Tableau des exceptions d'après SAM et MAX : http://sametmax.com/tableau-de-reference-des-exceptions-en-python/

Les exceptions suivantes sont levées en cas d'erreur. Elles héritent toutes de`StandardError` :

Exception	Cause	Résolution
NotImplementedError	Un développeur a volontairement levé cette exception dans une des méthodes de sa classe afin de signaler que c'est aux enfants de la classe de l'implémenter.	N'utilisez pas la classe directement, mais créez une classe enfant. Overridez la méthodes afin de lui donner un comportement. Si vous ne comprenez pas ce que je viens de dire, lisez le guide sur la POO.
IndentationError ou TabError	Le fichier mélange des tabs et des espaces ou n'utilisent pas le même nombre de tabs ou d'espaces partout.	Activez l'affichage des tabs et espaces dans votre éditeur de texte, et assurez-vous d'utiliser 4 espaces partout comme valeur d'indentation.
ImportError	Python ne peut pas importer un module.	Vérifiez que le nom du module ne comporte pas de fautes (Python est sensible à la casse). Assurez-vous que le module est importable (situé dans un dossier du PYTHON PATH) et qu'il ne contient pas d'erreurs empêchant son importation, telle qu'une référence cyclique. Si vous ne savez pas ce qu'est le PYTHON PATH, lisez l'article sur les imports.
AssertionError	Une expression `assert` est fausse.	Si c'est dans votre code, retirez le `assert`, ce mot clé n'est que pour les tests unittaires. Si vous avez la malchance de tomber sur une lib qui l'utilise comme garde fou pour le passage d'arguments valeur, lisez le code source, et passez une valeur qui rendra l'expression vraie. Si vous en avez dans les tests et que vous ne savez pas quoi en faire, lisez le guide sur le tests.
AttributeError	Vous demandez à un objet de vous fournir un attribut qu'il ne possède pas.	Vérifiez que le nom de l'attribut et le nom de l'objet ne contiennent pas de faute (Python est sensible à la casse). Vérifier que l'attribut a bien été créé avant son accès et pas supprimé entre temps (Python est dynamique, les attributs peuvent être créés et supprimés à la volée. Si le cas où l'attribut n'existe pas est un cas valide, vous pouvez tester cette existence avec hasattr() ou obtenir une valeur par défaut avec getattr().
NameError	Vous tentez d'utiliser un nom (de variable, fonction, classe, etc) qui n'existe pas.	Vérifiez que ce nom ne contient pas de faute (Python est sensible à la casse). Assurez-vous que ce que vous nommez a bien été créé avant cette ligne.
IndexError	Vous tentez d'accéder à une partie d'une indexable (souvent une liste ou un tuple) qui n'existe pas.	Assurez vous que l'indexable contient assez d'éléments. Si le cas d'un indexable trop court est normal, vous pouvez vérifier la longueur de l'indexable avec len(), ou utilisez un try/except.

KeyError	Vous tentez d'accéder à une clé d'un mapping (souvent un dictionnaire) qui n'existe pas.	Assurez vous que la clé existe. Si le cas d'une clé inexistante est normal, vous pouvez vérifier qu'une clé est dans le mapping avec 'in', utiliser try/except ou obtenir une valeur par défaut avec get(). Dans le cas où vous souhaitez aussi que la valeur par défaut soit ajoutée à la collection, utilisez setdefault() oucollection.defaultdict().
TypeError	Vous tentez une opération incompatible avec ce type.	Si l'erreur a lieu au niveau d'une fonction que vous appelez, assurez-vous de passer des paramètres du type attendu par la fonction. Vous pouvez vérifier le type d'un objet avec type(). Vérifiez également que vous n'utilisez pas un opérateur incompatible avec un type (par exemple, & ne fonctionne pas sur les strings) ou entre deux types incompatibles (par exemple, il n'est pas possible d'additionner une string avec un entier).
ValueError	Vous passez une valeur à une fonction qui n'a aucun sens dans ce contexte ou dont le sens est ambiguë.	Assurez-vous de ne pas passer une valeur aberrante et que le résultat attendu soit évident. Par exemple, si vous essayez de faire int('é'), la conversion de la lettre "é" en entier n'a pas de résultat évident.
UnicodeDecodeError UnicodeEncodeError	Vous gérez votre texte comme un porc.	Lisez le guide sur l'encoding.
OverflowError	Vous faites des calculs trop gros pour les types numériques de base	Utilisez le module decimal
ZeroDivisionError	Vous faites une division par zéro	Assurez-vous que sous aucune condition aucun dénominateur n'est égal à 0
IOError	Erreur d'entrée / sortie	Vérifiez que vous pouvez lire / écrire depuis et vers la ressource que vous utilisez. Parmi les problèmes récurrents : disque dur plein, système corrompu, absence de permissions, fichier inexistant, etc.
OSError	L'OS retourne une erreur	Les causes peuvent être très variées, mais concernent souvent le système de fichier ou l'utilisation d'un sous-process. Vérifier les lignes où vous utiliser les modules os, shutils, popen, subprocess, multiprocessing, etc.
MemoryError	Vous utilisez trop de mémoire	Vérifiez vos remplissages de listes et dictionnaires, particulièrement si vous en déclarez un comme valeur par défaut d'un paramètre.

Ne sont pas mentionnées quelques exceptions beaucoup plus rares, mais vous pouvez trouver la liste complète ici. Toutes les exceptions héritent de `BaseException`, y compris `Exception`.

Il existe 3 exceptions qui n'héritent pas de `Exception`, et ne représentent PAS des erreurs :

- `SystemExit` : levé par sys.exit() qui par nature met fin au programme. N'affiche pas de stack trace quand la VM s'arrête si elle n'est pas attrapée.
- `KeyboardInterrupt` : levé par des combinaison de touches au clavier qui par nature mettent fin au programme (comme `Ctrl` + `C`).
- `GeneratorExit` : levé automatiquement quand on appelle `close()` sur un générateur.

Bien qu'héritant d'`Exception`, les warnings sont des mécanismes un peu particulier. Ils sont tous notés XxxWarning (ex: `DeprecationWarning` pour signaler l'usage d'une fonctionnalité en cours de dépréciation) et sont des enfants de `Warning`.

Généralement les warnings ne sont pas fait pour être levés avec `raise`, mais appelé avec warnings.warn. Par défaut ces warnings sont affichés, mais peuvent être réduits au silence, filtrés ou levés comme exception selon le désir du développeur. Enfin, `StopIteration` est levée quand on appelle `next()` sur un itérateur vide. C'est le seul enfant de `Exception` (avec `Warning`) qui n'hérite pas de `StandardError` car ce n'est pas une erreur en soit, mais simplement un mécanisme de contrôle de flux qui dit à la boucle `for` quand s'arrêter.