Mécanisme de gestion des exceptions

Même si un code est syntaxiquement correct, il peut générer des erreurs. Quand Python rencontre une erreur lors de l'exécution d'un code, on dit qu'il **lève une exception**.

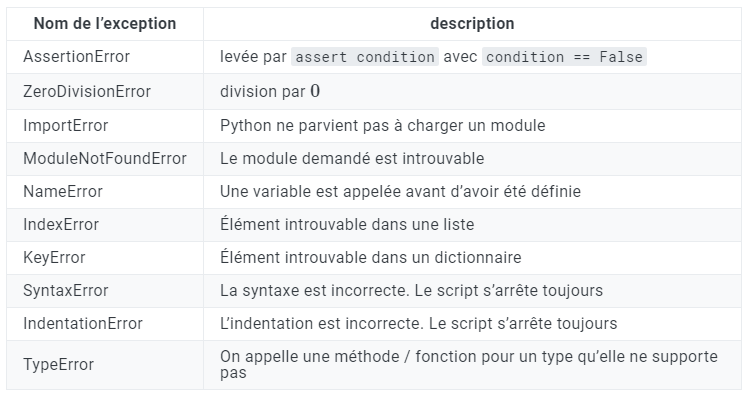
Cela génère l'affichage d'un message explicite et **l'arrêt** du programme si ces exceptions ne sont pas gérées.

**Gérer les exceptions** consiste à **anticiper les erreurs** (hors de contrôle du concepteur) de manière à ce que le programme ne s'arrête pas. Contrairement aux assertions, les exceptions permettent ainsi de ne pas planter le programme en proposant une alternative de traitement des cas problématiques avec les instructions try/except/else/finally. Elles doivent **demeurer dans le code final**.

En effet, la plupart des programmes ont vocation à être utilisés par d'autres utilisateurs que leur concepteur, et ceux-ci peuvent entrer des valeurs de variables provoquant des erreurs lors de l'exécution, même si le code du concepteur est initialement correct. Le concepteur doit anticiper ces erreurs dues à l'utilisateur.

|  |  |
| --- | --- |
| **Message d’erreur :** Le message affiché contient le traceback, c’est à dire la pile d’appel — le chemin parcouru par l’interpréteur pour atteindre l’erreur (soit la liste des fonctions traversées pour atteindre l’erreur). Ce message comporte le **type d’exception** levée (ZeroDivisionError, IOError, NameError,  SyntaxError, ValueError, TypeError, etc.) et un **message** qui décrit le problème rencontré.  *Remarque : Dans le cas d’une erreur de syntaxe, le message d’erreur indique même où est détectée l’erreur à l’aide d’une flèche (en fait, elle se trouve généralement juste avant l’endroit pointé par la flèche).* |  |

**Liste des exceptions courantes :**



***Exercice 1 :*** *Dans les lignes ci-dessous, un certain nombre de codes erronés sont écrits. Identifier l'exception correspondante parmi celles proposées ci-dessous et associées les au code.*

|  |  |  |
| --- | --- | --- |
|  |  | 1 : … 2 : … 3 : ….  4 :… 5 : … 6 : …. 7 : …. |

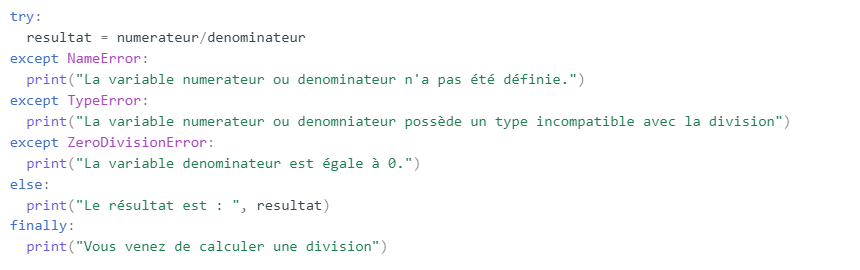
Pour éviter que le programme ne se termine définitivement ou qu’il se termine sans que la raison n’en soit explicite, il est conseillé de **gérer les exceptions**.

**Comment gérer les exceptions?**

Il suffit pour cela d'utiliser les instructions try/except et éventuellement else et finally.

* **Le bloc try** permet de tester un bloc de code et ne l'exécute que s'il ne contient aucune erreur. Dans le cas contraire, le programme ignore la totalité du code dans ce bloc et passe au bloc suivant except
* **Le bloc except** sera exécuté en cas d'erreur
* **Le bloc else** permet d'exécuter une action si aucune erreur ne survient dans le bloc try
* **Le bloc finally** vous permet d'exécuter du code, quel que soit le résultat des blocs try et except.

Exemple : structure complète possible



**Exception utilisant l’instruction raise :**

|  |  |
| --- | --- |
| *On peut se demander maintenant****s'il est possible de lever une exception sans rencontrer une erreur****.*  *Exemple pour un programme qui demande à l'utilisateur de tapez son age et de lever une exception si l'****age est < 18 ans !*** *Bien entendu un age tapé qui est inférieur à 18 ans est une opération qui ne contient aucune erreur, et pourtant on peut quand même en lever une :* |  |

***A faire avec Spyder :***

|  |  |
| --- | --- |
| ***Exercice 2 :*** *Écrire un script qui demande un nombre et qui calcule sa racine carrée avec gestion des exceptions.*  *Par exemple : >>> Entrer un nombre : go go n'est pas un nombre valide ! Entrer un nombre : -5.26 -5.26 n'est pas un nombre positif ! Entrer un nombre : 16 La racine carrée de 16.0 est : 4.0 >>>* | ***Exercice 3 :*** *Compléter le code suivant pour gérer les exceptions.*  *De plus, on ne veut pas de notes négatives ou supérieur à 20.* |