Моя IT Школа

сертифицированные IT курсы

Исключения

План занятия

Моя IT Школа
сертифицированные IT курсы

1 Исключения и их иерархия

2 Конструкция try-except-finally

Исключения

Разберём это сообщение подробнее: интерпретатор нам сообщает о том, что он поймал исключение и напечатал информацию **(Traceback (most recent call last)).**

Далее имя файла **(File "").**

Строка в файле **(line 1).**

Выражение, в котором произошла ошибка **(100 / 0).**

Название исключения **(ZeroDivisionError)** и краткое описание исключения **(division by zero)**.

Исключения

Моя IT Школа

BaseException - базовое исключение, от которого берут начало все остальные.
SystemExit - исключение, порождаемое функцией sys.exit при выходе из программы.
KeyboardInterrupt - порождается при прерывании программы пользователем (обычно сочетанием клавиш Ctrl+C).
Exception – то, на чем фактически строятся все остальные ошибки;
StopIteration - порождается встроенной функцией next, если в итераторе больше нет элементов.
ArithmeticError - арифметическая ошибка.
FloatingPointError - порождается при неудачном выполнении операции с плавающей запятой. На практике встречается нечасто.
OverflowError - возникает, когда результат арифметической операции слишком велик для представления. Не появляется при обычной работе с целыми числами (так как python поддерживает длинные числа), но может возникать в некоторых других случаях.
ZeroDivisionError - деление на ноль.

Конструкция try - except

Моя Школа

Теперь, зная, когда и при каких обстоятельствах могут возникнуть исключения, мы можем их обрабатывать.
Для обработки исключений используется конструкция try - except.

Первый пример применения этой конструкции:

```
try:
    k = 1 / 0
except ZeroDivisionError:
    k = 0

print(k)
```

Конструкция try - except

Моя Школа

В блоке try мы выполняем инструкцию, которая может породить исключение, а в блоке except мы перехватываем их. При этом перехватываются как само исключение, так и его потомки.
Например, перехватывая **ArithmeticError**, мы также перехватываем **FloatingPointError**, **OverflowError** и **ZeroDivisionError**.

```
try:
    k = 1 / 0
except ArithmeticError:
    k = 0

print(k)
```

Конструкция try - except

Моя IT Школа

Также возможна инструкция **except** без аргументов, которая перехватывает вообще всё (и прерывание с клавиатуры, и системный выход и т. д.). Поэтому в такой форме инструкция **except** практически не используется, а используется **except Exception**. Однако чаще всего перехватывают исключения по одному, для упрощения отладки (вдруг вы ещё другую ошибку сделаете, а **except** её перехватит).

Конструкция try - except

Моя Школа

Давайте взглянем еще на несколько примеров:

```
my_dict = {"a": 1, "b": 2, "c": 3}

try:
    value = my_dict["d"]
except KeyError:
    print("Ключа не существует!")
```

```
my_list = [1, 2, 3, 4, 5]

try:
    my_list[6]
except IndexError:
    print("Этого индекса нет в списке!")
```

Конструкция try - except

В первом примере, мы создали словарь из трех элементов. После этого, мы попытались открыть доступ ключу, которого в словаре нет. Так как ключ не в словаре, возникает **KeyError**, которую мы выявили. Второй пример показывает список, длина которого состоит из пяти объектов. Мы попытались взять седьмой объект из индекса.

Помните, что списки в Пайтоне начинаются с нуля, так что когда вы говорите 6, вы запрашиваете 7. В любом случае, в нашем списке только пять объектов, по этой причине возникает **IndexError**, которую мы выявили. Вы также можете выявить несколько ошибок за раз при помощи одного оператора. Для этого существует несколько различных способов.

Конструкция try - except

Моя Школа

```
my_dict = {"a": 1, "b": 2, "c": 3}

try:
    value = my_dict["d"]
except IndexError:
    print("Такого индекса не существует!")
except KeyError:
    print("Этого ключа нет в словаре!")
except:
    print("Произошла другая ошибка!")
```

Конструкция try - except

Моя IT Школа

Это самый стандартный способ **выявить несколько исключений**. Сначала мы попробовали открыть доступ к несуществующему ключу, которого нет в нашем словаре. При помощи **try/except** мы проверили код на наличие **ошибки KeyError**, которая находится во втором операторе **except**. Обратите внимание на то, что в конце кода у нас появилась «голое» исключение. Обычно, это не рекомендуется, но вы, возможно, будете сталкиваться с этим время от времени, так что лучше быть проинформированным об этом. Кстати, также обратите внимание на то, что вам не нужно использовать целый блок кода для обработки нескольких исключений. Обычно, целый блок используется для выявления одного единственного исключения.

Конструкция try - except

Моя Школа

Изучим второй способ выявления нескольких исключений:

```
my_dict = {"a": 1, "b": 2, "c": 3}

try:
    value = my_dict["d"]
except (IndexError, KeyError):
    print("Произошла ошибка IndexError или KeyError!")
```

Конструкция try - except

Моя Школа

Обратите внимание на то, что в данном примере мы помещаем ошибки, которые хотим выявить, внутри круглых скобок. Проблема данного метода в том, что трудно сказать какая именно ошибка произошла, так что предыдущий пример, рекомендую больше чем этот. Зачастую, когда происходит ошибка, вам нужно уведомить пользователя, при помощи сообщения.
В зависимости от сложности данной ошибки, вам может понадобиться выйти из программы. Иногда вам может понадобиться выполнить очистку, перед выходом из программы.

Оператор finally

Моя Школа

Оператор **finally** очень прост в использовании. Давайте взглянем на пример:

```
my_dict = {"a": 1, "b": 2, "c": 3}

try:
    value = my_dict["d"]
except KeyError:
    print("Произошла ошибка KeyError!")
finally:
    print("Оператор finally выполнен!")
```

Finally выполняет блок инструкций в любом случае, было ли исключение, или нет (применима, когда нужно непременно что-то сделать, к примеру, закрыть файл).

except или else

Моя IT Школа

Оператор try/except также имеет пункт else. Он работает только в том случае, если в вашем коде нет ни единой ошибки.

```
my_dict = {"a": 1, "b": 2, "c": 3}

try:
    value = my_dict["a"]
except KeyError:
    print("Произошла ошибка KeyError!")
else:
    print("Ошибок не произошло!")
```

Мы видим словарь, состоящий из трех элементов, и в операторе try/except мы открываем доступ к существующему ключу. Это работает, так что ошибка KeyError не возникает. Так как ошибки нет, else работает, и надпись “Ошибок не произошло!” появляется на экране.

except или else

Моя Школа

Теперь добавим оператор finally:

```
my_dict = {"a": 1, "b": 2, "c": 3}

try:
    value = my_dict["a"]
except KeyError:
    print("Произошла ошибка KeyError!")
else:
    print("Ошибок не произошло!")
finally:
    print("Оператор finally выполнен!")
```

except или else

В данном коде работают и оператор else и finally. Большую часть времени вы не будете сталкиваться с оператором else, используемый в том или ином коде, который следует за оператором try/except, если ни одна ошибка не была найдена. Единственное полезное применение оператора else, которое я видела, это когда вы хотите запустить вторую часть кода, в которой может быть ошибка. Конечно, если ошибка возникает в else, то она не будет поймана.

Задание №3

Введите два числа с клавиатуры. Поделите одно на другое. Обработайте ошибку деления на ноль, если ошибок нет, то результат деления возвести в квадрат. Также используйте оператор finally.

Решение

Моя Школа

```
try:
    a = input("Введите первое число: ")
    b = input("Введите второе число: ")
    result = int(a)/int(b)
except ZeroDivisionError:
    print("Что-то пошло не так...")
else:
    print("Результат в квадрате: ", result**2)
finally:
    print("Конец.")
```

Задание №4

Введите два числа с клавиатуры. Поделите одно на другое.
Обработайте деления на ноль, преобразования и общее исключение.

Решение

Моя Школа

```
try:
    number1 = int(input("Введите первое число: "))
    number2 = int(input("Введите второе число: "))
    print("Результат деления:", number1 / number2)
except ValueError:
    print("Преобразование прошло неудачно")
except ZeroDivisionError:
    print("Попытка деления числа на ноль")
except Exception:
    print("Общее исключение")
print("Завершение программы")
```

Домашнее задание

Задача №1

Ввод с клавиатуры. Если строка введённая с клавиатуры – это числа, то поделить первое на второе. Обработать ошибку деления на ноль. Если второе число 0, то программа запрашивает ввод чисел заново. Также если были введены буквы, то обработать исключение.