**1. Основные библиотеки**

В этом коде используются несколько ключевых библиотек для работы с Telegram API, обработки данных и работы с Excel:

**Библиотеки:**

* **logging** — стандартная библиотека для логирования сообщений. Мы используем её для отслеживания событий и ошибок в работе бота.
* **pandas** — это библиотека для обработки и анализа данных, в данном случае, для работы с таблицами (Excel). Мы используем её для хранения, обновления и чтения данных о расходах.
* **datetime** — стандартная библиотека для работы с датами и временем, например, для получения текущей даты.
* **telegram** — библиотека для работы с Telegram Bot API. Она используется для взаимодействия с пользователями через Telegram, отправки сообщений, обработки команд и работы с медиафайлами (в нашем случае — Excel файлами).
* **telegram.ext** — часть библиотеки python-telegram-bot, предоставляющая расширенные возможности для создания и обработки ботов в Telegram.
* **io.BytesIO** — используется для работы с байтовыми потоками в памяти, что позволяет отправлять файлы без необходимости их сохранения на диск.

**2. Основные функции и их работа**

**create\_expenses\_file()**

Эта функция проверяет наличие файла с расходами (expenses.xlsx). Если файл не существует, она создает новый Excel файл с необходимыми колонками: **Дата**, **Сумма** и **Описание**.

python

КопироватьРедактировать

def create\_expenses\_file():

try:

df = pd.read\_excel(EXCEL\_FILE)

except FileNotFoundError:

df = pd.DataFrame(columns=["Дата", "Сумма", "Описание"])

df.to\_excel(EXCEL\_FILE, index=False)

**record\_expense(date, amount, description)**

Эта функция записывает новый расход в Excel файл. Она считывает текущие данные из файла, добавляет новую строку с переданными значениями (дата, сумма и описание) и сохраняет обновленный файл.

python

КопироватьРедактировать

def record\_expense(date: str, amount: float, description: str):

df = pd.read\_excel(EXCEL\_FILE)

new\_row = pd.DataFrame({"Дата": [date], "Сумма": [amount], "Описание": [description]})

df = pd.concat([df, new\_row], ignore\_index=True)

df.to\_excel(EXCEL\_FILE, index=False)

**show\_expenses(date=None)**

Функция для получения расходов за определённую дату или всех расходов за весь период. Если параметр **date** передан, фильтруется таблица по этой дате. Если **date** не передан, показываются все расходы.

python

КопироватьРедактировать

def show\_expenses(date: str = None):

df = pd.read\_excel(EXCEL\_FILE)

df['Дата'] = pd.to\_datetime(df['Дата'])

if date:

filtered\_df = df[df['Дата'].dt.strftime('%Y-%m-%d') == date]

else:

filtered\_df = df

if filtered\_df.empty:

return "Нет расходов за указанную дату."

return "\n".join([f"{row['Дата'].strftime('%Y-%m-%d')} - {row['Сумма']} - {row['Описание']}" for index, row in filtered\_df.iterrows()])

**send\_expense\_file(update)**

Функция отправляет файл Excel с расходами в виде документа пользователю. Для этого данные из файла читаются в память и сохраняются в **BytesIO** (память вместо файла). Затем этот файл отправляется через Telegram API.

python

КопироватьРедактировать

async def send\_expense\_file(update: Update):

df = pd.read\_excel(EXCEL\_FILE)

with BytesIO() as output:

df.to\_excel(output, index=False)

output.seek(0)

await update.message.reply\_document(document=output, filename="expenses.xlsx")

**Команды и обработчики**

Основные команды бота:

1. **/start** — Эта команда запускает бота, отправляя приветственное сообщение с инструкциями по использованию бота.

python

КопироватьРедактировать

async def start(update: Update, context: CallbackContext) -> None:

welcome\_text = "..."

await update.message.reply\_text(welcome\_text)

1. **/add <сумма> <описание>** — Эта команда добавляет новый расход. Пользователь передает сумму и описание, и они записываются в Excel файл.

python

КопироватьРедактировать

async def add\_expense(update: Update, context: CallbackContext) -> None:

...

record\_expense(date, amount, description)

await update.message.reply\_text(f"Расход {amount} за {date} успешно добавлен!")

1. **/show <дата>** — Эта команда показывает все расходы за определённую дату. Если дата не передана, бот покажет все расходы.

python

КопироватьРедактировать

async def show\_expenses\_command(update: Update, context: CallbackContext) -> None:

...

expenses = show\_expenses(date)

await update.message.reply\_text(expenses)

1. **/file** — Эта команда отправляет пользователю файл Excel с его расходами.

python

КопироватьРедактировать

async def file(update: Update, context: CallbackContext) -> None:

await send\_expense\_file(update)

**3. Как работает основная логика**

1. **Создание и обновление Excel файла:**
   * Файл расходов создается при первом запуске бота (если его нет). Он хранит все данные о расходах, которые вводит пользователь.
   * При каждом добавлении нового расхода данные записываются в файл с помощью библиотеки pandas.
2. **Интерактивность с пользователем:**
   * Пользователь может использовать команды бота для записи расходов, отображения расходов за конкретную дату или период, а также для получения файла с расходами в формате Excel.
3. **Отправка файлов:**
   * Когда пользователь вызывает команду /file, бот отправляет ему файл Excel с его расходами. Для этого данные загружаются в память и передаются пользователю через Telegram API.

**4. Как бот работает с библиотеками**

* **python-telegram-bot** — используется для взаимодействия с Telegram API. В коде используется **Application** для асинхронного запуска бота, а также **CommandHandler** и **MessageHandler** для обработки команд.
* **pandas** — библиотека используется для удобной работы с таблицами Excel, обработки данных о расходах (чтение и запись в файл Excel).

**5. Основные шаги работы бота:**

1. Бот запускается с командой /start, приветствует пользователя и отправляет информацию о доступных командах.
2. Пользователь может использовать команду /add для добавления нового расхода, который сохраняется в Excel файл.
3. Пользователь может вызвать команду /show для получения списка всех расходов или расходов за конкретную дату.
4. При команде /file бот отправляет файл Excel с данными о расходах.

**6. Что нужно для работы с ботом:**

1. **Телеграм-бот токен** — для работы с Telegram API необходимо создать бота через BotFather в Telegram и получить токен.
2. **Python версии 3.7 и выше** — для работы с этим кодом требуется Python.
3. **Необходимые библиотеки** — для установки всех библиотек можно использовать команду:

bash

КопироватьРедактировать

pip install python-telegram-bot pandas openpyxl

**Заключение**

Этот бот помогает пользователю вести учёт своих расходов, используя Telegram для взаимодействия и pandas для хранения данных в Excel. Вы можете добавлять расходы, просматривать их за определённые даты, а также получать полные отчёты в виде Excel-файлов.