Python
Les exceptions

Nicolas Delestre

INSA
ROUEN NORMANDIE

Rappels 1 / 3

Qu'est ce qu'une exception ?

- C'est un mécanisme de gestion des erreurs :
  - Certaines utilisations d'instructions, d'opérations, de fonctions, de méthodes sont susceptibles de générer des erreurs (erreurs systèmes, valeurs incompatibles, types incompatibles, etc.)
  - Plutôt que de mélanger le code de gestion des erreurs avec le code métier, on sépare les deux pour rendre le code plus lisible
  - Plutôt que déterminer le type d'erreur en fonction d'une valeur particulière, ce sont des classes (une exception est donc un objet) qui vont indiquer la sémantique de l'erreur. Les langages proposent une hiérarchie d'exceptions, que le programmeur peut étendre.
- Une exception est dite levée lorsqu'une erreur apparaît
- Une exception est dite capturée lorsqu'elle est est gérée et traitée. Ce cas peut lever d'autres exceptions

Rappels 2 / 3

Comment gérer une exception ?

- Lorsqu'on utilise une suite d'instructions qui est susceptible de lever une exception :
  - soit on capture l'exception et on exécute une suite d'instructions qui a pour objectif de gérer l'erreur
  - soit on ne capture pas l'exception et dans ce cas, la fonction ou méthode en cours s'arrête et propage l'exception levée à la fonction appelante. Si cette propagation remonte jusqu'au programme principal sans que l'exception soit capturée, le programme s'arrête

INSA ROUEN NORMANDIE

Rappels 3 / 3

Quand lever une exception ?

Deux cas :

1. Une erreur « système » est détectée. Par exemple
   - ouverture d'un fichier qui n'existe pas
   - connexion réseau qui n'a pu être établie
   - etc.
2. L'appelant de la fonction ne respecte pas les règles définies par sa « documentation »[a]. Le corps de la fonction commence donc par une suite de tests susceptible de lever une exception.
   - Il y a un débat sur la vérification des types. Python adopte le principe du *duck typing*, il ne devrait pas y avoir de vérification de type. Mais on voit apparaître différents packages de vérification de type à partir des annotations (par exemple typecheck : `https://github.com/prechelt/typecheck-decorator`).

a. les langages compilés à typage statique vérifient une partie de cette documentation en vérifiant l'adéquation des types. Plus les types sont bien définis plus cette vérification à la compilation est utile

Les exceptions standards

https://docs.python.org/3/library/exceptions.html

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
           +-- ModuleNotFoundError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
```

```
BaseException
 +-- Exception
      +-- OSError
      |    +-- BlockingIOError
      |    +-- ChildProcessError
      |    +-- ConnectionError
      |    |    +-- BrokenPipeError
      |    |    +-- ConnectionAbortedError
      |    |    +-- ConnectionRefusedError
      |    |    +-- ConnectionResetError
      |    +-- FileExistsError
      |    +-- FileNotFoundError
      |    +-- InterruptedError
      |    +-- IsADirectoryError
      |    +-- NotADirectoryError
      |    +-- PermissionError
      |    +-- ProcessLookupError
      |    +-- TimeoutError
      +-- ReferenceError
      +-- RuntimeError
      |    +-- NotImplementedError
      |    +-- RecursionError
      +-- SyntaxError
      |    +-- IndentationError
      |         +-- TabError
      +-- SystemError
      +-- TypeError
```

```
BaseException
 +-- Exception
      +-- ValueError
      |    +-- UnicodeError
      |         +-- UnicodeDecodeError
      |         +-- UnicodeEncodeError
      |         +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
           +-- ResourceWarning
```

Lever d'une exception

raise

- On lève une exception à l'aide l'instruction `raise` suivie de la création d'une instance d'une sous classe de `Exception`
- Le constructeur prend au moins en paramètre une chaîne de caractères explicitant l'erreur

Exemple : simuler une classe abstraite

```
class ClasseAbstraite:
    def m1(self):
        raise NotImplementedError("Classe abstraite")

    def m2(self):
        self.m1()

class ClasseConcrete(ClasseAbstraite):
    def m1(self):
        pass
```

Attraper une exception

try...except

- La suite d'instructions susceptible de lever une exception, que l'on veut capturer, est :
  - précédée de l'instruction `try:`
  - suivie d'au moins une clause `except` suivie de la classe (ou d'un tuple de classes) de l'exception devant être attrapée avec possibilité de récupérer l'exception (mot clé `as`)
  - suivie optionnellement d'une clause `else`. Cette clause doit être positionnée après toutes les clauses `except` (il ne peut pas y avoir de clause `except` après un `else`), son code est exécuté lorsqu'aucune exception a été levée
  - suivie optionnellement d'une clause `finally`. Cette clause doit être positionnée après toutes les clauses `except` et la clause `else`, son code est toujours exécuté (qu'il y ait eu une exception ou pas).

```
...
try:
    instr1
    ...
except Exception1:
    instr1
except (Exception2, \
        Exception2) as ex:
    ...
else:
    ...
finally:
    ...
...
```

Création d'un type d'exception

Exception

- Pour créer un type d'exception il faut créer une sous classe (ou sous-sous classe) de la classe `Exception` (importée par défaut en python 3)
- On peut la configurer comme toute classe (redéfinir `__init__`, ajouter des attributs, ajouter des méthodes) mais elles sont souvent très simples (voire vides, utilisation de `pass`)
- La bonne pratique veut que le nom d'une exception se termine par `Error` (ou `Erreur` si codage en français).

INSA ROUEN NORMANDIE

Exemple : polyligne.py 1 / 6

Règles

- Un polyligne n'a de sens que s'il y a au moins deux points
- On interdit qu'une polyligne possède deux fois le même point

Définition des exceptions

```
4 class MemePointInterditErreur(Exception):
5     pass
6
7 class AuMoinsDeuxPointsErreur(Exception):
8     pass
```

INSA ROUEN NORMANDIE

Exemple : polyligne.py 2 / 6

Une méthode statique pour vérifier l'unicité des points (au sens de l'égalité)

```
12      @staticmethod
13      def _verifier_points_differents_deux_a_deux(pts):
14          if pts:
15              if pts[0] in pts[1:]:
16                  raise MemePointInterditErreur(f"{pts[0]} est présent au moins deux fois")
17              return Polyligne._verifier_points_differents_deux_a_deux(pts[1:])
```

Une méthode statique pour vérifier le nombre de points minimal

```
19      @staticmethod
20      def _verifier_assez_points(pts):
21          if len(pts) < 2:
22              raise AuMoinsDeuxPointsErreur("Une polyligne doit avoir au moins deux points")
```

Exemple : polyligne.py 3 / 6

Redéfinition de __init__

```
24    def __init__(self, est_fermee, pt1, pt2, *args):
25        pts = [pt1, pt2] + list(args)
26        self._verifier_points_differents_deux_a_deux(pts)
27        self._est_fermee = est_fermee
28        self._points = pts
```

Redéfinition de ajouter

```
40    def ajouter(self, *args):
41        pts = self._points + list(args)
42        self._verifier_points_differents_deux_a_deux(pts)
43        self._points = pts
```

INSA ROUEN NORMANDIE

Exemple : polyligne.py 4 / 6

Redéfinition de __setitem__

```
50    def __setitem__(self, indice_ou_slice, pt_ou_pts):
51        pts = list(self._points)
52        if isinstance(indice_ou_slice, int):
53            pts[indice_ou_slice] = pt_ou_pts
54        elif isinstance(indice_ou_slice, slice):
55            if isinstance(pt_ou_pts, Polyligne):
56                pts[indice_ou_slice] = pt_ou_pts._points
57            elif isinstance(pt_ou_pts, list) or \
58                 isinstance(pt_ou_pts, tuple):
59                pts[indice_ou_slice] = pt_ou_pts
60
61        self._verifier_points_differents_deux_a_deux(pts)
62        self._verifier_assez_points(pts)
63
64        self._points = pts
```

Exemple : polyligne.py 5 / 6

Redéfinition de __delitem__

```
66      def __delitem__(self, indice_ou_slice):
67          pts = list(self._points)
68          del(pts[indice_ou_slice])
69          self._verifier_assez_points(pts)
70          self._points = pts
```

Exemple : polyligne.py 6 / 6

```
>>> from point import Point2D
>>> from polyligne import Polyligne
>>> pl=Polyligne(True, Point2D(1,2), Point2D(3,4), Point2D(2,6))
>>> pl.ajouter(Point2D(1,2))
---------------------------------------------------------------------------
MemePointInterditErreur               Traceback (most recent call last)
<ipython-input-4-019abfa3dcb3> in <module>
----> 1 pl.ajouter(Point2D(1,2))
...
MemePointInterditErreur: (1,2) est présent au moins deux fois

>>> pl[0:2]=[Point2D(0,0), Point2D(1,2)]
>>> pl
Polyligne(True, Point2D(0, 0), Point2D(1, 2), Point2D(2, 6))
>>> pl[0]=Point2D(1,2)
---------------------------------------------------------------------------
MemePointInterditErreur               Traceback (most recent call last)
<ipython-input-10-733cd8ccec92> in <module>
----> 1 pl[0]=Point2D(1,2)
...
MemePointInterditErreur: (1,2) est présent au moins deux fois
>>> del(pl[0:2])
---------------------------------------------------------------------------
AuMoinsDeuxPointsErreur               Traceback (most recent call last)
<ipython-input-12-0ffab3dd44e2> in <module>
----> 1 del(pl[0:2])
...
AuMoinsDeuxPointsErreur: Une polyligne doit avoir au moins deux points
```

Conclusion

- Nous avons rappelé ce que sont les exceptions, ce que signifie lever et attraper une exception
- Nous avons rapidement vu les exceptions standards de Python et comment on lève ou attrape une exception
- Nous avons mis en pratique cela en ajoutant des exceptions à la classe `Polyligne`

INSA ROUEN NORMANDIE