МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №9

з дисципліни “Спеціалізовані мови програмування”

на тему

“Візуалізація та обробка даних за допомогою спеціалізованих бібліотек Python”

Виконав:

студент групи РІ-32

Мар’ян МІЛЯНЕЦЬ

Прийняв:

к.т.н.,

доц. кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

**Мета роботи:** Розробка додатка для візуалізації CSV-наборів даних за допомогою Matplotlib та базових принципів ООП (наслідування, інкапсуляція, поліморфізм)

**План роботи**

Завдання 1: Вибір CSV-набору даних

Оберіть CSV-набір даних, який ви хочете візуалізувати. Переконайтеся, що він містить відповідні дані для створення змістовних візуалізацій.

Завдання 2: Завантаження даних з CSV

Напишіть код для завантаження даних з CSV-файлу в ваш додаток Python. Використовуйте бібліотеки, такі як Pandas, для спрощення обробки даних.

Завдання 3: Дослідження даних

Визначте екстремальні значення по стовцям

Завдання 4: Вибір типів візуалізацій

Визначте, які типи візуалізацій підходять для представлення вибраних наборів даних. Зазвичай це може бути лінійні графіки, стовпчикові діаграми, діаграми розсіювання, гістограми та секторні діаграми.

Завдання 5: Підготовка даних

Попередньо обробіть набір даних за необхідністю для візуалізації. Це може включати виправлення даних, фільтрацію, агрегацію або трансформацію.

Завдання 6: Базова візуалізація

Створіть базову візуалізацію набору даних, щоб переконатися, що ви можете відображати дані правильно за допомогою Matplotlib. Розпочніть з простої діаграми для візуалізації однієї змінної.

Завдання 7: Розширені візуалізації

Реалізуйте більш складні візуалізації, виходячи з характеристик набору. Поекспериментуйте з різними функціями Matplotlib та налаштуваннями.

Завдання 8: Декілька піддіаграм

Навчіться створювати кілька піддіаграм в межах одного малюнка для відображення декількох візуалізацій поруч для кращого порівняння.

Завдання 9: Експорт і обмін

Реалізуйте функціональність для експорту візуалізацій як зображень (наприклад, PNG, SVG) або інтерактивних веб-додатків (наприклад, HTML)

Програмний код:

import pandas as pd

import matplotlib.pyplot as plt

from abc import ABC, abstractmethod

import plotly.express as px

class Command(ABC):

@abstractmethod

def execute(self):

pass

class LoadDataCommand(Command):

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

self.filename = filename

self.data = None

def execute(self):

self.data = pd.read\_csv(self.filename, delimiter=';')

return self.data

class ExtremeValuesCommand(Command):

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

self.data = data

self.extreme\_values = None

def execute(self):

self.extreme\_values = self.data.describe()

return self.extreme\_values

def undo(self):

self.extreme\_values = None

class VisualizationReceiver:

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

self.data = data

def basic\_visualization(self):

self.data.plot(kind='bar')

plt.show()

def extended\_visualization(self):

if len(self.data.columns) >= 2:

col1 = self.data.columns[0]

col2 = self.data.columns[1]

plt.scatter(self.data[col1], self.data[col2])

plt.xlabel(col1)

plt.ylabel(col2)

plt.title('Extended Visualization: Scatter Plot')

plt.show()

else:

print("Not enough columns for extended visualization.")

def save\_extended\_visualization\_html(self, filename):

if len(self.data.columns) >= 2:

fig = px.scatter(self.data, x=self.data.columns[0], y=self.data.columns[1],

title='Extended Visualization: Scatter Plot')

fig.write\_html(filename)

print(f"Extended visualization saved as HTML: {filename}")

else:

print("Not enough columns for extended visualization.")

def save\_basic\_visualization\_html(self, filename):

long\_form\_data = pd.melt(self.data, id\_vars=self.data.columns[0], value\_vars=self.data.columns[1:])

fig = px.bar(long\_form\_data, x='variable', y='value', title='Basic Visualization: Bar Plot')

fig.write\_html(filename)

print(f"Basic visualization saved as HTML: {filename}")

def save\_extended\_visualization\_png(self, filename):

if len(self.data.columns) >= 2:

col1 = self.data.columns[0]

col2 = self.data.columns[1]

plt.scatter(self.data[col1], self.data[col2])

plt.xlabel(col1)

plt.ylabel(col2)

plt.title('Extended Visualization: Scatter Plot')

plt.savefig(filename)

plt.show()

print(f"Extended visualization saved as PNG: {filename}")

else:

print("Not enough columns for extended visualization.")

def save\_basic\_visualization\_png(self, filename):

long\_form\_data = pd.melt(self.data, id\_vars=self.data.columns[0], value\_vars=self.data.columns[1:])

plt.bar(long\_form\_data['variable'], long\_form\_data['value'])

plt.title('Basic Visualization: Bar Plot')

plt.savefig(filename)

plt.show()

print(f"Basic visualization saved as PNG: {filename}")

class BasicVisualizationCommand(Command):

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

self.receiver = receiver

def execute(self):

self.receiver.basic\_visualization()

class ExtendedVisualizationCommand(Command):

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

self.receiver = receiver

def execute(self):

self.receiver.extended\_visualization()

class Client:

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

self.data = None

self.receiver = VisualizationReceiver(None)

self.command\_stack = []

def run\_command(self, command):

command.execute()

self.command\_stack.append(command)

def undo\_last\_command(self):

if self.command\_stack:

undone\_command = self.command\_stack.pop()

undone\_command.undo()

print(f"Undoing last command: {undone\_command}")

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

load\_command = LoadDataCommand("weather.csv")

data = load\_command.execute()

extreme\_values\_command = ExtremeValuesCommand(data)

extreme\_values = extreme\_values\_command.execute()

print("Extreme Values:")

print(extreme\_values)

visualization\_receiver = VisualizationReceiver(data)

basic\_visualization\_command = BasicVisualizationCommand(visualization\_receiver)

basic\_visualization\_command.execute()

extended\_visualization\_command = ExtendedVisualizationCommand(visualization\_receiver)

extended\_visualization\_command.execute()

visualization\_receiver.save\_basic\_visualization\_html("basic\_visualization.html")

visualization\_receiver.save\_extended\_visualization\_html("extended\_visualization.html")

visualization\_receiver.save\_basic\_visualization\_png("basic\_visualization.png")

visualization\_receiver.save\_extended\_visualization\_png("extended\_visualization.png")

client = Client()

client.run\_command(load\_command)

Висновок: Виконуючи ці завдання, я створив багатофункціональний додаток для візуалізації CSV-наборів даних за допомогою Matplotlib. Цей проект покращив навички візуалізації даних, дозволяючи досліджувати результати з різноманітними наборами даних