6. Exceptions, assertions

6. Exceptions, assertions

Les exceptions correspondent à des erreurs ou des cas particuliers qui empêchent (a priori) la suite du déroulement du programme ou d'une fonction.

Exemple

```
>>> int("test")
[...]
ValueError: invalid literal for int() with base 10: 'test'
```

6. Exceptions, assertions

6.1 `try`/`except`

En Python, il est courant d'« essayer » des opérations puis de gérer les exceptions.

On utilise pour cela des `try: ... except: ...`. On peut déclencher nos propres exceptions avec `raise Exception("message")`

Exemple

```
try:
    str = input("Entrez un entier svp !")
    n = int(str)
except:
    raise Exception("Ce n'est pas un entier !")
```

6. Exceptions, assertions

Utilisation différente

```
try:
    str = input("Entrez un entier svp !")
    n = int(str)
except:
    n = -1
```

6. Exceptions, assertions

The "python way"

« Better to ask forgiveness than permissions »

6. Exceptions, assertions

6.2 Assertions

Il est possible d'utiliser des `assert`ions pour expliciter certaines hypothèses qui sont faites à certains endroit du code. Si l'hypothèse n'est pas validée, une exception est déclenchée.

```
def distance(x=0, y=0):
    assert isinstance(x, int) or isinstance(x, float), "Cette fonction ne prei
    assert isinstance(y, int) or isinstance(y, float), "Cette fonction ne prei

    return math.sqrt(x*x + y*y)
```

6. Exceptions, assertions

6.2 Assertions (autre exemple)

```
def some_function(n):
    assert isinstance(n, int), "Cette fonction ne prends que des int en argume
    assert n % 2 == 0, "Cette fonction ne prends que des entiers pairs en argu

    [...]
```

6. Exceptions, assertions

6.3 Pour aller plus loin : type hints ! (Python >= 3.6)

```
from typing import Union
float_or_int = Union[float, int]

def distance(x: float_or_int, y: float_or_int) -> float:
    return math.sqrt(x*x + y*y)
```