МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж



Лабораторна робота №9

з дисципліни Спеціалізовані мови програмування

на тему

Створення та рефакторінг програмно-інформаційного продукту засобами Python

Виконав:

студент групи РІ-21сп

Назарій КУРИЩУК

Львів – 2024

**Мета виконання лабораторної роботи:** розробка програмно-інформаційного продукту засобами Python

**План роботи**

**Завдання 1.**

Створити скрипт запуску лабораторних робіт 1-8 (Runner) з єдиним меню для управління додатками використовуючи патерн FACADE https://refactoring.guru/uk/design-patterns/facade

**Завдання 2.**

Зробити рефакторінг додатків, які були зроблені в лб 1-8, для підтримки можливості запуску через Runner

**Завдання 3.**

Зробити рефакторинг додатків, які були зроблені в лб 1-8, використовуючи багаторівневу архітектуру додатків (див. приклад нижче) та всі принципи об’єктно-орієнтованого підходу

**Завдання 4.**

Створити бібліотеку класів, які повторно використовуються у всіх лабораторних роботах та зробити рефакторінг додатків для підтримки цієї бібліотеки. Таких класів в бібліотеці має буде як найменш 5

**Завдання 5.**

Додати логування функцій в класи бібліотеки програмного продукту використовуючи https://docs.python.org/uk/3/howto/logging.html

**Завдання 6.**

Додати коментарі до програмного коду та сформувати документацію програмного продукту засобами pydoc. Документація має бути представлена у вигляді сторінок тексту на консолі, подана у веб-браузері та збережена у файлах HTML

**Завдання 7.**

Документація та код програмного продукту має бути розміщено в GIT repo

**Завдання 8.**

Проведіть статичний аналіз коду продукту засобами PYLINT https://pylint.readthedocs.io/en/stable/ та виправте помилки, які були ідентифіковані. Первинний репорт з помилками додайте до звіту лабораторної роботи

**Результати тестування:**

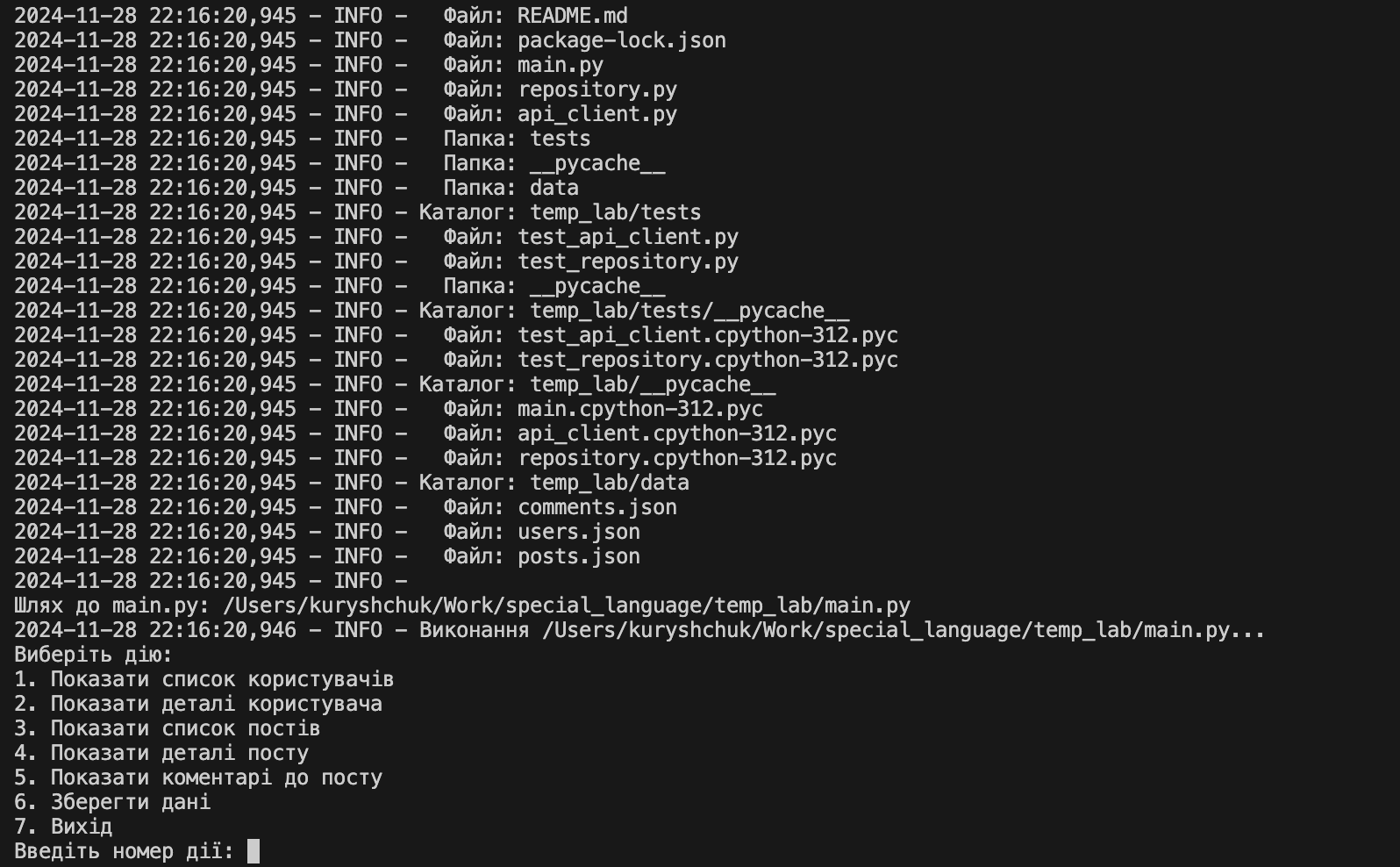
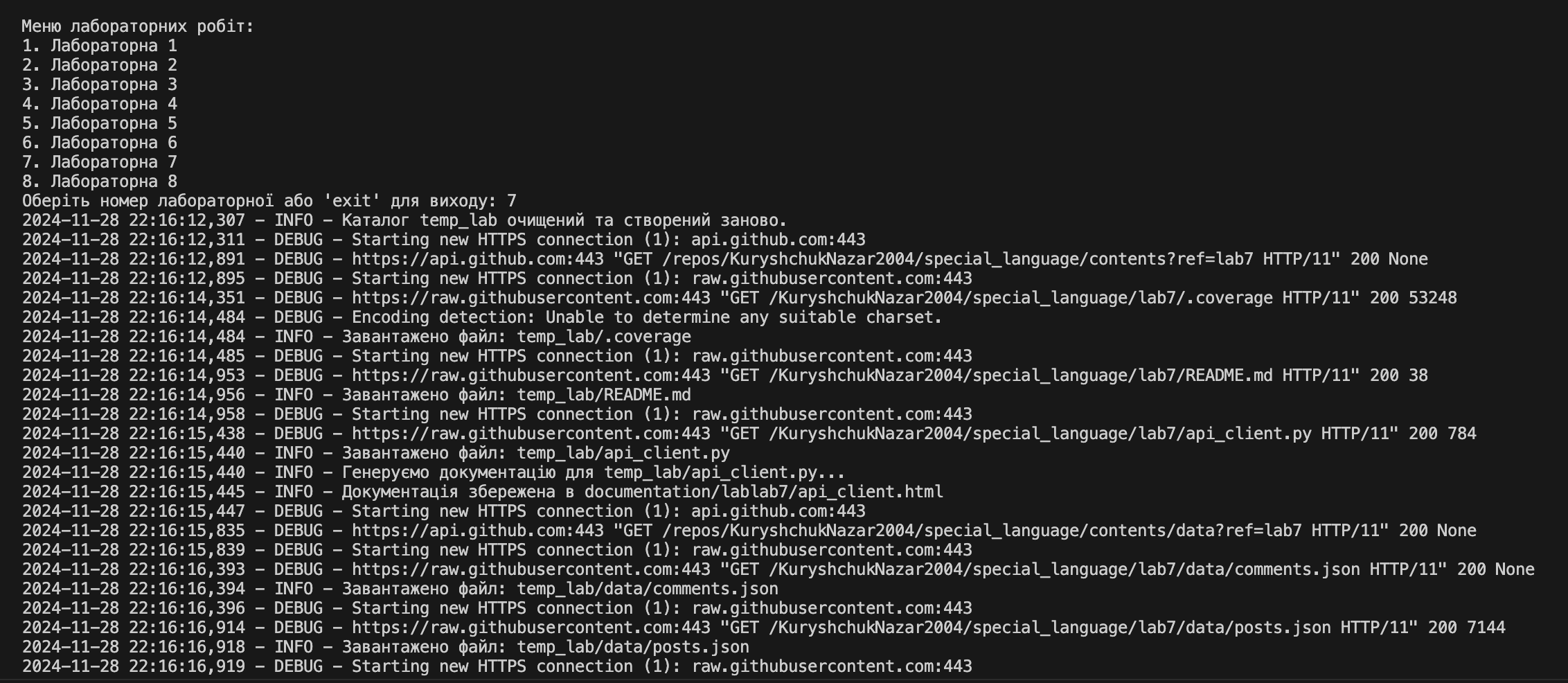


Рис. 1. Результати виконання

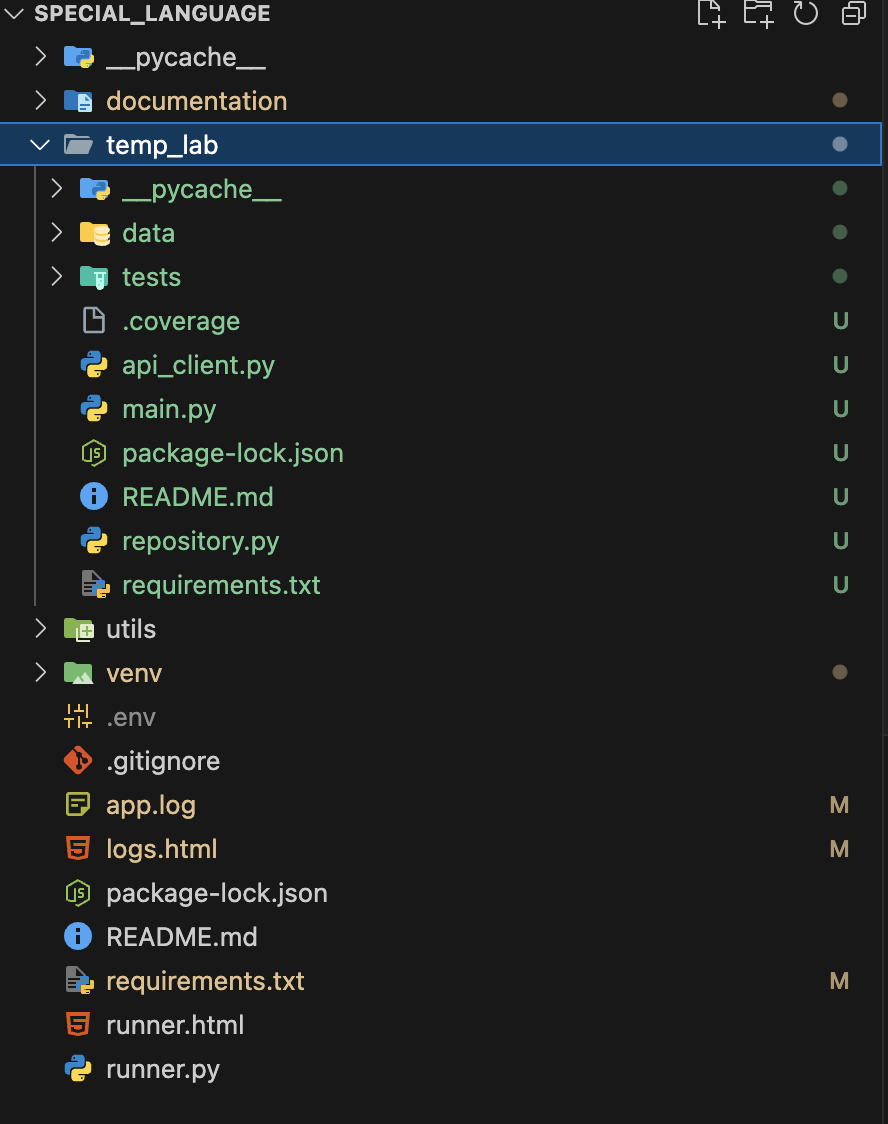


Рис. 2. Структура папок

# Завантажуємо змінні середовища з файлу .env

load\_dotenv()

# Налаштування логування

logging.basicConfig(level=logging.DEBUG,

format="%(asctime)s - %(levelname)s - %(message)s",

handlers=[

logging.StreamHandler(),

logging.FileHandler('app.log', mode='w') # Логування в файл

])

# Створення HTML логів

class HtmlFormatter(logging.Formatter):

"""Форматування для логів у HTML форматі."""

def format(self, record):

log\_message = super().format(record)

return f"<tr><td>{record.asctime}</td><td>{record.levelname}</td><td>{log\_message}</td></tr>"

html\_handler = handlers.RotatingFileHandler("logs.html", maxBytes=10\*1024\*1024, backupCount=5)

html\_handler.setFormatter(HtmlFormatter("%(message)s"))

logging.getLogger().addHandler(html\_handler)

class Runner:

"""

Клас для завантаження лабораторних робіт з GitHub, виконання та генерації документації.

Атрибути:

repo\_owner (str): Власник репозиторію на GitHub.

repo\_name (str): Назва репозиторію на GitHub.

base\_url (str): URL для доступу до вмісту репозиторію.

branches (dict): Словник з номерами лабораторних робіт та їх відповідними гілками.

"""

def \_\_init\_\_(self):

self.repo\_owner = "KuryshchukNazar2004"

self.repo\_name = "special\_language"

self.base\_url = f"https://api.github.com/repos/{self.repo\_owner}/{self.repo\_name}/contents"

self.branches = {

"1": "lab1",

"2": "lab2",

"3": "lab3",

"4": "lab4",

"5": "lab5",

"6": "lab6",

"7": "lab7",

"8": "lab8",

}

self.github\_token = os.getenv("GITHUB\_TOKEN")

self.excluded\_paths = [".DS\_Store", "\_\_pycache\_\_", ".git", ".github", "venv"]

def make\_request(self, url):

headers = {"Authorization": f"token {self.github\_token}"}

try:

response = requests.get(url, headers=headers)

response.raise\_for\_status() # Піднімаємо виключення, якщо статус код 4xx або 5xx

return response

except requests.exceptions.RequestException as e:

logging.error(f"Помилка запиту: {e}")

return None

def clear\_temp\_directory(self, temp\_dir):

if os.path.exists(temp\_dir):

shutil.rmtree(temp\_dir)

os.makedirs(temp\_dir, exist\_ok=True)

logging.info(f"Каталог {temp\_dir} очищений та створений заново.")

def fetch\_files(self, branch\_name, temp\_dir, doc\_dir):

self.clear\_temp\_directory(temp\_dir)

response = self.make\_request(f"{self.base\_url}?ref={branch\_name}")

if response.status\_code == 200:

for file\_info in response.json():

if not any(excluded in file\_info["path"] for excluded in self.excluded\_paths):

self.download\_file(file\_info, branch\_name, temp\_dir, doc\_dir)

else:

logging.error(f"Помилка завантаження: {response.status\_code}")

logging.error(f"URL: {self.base\_url}?ref={branch\_name}")

def download\_file(self, file\_info, branch\_name, temp\_dir, doc\_dir):

file\_url = file\_info.get("download\_url")

local\_path = os.path.join(temp\_dir, file\_info["path"])

if file\_info["type"] == "file" and file\_url:

os.makedirs(os.path.dirname(local\_path), exist\_ok=True)

response = self.make\_request(file\_url)

if response.status\_code == 200:

with open(local\_path, "w", encoding="utf-8") as file:

file.write(response.text)

logging.info(f"Завантажено файл: {local\_path}")

# Генеруємо документацію для Python файлів

if file\_info["path"].endswith(".py"): # Якщо файл Python

self.generate\_documentation(local\_path, doc\_dir)

else:

logging.error(f"Помилка завантаження файлу {file\_info['path']}: {response.status\_code}")

elif file\_info["type"] == "dir":

sub\_dir\_url = f"{self.base\_url}/{file\_info['path']}?ref={branch\_name}"

response = self.make\_request(sub\_dir\_url)

if response.status\_code == 200:

for sub\_file\_info in response.json():

if not any(excluded in sub\_file\_info["path"] for excluded in self.excluded\_paths):

self.download\_file(sub\_file\_info, branch\_name, temp\_dir, doc\_dir)

def generate\_documentation(self, file\_path, doc\_dir):

logging.info(f"Генеруємо документацію для {file\_path}...")

try:

if file\_path.endswith('.py'):

sys.path.insert(0, os.path.dirname(file\_path)) # Додаємо шлях до файлу

# Створюємо назву модуля для імпорту

module\_name = os.path.basename(file\_path).replace('.py', '')

try:

# Імпортуємо модуль

module\_spec = importlib.util.spec\_from\_file\_location(module\_name, file\_path)

module = importlib.util.module\_from\_spec(module\_spec)

module\_spec.loader.exec\_module(module)

if not os.path.exists(doc\_dir):

os.makedirs(doc\_dir)

# Генерація документації та збереження її в HTML

doc\_file\_path = os.path.join(doc\_dir, f"{module\_name}.html")

with open(doc\_file\_path, 'w', encoding='utf-8') as doc\_file:

doc\_file.write(pydoc.render\_doc(module))

logging.info(f"Документація збережена в {doc\_file\_path}")

except Exception as e:

logging.error(f"Помилка при імпорті {module\_name}: {e}")

finally:

sys.path.pop(0) # Повертаємо старий sys.path

else:

logging.warning(f"{file\_path} не є Python файлом. Пропускаємо...")

except Exception as e:

logging.error(f"Помилка при генерації документації для {file\_path}: {e}")

def fetch\_and\_execute(self, branch\_name):

temp\_dir = "temp\_lab"

doc\_dir = os.path.join("documentation", f"lab{branch\_name}") # Зберігаємо документацію в папці для кожної лабораторної

self.fetch\_files(branch\_name, temp\_dir, doc\_dir)

logging.info(f"\nВміст папки {temp\_dir}:")

for root, dirs, files in os.walk(temp\_dir):

logging.info(f"Каталог: {root}")

for file in files:

logging.info(f" Файл: {file}")

for directory in dirs:

logging.info(f" Папка: {directory}")

sys.path.insert(0, os.path.abspath(temp\_dir))

main\_path = os.path.abspath(os.path.join(temp\_dir, "main.py"))

logging.info(f"\nШлях до main.py: {main\_path}")

if os.path.exists(main\_path):

try:

logging.info(f"Виконання {main\_path}...")

current\_dir = os.getcwd()

os.chdir(temp\_dir)

with open(main\_path, "r", encoding="utf-8") as f:

exec(compile(f.read(), main\_path, "exec"), {"\_\_name\_\_": "\_\_main\_\_"})

except Exception as e:

logging.error(f"Помилка виконання main.py: {e}")

finally:

os.chdir(current\_dir)

sys.path.pop(0)

else:

logging.error(f"Файл main.py не знайдено в {temp\_dir}! Перевірте структуру файлів.")

if os.path.exists(temp\_dir):

shutil.rmtree(temp\_dir)

logging.info(f"Папка {temp\_dir} очищена.")

def run(self):

while True:

print("\nМеню лабораторних робіт:")

for num, branch in self.branches.items():

print(f"{num}. Лабораторна {num}")

choice = input("Оберіть номер лабораторної або 'exit' для виходу: ")

if choice.lower() == "exit":

print("Вихід з програми.")

break

elif choice in self.branches:

self.fetch\_and\_execute(self.branches[choice])

else:

print("Невірний вибір, спробуйте ще раз.")

if \_\_name\_\_ == "\_\_main\_\_":

Runner().run()

**Висновки:** На цій лабораторній роботі я зробив рефакторинг коду та розробив раннер для всіх попередніх лабораторних, розробивши API яке стягує динамічно дані про всі лабораторні з віток мого репозиторію.