Gestion des exceptions

Définition

- Les exceptions sont les opérations qu'effectue un interpréteur ou un compilateur lorsqu'une erreur est détectée au cours de l'exécution d 'un programme.
- En règle générale, l'exécution du programme est alors interrompue, et un message d'erreur plus ou moins explicite est affiché.
- Exemple :

```
>>> print(45/0)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print(45/0)
ZeroDivisionError: division by zero
```

On dit:
une exception est levée

Exemple

```
# script inverse.py
chaine = input('Entrer un nombre : ')
nombre = float(chaine)
inverse = 1.0/nombre
print("L'inverse de",nombre,"est : ",inverse)
```

Quand vous entrez un nombre, tout se déroule normalement :

```
Entrer un nombre : 12
L'inverse de 12.0 est :  0.08333333333333333
```

Quand vous saisissez autre chose qu'un nombre, une exception de type **ValueError est levée**

```
Entrer un nombre : bonjour
Traceback (most recent call last):
  File "F:/ILHAM/Algorithmiques 2015/Python/exemples/exception/test.py", line 3,
 in <module>
    nombre = float(chaine)
ValueError: could not convert string to float: 'bonjour'
```

Exemple

```python
# script inverse.py
chaine = input('Entrer un nombre : ')
nombre = float(chaine)
inverse = 1.0/nombre
print("L'inverse de",nombre,"est : ",inverse)
```

Quand vous saisissez 0 , une exception de type **ZeroDivisionError** est levé

```
Entrer un nombre : 0
Traceback (most recent call last):
  File "F:/ILHAM/Algorithmiques 2015/Python/exemples/exception/test.py", line 4,
 in <module>
    inverse = 1.0/nombre
ZeroDivisionError: float division by zero
```

Gestion des exceptions

- il est possible de gérer les exceptions pour éviter l'arrêt brutal du programme.
- on utilise conjointement les instructions **try** et **except**.
- L'instruction **else** est optionnelle :

```
try:
    chaine = input('Entrer un nombre : ')
    nombre = float(chaine)
    inverse = 1.0/nombre
except:
    #ce bloc est exécuté si une exception est levée dans le bloc try
    print("Erreur !")
else:
    #on arrive ici si aucune exception n'est levée dans le bloc try
    print("L'inverse de",nombre,"est : ",inverse)
```

Distinguer les différents types d'exceptions

```
try:
    chaine = input('Entrer un nombre : ')
    nombre = float(chaine)
    inverse = 1.0/nombre
except ValueError:
    #ce bloc est exécuté si une exception de
    #type ValueError est levée dans le bloc try
    print(chaine,"n'est pas un nombre !")
except ZeroDivisionError:
    #ce bloc est exécuté si une exception de
    #type ZeroDivisionError est levée dans le bloc try
    print("Division par zéro !")
else:
    #on arrive ici si aucune exception n'est levée dans le bloc try
    print("L'inverse de",nombre,"est : ",inverse)
```

Distinguer les différents types d'exceptions

```
Entrer un nombre : BONJOUR
BONJOUR n'est pas un nombre !
```

```
Entrer un nombre : 0
Division par zéro !
```

Boucler le programme

```
while True:
    try:
        chaine = input('Entrer un nombre : ')
        nombre = float(chaine)
        inverse = 1.0/nombre
    except ValueError:
        #ce bloc est exécuté si une exception de
        #type ValueError est levée dans le bloc try
        print(chaine,"n'est pas un nombre !")
    except ZeroDivisionError:
        #ce bloc est exécuté si une exception de
        #type ZeroDivisionError est levée dans le bloc try
        print("Division par zéro !")
    else:
        #on arrive ici si aucune exception n'est levée dans le bloc try
        print("L'inverse de",nombre,"est : ",inverse)
        break
print("fin du test")
```

Résultat

```
Entrer un nombre : test
test n'est pas un nombre !
Entrer un nombre : 0
Division par zéro !
Entrer un nombre : 123
L'inverse de 123.0 est :  0.008130081300813009
fin du test
```