Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет Технологического Менеджмента и Инноваций

**Лабораторная работа №10**

Выполнили:

Савинова А.К.

Проверил

Мусаев А.А.

Санкт-Петербург, 2024

**Задание 7**

Представьте, что вы разрабатываете программу для анализа погодных данных с

различных метеостанций. У вас есть набор данных о температуре и осадках за период в

нескольких городах.

Ваша задача - написать программу на Python, выполняющую следующие действия:

1. Создать функцию для определения средней температуры и суммарного количества

осадков по каждому городу за весь период данных.

2. Реализовать функцию для определения того, какие дни в каждом городе были с

наибольшей и наименьшей температурой.

3. Создать функцию для анализа того, какие города были самыми холодными и

самыми теплыми в среднем за весь период.

4. Вывести на экран сводную информацию: среднюю температуру и количества

осадков по каждому городу, дни с наибольшей и наименьшей температурой, а также

информацию о самых холодных и теплых городах.

**Задание 8**

Представьте, что вы разрабатываете программу для книжного магазина, которая

помогает вести учет продаж и оценивать популярность книг различных жанров. У вас есть

данные о продажах книг за определенный период.

Ваша задача - написать программу на Python, выполняющую следующие действия:

1. Создать функцию для определения самых продаваемых книг по общему

количеству проданных экземпляров за весь период данных.

2. Реализовать функцию для расчета общей выручки от продаж книг каждого автора.

3. Создать функцию для анализа популярности книг различных жанров с помощью

подсчета количества проданных книг каждого жанра.

4. Вывести на экран сводную информацию: самые продаваемые книги, авторы с

наибольшей выручкой, популярность книг различных жанров.

**Задание 9**

Представьте, что вы создаете программу для компании, занимающейся логистикой,

которая хочет оптимизировать маршруты доставки грузов. У вас есть данные о

расположении складов, клиентов и протяженности дорог между ними.

Ваша задача - написать программу на Python, выполняющую следующие действия:

1. Создать функцию для построения оптимальных маршрутов доставки из каждого склада к каждому клиенту с учетом расстояний и времени в пути.

2. Реализовать функцию для расчета общего времени, затраченного на доставкугрузов из каждого склада.

3. Создать функцию для определения самого удаленного клиента от каждого склада.

4. Вывести на экран сводную информацию: оптимальные маршруты доставки, общее время доставки из каждого склада, информацию о самых удаленных клиентах от складов.

**Выполнение лабораторной**

Разделим нашу реализацию на несколько пунктов. То есть будем решать каждую поставленную задачу в виде отдельного файла с предоставлением функций её выполнения. У нас будет главный файл (*main.py*), который мы запускаем и где будем импортировать решенные задания.

Файл exercise7.py

def average\_temperature\_and\_precipitation(data):  
 city\_data = {}  
 for city, city\_weather in data.items():  
 temperatures = [day['temperature'] for day in city\_weather]  
 precipitation = [day['precipitation'] for day in city\_weather]  
 avg\_temperature = sum(temperatures) / len(temperatures)  
 total\_precipitation = sum(precipitation)  
 city\_data[city] = {'avg\_temperature': avg\_temperature, 'total\_precipitation': total\_precipitation}  
 return city\_data  
  
def min\_max\_temperature\_days(data):  
 min\_max\_days = {}  
 for city, city\_weather in data.items():  
 temperatures = [day['temperature'] for day in city\_weather]  
 min\_temp\_day = city\_weather[temperatures.index(min(temperatures))]['date']  
 max\_temp\_day = city\_weather[temperatures.index(max(temperatures))]['date']  
 min\_max\_days[city] = {'min\_temp\_day': min\_temp\_day, 'max\_temp\_day': max\_temp\_day}  
 return min\_max\_days  
  
def coldest\_and\_warmest\_cities(data):  
 city\_temperatures = {city: sum([day['temperature'] for day in city\_weather]) / len(city\_weather) for city, city\_weather in data.items()}  
 coldest\_city = min(city\_temperatures, key=city\_temperatures.get)  
 warmest\_city = max(city\_temperatures, key=city\_temperatures.get)  
 return coldest\_city, warmest\_city  
  
def print\_summary(data):  
 avg\_temp\_precipitation = average\_temperature\_and\_precipitation(data)  
 min\_max\_days = min\_max\_temperature\_days(data)  
 coldest\_city, warmest\_city = coldest\_and\_warmest\_cities(data)  
  
 print("Average Temperature and Total Precipitation:")  
 for city, city\_data in avg\_temp\_precipitation.items():  
 print(f"{city}: Avg Temp: {city\_data['avg\_temperature']}, Total Precipitation: {city\_data['total\_precipitation']}")  
  
 print("\nDays with Minimum and Maximum Temperature:")  
 for city, days in min\_max\_days.items():  
 print(f"{city}: Min Temp Day: {days['min\_temp\_day']}, Max Temp Day: {days['max\_temp\_day']}")  
  
 print("\nColdest and Warmest Cities:")  
 print(f"Coldest City: {coldest\_city}")  
 print(f"Warmest City: {warmest\_city}")  
  
# Пример данных о погоде  
weather\_data = {  
 'City1': [  
 {'date': '2024-05-01', 'temperature': 20, 'precipitation': 5},  
 {'date': '2024-05-02', 'temperature': 22, 'precipitation': 2},  
 {'date': '2024-05-03', 'temperature': 18, 'precipitation': 7}  
 ],  
 'City2': [  
 {'date': '2024-05-01', 'temperature': 15, 'precipitation': 8},  
 {'date': '2024-05-02', 'temperature': 17, 'precipitation': 3},  
 {'date': '2024-05-03', 'temperature': 14, 'precipitation': 6}  
 ]  
}  
def print\_solution():  
 print\_summary(weather\_data)

Файл exercise8.py

def top\_selling\_books(data):  
 book\_sales = {}  
 for book in data:  
 title = book['title']  
 if title in book\_sales:  
 book\_sales[title] += book['copies\_sold']  
 else:  
 book\_sales[title] = book['copies\_sold']  
 top\_books = sorted(book\_sales.items(), key=lambda x: x[1], reverse=True)  
 return top\_books  
  
def author\_revenue(data):  
 author\_sales = {}  
 for book in data:  
 author = book['author']  
 if author in author\_sales:  
 author\_sales[author] += book['price'] \* book['copies\_sold']  
 else:  
 author\_sales[author] = book['price'] \* book['copies\_sold']  
 top\_authors = sorted(author\_sales.items(), key=lambda x: x[1], reverse=True)  
 return top\_authors  
  
def genre\_popularity(data):  
 genre\_sales = {}  
 for book in data:  
 genre = book['genre']  
 if genre in genre\_sales:  
 genre\_sales[genre] += book['copies\_sold']  
 else:  
 genre\_sales[genre] = book['copies\_sold']  
 return genre\_sales  
  
def print\_summary(data):  
 top\_books = top\_selling\_books(data)  
 top\_authors = author\_revenue(data)  
 genre\_sales = genre\_popularity(data)  
  
 print("Top Selling Books:")  
 for book, copies\_sold in top\_books:  
 print(f"{book}: {copies\_sold} copies sold")  
  
 print("\nTop Authors by Revenue:")  
 for author, revenue in top\_authors:  
 print(f"{author}: ${revenue}")  
  
 print("\nGenre Popularity:")  
 for genre, copies\_sold in genre\_sales.items():  
 print(f"{genre}: {copies\_sold} copies sold")  
  
# Пример данных о продажах книг  
book\_sales\_data = [  
 {'title': 'Book1', 'author': 'Author1', 'genre': 'Fiction', 'price': 20, 'copies\_sold': 100},  
 {'title': 'Book2', 'author': 'Author2', 'genre': 'Fantasy', 'price': 25, 'copies\_sold': 150},  
 {'title': 'Book3', 'author': 'Author1', 'genre': 'Non-Fiction', 'price': 18, 'copies\_sold': 80},  
 {'title': 'Book4', 'author': 'Author3', 'genre': 'Fantasy', 'price': 22, 'copies\_sold': 120},  
 {'title': 'Book5', 'author': 'Author2', 'genre': 'Fiction', 'price': 20, 'copies\_sold': 90}  
]  
  
def print\_solution():  
 print\_summary(book\_sales\_data)

Файл exercise9.py

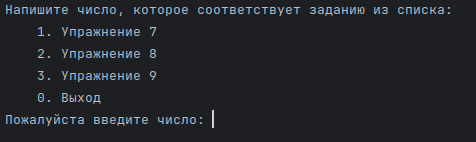
import networkx as nx  
  
def build\_optimal\_routes(warehouses, clients, road\_distances):  
 optimal\_routes = {}  
 for warehouse in warehouses:  
 optimal\_routes[warehouse] = {}  
 for client in clients:  
 shortest\_path = nx.dijkstra\_path(road\_distances, warehouse, client)  
 optimal\_routes[warehouse][client] = shortest\_path  
 return optimal\_routes  
  
def total\_delivery\_time(warehouses, clients, road\_distances, average\_speed):  
 delivery\_times = {}  
 for warehouse in warehouses:  
 total\_time = 0  
 for client in clients:  
 shortest\_path = nx.dijkstra\_path\_length(road\_distances, warehouse, client)  
 total\_time += shortest\_path / average\_speed  
 delivery\_times[warehouse] = total\_time  
 return delivery\_times  
  
def furthest\_clients(warehouses, clients, road\_distances):  
 furthest = {}  
 for warehouse in warehouses:  
 max\_distance = 0  
 furthest\_client = None  
 for client in clients:  
 distance = nx.dijkstra\_path\_length(road\_distances, warehouse, client)  
 if distance > max\_distance:  
 max\_distance = distance  
 furthest\_client = client  
 furthest[warehouse] = furthest\_client, max\_distance  
 return furthest  
  
def print\_summary(warehouses, optimal\_routes, delivery\_times, furthest\_clients):  
 print("Optimal Routes:")  
 for warehouse, routes in optimal\_routes.items():  
 print(f"From Warehouse {warehouse}:")  
 for client, route in routes.items():  
 print(f"To Client {client}: {' -> '.join(route)}")  
  
 print("\nTotal Delivery Time from Each Warehouse:")  
 for warehouse, time in delivery\_times.items():  
 print(f"Warehouse {warehouse}: {time} hours")  
  
 print("\nFurthest Clients from Each Warehouse:")  
 for warehouse, (client, distance) in furthest\_clients.items():  
 print(f"From Warehouse {warehouse} to Client {client}: {distance} units")  
  
# Пример данных о расположении складов, клиентов и расстояниях между ними  
warehouses = ['Warehouse1', 'Warehouse2']  
clients = ['Client1', 'Client2', 'Client3']  
road\_distances = nx.Graph()  
road\_distances.add\_edge('Warehouse1', 'Client1', weight=10)  
road\_distances.add\_edge('Warehouse1', 'Client2', weight=15)  
road\_distances.add\_edge('Warehouse1', 'Client3', weight=20)  
road\_distances.add\_edge('Warehouse2', 'Client1', weight=12)  
road\_distances.add\_edge('Warehouse2', 'Client2', weight=18)  
road\_distances.add\_edge('Warehouse2', 'Client3', weight=25)  
  
average\_speed = 50 # средняя скорость доставки в км/ч  
  
# Вызываем функции для получения сводной информации  
optimal\_routes = build\_optimal\_routes(warehouses, clients, road\_distances)  
delivery\_times = total\_delivery\_time(warehouses, clients, road\_distances, average\_speed)  
furthest\_clients\_info = furthest\_clients(warehouses, clients, road\_distances)  
  
def print\_solution():  
 print\_summary(warehouses, optimal\_routes, delivery\_times, furthest\_clients\_info)

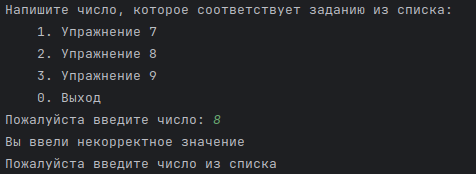
Главный файл (*main.py*)

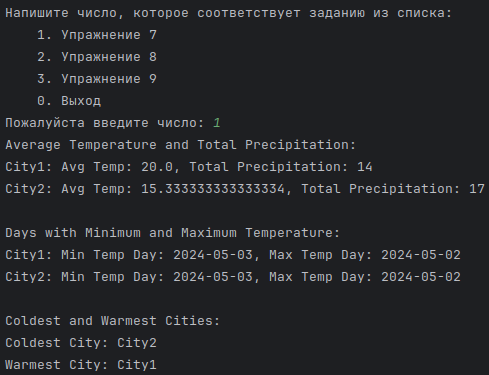
from Exercise import exercise8 as eight, exercise9 as nine, exercise7 as seven  
  
info = '''Напишите число, которое соответствует заданию из списка:  
\t1. Упражнение 7  
\t2. Упражнение 8  
\t3. Упражнение 9  
\t0. Выход'''  
while True:  
 print(info)  
 a = input('Пожалуйста введите число: ')  
 match a:  
 case '1':  
 seven.print\_solution()  
 print("\n\n")  
 case '2':  
 eight.print\_solution()  
 print("\n\n")  
 case '3':  
 nine.print\_solution()  
 print("\n\n")  
 case '0':  
 print('Bye-bye :(')  
 break  
 case \_:  
 print("Вы ввели некорректное значение\nПожалуйста введите число из списка")  
 print("\n\n")е

**Результаты выполнения каждого файла**

При запуске главного файла (*main.py*) мы можем видеть следующую картину:

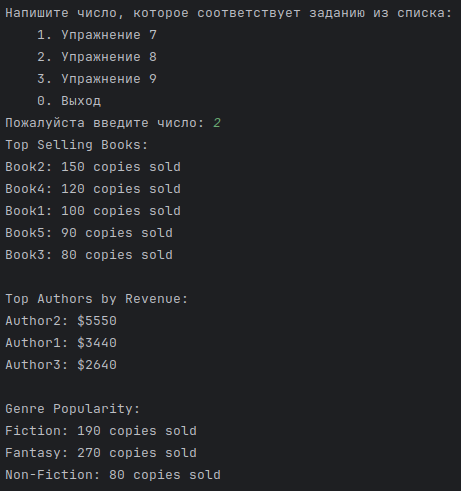
  
Сразу видим информацию о том, какой набор функций есть у нашей программы. При неверном вводе будет следующее:

  
Теперь введем номер с соответствующем ему решением задачи и посмотрим какой будет вывод:

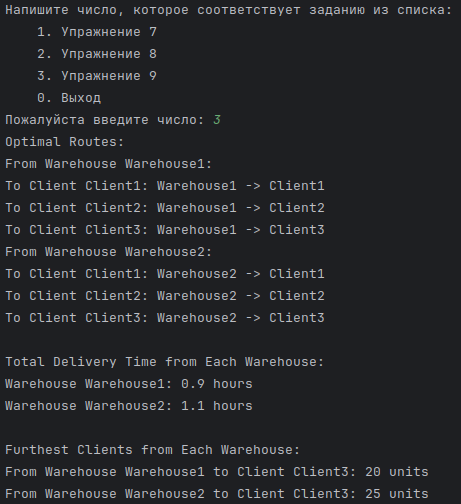


Успешное выполнение, мы можем лицизреть выполнение всех пунктов задания.

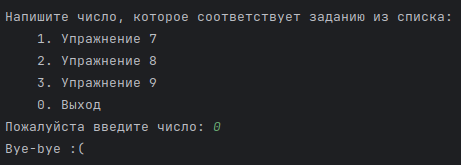
Теперь посмотрим какой вывод будет при избрании следующего пункта из списка:

  
И тут мы видим успешное выполнение. Каждый из пунктов был выполнен, а также есть приглядный глазу вывод.

Теперь посмотрим на предпоследний пункт нашего списка:

  
И снова видим реализацию с вполне читабельной информацией и понятными данными.

Ну и наконец последний пункт – выход из нашего приложения, посмотрим каков будет вывод:

Ну и вот такой вывод мы имеем в конце.

**Вывод**

В ходе проделанной работы я ознакомилась: с последовательным выполнением поставленных задач, с продумыванием архитектуры приложения и поиском оптимальных решений. Поиском библиотек, которые могли упростить решение поставленной задачи и оптимизировать работу приложения.

