except 節による例外の捕捉と対処

例外の捕捉と対処を行う except 節に関しては、次の規則があります。

1. except 節は 1 個以上置ける。
2. 捕捉できるのは、except 節で指定した例外と互換性のある例外である。
3. 単一の except 節で複数の例外をタプルとして指定できる。
4. 捕捉した例外を変数に受け取れる。
5. 捕捉する例外を指定しないこともできる。

1は学習しました。

2は、except 節で指定したクラスだけでなく、その派生クラスも捕捉できる、ということです。すなわち、例外 *B* が例外 *A* の派生クラスであれば、“except *A*:” は、例外 *A* だけでなく、例外 *B* も捕捉します。というのも、“*B* は *A* の一種” だからです。

さて、3を利用して書きかえたのが、**List 12-3** のプログラムです。

List 12-3 chap12/list1203.py

```
# 乗算と除算を行うプログラム（例外処理：その２）

try:
    a = int(input('整数a：'))
    b = int(input('整数b：'))

    print('a * b は', a * b, 'です。')
    print('a / b は', a / b, 'です。')

except (ValueError, ZeroDivisionError):
    print('認識不能orゼロ除算！')

else:
    print('正常終了！')

finally:
    print('お疲れさまでした。')
```

実行例

```
1 整数a：12
  整数b：5
  a * b は 60 です。
  a / b は 2.4 です。
  正常終了！
  お疲れさまでした。

2 整数a：12
  整数b：3.14
  認識不能orゼロ除算！
  お疲れさまでした。

3 整数a：12
  整数b：0
  a * b は 0 です。
  認識不能orゼロ除算！
  お疲れさまでした。
```

単一の except 節で二つの例外を捕捉します。そのため、**複数の例外に対して、同一の対処が行える**ようになっています。

▶ 本プログラムは、文法的な理解を目的としたものです。性格の異なる例外に対して、同一の対処をすべきかどうかは、ケースバイケースです。

次は4です。すべてをオブジェクトと考える Python ですから、例外もオブジェクトの一種です。捕捉した例外オブジェクトを変数に受け取る際に、名前を与えることができます（関数での引数の受取りに似ています）。

捕捉した例外オブジェクトに名前を与える except 節の構文は、

```
except 例外 as 変数名： スイート
```

という形式です。

この形式を使って書きかえたのが、**List 12-4** のプログラムです。

List 12-4 chap12/list1204.py

```
# 乗算と除算を行うプログラム（例外処理：その３）

except (ValueError, ZeroDivisionError) as e:
    print(type(e))
    print('認識不能orゼロ除算！')
```

実行例

```
2 整数a：12
整数b：3.14
<class 'ValueError'>
認識不能orゼロ除算！
お疲れさまでした。
```

```
3 整数a：12
整数b：0
a * b は 0 です。
<class 'ZeroDivisionError'>
認識不能orゼロ除算！
お疲れさまでした。
```

except 節の頭部に“as e”が加えられていますので、変数 e は、捕捉した例外オブジェクトを参照することになります。

スイートの先頭行では、type(e) を出力しますので、捕捉した例外の型が表示されます。

ここに示す実行例では、『<class 'ValueError'>』と『<class 'ZeroDivisionError'>』が表示されています。

▶ ここでの e のように、except 節の頭部で宣言した変数は、その except 節でのみ有効です。

最後は5です。例外の種類を指定しない except 節は、すべての例外を捕捉します。この形式の except 節は、（except 節が複数あるのであれば）最後に置かなければなりません。

この形式を利用して書きかえたのが、**List 12-5** のプログラムです。

List 12-5 chap12/list1205.py

```
# 乗算と除算を行うプログラム（例外処理：その４）

except ValueError:
    print('整数と認識できません！')
except ZeroDivisionError:
    print('ゼロによる除算！')
except:
    print('例外発生！')
```

実行例

```
2 整数a：12
整数b：3.14
整数と認識できません！
お疲れさまでした。
```

```
3 整数a：12
整数b：0
a * b は 0 です。
ゼロによる除算！
お疲れさまでした。
```

```
4 整数a：[Control] + [C]
例外発生！
お疲れさまでした。
```

最後の except 節では、それより前の except 節で明示されている例外（および、それらの例外と互換性のある例外）以外のすべての例外が捕捉可能です。

実行例2と3は、**List 12-2**（p.342）と同じです。

実行例4では、入力中に [Control] キーを押しながら [C] キーを押すことによって発生する KeyboardInterrupt 例外が捕捉されています。

▶ サードパーティー製の統合開発環境では、入力時に [Control]+[C] の操作を受け付けないような仕組みがとられていることがあります（というよりも、そうなっているのが一般的です）。そのような環境では、例外は発生しませんし、捕捉も行われません。