МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

**Звіт**

До лабораторної роботи №7

**Розробка з API та веб-сервісами**

Виконав:

ст. гр. ІТ-21сп

**Олександр БІЗЯЄВ**

Прийняв:

доц. каф. ІСМ

**Сергій ЩЕРБАК**

Львів ‒ 2023

**Лабораторна робота №7**

**Мета:** створення консольного об’єкто-орієнтованого додатка з використанням API.

**Хід роботи**

Завдання 1: Вибір провайдера API

Виберіть надійний API, який надає через HTTP необхідні дані для віддаленого зберігання, вивантаження або реалізуйте свій. Для прикладу це може бути jsonplaceholder.org

Завдання 2: Інтеграція API

Виберіть бібліотеку для роботи з API та обробки HTTP запитів (для прикладу це може бути бібліотека Requests). Інтегруйте обраний API в ваш консольний додаток на Python. Ознайомтеся з документацією API та налаштуйте необхідний API-ключ чи облікові дані.

Завдання 3: Введення користувача

Розробіть користувальницький інтерфейс, який дозволяє користувачам візуалізувати всі доступні дані в табличному вигляді та у вигляді списку. Реалізуйте механізм для збору та перевірки введеного даних користувачем.

Завдання 4: Розбір введення користувача

Створіть розбірник для видобування та інтерпретації виразів користувача на основі регулярних виразів, наприклад, для візуалізації дат, телефонів, тощо. Переконайтеся, що розбірник обробляє різні формати введення та надає зворотний зв'язок про помилки.

Завдання 5: Відображення результатів

Реалізуйте логіку для візуалізації даних через API в консолі. Обробляйте відповіді API для отримання даних у вигляді таблиць, списків. Заголовки таблиць, списків мають виділяться кольором та шрифтом, які задається користувачем

Завдання 6: Збереження даних

Реалізуйте можливості збереження даних у чіткому та читабельному форматі JSON, CSV та TXT

Завдання 7: Обробка помилок

Розробіть надійний механізм обробки помилок для керування помилками API, некоректним введенням користувача та іншими можливими проблемами. Надавайте інформативні повідомлення про помилки.

Завдання 8: Ведення історії обчислень

Включіть функцію, яка реєструє запити користувача, включаючи введені запити та відповідні результати. Дозвольте користувачам переглядати та рецензувати історію своїх запитів.

Завдання 9: Юніт-тести

Напишіть юніт-тести для перевірки функціональності вашого додатку. Тестуйте різні операції, граничні випадки та сценарії помилок.

Файл **data\_display.py**

import re  
import pandas as pd  
from colorama import Fore, Style  
from tabulate import tabulate  
  
# from labwork\_7.api\_data import APIData  
from labwork\_7.errors import Error  
from labwork\_7 import variables  
  
  
class DataDisplay:  
  
 def display\_data\_as\_table(self, data, color, choose=1):  
 if data:  
 pd.set\_option('display.max\_columns', None)  
 pd.set\_option('display.max\_rows', None)  
 pd.set\_option('display.width', None)  
 pd.set\_option('display.max\_colwidth', 20)  
 df = pd.DataFrame(data)  
 if choose == 1:  
 df['login'] = df['login'].apply(lambda x: x['username'])  
 df['address'] = df['address'].apply(lambda x: x['street'])  
 df['company'] = df['company'].apply(lambda x: x['name'])  
  
 return self.columns\_table(df, color)  
 if choose == 2:  
 return self.columns\_table(df, color)  
 else:  
 return variables.no\_data  
  
 def columns\_table(self, df, color):  
 colored\_columns = [f"{color}{col}{Style.RESET\_ALL}" for col in df.columns]  
 df = df.map(lambda x: x if isinstance(x, float) or len(str(x)) <= 20 else str(x)[:20 - 3] + '...')  
 return tabulate(df, headers=colored\_columns, tablefmt='psql', showindex=False)  
  
 def display\_data\_as\_list(self, data, color, choose=1):  
 if data:  
 df = pd.DataFrame(data)  
  
 if choose == 1:  
 df['login'] = df['login'].apply(lambda x: x['username'])  
 df['address'] = df['address'].apply(lambda x: x['street'])  
 df['company'] = df['company'].apply(lambda x: x['name'])  
 return self.columns\_list(df, color)  
 if choose == 2:  
 for col in df.columns:  
 if df[col].dtype == 'object':  
 df[col] = df[col].astype(str).str.slice(0, 50)  
 return self.columns\_list(df, color)  
 else:  
 return variables.no\_data  
  
 def columns\_list(self, df, color):  
 result = ""  
 for index, row in df.iterrows():  
 for col, value in row.items():  
 result += f"{color}{col}{Style.RESET\_ALL}: {value}\n"  
 result += "\n"  
 return result  
  
  
 def remove\_color\_tags(self, text):  
 clean\_text = re.sub(r'\x1b\[[0-9;]\*m', '', text)  
 return clean\_text

Файл **variables.py**

menu\_input = "\nWhat you want? \n1. Do API request\n2. Show history\n3. Exit\n Your choose: "  
url\_input = "Please write url: "  
error\_url = "Error: You have selected the wrong Url"  
  
color\_input = "Colors: \n 1. 'black'\n 2. 'red'\n 3. 'green'\n 4. 'yellow'\n 5. 'blue'\n 6. 'white'\n 7. 'cyan'\n 8. 'magenta'\nPlease choose color : "  
error\_color = "Error: You have selected the wrong color number"  
ask\_save\_to\_file = "Do you want to save the result? (Yes/No):"  
file\_name = "Please enter file name: "  
choice\_yes = "yes"  
choice\_y = "y"  
no\_data = "No data to display."  
  
request\_exception = "Request failed. Please enter correct url "  
  
method\_input = "Methods: \n 1. 'Table'\n 2. 'List'\nPlease choose method : "  
error\_method = "Error: You have selected the wrong method number"  
  
format\_input = "Formats: \n 1. 'Txt'\n 2. 'Json'\n 3. 'Csv'\nPlease choose method : "  
  
history = "\nUser Requests History:"  
empty\_history = "\nThe request history is empty"

Файл **errors.py**

import requests  
from colorama import Fore, Style  
from labwork\_7 import variables  
  
  
class Error:  
 def check\_api\_data(self, url):  
 try:  
 response = requests.get(url)  
 response.raise\_for\_status()  
 return response.json()  
 except requests.RequestException as e:  
 raise requests.HTTPError() from e  
  
  
 def check\_color(self, number):  
 try:  
 number = int(number)  
 if number < 1 or number > 8:  
 raise ValueError  
 except ValueError:  
 return ValueError()  
  
 def check\_method(self, number):  
 try:  
 number = int(number)  
 if number < 1 or number > 2:  
 raise ValueError  
 except ValueError:  
 return ValueError()  
  
 def check\_fomat(self, number):  
 try:  
 number = int(number)  
 if number < 1 or number > 3:  
 raise ValueError  
 except ValueError:  
 return ValueError()

Файл **interface.py**

from colorama import Fore, Style  
import requests  
  
from labwork\_7.data\_display import DataDisplay  
from labwork\_7.errors import Error  
from labwork\_7 import variables  
  
  
class Interface:  
 def \_\_init\_\_(self):  
 self.error = Error()  
 self.user\_requests = []  
 self.api\_data = DataDisplay()  
  
 def \_choose\_color(self, user\_color):  
 if user\_color == '1':  
 return Fore.BLACK  
 elif user\_color == '2':  
 return Fore.RED  
 elif user\_color == '3':  
 return Fore.GREEN  
 elif user\_color == '4':  
 return Fore.YELLOW  
 elif user\_color == '5':  
 return Fore.BLUE  
 elif user\_color == '6':  
 return Fore.WHITE  
 elif user\_color == '7':  
 return Fore.CYAN  
 elif user\_color == '8':  
 return Fore.MAGENTA  
  
 def \_choose\_method(self, method):  
 if method == '1':  
 return 'table'  
 else:  
 return 'list'  
  
 def \_choose\_format(self, format):  
 if format == '1':  
 return 'txt'  
 elif format == '2':  
 return 'json'  
 else:  
 return 'csv'  
  
 def user\_input(self):  
 while True:  
 # url = "https://jsonplaceholder.org/users"  
 # url = "https://jsonplaceholder.org/posts"  
 # url = "https://jsonplaceholder.org/comments"  
 menu\_input = input(variables.menu\_input)  
 if menu\_input == "1":  
 url, url\_good = self.get\_input\_api(self.error.check\_api\_data, variables.error\_url)  
 method = self.\_choose\_method(self.get\_input\_settings(variables.method\_input, self.error.check\_method,  
 variables.error\_method))  
 color = self.\_choose\_color(self.get\_input\_settings(variables.color\_input, self.error.check\_color,  
 variables.error\_color))  
 data = self.display\_data(url, url\_good, color, method)  
 self.add\_to\_request\_history(url)  
 change\_selection = input(variables.ask\_save\_to\_file)  
 if self.\_choice(change\_selection):  
 name\_file = input(variables.file\_name)  
 format = self.\_choose\_format(self.get\_input\_settings(variables.format\_input, self.error.check\_fomat,  
 variables.error\_method))  
 self.save\_to\_file(name\_file, data, format)  
 elif menu\_input == '2':  
 self.show\_request\_history()  
 else:  
 break  
  
 def display\_data(self, url, url\_data, color, method="list"):  
 print(f"Data displayed as a {method}:")  
 if url.endswith("users"):  
 if method == "table":  
 data = self.api\_data.display\_data\_as\_table(url\_data, color, 1)  
 else:  
 data = self.api\_data.display\_data\_as\_list(url\_data, color, 1)  
 else:  
 if method == "table":  
 data = self.api\_data.display\_data\_as\_table(url\_data, color, 2)  
 else:  
 data = self.api\_data.display\_data\_as\_list(url\_data, color, 2)  
 print(data)  
 return data  
  
 def show\_request\_history(self):  
 if not self.user\_requests:  
 print(f"{Fore.RED + Style.BRIGHT}{variables.empty\_history}{Style.RESET\_ALL}")  
 else:  
 print(f"{variables.history}")  
 for index, request in enumerate(self.user\_requests, start=1):  
 print(f"{index}. {request}")  
  
 def add\_to\_request\_history(self, url):  
 return self.user\_requests.append(url)  
  
 def save\_to\_file(self, file\_name, data, format):  
 file\_name = f"{file\_name}.{format}"  
  
 cleaned\_text = self.api\_data.remove\_color\_tags(data)  
  
 with open(file\_name, "w") as file:  
 file.write(cleaned\_text)  
  
 def get\_input\_settings(self, number, error\_check, error\_message):  
 while True:  
 value = input(number)  
 if not error\_check(value):  
 return value  
 print(Fore.RED + Style.BRIGHT + error\_message + Style.RESET\_ALL)  
  
 def get\_input\_api(self, error\_check, error\_message):  
 while True:  
 try:  
 url = input(variables.url\_input)  
 if error\_check(url):  
 data = requests.get(url)  
 data = data.json()  
 return url, data  
 except requests.HTTPError as e:  
 print(Fore.RED + Style.BRIGHT + error\_message + Style.RESET\_ALL)  
  
 def \_choice(self, choice):  
 if choice.lower() in [variables.choice\_yes, variables.choice\_y]:  
 return True

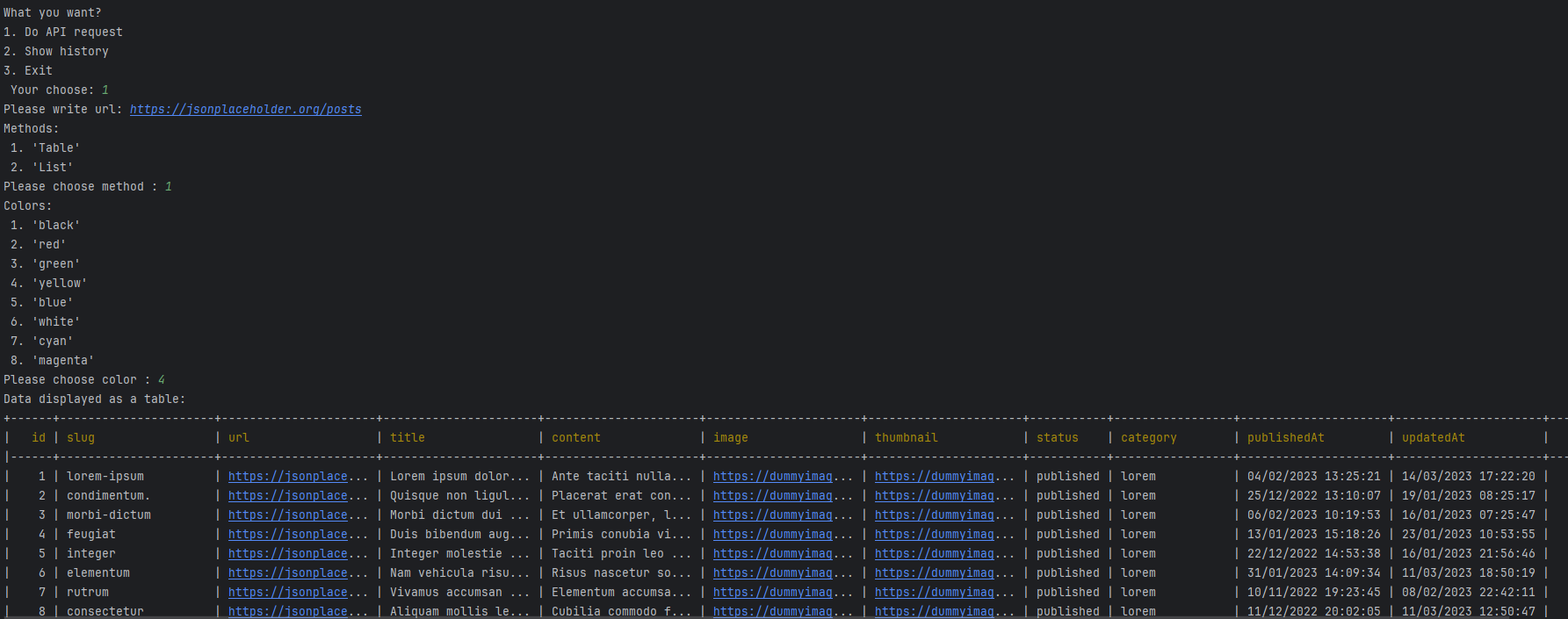
Файл **test.py**

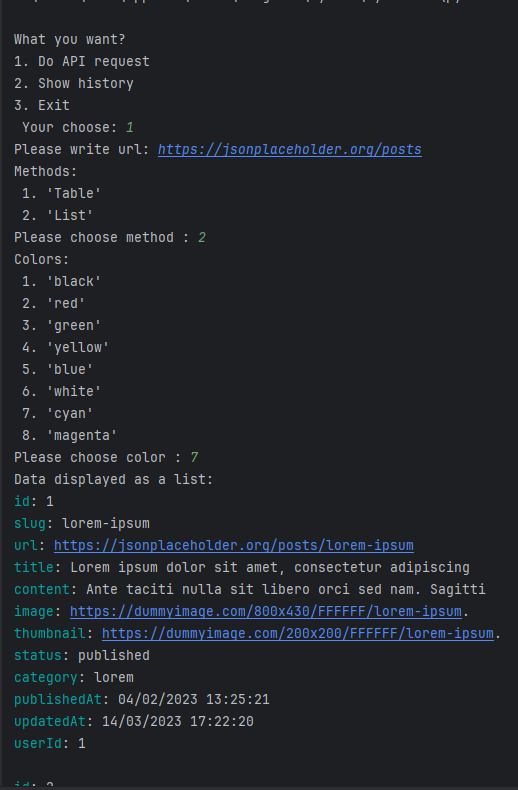
import unittest  
  
import requests  
  
from labwork\_7.errors import Error  
  
  
class Test(unittest.TestCase):  
  
 def setUp(self):  
 self.error = Error()  
  
 def test\_without\_errors(self):  
 urls = [  
 "https://jsonplaceholder.org/users",  
 "https://jsonplaceholder.org/posts",  
 "https://jsonplaceholder.org/comments"  
 ]  
  
 for url in urls:  
 with self.subTest(url=url):  
 with self.assertRaises(requests.HTTPError):  
 self.error.check\_api\_data(url)  
  
 def test\_with\_errors(self):  
 urls = [  
 None,  
 1234567689,  
 "https://empty"  
 ]  
  
 for url in urls:  
 with self.subTest(url=url):  
 with self.assertRaises(requests.HTTPError):  
 self.error.check\_api\_data(url)

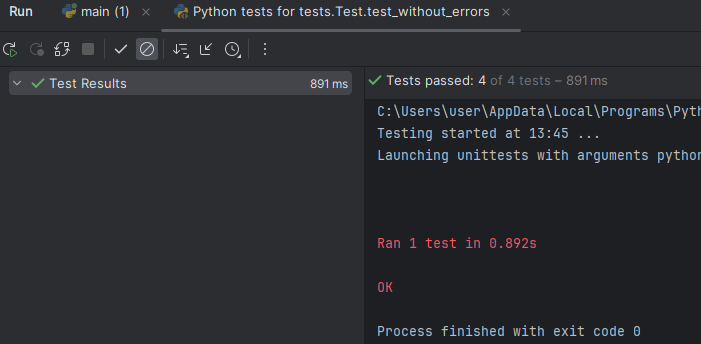
Файл **main.py**

from labwork\_7.interface import Interface  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 interface = Interface()  
 interface.user\_input()

Результат виконання всіх завдань:







**Рис.1** Результат виконання

**Висновок:** Виконавши ці завдання, я розробив консольний об’єкто-орієнтований додаток з використанням API