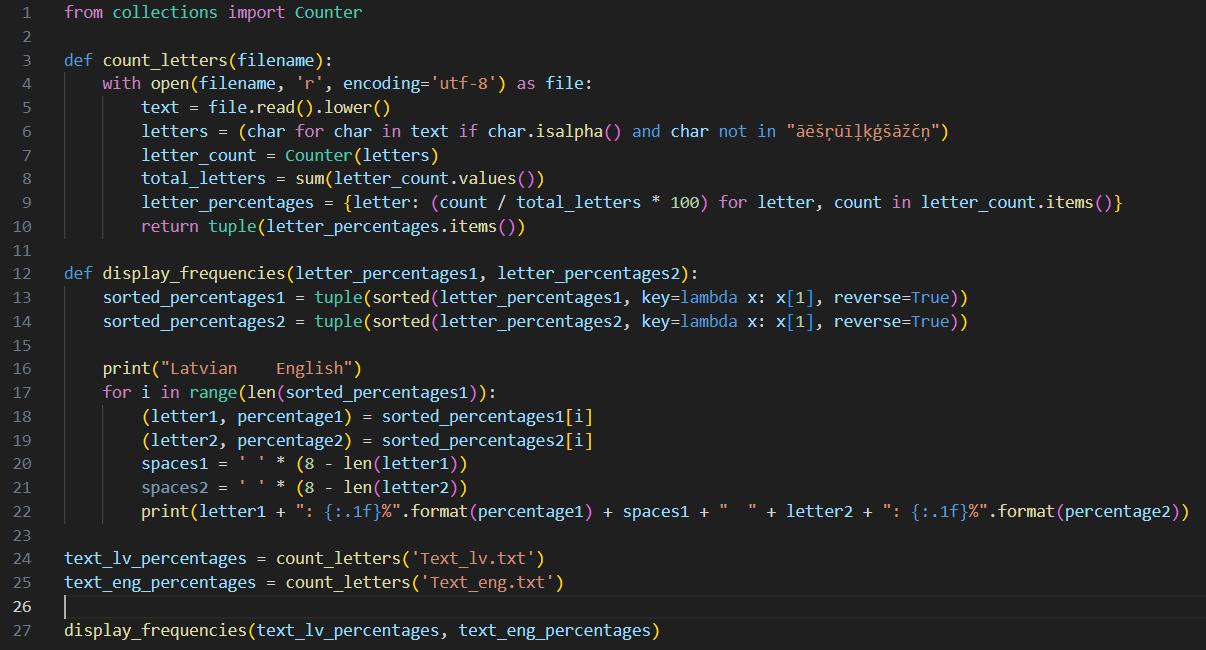


Image: 1.3. Code of the third task (Task\_4.3\_Gr.06.py)

# References

Osis, K. (2024, Mar). *Moodle.va.lv.* Retrieved from

https://moodle.va.lv/pluginfile.php/31788/mod\_resource/content/8/2024\_practical\_work\_4.pdf

Osis, K. (2024, Mar). *Moodle.va.lv.* Retrieved from Template of Document: https://moodle.va.lv/mod/resource/view.php?id=11925

# Appendix (Code)

## Task\_4.1\_Gr.06.py

fname = input("Enter a file name: ")

try:

    fhand = open(fname)

except:

    print("File cannot be opened:", fname)

    exit()

senders = []

for line in fhand:

    if line.startswith("From "):

        senders.append(line.split()[1])

for sender in sorted(senders):

    print(sender)

print("There were", len(senders), "lines in the file with From as the first word")

## Task\_4.2\_Gr.06.py

filename = input("Enter the file name: ")

try:

    file\_handle = open(filename)

except:

    print("File cant  be opened:", filename)

    exit()

domain\_dict = {}

for line in file\_handle:

    if line.startswith("From "):

        words = line.split()

        email = words[1]

        domain = email.split('@')[1]

        domain\_dict[domain] = domain\_dict.get(domain, 0) + 1

sorted\_domains = sorted(domain\_dict.items())

for domain, count in sorted\_domains:

    print(f"{domain:25s}: {count} {'\*' \* count}")

## 4.3 Task\_4.3\_Gr.06.py

from collections import Counter

def count\_letters(filename):

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

        text = file.read().lower()

        letters = (char for char in text if char.isalpha() and char not in "āēšŗūīļķģšāžčņ")

        letter\_count = Counter(letters)

        total\_letters = sum(letter\_count.values())

        letter\_percentages = {letter: (count / total\_letters \* 100) for letter, count in letter\_count.items()}

        return tuple(letter\_percentages.items())

def display\_frequencies(letter\_percentages1, letter\_percentages2):

    sorted\_percentages1 = tuple(sorted(letter\_percentages1, key=lambda x: x[1], reverse=True))

    sorted\_percentages2 = tuple(sorted(letter\_percentages2, key=lambda x: x[1], reverse=True))

    print("Latvian    English")

    for i in range(len(sorted\_percentages1)):

        (letter1, percentage1) = sorted\_percentages1[i]

        (letter2, percentage2) = sorted\_percentages2[i]

        spaces1 = ' ' \* (8 - len(letter1))

        spaces2 = ' ' \* (8 - len(letter2))

        print(letter1 + ": {:.1f}%".format(percentage1) + spaces1 + "  " + letter2 + ": {:.1f}%".format(percentage2))

text\_lv\_percentages = count\_letters('Text\_lv.txt')

text\_eng\_percentages = count\_letters('Text\_eng.txt')

display\_frequencies(text\_lv\_percentages, text\_eng\_percentages)