Python
Méthodes spéciales (fin)

Nicolas Delestre

INSA
ROUEN NORMANDIE

Rappels

Les conteneurs

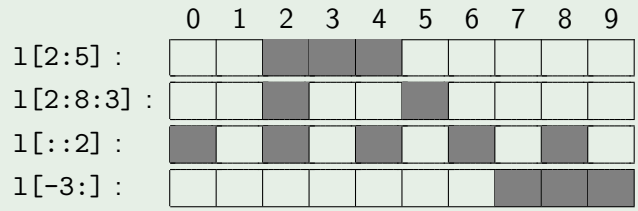
- Définition :
  - « Les conteneurs sont habituellement des séquences (telles que les tuples ou les listes) ou des tableaux de correspondances (comme les dictionnaires)... »[a]
- Un certain nombre d'opérations et de fonctions standards permettent de manipuler les conteneurs :
  - `+`, `in`, `[i]`, `[i:j]`, `[i:j:k]`, etc.
  - `len`, `del`, `reversed`

a. `https://docs.python.org/fr/3/reference/datamodel.html`

INSA ROUEN NORMANDIE

Les *slices*

Les *slices* par l'exemple



Des objets de type `slice`

- Les notations précédentes sont transformées en objet de type `slice`
- Signature de `__init__` : `slice(debut, fin[, pas])`, avec utilisation de la valeur `None`

$$\texttt{l[::2]} \iff \texttt{l[slice(None,None,2)]}$$

- Les objets de type `slice` ont trois attributs en lecture seule : `start`, `stop` et `step`

Méthode spéciales

- `__getitem__(self, cle)` pour l'opérateur `[ ]` (en lecture)
- `__setitem__(self, cle, valeur)` pour l'opérateur `[ ]` (en écriture)
- `__delitem__(self, cle)` pour la suppression (≪ fonction ≫ del) de la valeur associée à la clé

- `__add__(self, autre)` pour l'opérateur `+`
- `__contains__(self, element)` pour l'opérateur `in`
- `__len__(self)` pour la fonction `len`
- `__reversed__(self)` pour la fonction `reversed`

- `__iter__(self)` pour la fonction `iter` ou lorsque le conteneur est utilisé comme itérateur

Un exemple 1 / 5

polyligne.py

```
 5 class Polyligne(object):
 6     def __init__(self, est_fermee, pt1, pt2, *args):
 7         self._est_fermee = est_fermee
 8         self._points = [pt1, pt2]
 9         for pt in args:
10             self.ajouter(pt)
11
12     @property
13     def est_fermee(self):
14         return self._est_fermee
15
16     def ouvrir(self):
17         self._est_fermee = False
18
19     def fermer(self):
20         self._est_fermee = True
21
22     def ajouter(self, *args):
23         for pt in args:
24             self._points.append(pt)
```

Un exemple 2 / 5

polyligne.py

```
26      def __getitem__(self, indice_ou_slice):
27          if isinstance(indice_ou_slice, slice):
28              return Polyligne(self.est_fermee, *self._points[indice_ou_slice])
29          return self._points[indice_ou_slice]
30
31      def __setitem__(self, indice_ou_slice, pt_ou_pts):
32          if isinstance(indice_ou_slice, int):
33              self._points[indice_ou_slice] = pt_ou_pts
34          elif isinstance(indice_ou_slice, slice):
35              if isinstance(pt_ou_pts, Polyligne):
36                  self._points[indice_ou_slice] = pt_ou_pts._points
37              elif isinstance(pt_ou_pts, list) or \
38                   isinstance(pt_ou_pts, tuple):
39                  self._points[indice_ou_slice] = pt_ou_pts
40
41      def __delitem__(self, indice):
42          del(self._points[indice])
```

ROUEN NORMANDIE

Un exemple 3 / 5

polyligne.py

```
44      def __add__(self, autre):
45          return Polyligne(self.est_fermee, *(self._points + autre._points))
46
47      def __len__(self):
48          return len(self._points)
49
50      def __contains__(self, pt):
51          return pt in self._points
52
53      def __iter__(self):
54          return iter(self._points)
55
56      def __repr__(self):
57          return f"Polyligne({self.est_fermee}, {', '.join([repr(pt) for pt in self])})"
58
59      def __str__(self):
60          return f"({', '.join([str(pt) for pt in self])})"
```

ROUEN NORMANDIE

Utilisation de la classe Polyligne

```
>>> from point import Point2D
>>> from polyligne import Polyligne
>>> pl1 = Polyligne(True, Point2D(1,2), Point2D(3,4), Point2D(2,6))
>>> len(pl1)
3
>>> pl1[1:]
Polyligne(True, Point2D(3, 4), Point2D(2, 6))
>>> for pt in pl1:print(pt)
(1, 2)
(3, 4)
(2, 6)
>>> pl2 = Polyligne(True, Point2D(5, 2), Point2D(7, 4), Point2D(8, 6))
>>> pl1[0:2] = pl2
>>> pl1
Polyligne(True, Point2D(5, 2), Point2D(7, 4), Point2D(8, 6),
Point2D(2, 6))
```

INSA ROUEN NORMANDIE

Utilisation de la classe Polyligne

```
>>> l = [Point2D(9,4), Point2D(10,8)]
>>> pl1[-1:]=l
>>> pl1
Polyligne(True, Point2D(5, 2), Point2D(7, 4), Point2D(8, 6),
Point2D(9, 4), Point2D(10, 8))
>>> del(pl1[2:])
>>> pl1
Polyligne(True, Point2D(5, 2), Point2D(7, 4))
>>> Point2D(5,2) in pl1
True
```

INSA ROUEN NORMANDIE

Conclusion

- Rappels sur la notion de conteneur et les *slices*
- Les opérateurs et fonctions standards sur les conteneurs font appel à des méthodes spéciales

INSA ROUEN NORMANDIE