CHAPITRE 8
GESTION DES EXCEPTIONS

Programmation en Python

Exceptions

- L'exécution de programmes Python peut déboucher sur divers types d'erreurs. Par exemple, des erreurs surviennent pour signaler :
  - une interdiction de diviser un nombre par 0
  - une interdiction d'additionner un nombre et une chaîne
  - plus généralement, une interdiction d'appliquer certains opérateurs sur des variables de types différents
  - un appel à des indices de listes ou de chaînes inappropriés
  - des utilisations de méthodes inexistantes ou inadéquates
  - etc.

Exceptions


```
>>> 5 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
>>> 1 + "a"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> l = [1,2,3]
>>> l = ["a", "b", "c"]
>>> l[7]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Exceptions


```
>>> l.ma_methode()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute 'ma_methode'
>>> d = {1 : "a", 2 : "b"}
>>> d[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 3
```

Exceptions

- Dans ces exemples, on voit apparâitre des erreurs de types différents :
  - `ZeroDivisionError`
  - `TypeError`
  - `IndexError`
  - `AttributeError`
  - `KeyError`
  - `...`
- L'ensemble et la hiérarchisation des types d'erreurs est donnée à l'adresse :

  https://docs.python.org/2/library/exceptions.html

Exceptions

- Une **exception** est une opération qu'effectue un interpréteur ou un compilateur lorsqu'une erreur d'un certain type est détectée au cours de l'exécution d'un programme.
- Pour simplifier le traitement des erreurs, il est possible d'associer un mécanisme de surveillance à tout ensemble d'instructions.
- Ce sont des mécanismes de traitement ou gestion des exceptions.

Les instructions « try – except »

- Le traitement d'une exception s'effectue via la synaxe suivante :
- Pour cela, ou utilise la syntaxe :

```
try:
        bloc d'istructiions 1
except [type d'erreur]:
        bloc d'instructions 2
```

- La spécification du type d'erreur est facultative.

Les instructions « try – except »


```
try:
        bloc d'istructiions 1
except [type d'erreur]:
        bloc d'instructions 2
```

- Ces instruction se comprennent ainsi :
  - essayer d'effectuer le bloc d'instruction qui suit le `try`;
  - si une exception – d'un certain type `type d'erreur` spécifié ou non – est générée, exécuter le bloc d'instructions qui suit le `except`.

Les instructions « try – except »

Programme

```
a = input("Entrer un nombre: ")
b = input("Entrer un nombre: ")
try:
    print a / float(b)
except:
    print "Division par 0 interdite..."
```

Les instructions « try – except »

Executions

```
Entrer un nombre: 1
Entrer un nombre: 2
0.5

Entrer un nombre: 1
Entrer un nombre: 0
Division par 0 interdite...
```

Les instructions « try – except »

Programme

```
# On peut spécifier le type d'exception à gérer
a = input("Entrer un nombre: ")
b = input("Entrer un nombre: ")
try:
    print a / float(b)
except ZeroDivisionError:
    print "Division par 0 interdite..."
```

Les instructions « try – except »

Executions

```
Entrer un nombre: 1
Entrer un nombre: 0
Division par 0 interdite...

Entrer un nombre: 1
Entrer un nombre: [1,2,3] # on entre une liste...
Traceback (most recent call last):
  File "code7.py", line 44, in <module>
    print a / float(b)
TypeError: float() argument must be a string or a number
# Ceci est une erreur de type autre que ZeroDivisionError
# Elle n'a donc pas été gérée...
```

Les instructions « try – except »

Programme

```
# On peut spécifier le type d'exception à gérer
# et multiplier les blocs "except"
a = input("Entrer un nombre: ")
b = input("Entrer un nombre: ")
try:
    print a / float(b)
except ZeroDivisionError:
    print "Division par 0 interdite..."
except:
    print "Autre type d'erreur..."
```

Les instructions « try – except »

Executions

```
Entrer un nombre: 1
Entrer un nombre: 0
Division par 0 interdite...

Entrer un nombre: 1
Entrer un nombre: [1,2,3] # on entre une liste...
Autre type d'erreur...
# Toute erreur de type autre que ZeroDivisionError
# est capturée par le dernier bloc except...
```

Les instructions « try – except »

Programme

```
def print_sorted(collection):
    try:
        collection.sort()
    except AttributeError:
        pass
    print collection
```

Executions

```
>>> print_sorted([1, 3, 2])
[1, 2, 3]
>>> print_sorted("acb")
'acb'
```