## 基本

例外がどの階層でも補足されず、コードの最上位レベルまで raise されたとき、実行時エラーが発生してプログラムが動作を停止する。
トレースバックが表示される。

except で例外が補足されると、try 〜 except 節の次の文から引き続きプログラムが実行される。
つまり、raise で上位に例外を送出しない限り、例外はプログラム中でどこかに消えてなくなり、何事も無かったかのようにスクリプトの実行が継続される。

```
try:
    ...
except:
    ...
else:
    ...
finally:
    ...

try:
    ...
except (tuple_of_exceptions) as e:
    ...
```

すべての例外を補足する：

```
try:
    ...
except Exception as e:
    ...
```

## トレースバック

例外を補足すると、デフォルトのトレースバック表示は行われなくなる。
補足した例外のトレースバックを表示したい場合どうするか。
traceback モジュールを使う。

```
import traceback

try:
    ...
except Exception as e:
    traceback.print_exception(type(e), e, e.__traceback__)
```

print_exception() 以外にも、例外情報の文字列リストを得る format_exception() メソッドが定義されている。

## 例外を使うべき場面

実行してみなければエラーが発生するかどうかわからない、というときに使うべき。たとえば、ファイルの存在確認をすれば実行前に問題があるかどうかわかる、というときには使う必要はないのでは？
=> ファイルアクセスは OS の領分に踏み込むので、例が適切ではないかもしれないが。。。
=> いや。たとえばファイルが存在しないことはスクリプトのエラー検知で処理したいが、『存在

するのに開けない』という想定外のことはスクリプトの範疇じゃない、というのであれば例外を使わなくてよいと思う。システム管理用のスクリプトなんかはだいたいこれに当てはまると思う。
=> ループの中で外部コマンドを呼び出したりソケットを使ったりしていて、例外が発生しても無視して次のループに行ってほしい、というようなときには使いでがあるだろう。