Exceptionhandlung Teil 01

Bevor wir beginnen, werden wir uns so ausführlich wie möglich mit der Ausnahmebehandlung befassen. Achten Sie darauf, zwischen **Exception** und **Error** zu unterscheiden

Was ist ein Error? Was ist eine Exception?

Schauen wir uns zunächst an, was ein Error ist

Error wird auch als Analysefehler bezeichnet. Dies bedeutet, dass der Riemen nicht den Spezifikationen entspricht und überhaupt nicht ausgeführt werden kann.

In [4]:

```
# demo
while True print('Hello world')
```

```
  File "<ipython-input-4-42ea4a5d202b>", line 2
    while True print('Hello world')
                    ^
SyntaxError: invalid syntax
```

Der Parser gibt die Zeile mit dem Syntaxfehler aus und zeigt einen "Pfeil" an, der auf den ersten in dieser Zeile erkannten Fehler zeigt. Der Fehler wird durch das Token über der durch den Pfeil angegebenen Position verursacht (oder zumindest hier erkannt): Im Beispiel wurde der Fehler in der Funktion print () erkannt, da kein Doppelpunkt (':') vor Ihnen steht davon. Der Dateiname und die Zeilennummer werden ebenfalls ausgegeben, damit Sie wissen, wo Sie überprüfen müssen, wann die Eingabe aus der Skriptdatei stammt.

Schauen wir uns als nächstes an, was eine Exception ist.

Selbst wenn der Code korrekt ausgedrückt wird, kann es zur Laufzeit zu Problemen kommen. Genau wie "Ich esse gern Fernsehen." Dieser Satz hat keinen Grammatikfehler, macht aber keinen Sinn. Da ist Fernsehen kein Essen.

In [1]:

```
# demo
b=123/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-5f59c11e5a86> in <module>
      1 # demo
----> 2 b=123/0

ZeroDivisionError: division by zero
```

Der obige Code verursacht **ZeroDivisionError** , eine Exception. Es gibt keinen Fehler im Ausdruck von b = 123/0, aber es muss ein Problem beim Ausführen geben, dies ist die Ausnahme
Verschiedene Informationen zu integrierten Ausnahmen finden Sie unter: https://docs.python.org/zh-cn/3/library/exceptions.html#bltin-exceptions (https://docs.python.org/zh-cn/3/library/exceptions.html#bltin-exceptions)

Exceptionhandlung

Python bietet Methoden, die Exception behandeln können. Ich werde ein Beispiel bereitstellen, bei dem der Benutzer so lange tippen muss, bis eine gültige Ganzzahl eingegeben wird (natürlich kann der Benutzer den Vorgang unterbrechen, indem er einfach Strg + C verwendet).

In [1]:

```
while True:
    try:
        x = int(input('Bitte geben Sie eine Ganzzahl ein:'))
        break
    except ValueError:
        print('Dies ist keine Ganzzahl, bitte versuchen Sie es erneut!')
    except KeyboardInterrupt:
        print('Dies ist keine Ganzzahl, bitte versuchen Sie es erneut!')
```

```
Bitte geben Sie eine Ganzzahl ein:a
Dies ist keine Ganzzahl, bitte versuchen Sie es erneut!
Bitte geben Sie eine Ganzzahl ein:1.2
Dies ist keine Ganzzahl, bitte versuchen Sie es erneut!
Bitte geben Sie eine Ganzzahl ein:3
```

So funktioniert **try...except** :
-Führen Sie zunächst die Subsentenz unter try aus.(Dies ist der Code zwischen **try** und mit **except** von)
-Wenn es keine Exception gibt, wird **except** ignoriert und die Subsentenz vervollständigt -Wenn dies eine Exception ist, ignorieren Sie die verbleibende Folge vor **except** . Wenn die Exception vom selben Typ ist, den **except** wollte, führen Sie den Code unter **except** aus.
-Wenn die Exception nicht derselbe Typ ist, den **except** wollte, wird diese Exception außerhalb des **try ... except** -Satzes ausgelöst. Wenn kein Code diese Exception behandelt, handelt es sich um eine nicht behandelte Exception. Das gesamte Programm wird gestoppt und löst diese Exception aus.

Weitere Informationen finden Sie unter https://docs.python.org/zh-cn/3/tutorial/errors.html (https://docs.python.org/zh-cn/3/tutorial/errors.html)