Обработка исключений

Date: 08/04/2024

Tochilkina Ksenya Stanislavovna

При выполнении программы могут возникнуть различного рода ошибки (исключения). Нам необходимо уметь отслеживать подобные ошибки и предотвращать их. В ходе урока мы изучим конструкцию «try - except» для отлова и обработки исключений.

Что такое исключение?

Предположим, что вы разработали программу «Текстовый редактор». В программе пользователь может создать новый файл, вписать в него данные и далее сохранить файл в системе.

Если код прописан корректно, то никаких ошибок возникать не будет. Но давайте представим ситуацию, что пользователь открыл редактор, открыл нужный файл, записал в него данные, далее вручную удалил файл с компьютера и потом попытался сохранить файл через вашу программу.

При таком раскладе у вас получиться ошибка, которая сломает программу и отобьет любое желание у пользователя работать в вашей программе.

Получается, **исключение** - это ошибка, что возникает в ходе работы самой программы. Отслеживать такие ошибки при помощи условных операторов не всегда возможно, ведь программа уже запущена, поведение пользователя нам неизвестно заранее, а значит и «ловить» ошибку нам нужно в момент её создания.

Обработка исключений в языке программирования Python - это механизм, который позволяет программистам обрабатывать и управлять ошибками, возникающими во время выполнения программы. Когда возникает ошибка, которая может привести к прерыванию выполнения программы, исключение генерируется, и программа может перехватывать и обрабатывать это исключение.

Конструкция **try-except** в языке программирования Python используется для обработки исключений, то есть ошибок, которые могут возникнуть во время выполнения программы. Она позволяет программистам предусмотреть возможные ошибки и определить, как обрабатывать их, чтобы избежать прерывания выполнения программы.

Основная структура конструкции try-except выглядит следующим образом:

```
try:
    # Код, который может вызвать исключение
except Имя_исключения:
    # Код, который обрабатывает исключение
```

В блоке **try** помещается код, который может вызвать исключение. Если во время выполнения этого кода происходит исключение, оно перехватывается блоком **except**, и код внутри блока **except** выполняется. Блоков с исключениями (except) может быть сколько угодно. Имя_исключения - это тип исключения, которое мы хотим перехватить и обработать. Их существует огромное множество, некоторые из них:

- **SyntaxError**: возникает, когда обнаружена синтаксическая ошибка в коде.
- **ZeroDivisionError:** возникает, когда попытка деления на ноль.
- **TypeError:** возникает, когда операция применяется к объекту неправильного типа.
- **ValueError:** возникает, когда функция получает аргумент правильного типа, но с недопустимым значением.
- **FileNotFoundError:** возникает, когда файл, на который ссылается программа, не найден.
- **IndexError:** возникает, когда индекс находится вне диапазона элементов списка или строки.
- **KeyError:** возникает, когда указанный ключ отсутствует в словаре.
- **NameError**: возникает, когда имя переменной не определено.
- **AttributeError:** возникает, когда пытаемся обратиться к атрибуту или методу объекта, которого не существует.
- **IOError:** возникает, когда происходит ошибка ввода-вывода, например, при попытке открыть несуществующий файл.

Конструкция **try-except** может также содержать блок **finally**, который выполняется независимо от того, возникло исключение или нет. Например:

```
try:
    # Код, который может вызвать исключение
except Имя_исключения:
    # Код, который обрабатывает исключение
finally:
    # Код, который будет выполнен в любом случае
```

Блок **finally** используется для освобождения ресурсов или выполнения других завершающих действий, независимо от того, произошло исключение или нет.

Вот несколько примеров использования конструкции try-except:

Пример 1: Обработка деления на ноль

```
try:
    a = 10
    b = 0
    result = a / b
except ZeroDivisionError:
    print("Ошибка: деление на ноль")
```

Пример 2: Обработка ошибки преобразования типов

```
try:
    value = int("abc")
except ValueError:
    print("Ошибка: невозможно преобразовать строку в число")
```

Пример 3: Обработка нескольких типов исключений

```
try:
    a = 10
    b = 0
    result = a / b
except ZeroDivisionError:
    print("Ошибка: деление на ноль")
except TypeError:
    print("Ошибка: неправильный тип данных")
```

Пример 4: С использованием finally

```
try:
    file = open("example.txt", "r")
    # код для чтения файла
except FileNotFoundError:
    print("Файл не найден")
finally:
    file.close()
    print("Операция завершена")
```

В этом примере мы открываем файл "example.txt" для чтения. Если файл не найден, возникает исключение FileNotFoundError, которое перехватывается блоком except и выводится

сообщение об ошибке. Независимо от того, возникло исключение или нет, блок finally гарантирует, что файл будет закрыт после чтения. В этом блоке мы также выводим сообщение "Операция завершена".

ПРАКТИЧЕСКАЯ РАБОТА

1. Задача: Чтение данных из файла
   - Создайте файл "data.txt" и запишите в него несколько строк текста.
   - Напишите программу, которая открывает файл "data.txt" и выводит содержимое на экран.
   - Если файл не найден, необходимо вывести ошибку "Ошибка: файл не найден"
2. Задача: Запись данных в файл
   - Напишите программу, которая запрашивает у пользователя несколько строк текста или ввести слово exit для выхода.
   - Если пользователь ввел exit прерываем работы программы
   - Откройте файл "output.txt" для записи и запишите в него введенные пользователем строки.
   - Произвести обработку исключения "Ошибка: невозможно записать файл"
3. Задача: Обработка ошибок при чтении файла
   - Напишите программу, которая открывает файл "data.txt" для чтения.
   - Если файл не найден, выведите сообщение об ошибке.
4. Задача: Обработка ошибок при делении чисел
   - Напишите программу, которая запрашивает у пользователя два числа.
   - Попробуйте поделить первое число на второе.
   - Если возникает ошибка деления на ноль, выведите сообщение об ошибке.
5. Задача: Обработка ошибок при преобразовании типов данных
   - Напишите программу, которая запрашивает у пользователя число.
   - Попробуйте преобразовать введенное значение в целое число.
   - Если возникает ошибка преобразования типов, выведите сообщение об ошибке.
6. Задача: Использование блока finally
   - Напишите программу, которая открывает файл "data.txt" для чтения.
   - Если файл не найден, выведите сообщение об ошибке.
   - В блоке finally закройте файл, чтобы гарантировать его закрытие независимо от того, возникла ошибка или нет.