Introduction

En programmation, une exception est une anomalie qui se produit pendant l'exécution d'un programme. Les exceptions interrompent le flux normal d'exécution d'un programme et fournissent un moyen de gérer les erreurs et les situations exceptionnelles.

Définition de l'Exception

Une exception est un objet qui représente une erreur ou une situation exceptionnelle. En Python, les exceptions sont des objets qui héritent de la classe `BaseException`.

Tous les Types d'Exceptions

Python offre de nombreux types d'exceptions prédéfinies pour gérer différentes situations. Voici quelques-unes des exceptions courantes :

Type d'Exception	Description
`ZeroDivisionError`	Se produit lorsqu'une division par zéro est tentée
`TypeError`	Se produit lorsqu'une opération est effectuée sur un objet d'un type incorrect
`NameError`	Se produit lorsque le nom d'une variable ou d'une fonction n'est pas trouvé
`FileNotFoundError`	Se produit lorsque le fichier spécifié n'est pas trouvé
`IndexError`	Se produit lorsqu'une tentative est faite pour accéder à un index inexistant dans une séquence
`ValueError`	Se produit lorsqu'une fonction reçoit un argument du bon type, mais avec une valeur inappropriée

Try-Except

Pour gérer les exceptions, vous pouvez utiliser la structure `try-except`. Le code à risque est placé dans le bloc `try`, et en cas d'exception, le code du bloc `except` est exécuté.

```
try:
    resultat = 10 / 0
except ZeroDivisionError:
    print("Division par zéro détectée.")
```

La Clause Else

La clause `else` peut être utilisée pour exécuter un code en cas d'absence d'exception.

```
try:
    resultat = 10 / 2
except ZeroDivisionError:
    print("Division par zéro détectée.")
else:
    print("Aucune exception détectée.")
```

Le Finally

La clause `finally` permet d'exécuter un code quel que soit le résultat (qu'il y ait eu ou non une exception).

```
try:
    resultat = 10 / 0
except ZeroDivisionError:
    print("Division par zéro détectée.")
else:
    print("Aucune exception détectée.")
finally:
    print("Fin de l'exécution.")
print(resultat)
```

Classe Personnalisée d'Exception

Vous pouvez créer votre propre classe d'exception en héritant de la classe `Exception`.

```
class MonExceptionPersonnalisee(Exception):
    def __init__(self, message="Une erreur personnalisée s'est produite."):
        super().__init__(message)

try:
    raise MonExceptionPersonnalisee
except MonExceptionPersonnalisee as e:
    print(e)
```

Exercice