



# Python - Test, exceptions, assertions et module doctest

[Mise à jour le : 28/10/2022]

- **Sources**

- **Documentation** sur Python.org : [référence du langage](#), [erreurs et exceptions](#), [fonctions natives](#) (built-in)
- Framework sur Github - [testbook](#)

- **Mots-clés** : exceptions, assertions

Les mots ci-dessous sont dits “réservés”. Ils ne peuvent pas être utilisés comme nom de variable. Les mots soulignés sont une nouveauté de Python 3. Les mots en **gras** sont utilisés dans cette page.

and	continue	<b>finally</b>	is	<b>raise</b>
as	def	for	lambda	return
<b>assert</b>	del	from	<u>None</u>	<u>True</u>
<u>async</u>	elif	global	<u>nonlocal</u>	<b>try</b>
<u>await</u>	<b>else</b>	if	not	while
break	<b>except</b>	import	or	with
class	<u>False</u>	in	<b>pass</b>	yield

- [Fonctions natives \(built-in\)](#)<sup>1)</sup> utilisées dans les exemples : **enumerate()**, **flush()**, **open()**, **print()**, **ranges()**, **repr()**.

## Introduction

Quand Python rencontre une erreur dans le code, il **lève une exception**. Le traitement de ces exceptions permettra notamment de donner une réponse à une erreur produite par l'action de l'utilisateur (entrée de valeurs, etc.).

## 1. Exceptions

### 1.1 Forme minimale try ... except

\*.py

```
try:
    # Bloc de code à essayer
```

```
pass
except:
    # Bloc qui sera exécuté en cas d'erreur
pass
```

### Exemple

\*.py

```
age = input("Quel est votre âge ? ")
try:
    age = int(age)
    print("Vous avez", age, "ans")
except:
    print("Erreur lors de la conversion de votre âge !")
```

### Résultat attendu

Quel est votre âge ? 35  
Vous avez 35 ans

Quel est votre âge ? ldkfjg  
Erreur lors de la conversion de votre âge !

On intercepte toutes les erreurs sans les distinguer.

## 1.2 Forme complète try ... except ... finally

\*.py

```
try:
    # Bloc de code à essayer
    pass
except <error_1>:
    # Ce bloc sera exécuté si l'erreur = <error_1>
    pass
except <error_2>:
    # Ce bloc sera exécuté si l'erreur = <error_2>
    pass
except <error_n>:
    # Ce bloc sera exécuté si l'erreur = <error_n>
    pass
else:
    # Ce bloc sera exécuté si le code du bloc try s'est exécuté sans
    erreur
```

```
# else souvent omis et bloc placé dans try  
pass  
finally:  
# Bloc exécuté qu'il y ait eu des erreurs ou non
```

## Exemple

*\*.py*

```
num = input("Entrée le numérateur ")  
denom = input("Entrée le dénominateur ")  
resultat = 0  
try:  
    numérateur = int(num)  
    dénominateur = int(denom)  
    resultat = numérateur / dénominateur  
except NameError:  
    print("Le numérateur ou le dénominateur n'ont pas été définis")  
except ValueError:  
    print("Le numérateur ou le dénominateur possède un type  
incompatible avec la division.")  
except ZeroDivisionError:  
    print("Le dénominateur est égal à 0.")  
else:  
    print("Le résultat de la division est", resultat)  
finally:  
    print("On poursuit l'exécution du programme")
```

## Résultat attendu

Entrer le numérateur 12  
Entrer le dénominateur 2  
Le résultat de la division est 6.0  
On poursuit l'exécution du programme

Entrer le numérateur 12  
Entrer le dénominateur klkl  
Le numérateur ou le dénominateur possède un type incompatible avec la division.  
On poursuit l'exécution du programme

Entrer le numérateur 12  
Entrer le dénominateur 0  
Le dénominateur est égal à 0.  
On poursuit l'exécution du programme

## 1.3 Lever une exception

Pour lever une exception on utilise le mot-clé **raise**.

\*.py

```
raise TypeError("Message à afficher")
```

## 2. Les assertions

Les assertions sont un moyen de s'assurer, avant de continuer , qu'une condition est respectée.

Syntaxe

\*.py

```
assert test
```

Exemple

\*.py

```
val = input("Saisissez une valeur supérieure à 0 : ")
try:
    val=int(val)
    assert val>0
except ValueError:
    print("Ce n'est pas un nombre")
except AssertionError:
    print("La valeur n'est pas supérieure à 0 !")
```

Résultat attendu Saisissez une valeur supérieure à 0 : 6

Saisissez une valeur supérieure à 0 : 0

La valeur n'est pas supérieure à 0 !

## 2. Le module doctest

- **Source** : [Test interactive Python examples](#) sur python.org

Le module doctest recherche des zones de texte ressemblant à des sessions Python interactives et les exécute pour en vérifier le bon fonctionnement. Principe et exemples [ici](#)

## Résumé

- On peut intercepter les erreurs (ou exceptions) levées par le code avec **try... except... finally...**
- La syntaxe d'une assertion est **assert test**.
- Les assertions lèvent une exception **AssertionError** si le test échoue.
- On peut lever une exception grâce au mot-clé **raise** suivi du type de l'exception et d'un message.



## Pour aller plus loin ...

- [Pytest for Beginners](#)
- [Python's assert: Debug and Test Your Code Like a Pro](#)
- [Getting Started With Testing in Python](#)
- [Test Your Python Apps](#)
- [ward : un framework de test pour Python](#)

<sup>1)</sup>

Fonctions toujours disponibles.

From:

<http://webge.fr/dokuwiki/> - **WEBGE Wikis**

Permanent link:

<http://webge.fr/dokuwiki/doku.php?id=python:bases:exceptions&rev=1666946694>

Last update: **2022/10/28 10:44**

