8 - зертханалық жұмыс: Функционалды бағдарламалау арқылы нақты

жағдайларды талдау

13. Функционалды жаңалықтар агрегаторы

- Дереккөздерді функционалды түрде өңдейтін және пайдаланушы интерфейсін

қамтамасыз ететін жаңалықтар агрегаторын әзірлеу.

RSS (Really Simple Syndication) - это стандартный формат для сбора и предоставления содержимого веб-сайтов в удобной для чтения форме. Он позволяет пользователям получать обновления содержимого веб-сайтов без необходимости посещать их непосредственно.

RSS-ленты - это файлы, содержащие список последних обновлений с веб-сайта. Они содержат заголовки статей, краткое содержание и ссылки на полный текст. Эти ленты могут быть использованы различными программами и сервисами для отслеживания обновлений и предоставления пользователю актуальной информации.

Feedparser - это модуль Python, который позволяет разбирать RSS-ленты и извлекать из них данные, такие как заголовки новостей, описания и ссылки. Он предоставляет простой способ получения и анализа содержимого RSS-ленты в вашем Python-коде.

import feedparser

import tkinter as tk

def fetch\_news(feed\_url):

"""

Функция для получения новостей из RSS-ленты.

"""

feed = feedparser.parse(feed\_url) #фридпэрс модулінен пэрс функциясын шакырамыз,

return [entry.title for entry in feed.entries] бул рсс лентадан жаналыктар такырыбынын тызымын кайтарады

def display\_news(news\_list): корсету

"""

Функция для отображения списка новостей в интерфейсе.

"""

for idx, news in enumerate(news\_list, start=1): реттік номерімен тызымдеу

label = tk.Label(window, text=f"{idx}. {news}")

label.pack()

def fetch\_and\_display\_news():

"""

Функция для получения и отображения новостей при нажатии кнопки.

"""

news\_list = fetch\_news(entry\_url.get())

display\_news(news\_list)

# Создание графического интерфейса

window = tk.Tk()

window.title("Функциональный агрегатор новостей")

# Виджеты

tk.Label(window, text="Введите URL RSS-ленты:").pack()

entry\_url = tk.Entry(window, width=50)

entry\_url.pack()

button\_fetch = tk.Button(window, text="Получить новости", command=fetch\_and\_display\_news)

button\_fetch.pack()

# Запуск главного цикла

window.mainloop()

Код представляет собой простое приложение на Python с использованием библиотеки `feedparser` для получения новостей из RSS-ленты и библиотеки `tkinter` для создания графического интерфейса.

1. `import feedparser`: Этот оператор импортирует модуль `feedparser`, который используется для парсинга RSS-ленты.

2. `import tkinter as tk`: Этот оператор импортирует модуль `tkinter` и позволяет обращаться к нему с использованием сокращения `tk`.

3. `def fetch\_news(feed\_url):`: Это определение функции `fetch\_news`, которая получает новости из RSS-ленты по заданному URL.

4. `feed = feedparser.parse(feed\_url)`: Здесь происходит вызов функции `parse` из модуля `feedparser`, которая выполняет парсинг RSS-ленты по указанному URL.

5. `return [entry.title for entry in feed.entries]`: Функция возвращает список заголовков новостей из полученных записей RSS-ленты.

6. `def display\_news(news\_list):`: Это определение функции `display\_news`, которая отображает список новостей в графическом интерфейсе.

7. `for idx, news in enumerate(news\_list, start=1):`: Здесь используется функция `enumerate`, чтобы перечислить каждую новость в списке `news\_list` с указанием их порядкового номера, начиная с 1.

8. `label = tk.Label(window, text=f"{idx}. {news}")`: Создается метка (`Label`), которая содержит порядковый номер новости и саму новость.

9. `label.pack()`: Метка добавляется в графический интерфейс с помощью метода `pack`, чтобы отобразить ее.

10. `def fetch\_and\_display\_news():`: Это определение функции `fetch\_and\_display\_news`, которая вызывает функцию `fetch\_news` для получения новостей и затем функцию `display\_news` для их отображения.

11. `window = tk.Tk()`: Создается основное окно приложения с помощью конструктора `Tk` из модуля `tkinter`.

12. `window.title("Функциональный агрегатор новостей")`: Устанавливается заголовок для окна приложения.

13. `tk.Label(window, text="Введите URL RSS-ленты:").pack()`: Создается текстовая метка, запрашивающая у пользователя ввести URL RSS-ленты, и она добавляется в главное окно.

14. `entry\_url = tk.Entry(window, width=50)`: Создается поле ввода для URL RSS-ленты.

15. `entry\_url.pack()`: Поле ввода добавляется в главное окно.

16. `button\_fetch = tk.Button(window, text="Получить новости", command=fetch\_and\_display\_news)`: Создается кнопка с названием "Получить новости", и при ее нажатии будет вызываться функция `fetch\_and\_display\_news`.

17. `button\_fetch.pack()`: Кнопка добавляется в главное окно.

18. `window.mainloop()`: Этот метод запускает основной цикл обработки событий, который ожидает событий от пользователей (например, нажатие кнопки или ввод текста) и реагирует на них, пока окно не будет закрыто.

Этот код создает простое окно с текстовым полем для ввода URL RSS-ленты и кнопку "Получить новости". При нажатии кнопки он парсит новости из указанной RSS-ленты и отображает их в виде меток на экране.

Дайындық сұрақтары:

1. Функционалдық бағдарламалау принциптерін жобаңызға қалай кіріктірдіңіз?

Функциональное программирование в проекте\*\*: В коде использованы функции высшего порядка (например, `fetch\_and\_display\_news` передается в качестве аргумента для `command` в `Button`) и функции первого класса (как `fetch\_news` и `display\_news`), что позволяет создавать более компактный и легко читаемый код.

2. Жобаны жасау кезінде қандай функционалдық үлгілерді немесе

конструкцияларды пайдаландыңыз?

Функциональные элементы\*\*: Проект использует функции для получения новостей и отображения их в пользовательском интерфейсе. Это упрощает управление и обслуживание кода.

3. Жобамен жұмыс істеу барысында функционалдық тәсілдің қандай

артықшылықтарын байқадыңыз?

Преимущества функционального подхода\*\*: Функции `fetch\_news` и `display\_news` обеспечивают отделение логики получения данных от логики их отображения, что делает код более модульным и гибким.

4. Функционалдық бағдарламалауды қолдану кезінде қандай қиындықтарға тап

болдыңыз және оларды қалай жеңдіңіз?

Сложности асинхронного программирования\*\*: В проекте нет явной асинхронной логики, но если бы требовалось асинхронно загружать новости из сети, это могло бы вызвать сложности, которые можно было бы решить с использованием асинхронных функций и библиотек, таких как `asyncio`.

5. Сіздің жобаңыздың қандай аспектілерінде функционалдық бағдарламалау

әсіресе пайдалы болды?

Практические преимущества функционального подхода\*\*: Функциональный подход делает код более модульным, легким для понимания и поддержки. Он также способствует повторному использованию кода и упрощает тестирование.

6. Функционалдық бағдарламалаудың тестілеуге және кодты жөндеуге қалай әсер

ететінін сипаттай аласыз ба?

Тестирование функционального кода\*\*: Функциональные элементы кода, такие как `fetch\_news` и `display\_news`, могут быть легко протестированы с помощью модульных тестов для проверки их корректности и надежности.

7. Жобаңызда модульдік пен композицияны қалай қамтамасыз еттіңіз?

Модульность и композиция кода\*\*: Проект использует модульную структуру, разделяя логику получения данных и их отображения, что упрощает изменения и расширение функциональности проекта.

8. Функционалдық бағдарламалау үшін қандай Python құралдары мен

кітапханаларын ең пайдалы деп таптыңыз?

Мақсаты: Оқушылар қандай ресурстарды пайдаланғанын және олардың

қаншалықты тиімді болғанын анықтаңыз.

9. Функционалдық бағдарламалаудың қандай аспектілерін түсіну немесе қолдану

сізге қиынға соғады?

Использование Python для функционального программирования\*\*: Python предоставляет множество инструментов и библиотек для функционального программирования, что делает его подходящим выбором для проектов такого типа.

10. Жобамен жұмыс істеу барысында функционалдық бағдарламалаудың қандай

шектеулерін таптыңыз?

Улучшение архитектуры проекта\*\*: Функциональное программирование способствует созданию чистой и модульной архитектуры, что делает проект более гибким и легким для поддержки и развития.

11. Функционалды бағдарламалау жобаңыздың өнімділігіне қалай әсер етті?

Мақсаты: Студенттердің функционалдық бағдарламалаудың өнімділікке әсері

туралы түсінігін бағалау.

12. Функционалдық тұжырымдамаларды пайдалана отырып, жобаңызға

жақсартуларды ұсына аласыз ба?

Мақсаты: Студенттерді рефлексия жасауға және өз жобаларын жақсартуды

ұсынуға ынталандыру.

13. Жобаңызды масштабтау немесе жаңа мүмкіндіктерді қосу үшін қалай

бейімдеуге болады?

Мақсаты: Студенттердің өз жобасының әрі қарай дамуын және масштабын

жоспарлауын қаншалықты жақсы меңгергенін түсіну.

14. Жобаңызда жанама әсерлерді басқару үшін қандай әдістерді қолдандыңыз?

Мақсаты: Студенттердің функционалдық бағдарламалаудағы жанама

әсерлерді басқару әдістері туралы білімдерін тексеру.

1. \*\*Функциональное программирование в проекте\*\*: В коде использованы функции высшего порядка (например, `fetch\_and\_display\_news` передается в качестве аргумента для `command` в `Button`) и функции первого класса (как `fetch\_news` и `display\_news`), что позволяет создавать более компактный и легко читаемый код.

2. \*\*Функциональные элементы\*\*: Проект использует функции для получения новостей и отображения их в пользовательском интерфейсе. Это упрощает управление и обслуживание кода.

3. \*\*Преимущества функционального подхода\*\*: Функции `fetch\_news` и `display\_news` обеспечивают отделение логики получения данных от логики их отображения, что делает код более модульным и гибким.

4. \*\*Сложности асинхронного программирования\*\*: В проекте нет явной асинхронной логики, но если бы требовалось асинхронно загружать новости из сети, это могло бы вызвать сложности, которые можно было бы решить с использованием асинхронных функций и библиотек, таких как `asyncio`.

5. \*\*Практические преимущества функционального подхода\*\*: Функциональный подход делает код более модульным, легким для понимания и поддержки. Он также способствует повторному использованию кода и упрощает тестирование.

6. \*\*Тестирование функционального кода\*\*: Функциональные элементы кода, такие как `fetch\_news` и `display\_news`, могут быть легко протестированы с помощью модульных тестов для проверки их корректности и надежности.

7. \*\*Модульность и композиция кода\*\*: Проект использует модульную структуру, разделяя логику получения данных и их отображения, что упрощает изменения и расширение функциональности проекта.

8. \*\*Использование Python для функционального программирования\*\*: Python предоставляет множество инструментов и библиотек для функционального программирования, что делает его подходящим выбором для проектов такого типа.

9. \*\*Сложности, возникающие в ходе разработки\*\*: В ходе разработки могут возникнуть сложности при управлении состоянием и обработке ошибок, но они могут быть успешно преодолены с помощью правильного применения функциональных принципов.

10. \*\*Улучшение архитектуры проекта\*\*: Функциональное программирование способствует созданию чистой и модульной архитектуры, что делает проект более гибким и легким для поддержки и развития.

Надеюсь, это ответило на ваши вопросы о функциональном подходе в предоставленном коде! Если у вас есть еще вопросы, не стесняйтесь задавать.