Обработка исключений

Иногда во время работы программы возникают ошибки. Такие ошибки в программировании называют исключениями. Мы уже сталкивались с исключением KeyError, которое вызывается, если мы обратились к словарю по несуществующему ключу.

Приведем пример. Предположим, мы пишем программу, которая бесконечно считывает два числа и

1 a = int(input('a = '))  
2 b = int(input('b = '))  
3  
4 result = a / b  
5 print('a / b = ', result)

Вывод на консоль:

a = **1**

b = **0**

Traceback (most recent call last):

File "examle.py", line 4, in <module>

result = a / b

ZeroDivisionError: division by zero

Работа программы завершилась из-за возникновения исключения («ZeroDivisionError» – ошибка деления на ноль) в 4 строке (result = a / b) нашего файла ("examle.py").

В данном случае ошибку можно предотвратить простой проверкой делителя, тогда при делении на 0 выведется сообщение 'Ошибка деления на 0!' и программа продолжит работу:

1 a = int(input('a = '))  
2 b = int(input('b = '))  
3

4 if b != 0:  
5 result = a / b  
6 print('a / b = ', result)

7 else:

8 print('Ошибка деления на 0!')

Вывод на консоль:

a = **1**

b = **0**

Ошибка деления на 0!

Иногда очень сложно предугадать все ошибки в программе, или условия становятся слишком сложными. Python предоставляет возможности для обработки исключений. Для этого существует конструкция try/except:

try:

# код

except [ИмяИсключения]:

# код

«ИмяИсключения» указывает, какое исключение обрабатывает данный блок except.

Рассмотрим код:

try:  
 a = int(input('a = '))  
 b = int(input('b = '))  
  
 result = a / b  
 print('a / b = ', result)  
  
except ZeroDivisionError:  
 print('Нельзя делить на 0!')  
  
print('Все отлично!')

В данном случае блок except обрабатывает ошибку «ZeroDivisionError».

Работа начинается с блока try, затем:

* Если код в блоке try вызывает исключение, которое указано в «ИмяИсключения», то выполняется код, который указан после выражения except;

Вывод на консоль:

a = **1**

b = **0**

Нельзя делить на 0!

Конец

* Если код в блоке try вызывает исключение, которое не указано в «ИмяИсключения», то программа выдаст сообщение об ошибке и завершит работу;

Вывод на консоль:

a = **1**

b = **пять**

Traceback (most recent call last):

File "C:/Users/thisi/PycharmProjects/untitled/book.py", line 3, in <module>

b = int(input('b = '))

ValueError: invalid literal for int() with base 10: 'пять'

* Если код в блоке try не вызывает исключение, то код в блоке except пропускается, и программа продолжает работу.

Вывод на консоль:

a = **1**

b = **5**

a / b = 0.2

Конец

Иногда есть необходимость обработать все исключения сразу. Если «ИмяИсключения» не указано, то блок except будет обрабатывать исключение любого типа, однако, мы не узнаем исключение какого типа было вызвано и не сможем грамотно обработать исключение.

Пример с try/except без «ИмяИсключения»:

while True:  
 try:  
 a = int(input('a = '))  
 b = int(input('b = '))  
  
 result = a / b  
 print('a / b = ', result)  
  
 except:  
 print('Произошла ошибка! Не знаю, какая именно.')

Вывод на консоль:

a = **девять**

Произошла ошибка! Не знаю, какая именно.

a = **2**

b = **0**

Произошла ошибка! Не знаю, какая именно.

Также, можно обрабатывать сразу несколько типов исключений, с помощью конструкции:

try:

# код

except ИмяИсключения1:

# код

except ИмяИсключения2:

# код

…

while True:  
 try:  
 a = int(input('a = '))  
 b = int(input('b = '))  
  
 result = a / b  
 print('a / b = ', result)  
  
 except ValueError:  
 print('Вы ввели значение, которое нельзя привести к числу.')  
  
 except ZeroDivisionError:  
 print('Нельзя делить на 0!')

Вывод на консоль:

a = **девять**

Вы ввели значение, которое нельзя привести к числу.

a = **2**

b = **0**

Нельзя делить на 0!

При написании программы не всегда можно обойтись стандартными исключениями. Например, мы пишем программу, которая спрашивает у пользователя длину и ширину прямоугольника и возвращает его площадь:

a = int(input('Длина прямоугольника = '))  
b = int(input('Ширина прямоугольника = '))  
  
s = a \* b  
print('Площадь прямоугольника =', s)

Как мы знаем длина и ширина прямоугольника должны быть больше 0. Если пользователь введёт некорректные данные, то программа посчитает результат, но ее работа будет некорректной. Поэтому мы можем возбудить собственное исключение с помощью ключевого слова raise:

a = int(input('Длина прямоугольника = '))  
if a <= 0:  
 raise Exception('Длина не может быть отрицательной')  
b = int(input('Ширина прямоугольника = '))  
if b <= 0:  
 raise Exception('Ширина не может быть отрицательной')  
  
s = a \* b  
print('Площадь прямоугольника =', s)

Вывод на консоль:

Длина прямоугольника = **10**

Ширина прямоугольника = **-10**

Traceback (most recent call last):

File "C:/Users/thisi/PycharmProjects/untitled/book.py", line 6, in <module>

raise Exception('Ширина не может быть отрицательной')

Exception: Ширина не может быть отрицательной