МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІКТА

Кафедра ЗІ

Звіт

до лабораторної роботи №8

з курсу: «Програмування скриптовими мовами» на тему: «Робота з файлами та обробка винятків»

Виконав:

студент групи КБ-305

Семенчук А.А.

Прийняв:

к.т.н., доцент

Совин Я. Р.

Львів 2024

Мета роботи – ознайомитись з можливостями і застосуванням функцій у мові Python.

Завдання

Задано каталог folder_1 з підкаталогами folder_2 і folder_3 (каталог є в матеріалах до лабораторної роботи). Написати програму маніпулювання з файлами і каталогами відповідно до завдання в табл. 1. Операції:

- **Вивести вміст каталогу `folder_1` і його підкаталогів.**
- **Створити в каталозі `folder_1` підкаталог `folder_4`.**
- **Скопіювати в підкаталог `folder_4` всі файли з каталогу `folder_2` з розширенням `.pdf`.**
- **Перемістити в підкаталог `folder_4` всі файли з каталогу `folder_3` з розширенням `.txt`.**
- **Перейменувати в каталозі `folder_1` всі файли з розширенням `.txt` в ім'я `файлу_2024.txt`**
  (Наприклад, файл `file_1.txt` має стати `file_1_2024.txt`)
- **Видалити в підкаталозі `folder_3` всі файли з розширенням `.doc`.**
- **Видалити підкаталог `folder_2`.**

2. Програму з лабораторної роботи № 7 доповнити таким чином, щоб список при запуску програми завантажувався з текстового файлу, а при виході зберігався у текстовий файл. Додати пункти меню і відповідні функції для таких операцій: "Зберегти список у текстовий файл", "Зберегти список у файл як об'єкт", "Завантажити список з текстового файлу", "Завантажити список як об'єкт". Також доповнити програму обробкою винятків, що виникають при неправильно введених даних, неправильно заданих індексах елементів, файлових помилках. Програма повинна коректно працювати при будь-яких діях користувача.

3. Програму валідації введеного паролю з Лабораторної роботи №7 доповнити додатковою перевіркою пароля: перевіряється чи є пароль у заданому текстовому файлі згідно табл. 2 – якщо так, то виводиться повідомлення, що пароль слабкий і валідація не проходить. Файли з паролями є в матеріалах до лабораторної роботи.

27.	10-million-password-list-top-1000000.txt

Виконання лабораторної роботи

1.

```python
import os
import shutil

# Довідник для каталогів
base_dir = 'folder_1'
folder_2 = os.path.join(base_dir, 'folder_2')
folder_3 = os.path.join(base_dir, 'folder_3')
```

```python
folder_4 = os.path.join(base_dir, 'folder_4')

# 1. Вивести вміст каталогу folder_1 і його підкаталогів.
def list_contents(directory: str) -> None:
    for root, dirs, files in os.walk(directory):
        print(f"Directory: {root}")
        for dir_name in dirs:
            print(f"  [DIR] {dir_name}")
        for file_name in files:
            print(f"  [FILE] {file_name}")

# 2. Створити в каталогу folder_1 підкаталог folder_4.
def create_folder(directory: str) -> None:
    os.makedirs(directory, exist_ok=True)
    print(f"Created folder: {directory}")

# 6. Скопіювати в підкаталог folder_4 всі файли з каталогу folder_2 з розширенням
*.pdf.
def copy_files_with_extension(source_folder: str, target_folder: str, extension:
str) -> None:
    for filename in os.listdir(source_folder):
        if filename.endswith(extension):
            shutil.copy(os.path.join(source_folder, filename), target_folder)
            print(f"Copied {filename} to {target_folder}")

# 18. Перемістити в підкаталог folder_4 всі файли з каталогу folder_3 з розширенням
*.txt.
def move_files_with_extension(source_folder: str, target_folder: str, extension:
str) -> None:
    for filename in os.listdir(source_folder):
        if filename.endswith(extension):
            shutil.move(os.path.join(source_folder, filename), target_folder)
            print(f"Moved {filename} to {target_folder}")

# 23. Перейменувати в каталогу folder_1 всі файли з розширенням *.txt в ім’я
файлу_2024.txt
def rename_files_in_folder(directory: str, extension: str, suffix: str) -> None:
    for filename in os.listdir(directory):
        if filename.endswith(extension):
            name, _ = os.path.splitext(filename)
            new_name = f"{name}_{suffix}{extension}"
            os.rename(os.path.join(directory, filename), os.path.join(directory,
new_name))
            print(f"Renamed {filename} to {new_name}")

# 52. Видалити в підкаталогу folder_3 всі файли з розширенням .doc.
def delete_files_with_extension(directory: str, extension: str) -> None:
    for filename in os.listdir(directory):
        if filename.endswith(extension):
            os.remove(os.path.join(directory, filename))
            print(f"Deleted {filename}")

# 56. Видалити підкаталог folder_2.
```

```python
def delete_folder(directory: str) -> None:
    shutil.rmtree(directory, ignore_errors=True)
    print(f"Deleted folder: {directory}")

# Виконання операцій
print("1. Вміст каталогу folder_1 та його підкаталогів:")
list_contents(base_dir)
print("\n2. Створення каталогу folder_4:")
create_folder(folder_4)
print("\n6. Копіювання всіх файлів з розширенням *.pdf з folder_2 до folder_4:")
copy_files_with_extension(folder_2, folder_4, '.pdf')
print("\n18. Переміщення всіх файлів з розширенням *.txt з folder_3 до folder_4:")
move_files_with_extension(folder_3, folder_4, '.txt')
print("\n23. Перейменування всіх файлів з розширенням *.txt в folder_1:")
rename_files_in_folder(base_dir, '.txt', '2024')
print("\n52. Видалення всіх файлів з розширенням .doc в folder_3:")
delete_files_with_extension(folder_3, '.doc')
print("\n56. Видалення каталогу folder_2:")
delete_folder(folder_2)
```

```
@Velovo123 →/workspaces/python-labs/lab8 (main) $ python manage_folders.py
1. Вміст каталогу folder_1 та його підкаталогів:
Directory: folder_1
  [DIR] folder_2
  [DIR] folder_3
  [FILE] file_1.txt
  [FILE] file_4.doc
  [FILE] file_6.docx
  [FILE] file_7.pdf
  [FILE] file_8.pdf
  [FILE] file_5.docx
  [FILE] file_9.xlsx
  [FILE] file_10.xlsx
  [FILE] file_2.txt
  [FILE] file_3.doc
Directory: folder_1/folder_2
  [FILE] file_16.docx
  [FILE] file_12.txt
  [FILE] file_17.pdf
  [FILE] file_19.xlsx
  [FILE] file_14.doc
  [FILE] file_15.docx
  [FILE] file_13.doc
  [FILE] file_18.pdf
  [FILE] file_11.txt
  [FILE] file_20.xlsx
Directory: folder_1/folder_3
  [FILE] file_30.xlsx
  [FILE] file_24.doc
  [FILE] file_28.pdf
  [FILE] file_29.xlsx
  [FILE] file_27.pdf
  [FILE] file_22.txt
  [FILE] file_21.txt
  [FILE] file_26.docx
  [FILE] file_23.doc
  [FILE] file_25.docx

2. Створення каталогу folder_4:
Created folder: folder_1/folder_4

6. Копіювання всіх файлів з розширенням *.pdf з folder_2 до folder_4:
Copied file_17.pdf to folder_1/folder_4
Copied file_18.pdf to folder_1/folder_4

18. Переміщення всіх файлів з розширенням *.txt з folder_3 до folder_4:
Moved file_22.txt to folder_1/folder_4
Moved file_21.txt to folder_1/folder_4

23. Перейменування всіх файлів з розширенням *.txt в folder_1:
Renamed file_1.txt to file_1_2024.txt
Renamed file_2.txt to file_2_2024.txt
```

```
52. Видалення всіх файлів з розширенням .doc в folder_3:
Deleted file_24.doc
Deleted file_23.doc

56. Видалення каталогу folder_2:
Deleted folder: folder_1/folder_2
```

2.

```python
import json
import pickle
from typing import List, Dict, Any

# Файл для збереження даних
TEXT_FILE = 'beer_database.txt'
OBJECT_FILE = 'beer_database.pkl'

# Створення бази даних (двовимірний список) з інформацією про пиво
beer_database: List[Dict[str, Any]] = [
    {"Назва": "Оболонь", "Виробник": "Україна", "Міцність": 5.0, "Ціна": 30,
"Термін зберігання": 180},
    {"Назва": "Львівське", "Виробник": "Україна", "Міцність": 4.5, "Ціна": 25,
"Термін зберігання": 150},
    {"Назва": "Heineken", "Виробник": "Нідерланди", "Міцність": 5.0, "Ціна": 50,
"Термін зберігання": 365},
    {"Назва": "Corona", "Виробник": "Мексика", "Міцність": 4.6, "Ціна": 45, "Термін
зберігання": 300},
    {"Назва": "Budweiser", "Виробник": "США", "Міцність": 5.0, "Ціна": 40, "Термін
зберігання": 180}
]

# Функція для виведення бази даних
def print_database(db: List[Dict[str, Any]]) -> None:
    print(f"{'Назва':<12}{'Виробник':<15}{'Міцність':<10}{'Ціна':<10}{'Термін
зберігання':<15}")
    for beer in db:
        print(f"{beer['Назва']:<12}{beer['Виробник']:<15}{beer['Міцність']:<10}{bee
r['Ціна']:<10}{beer['Термін зберігання']:<15}")

# Функція для додавання нового елемента
def add_beer(db: List[Dict[str, Any]], name: str, manufacturer: str, strength:
float, price: int, storage: int) -> None:
    db.append({"Назва": name, "Виробник": manufacturer, "Міцність": strength,
"Ціна": price, "Термін зберігання": storage})

# Функція для сортування за атрибутом
def sort_database(db: List[Dict[str, Any]], attribute: str) -> None:
    try:
        db.sort(key=lambda beer: beer[attribute])
    except KeyError:
        print("Неправильний атрибут для сортування.")

# Функція для видалення за атрибутом
```

```python
def delete_by_attribute(db: List[Dict[str, Any]], attribute: str, value: Any) ->
None:
    db[:] = [beer for beer in db if beer.get(attribute) != value]

# Функція для видалення за індексом
def delete_by_index(db: List[Dict[str, Any]], index: int) -> None:
    try:
        db.pop(index)
    except IndexError:
        print("Неправильний індекс.")

# Функція для фільтрації за атрибутом
def filter_by_attribute(db: List[Dict[str, Any]], attribute: str, value: Any) ->
None:
    print(f"{'Назва':<12}{'Виробник':<15}{'Міцність':<10}{'Ціна':<10}{'Термін
зберігання':<15}")
    for beer in db:
        if beer.get(attribute) == value:
            print(f"{beer['Назва']:<12}{beer['Виробник']:<15}{beer['Міцність']:<10}
{beer['Ціна']:<10}{beer['Термін зберігання']:<15}")

# Завантаження списку з текстового файлу
def load_from_text_file() -> None:
    try:
        with open(TEXT_FILE, 'r', encoding='utf-8') as f:
            global beer_database
            beer_database = json.load(f)
        print("Список завантажено з текстового файлу.")
    except (FileNotFoundError, json.JSONDecodeError) as e:
        print(f"Помилка завантаження з текстового файлу: {e}")

# Збереження списку у текстовий файл
def save_to_text_file() -> None:
    try:
        with open(TEXT_FILE, 'w', encoding='utf-8') as f:
            json.dump(beer_database, f, ensure_ascii=False, indent=4)
        print("Список збережено у текстовий файл.")
    except IOError as e:
        print(f"Помилка збереження у текстовий файл: {e}")

# Завантаження списку як об'єкт
def load_from_object_file() -> None:
    try:
        with open(OBJECT_FILE, 'rb') as f:
            global beer_database
            beer_database = pickle.load(f)
        print("Список завантажено як об'єкт.")
    except (FileNotFoundError, pickle.UnpicklingError) as e:
        print(f"Помилка завантаження як об'єкт: {e}")

# Збереження списку у файл як об'єкт
def save_to_object_file() -> None:
    try:
```

```
        with open(OBJECT_FILE, 'wb') as f:
            pickle.dump(beer_database, f)
        print("Список збережено як об'єкт.")
    except IOError as e:
        print(f"Помилка збереження як об'єкт: {e}")

# Основне меню
def main() -> None:
    load_from_text_file()  # Завантаження даних при запуску

    while True:
        print("\nМеню")
        print("1 - Друк списку")
        print("2 - Додати елемент до списку")
        print("3 - Відсортувати список за заданим атрибутом")
        print("4 - Видалити елемент за заданим атрибутом")
        print("5 - Видалити елемент за заданим індексом")
        print("6 - Вивести елементи із заданим атрибутом")
        print("7 - Зберегти список у текстовий файл")
        print("8 - Зберегти список у файл як об’єкт")
        print("9 - Завантажити список з текстового файлу")
        print("10 - Завантажити список як об’єкт")
        print("11 - Вихід")

        choice = input("Виберіть операцію натиснувши відповідну цифру: ")
        if choice == "1":
            print_database(beer_database)
        elif choice == "2":
            try:
                name = input("Назва: ")
                manufacturer = input("Виробник: ")
                strength = float(input("Міцність: "))
                price = int(input("Ціна: "))
                storage = int(input("Термін зберігання: "))
                add_beer(beer_database, name, manufacturer, strength, price,
storage)
            except ValueError:
                print("Неправильний формат введених даних.")
        elif choice == "3":
            attribute = input("Атрибут для сортування (Назва, Виробник, Міцність,
Ціна, Термін зберігання): ")
            sort_database(beer_database, attribute)
        elif choice == "4":
            attribute = input("Атрибут для видалення (Назва, Виробник, Міцність,
Ціна, Термін зберігання): ")
            value = input("Значення атрибуту для видалення: ")
            delete_by_attribute(beer_database, attribute, value)
        elif choice == "5":
            try:
                index = int(input("Введіть індекс для видалення: "))
                delete_by_index(beer_database, index)
            except ValueError:
                print("Неправильний індекс.")
```

```
        elif choice == "6":
            attribute = input("Атрибут для фільтрації (Назва, Виробник, Міцність,
Ціна, Термін зберігання): ")
            value = input("Значення атрибуту для фільтрації: ")
            filter_by_attribute(beer_database, attribute, value)
        elif choice == "7":
            save_to_text_file()
        elif choice == "8":
            save_to_object_file()
        elif choice == "9":
            load_from_text_file()
        elif choice == "10":
            load_from_object_file()
        elif choice == "11":
            print("Збереження перед виходом.")
            save_to_text_file()  # Зберігання при виході
            break
        else:
            print("Невірний вибір. Спробуйте ще раз.")

# Виконання програми
main()
```

```
@Velovo123 ➜/workspaces/python-labs/lab8 (main) $ python beer_management.py
Помилка завантаження з текстового файлу: [Errno 2] No such file or directory: 'beer_database.txt'

Меню
1 - Друк списку
2 - Додати елемент до списку
3 - Відсортувати список за заданим атрибутом
4 - Видалити елемент за заданим атрибутом
5 - Видалити елемент за заданим індексом
6 - Вивести елементи із заданим атрибутом
7 - Зберегти список у текстовий файл
8 - Зберегти список у файл як об’єкт
9 - Завантажити список з текстового файлу
10 - Завантажити список як об’єкт
11 - Вихід
Виберіть операцію натиснувши відповідну цифру: 1
Назва       Виробник       Міцність  Ціна      Термін зберігання
Оболонь     Україна        5.0       30        180
Львівське   Україна        4.5       25        150
Heineken    Нідерланди     5.0       50        365
Corona      Мексика        4.6       45        300
Budweiser   США            5.0       40        180

Меню
1 - Друк списку
2 - Додати елемент до списку
3 - Відсортувати список за заданим атрибутом
4 - Видалити елемент за заданим атрибутом
5 - Видалити елемент за заданим індексом
6 - Вивести елементи із заданим атрибутом
7 - Зберегти список у текстовий файл
8 - Зберегти список у файл як об’єкт
9 - Завантажити список з текстового файлу
10 - Завантажити список як об’єкт
11 - Вихід
Виберіть операцію натиснувши відповідну цифру: 7
Список збережено у текстовий файл.
```

```
Меню
1 - Друк списку
2 - Додати елемент до списку
3 - Відсортувати список за заданим атрибутом
4 - Видалити елемент за заданим атрибутом
5 - Видалити елемент за заданим індексом
6 - Вивести елементи із заданим атрибутом
7 - Зберегти список у текстовий файл
8 - Зберегти список у файл як об’єкт
9 - Завантажити список з текстового файлу
10 - Завантажити список як об’єкт
11 - Вихід
Виберіть операцію натиснувши відповідну цифру: 8
Список збережено як об'єкт.

Меню
1 - Друк списку
2 - Додати елемент до списку
3 - Відсортувати список за заданим атрибутом
4 - Видалити елемент за заданим атрибутом
5 - Видалити елемент за заданим індексом
6 - Вивести елементи із заданим атрибутом
7 - Зберегти список у текстовий файл
8 - Зберегти список у файл як об’єкт
9 - Завантажити список з текстового файлу
10 - Завантажити список як об’єкт
11 - Вихід
Виберіть операцію натиснувши відповідну цифру: 9
Список завантажено з текстового файлу.
```

3.

```
from colorama import Fore, Style

# Набори символів
upp_char = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
low_char = "abcdefghijklmnopqrstuvwxyz"
num_char = "0123456789"
spc_char = "!@#$%^&*_-"

# Функція для перевірки кожного правила
def check_rule(description: str, condition: bool) -> str:
    color = Fore.GREEN if condition else Fore.RED
    status = "OK!" if condition else "FAIL!"
    print(f"{description} - {color}{status}{Style.RESET_ALL}")
    return condition

# Функція для перевірки пароля на слабкість
def is_password_weak(password: str, filename: str = "weak_passwords.txt") -> bool:
    try:
        with open(filename, "r") as file:
            weak_passwords = {line.strip() for line in file}  # Зчитуємо слабкі
паролі у множину
        return password in weak_passwords
    except FileNotFoundError:
        print(Fore.RED + "Файл зі слабкими паролями не знайдено!" +
Style.RESET_ALL)
        return False

# Основна функція для валідації пароля
def validate_password(password: str) -> None:
    # Перевірка слабкості пароля
    if is_password_weak(password):
        print(Fore.RED + "Пароль слабкий! Він знаходиться у списку слабких
паролів." + Style.RESET_ALL)
        return

    # Перевірка довжини пароля
    length_check = check_rule("Довжина не менше 10 символів", len(password) >= 10)

    # Перевірка дозволених символів
    allowed_chars_check = check_rule("Пароль містить лише допустимі символи",
                                     all(c in (upp_char + low_char + num_char +
spc_char) for c in password))

    # Підрахунок кількості кожного типу символів
    low_count = sum(1 for c in password if c in low_char)
    upp_count = sum(1 for c in password if c in upp_char)
    num_count = sum(1 for c in password if c in num_char)
    spc_count = sum(1 for c in password if c in spc_char)

    # Валідація кожного правила
    rules = [
```

```
        ("Не менше 2 маленьких латинських літер", low_count >= 2),
        ("Не більше 4 маленьких латинських літер", low_count <= 4),
        ("Не менше 2 цифр", num_count >= 2),
        ("Не більше 4 цифр", num_count <= 4),
        ("Не менше 2 спеціальних символів", spc_count >= 2),
        ("Не більше 3 спеціальних символів", spc_count <= 3),
        ("Не менше 2 великих латинських літер", upp_count >= 2),
        ("Не більше 5 великих латинських літер", upp_count <= 5),
        ("Не більше 3 однакових великих латинських літер підряд",
         not any(password[i] == password[i+1] == password[i+2] for i in
range(len(password) - 2) if password[i] in upp_char)),
        ("Не більше 3 однакових спеціальних символів підряд",
         not any(password[i] == password[i+1] == password[i+2] for i in
range(len(password) - 2) if password[i] in spc_char))
    ]

    # Перевірка правил
    results = [check_rule(description, condition) for description, condition in
rules]

    # Перевірка загальної валідності пароля
    is_valid = all([length_check, allowed_chars_check] + results)
    print(Fore.GREEN + "Пароль валідний!" + Style.RESET_ALL if is_valid else
Fore.RED + "Пароль не валідний!" + Style.RESET_ALL)

# Запит на введення пароля
password = input("Введіть пароль довжиною не менше 10 символів: ")
validate_password(password)
```

```
@Velovo123 →/workspaces/python-labs/lab8 (main) $ python password_validator.py
Введіть пароль довжиною не менше 10 символів: UpperL!M!!123
Довжина не менше 10 символів - OK!
Пароль містить лише допустимі символи - OK!
Не менше 2 маленьких латинських літер - OK!
Не більше 4 маленьких латинських літер - OK!
Не менше 2 цифр - OK!
Не більше 4 цифр - OK!
Не менше 2 спеціальних символів - OK!
Не більше 3 спеціальних символів - OK!
Не менше 2 великих латинських літер - OK!
Не більше 5 великих латинських літер - OK!
Не більше 3 однакових великих латинських літер підряд - OK!
Не більше 3 однакових спеціальних символів підряд - OK!
Пароль валідний!
@Velovo123 →/workspaces/python-labs/lab8 (main) $ python password_validator.py
Введіть пароль довжиною не менше 10 символів: qwertyuiop
Пароль слабкий! Він знаходиться у списку слабких паролів.
```

Висновок:

Під час виконання лабораторної роботи я ознайомився з роботою з файлами та каталогами у Python, використовуючи модулі `os` і `shutil`, а також навчився ефективно застосовувати

обробку винятків для забезпечення стабільності програми. Завдяки цьому я зрозумів, як виконувати різноманітні операції з файлами (створення, копіювання, переміщення, перейменування та видалення) і обробляти можливі помилки, пов'язані з відсутніми файлами чи відсутністю прав доступу.

Досвід, отриманий під час роботи, дозволив мені підвищити рівень написання надійного та безпечного коду, що є важливим для розробки стійких програм.