

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Програмування

Лабораторна робота №7

«Обробка виключень та робота з файлами»

Виконав:
студент групи ІО-32
Крадожон М. Р.
Номер у списку групи: 16
Перевірів:
Пономаренко А. М.

Лабораторна робота №7

Тема: «Обробка виключень та робота з файлами».

Мета: вивчити основні способи роботи з виключеннями. Виключення користувача. Відкриття файлів, зчитування та запис у файл. Шляхи доступу до файлів. Функції, методи та атрибути для роботи з файлами.

Загальне завдання:

1. Вивчити матеріал лекцій 24, 25, 26 та 27.
2. Виконати індивідуальне завдання лабораторної роботи, вибране відповідно до варіанту.

Короткі теоретичні основи:

Відкриття файлу

Функція відкриття має наступний формат:

```
open(<Шлях до файлу>[, mode='r'] [, buffering=-1] [,
encoding=None) [,errors =None] [, newline=None) [,
closefd=True])
```

Необов'язковий параметр mode у функції open() може приймати різні значення:

r – тільки читання (значення за замовчуванням).

w – запис.

b – файл буде відкритий у бінарному режимі. Файлові методи приймають і повертають об'єкти типу bytes.

t – файл буде відкритий у текстовому режимі (за замовчуванням).

Методи для роботи з файлами

- write (<дані>) – записує рядок або послідовність байтів у файл.
- writelines (<Послідовність>) – записує послідовність у файл.
- read([<Кількість>]) – зчитує дані з файлу.
- truncate ([<Кількість>]) – обрізає файл до зазначеної кількості символів (якщо заданий текстовий режим) або байтів (у випадку бінарного режиму).
- seek(<Зсув>[, <Позиція>]) – установлює покажчик у позицію, що має зсув <Зсув> (може бути 0 – початок файлу).

Функція для маніпулювання файлами

```
move(<Шлях до файлу>, <Куди переміщуємо>)
```

Функція переміщує файл у зазначене місце з видаленням початкового файлу.

Функція move () як результат повертає шлях переміщеного файлу.

Перетворення шляху до файлу або каталогу

Перетворити шлях до файлу або каталогу дозволяє наступна функція з модуля `os.path`:

```
abspath ( <Відносний шлях> )
```

Функція перетворить відносний шлях в абсолютний, враховуючи місце розташування поточного робочого каталогу.

Завдання:

Використовуючи функції та методи мови програмування Python:

1. Написати програму створення каталогу зі шляхом та назвою: «C:\lab7\»
2. Написати програму створення підкаталогу «C:\lab7\<прізвище>»
3. Завантажити в даний підкаталог файл *.txt, де * – номер Вашого варіанту лабораторної роботи та виконати з ним дії, що описані в номері Вашого варіанту.
4. Зберегти об'єкти з даними, які створені Вами при виконанні лабораторної роботи №5, у файл, користуючись модулем `pickle`. Створений файл перемістити в попередньо створений каталог «C:\lab5». Зчитати файл, доповнити даними та записати в даний каталог з іншим ім'ям.
5. Зберегти об'єкти з даними, які створені вами при виконанні лабораторної роботи №6, у файл, користуючись модулем `shelve`. Файл перемістити в попередньо створений каталог «C:\lab6». Застосувати три відомі вам методи до модифікації файлу. Вивести на друк модифікований файл.

16	Зчитати файл «16.txt» та перетворити його у файл «161.txt», який складається з речень, кількість слів у яких є найближчою до середньої арифметичної кількості у реченнях даного тексту. На основі файлу «16.txt» створити також файл «162.txt», кожна кирилична буква у якому замінена на букву, номер якої у алфавіті дорівнює номеру даної буква за умови, що відлік відбувається у реверсному порядку.
----	---

Відтепер завдання будуть лише нумеруватися.

Завдання 1:

Код:

```
import os
```

```
path = "C:\\"  
name = "lab7"
```

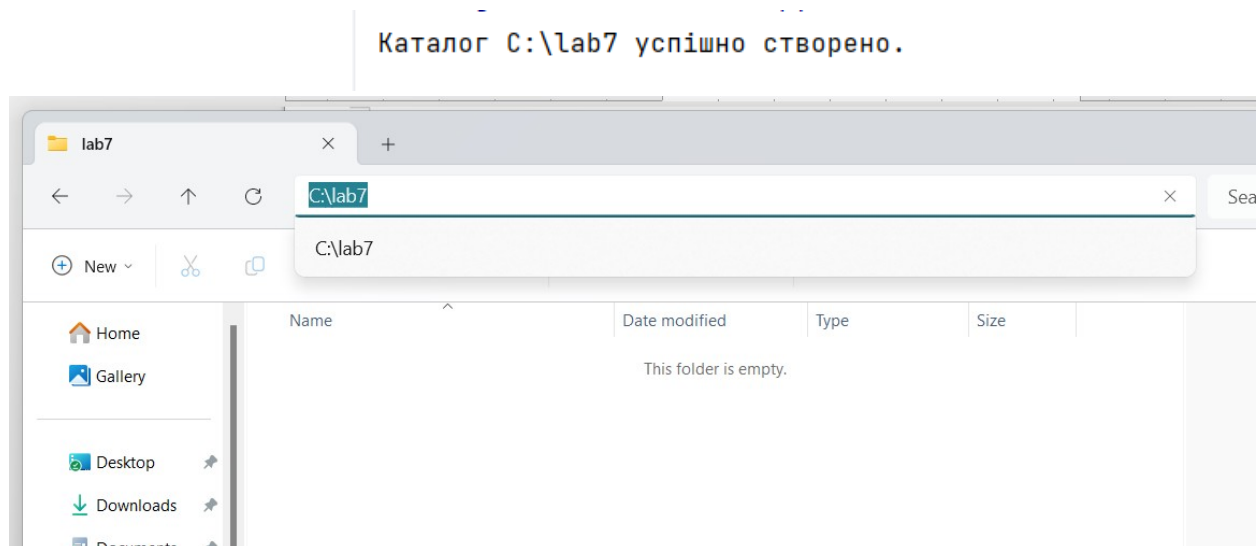
```
# Створюємо повний шлях до каталогу  
full_path = os.path.join(path, name)
```

```

if os.path.exists(full_path):
    print(f"Каталог {full_path} вже існує.")
else:
    os.mkdir(full_path) #Створюється каталог
    print(f"Каталог {full_path} успішно створено.")

```

Скріншоти:



Завдання 2:

Код:

```

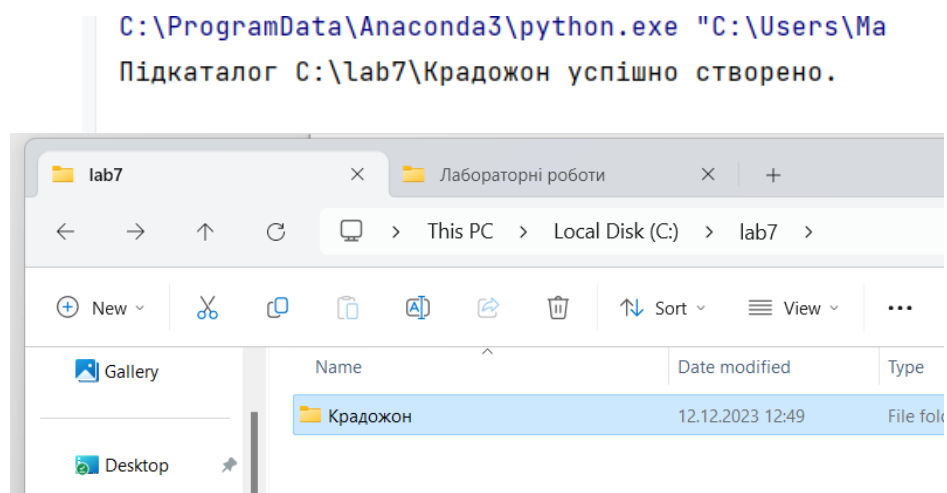
import os

subdir = "C:\\lab7\\Крадожон"

if not os.path.exists(subdir):
    os.makedirs(subdir)
    print(f"Підкаталог {subdir} успішно створено.")
else:
    print(f"Підкаталог {subdir} вже існує.")

```

Скріншоти:



Завдання 4:

Код:

```
import os
import shutil
import pickle

capitals = {"Київ": 3133712, "Варшава": 1863056, "Прага": 1179384,
            "Братислава": 424428, "Кишинів": 532513, "Бухарест": 1926334}

# Відкриваємо файл для запису в бінарному режимі
with open("capitals.pkl", "wb") as f:
    # Зберігаємо словник у файл за допомогою модуля pickle
    pickle.dump(capitals, f)

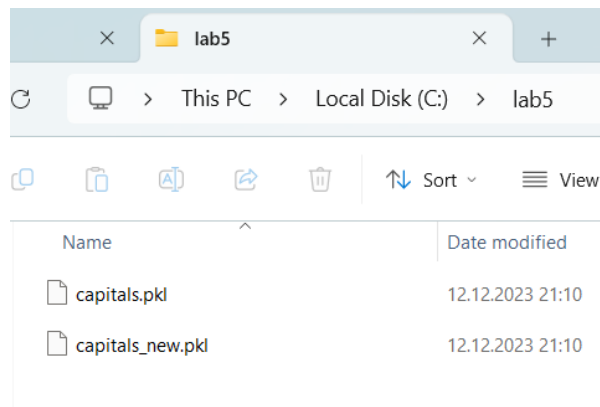
destination_path = "C:\\lab5\\"
shutil.move("capitals.pkl", destination_path)

# Відкриваємо файл для читання в бінарному режимі
with open(destination_path + "capitals.pkl", "rb") as f:
    # Зчитуємо словник з файлу за допомогою модуля pickle
    capitals = pickle.load(f)

# Доповнюємо словник новими даними
capitals["Берлін"] = 3644826
capitals["Париж"] = 2140526

with open(destination_path + "capitals_new.pkl", "wb") as f:
    pickle.dump(capitals, f)
```

Скріншоти:



Завдання 5:

Код:

```
import shelve

# Створюємо клас Countries, який приймає словник країн
class Countries:
    def __init__(self, data):
        self.data = data

# Метод для отримання кількості країн
def count(self):
    return len(self.data)
```

```

# Метод для отримання списку країн за алфавітом
def sort(self):
    return sorted(self.data.keys())

# Метод для отримання інформації про країну за назвою
def info(self, name):
    if name in self.data:
        return self.data[name]
    else:
        return "Немає такої країни"

countries = Countries({
    "Україна": ("Київ", 839, 2966000),
    "Франція": ("Париж", 105.4, 2148000),
    "Німеччина": ("Берлін", 891.8, 3600000),
    "Італія": ("Рим", 1285, 2873000),
    "Іспанія": ("Мадрид", 604.3, 3223000),
    "Польща": ("Варшава", 517.2, 1790000),
    "США": ("Вашингтон", 177, 705749),
    "Канада": ("Оттава", 2778, 934243),
    "Китай": ("Пекін", 16410.54, 21540000),
    "Японія": ("Токіо", 2187.66, 13929286),
    "Індія": ("Нью-Делі", 42.7, 257803),
    "Бразилія": ("Бразилія", 5802, 3015268),
    "Австралія": ("Канберра", 814.2, 397393)
})

shelve_file = shelve.open("C:\\lab6\\countries")
shelve_file["data"] = countries

# Застосовуємо три методи до модифікації файлу
# Додаємо нову країну
shelve_file["data"].data["Норвегія"] = ("Осло", 454, 681067)

# Видаляємо країну
del shelve_file["data"].data["Бразилія"]

# Змінюємо дані про країну
shelve_file["data"].data["Італія"] = ("Рим", 1285, 2865000)

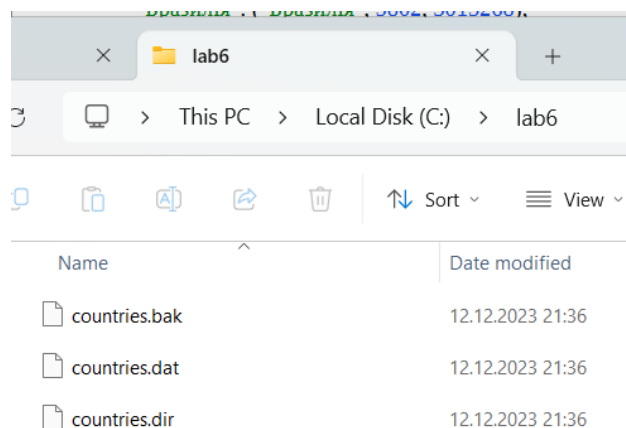
shelve_file.close()

```

Скріншоти:

Завдання 6:

Код:



```
import re
```

```
# Визначаємо клас для обробки файлів
```

```
class FileProcessor:
```

```
# Конструктор класу, приймає назву вхідного файлу
```

```
def __init__(self, input_file):
```

```
    self.input_file = input_file
```

```
# Метод для зчитування тексту з файлу
```

```
def read_text(self):
```

```
    with open(self.input_file, "r", encoding="utf-8") as f:
```

```
        text = f.read()
```

```
    return text
```

```
# Метод для розбиття тексту на речення
```

```
def split_sentences(self, text):
```

```
    # Використовуємо регулярний вираз для пошуку крапок, знаків оклику та питання
```

```
    sentences = re.split(r"[.!?]\s+", text)
```

```
    return sentences
```

```
# Метод для розбиття речення на слова
```

```
def split_words(self, sentence):
```

```
    # Використовуємо регулярний вираз для пошуку слів, що складаються з букв
```

```
    words = re.findall(r"\w+", sentence)
```

```
    return words
```

```
# Метод для обчислення середньої кількості слів у реченнях
```

```
def average_words(self, sentences):
```

```
    total_words = 0
```

```
    for sentence in sentences:
```

```
        # Розбиваємо речення на слова
```

```
        words = self.split_words(sentence)
```

```
        total_words += len(words)
```

```
    average = total_words / len(sentences)
```

```
    return average
```

```
# Метод для створення файлу з реченнями, кількість слів у яких найближча до середньої
```

```
def create_file_161(self, sentences, average):
```

```
    # Відкриваємо файл для запису
```

```
    with open("161.txt", "w", encoding="utf-8") as f:
```

```
        for sentence in sentences:
```

```
            words = self.split_words(sentence)
```

```
            if abs(len(words) - average) <= 1:
```

```
                # Записуємо речення в файл з крапкою в кінці
```

```
                f.write(sentence + ".\n")
```

```
# Метод для створення файлу з кириличними буквами, що замінені на букви в реверсному порядку алфавіту
```

```
def create_file_162(self, text):
```

```
    # Визначаємо кириличний алфавіт в нижньому регістрі
```

```
    alphabet = "абвгдежзиййклмнопрстуфхцчщьюя"
```

```
    # Визначаємо реверсний алфавіт в нижньому регістрі
```

```
    reverse = alphabet[::-1]
```

```
    substitution = dict(zip(alphabet, reverse))
```

```
    modified_text = ""
```

```
    for char in text:
```

```
        # Якщо символ є кириличною буквою в нижньому регістрі
```

```
        if char in alphabet:
```

```
            # Замінюємо його на відповідну букву з реверсного алфавіту
```

```
            modified_text += substitution[char]
```

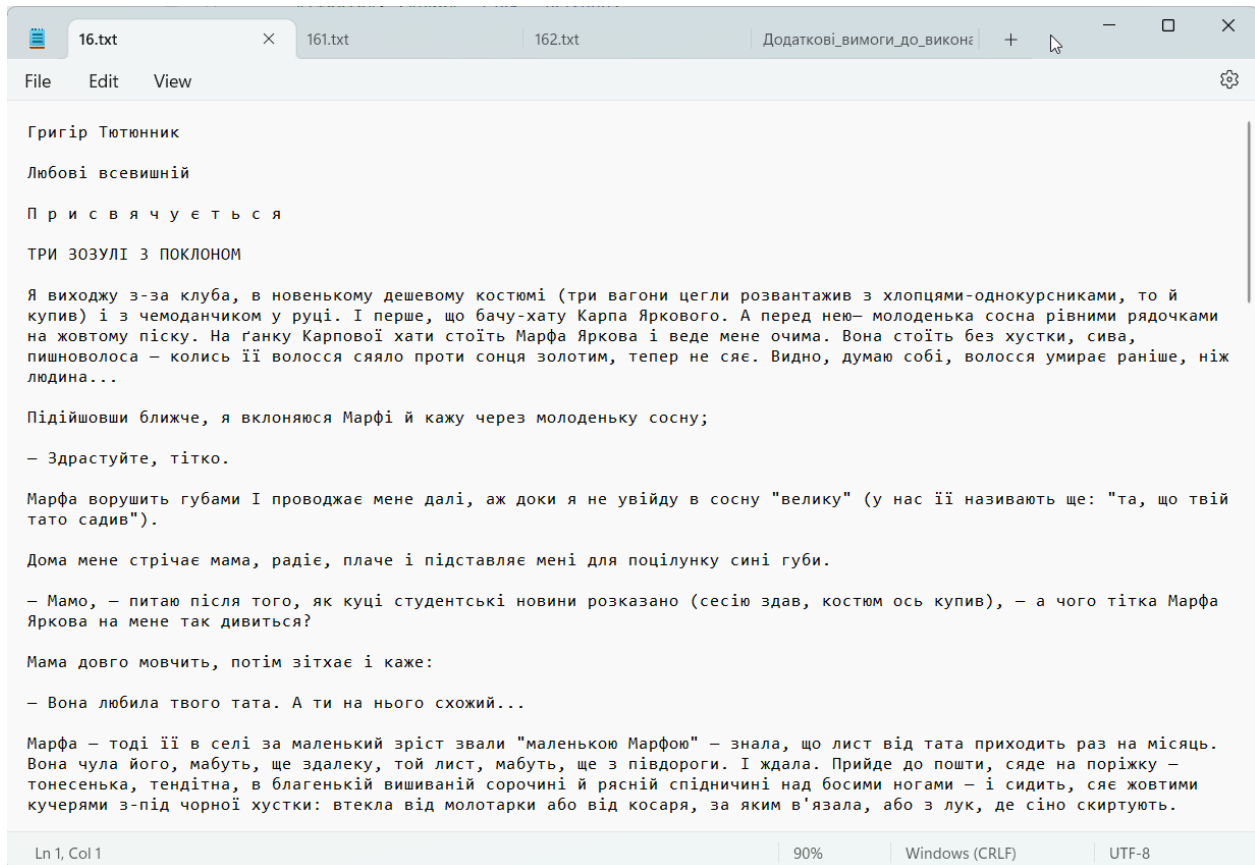
```

# Якщо символ є кириличною буквою в верхньому регістрі
elif char.lower() in alphabet:
    # Замінюємо його на відповідну букву з реверсного алфавіту в верхньому регістрі
    modified_text += substitution[char.lower()].upper()
else:
    # Залишаємо символ без змін
    modified_text += char
# Відкриваємо файл для запису
with open("162.txt", "w", encoding="utf-8") as f:
    f.write(modified_text)

# Створюємо об'єкт класу FileProcessor з назвою вхідного файлу
fp = FileProcessor("16.txt")
text = fp.read_text()
sentences = fp.split_sentences(text)
# Обчислюємо середню кількість слів у реченнях
average = fp.average_words(sentences)
fp.create_file_161(sentences, average)
fp.create_file_162(text)

```

Скрін файла “16.txt”:



```

Григорію Тютюннику
Любові всевишній
П р и с в я ч у є т ь с я
ТРИ ЗОЗУЛІ З ПОКЛОНОМ

Я виходжу з-за клуба, в новенькому дешевому костюмі (три вагони цегли розвантажив з хлопцями-однокурсниками, то й купив) і з чемоданчиком у руці. І перше, що бачу-хату Карпа Яркового. А перед нею- молоденька сосна рівними рядочками на жовтому піску. На ганку Карпової хати стоїть Марфа Яркова і веде мене очима. Вона стоїть без хустки, сива, пишноволоса – колись її волосся сяяло проти сонця золотим, тепер не сяє. Видно, думаю собі, волосся умирає раніше, ніж людина...

Підійшовши ближче, я вклоняюся Марфі й кажу через молоденьку сосну;

– Здрастуйте, тітко.

Марфа ворухить губами і проводить мене далі, аж доки я не увійду в сосну "велику" (у нас її називають ще: "та, що твій тато садив").

Дома мене стрічає мама, радіє, плаче і підставляє мені для поцілунку сині губи.

– Мамо, – питаю після того, як куці студентські новини розказано (сесію здав, костюм ось купив), – а чого тітка Марфа Яркова на мене так дивиться?

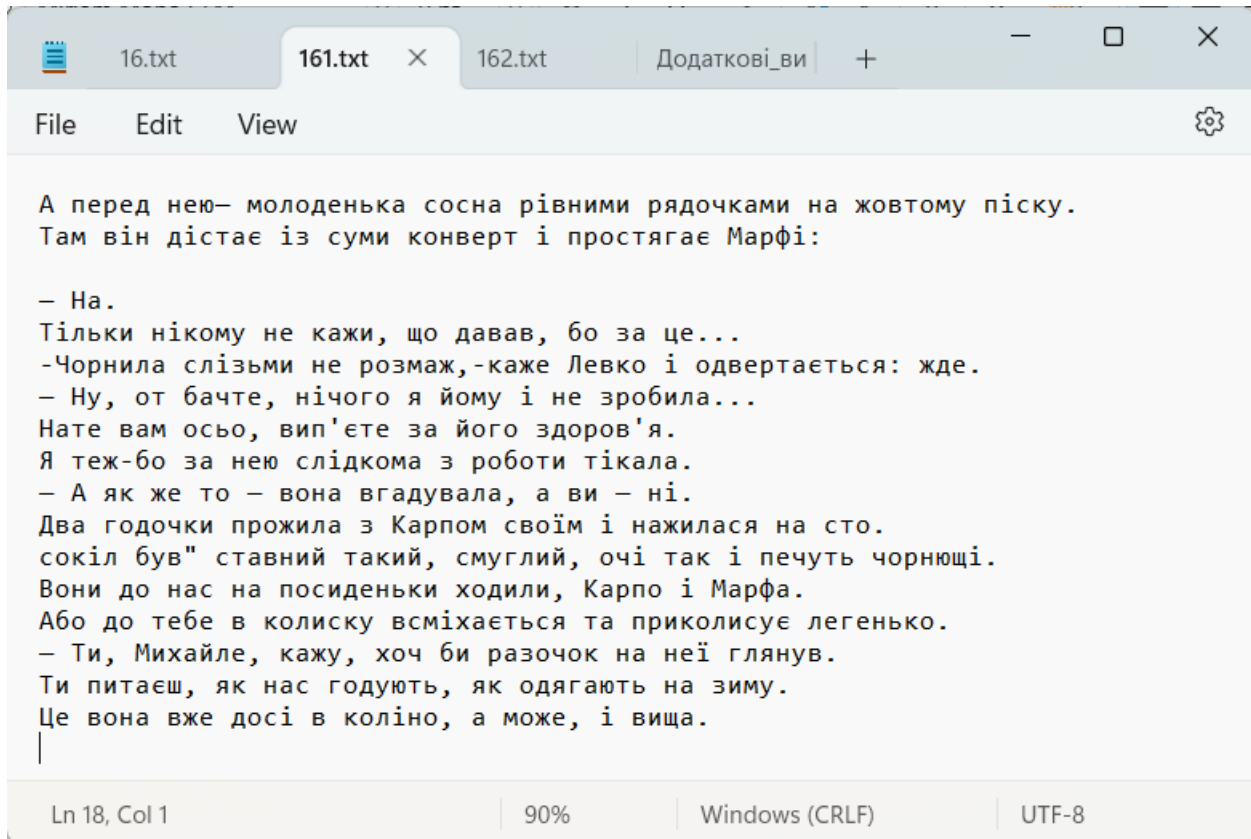
Мама довго мовчить, потім зітхає і каже:

– Вона любила твого тата. А ти на нього схожий...

Марфа – тоді її в селі за маленький зріст звали "маленькою Марфю" – знала, що лист від тата приходить раз на місяць. Вона чула його, мабуть, ще здалеку, той лист, мабуть, ще з півдороги. І ждала. Прийде до пошти, сяде на поріжку – тонесенька, тендітна, в блаженській вишиваній сорочині й рясній спідничині над босими ногами – і сидить, сяє жовтими кучерями з-під чорної хустки: втекла від молотарки або від косаря, за яким в'язала, або з лук, де сіно скиртують.

```


Скрін файла "161.txt":



```
А перед нею— молоденька сосна рівними рядочками на жовтому піску.  
Там він дістає із суми конверт і простягає Марфі:  
  
— На.  
Тільки нікому не кажи, що давав, бо за це...  
-Чорнила слізьми не розмаж,-каже Левко і одвертається: жде.  
— Ну, от бачте, нічого я йому і не зробила...  
Нате вам осьо, вип'єте за його здоров'я.  
Я теж-бо за нею слідкома з роботи тікала.  
— А як же то — вона вгадувала, а ви — ні.  
Два годочки прожила з Карпом своїм і нажилася на сто.  
сокіл був" ставний такий, смуглий, очі так і печуть чорнючі.  
Вони до нас на посиденьки ходили, Карпо і Марфа.  
Або до тебе в колиску всміхається та приколисує легенько.  
— Ти, Михайле, кажу, хоч би разочок на неї глянув.  
Ти питаєш, як нас годують, як одягають на зиму.  
Це вона вже досі в коліно, а може, і вища.  
|
```

Скрін файла "162.txt":



```
Щітсї Ибиблто  
Нбюкьс Ыцѣтглсп  
Й і т і ь а д з х и в і а  
иІТ УКУЗНС з Йконклкм  
  
А ьтекчфз у-уя онэмя, ь лкьцлвокмз чѣгцькмз окіибмс (иіт ьашклт ещнт ікуьялияфть у енкийеамт-кчлкозіілтоямт, ик п озйть) с  
у дцмкчялдокм з ізес. С йцігц, гк юадз-еяиз Ояїя Аіокькшк. Я йціцч лцб— мкнкчлвоа ікіля ісьлтмт іачкдоямт ля фкьикмз  
йсіоз. Ля шялоз Ояїйкькр еяит іікрив Мяїжа Аіокьс с ьццч мцлц кдтмя. Ькля іікрив юцу езііот, ітья, йтглкькнкія — окнтів рр  
ькнкііа іаанк йікит іклеа ункнитм, иційці лц іах. Ьтчлк, чэмяб ікюс, ькнкііа эмтіях іялсгц, лсф нбчля...  
  
Йсчспгкьгт юнтфдц, а ьонклабіа Мяїжс п ояфз дціцу мкнкчлвлоз ікілз;  
  
— Учїяіизпиц, исиок.  
  
Мяїжа ькїзгтив щэямт С йїкькчфях мцлц чянс, яф чкот а лц зьспчз ь ікілз "ьцнтоз" (з ляі рр ляутьябив гц: "ия, гк иьсп  
ияик іячть").  
  
Чкмь мцлц ііісдях мямя, іячсх, йнядц с йсчіяьнах мцлс чна йкеснзлос ітлс щэют.  
  
— Мямк, — йтияб йсіна икшк, ао озес іизчцліivos лкьтлт ікуоуаялк (іісіб учяь, окіибм ків озйть), — я дкшк исиоя Мяїжа  
Аіокьс ля мцлц ияо чтьтивіа?  
  
Мямя чкьшк мкьдтив, йкисм усиеях с ояфц:  
  
— Ькля нбютня иькшк иия. Я ит ля лвкшк іекфтп...  
  
Мяїжа — икчс рр ь іцнс уя мянцлвотп уїсіи уьянт "мянцлвокб Мяїжкб" — уляня, гк нтіи ьсч иия йїтекчтив іяу ля мсіаев. Ькля  
дзня пкшк, мяюзив, гц учянцоз, икп нтіи, мяюзив, гц у йсьчкїкшт. С фчяня. Йїтпчч чк йкгит, іачч ля йкїсфоз — иклцілвоа,  
ицлчсила, ь юнящлвосп ьтгтьялсп ікїкдтлс п іаілсп ійсчлтдтлс ляч юкітмт лкшямт — с ітчтив, іах фкьитмт оздціамт у-йсч  
дкїлкр езііот: ьицоня ьсч мкнкияіот яюк ьсч окіяіа, уя аотм ь'ауяня, яюк у нзо, чч іслк іотїизбив.
```

Висновок: Виконавши цю лабораторну роботу, я зміг здобути відповідні навички в роботі зі обробці виключень та робота з файлами в Python. Під час виконання лабораторної роботи проблем не виникало, а складність була в структуруванні коду та приведенні його до більш гарного вигляду.