

Практическая работа 8.3 - Менеджер «With ... as» для работы с файлами

При работе с файлами зачастую нужно отслеживать исключения. Делать это лишь при помощи «try – except» не особо удобно.

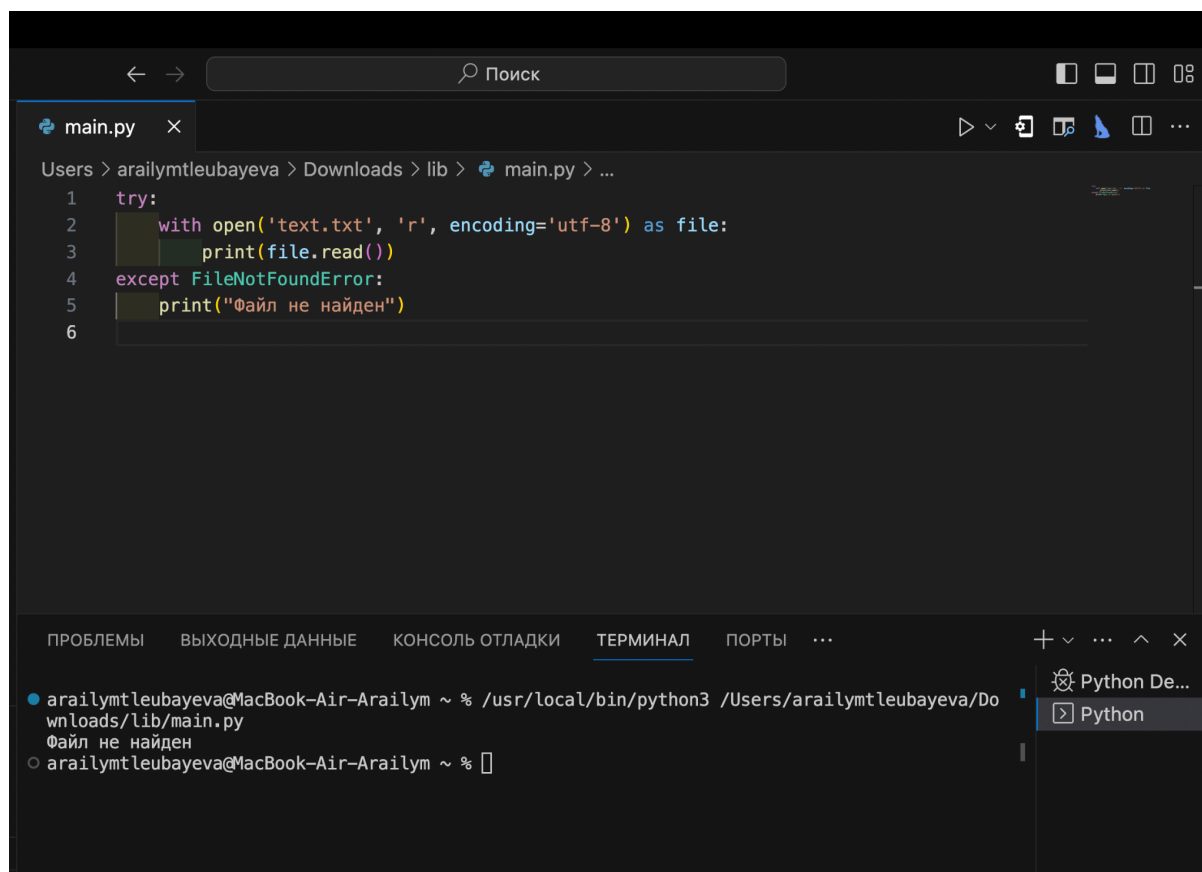
With as – конструкция, которая способна выполнить код сто процентов. Что это означает? Если при работе с файлами мы используем метод open, то дополнительно нам необходимо ещё и закрывать файл, иначе это чревато последствиями.

При работе с With as мы можем быть уверены что файл точно будет закрыт, даже в случае неправильного выполнения функции.

Конструкция with ... as в Python действительно обеспечивает удобный и безопасный способ работы с файлами, поскольку она гарантирует автоматическое закрытие файла после выхода из блока with. Это происходит даже в случае возникновения исключения внутри этого блока. Такой подход называется "менеджером контекста" и является примером паттерна проектирования, который называется "Освобождение ресурса есть инициализация" (Resource Acquisition Is Initialization, RAII).

Пример работы с файлом без использования with ... as может выглядеть следующим образом:

Пример, Оператор «With ... as»



```
main.py x
Users > arailymtleubayeva > Downloads > lib > main.py > ...
1  try:
2      with open('text.txt', 'r', encoding='utf-8') as file:
3          print(file.read())
4  except FileNotFoundError:
5      print("Файл не найден")
6

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ  ...
● arailymtleubayeva@MacBook-Air-Araily ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Do
wnloads/lib/main.py
Файл не найден
○ arailymtleubayeva@MacBook-Air-Araily ~ %
```

В этом случае нам нужно явно закрыть файл в блоке `finally`, чтобы не допустить утечек ресурсов.

С другой стороны, использование `with ... as` упрощает управление ресурсами и делает код более чистым и безопасным:

```
1 try:
2     file = open('/Users/arailymtleubayeva/Documents/example.txt', 'r')
3     data = file.read()
4     # Делаем что-то с данными
5 except IOError as e:
6     # Обрабатываем возможные исключения во время работы с файлом
7     print(f'Произошла ошибка: {e}')
8 finally:
9     # Убеждаемся, что файл закрывается
10    file.close()
11
```

PROБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ **ТЕРМИНАЛ** ... + - Python

```
/usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
○ arailymtleubayeva@MacBook-Air-Arilym ~ %
```

```
1 try:
2     with open('/Users/arailymtleubayeva/Documents/example.txt', 'r') as file:
3         data = file.read()
4         # Делаем что-то с данными
5 except IOError as e:
6     # Файл автоматически закрывается после выхода из блока with, даже если возникло исключение
7     print(f'Произошла ошибка: {e}')
8
```

PROБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ **ТЕРМИНАЛ** ПОРТЫ ... + - Python

```
/usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
○ arailymtleubayeva@MacBook-Air-Arilym ~ %
○ arailymtleubayeva@MacBook-Air-Arilym ~ %
```

В этом случае файл будет закрыт автоматически, как только выполнение кода выйдет из блока with, независимо от того, возникло исключение или выполнение кода прошло успешно. Это избавляет программиста от необходимости явно вызывать file.close(), что делает код не только короче, но и уменьшает вероятность ошибок, связанных с управлением файловыми дескрипторами.

Самостоятельная работа

Задание 1. Открытие файла

Откройте файл для записи и пропишите в него «Наш новый файл».

Выполните работу с файлом через менеджер With ... as.

Задание 2. Чтение файла

Выполните запись и чтение файла «some.txt» при помощи менеджера with .. as.

Задание 3. Запись в файл

Напишите скрипт, который запрашивает у пользователя ввод строки и записывает эту строку в текстовый файл user_data.txt. Если файл уже существует, ваш скрипт должен добавлять текст в конец файла, не удаляя его предыдущее содержимое.

Задание 4. Безопасное деление

Реализуйте функцию для безопасного выполнения операции деления. Функция должна запрашивать у пользователя ввод двух чисел и выводить результат деления первого числа на второе. Обработайте возможные исключения, такие как деление на ноль и ввод нечисловых значений.

Задача 5. Перехват специфичных исключений

Создайте список из нескольких различных типов данных (строки, числа, списки и т.д.). Напишите функцию, которая проходит по списку и выполняет определенную операцию с каждым элементом (например, деление на число, добавление строки и т.п.). Ваша функция должна корректно обрабатывать исключения для каждого типа данных и сообщать о них пользователю.

Задача 6. Работа с JSON

Используйте модуль json для работы с данными в формате JSON. Напишите код, который читает JSON из файла data.json, модифицирует данные и сохраняет их обратно в файл. Обработайте возможные исключения, например, если файл не существует или содержит некорректный JSON.

Задача 7. Менеджер контекста

Реализуйте собственный менеджер контекста, который будет работать как таймер: он запоминает время начала блока и печатает, сколько времени прошло, когда выполнение блока with завершено.