1 例外処理の使い道

以下のコードを実行してみましょう。

ソースコード 1: 例外処理の使い道

```
hello = "Hello, World!"
number = int(hello)
print(number)
```

このコードを実行すると、以下のようなエラーが出力されます。

ソースコード 2: エラー

```
ValueError: invalid literal for int() with base 10: 'Hello, World!'
```

このようなエラーが出力されると、プログラムが停止してしまいます。しかし、例外処理を使うことで、エラーが出力されてもプログラムを停止させずに処理を続けることができます。

2 例外処理の書き方

例外処理の書き方は以下の通りです。

ソースコード 3: 例外処理の書き方

```
try:
  hello = "Hello, World!"
  number = int(hello)
  print(number)
except ValueError:
  printエラーが発生しました ("")
print処理が終了しました ("")
```

このコードを実行すると、以下のようにエラーが出力されずに処理が続行されます。

ソースコード 4: エラー

```
エラーが発生しました処理が終了しました
```

3 例外処理の種類

例外処理には以下の種類があります。

- ValueError：値が不正な場合に発生するエラー
- ZeroDivisionError：0 で割った場合に発生するエラー
- IndexError：インデックスが範囲外の場合に発生するエラー
- TypeError：型が不正な場合に発生するエラー
- KeyError：辞書のキーが不正な場合に発生するエラー
- FileNotFoundError：ファイルが見つからない場合に発生するエラー
- ModuleNotFoundError：モジュールが見つからない場合に発生するエラー

4 前回の課題

前回の課題の部分を例外処理を使って書き換えてみましょう。