Python
Les flux

Nicolas Delestre

INSA
ROUEN NORMANDIE

Plan

INSA ROUEN NORMANDIE

Séquences d'octets

Qu'est-ce ?

- Jusqu'à présent les séquences étaient des séquences d'objets
- Python propose deux types séquences d'octets :
  - `bytes` : des séquences immuables d'octets
  - `bytearray` : des séquences d'octets
- Les valeurs des éléments de ces séquences sont donc des naturels compris entre 0 et 255 (levée d'un exception `ValueError` sinon)

INSA ROUEN NORMANDIE

bytes 1 / 3

Constantes

- Les constantes `bytes` sont représentées comme des chaînes de caractères avec la lettre `b` comme préfixe telles qu'elles contiennent :
  - des caractères ASCII pour des valeurs comprises entre 32 et 127
  - le caractère d'échappement \ suivi de la valeur (en décimal, octal ou hexadécimal) pour les autres

```
In [1]: a=b"abc"
In [2]: a
Out[2]: b'abc'
In [3]: a[0]
Out[3]: 97
In [4]: a=b"abcé"
  File "<ipython-input-4-3490d5b33a37>", line 1
    a=b"abcé"
      ^
SyntaxError: bytes can only contain ASCII literal characters.
In [5]: a=b"abc\xF10"

In [6]: a
Out[6]: b'abc\xf10'
```

bytes 2 / 3

Méthodes de classe `fromhex` et d'instance `hex`

- Elles permettent de créer un `bytes` à partir d'une chaîne de caractères contenant des représentations hexadécimales des octets (méthode `fromhex`) ou d'obtenir cette chaîne à partir d'un `bytes` (méthode `hex`)
- Pour `fromhex` les espaces ne sont pas pris en compte et une exception `ValueError` peut être levée

```
In [1]: bytes.fromhex('2Ef0 F1f2 ')
Out[1]: b'.\xf0\xf1\xf2'

In [2]: b'.\xf0\xf1\xf2'.hex()
Out[2]: '2ef0f1f2'
```

INSA ROUEN NORMANDIE

bytes 3 / 3

Constructeur

- Le constructeur de `bytes` peut être :
  - un entier positif n, on obtient alors une séquence de longueur n contenant que des 0
  - un itérable produisant des valeurs comprises entre 0 et 255, on obtient alors un `bytes` de même longueur avec les octets correspondants

```
In [1]: bytes(10)
Out[1]: b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'

In [2]: bytes(range(10))
Out[2]: b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t'
```

INSA ROUEN NORMANDIE

Classe bytearray

- Les instances de la classe bytearray s'utilisent comme ceux la classe bytes mais elles sont en plus mutables

```
In [1]: a=bytearray(10)

In [2]: a
Out[2]: bytearray(b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00')

In [3]: a.append(1)

In [4]: a
Out[4]: bytearray(b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01')

In [5]: a[0]=1

In [6]: a
Out[6]: bytearray(b'\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01')
```

Méthodes communes

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

- Ces deux classes proposent un grand nombre de méthodes permettant :
  - de compter le nombre d'occurrences d'une suite d'octets
  - de retrouver un suite d'octets
  - d'ajouter, de remplacer, de découper une suite d'octets
  - de modifier une suite d'octets
  - de questionner une suite d'octets

INSA ROUEN NORMANDIE

Le module io 1 / 6

Le module io

- Le module io permet de gérer trois catégories de flux d'entrée/sortie :

  Text I/O des flux qui attendent et qui produisent des chaîne de caractères (str)

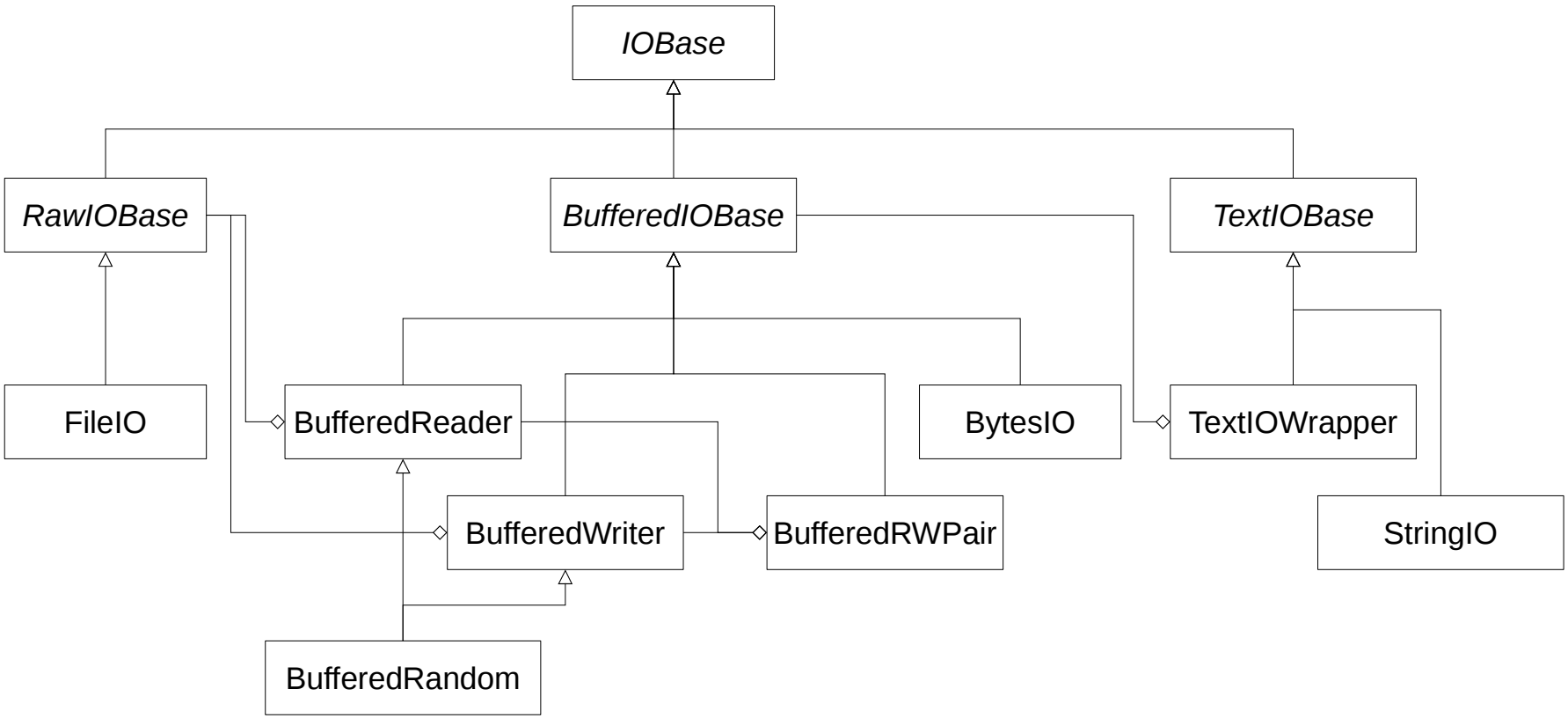
  Binary I/O des flux qui attendent et qui produisent des séquences d'octets (bytes et bytearray)

  Raw I/O des flux bas niveau, orientés bloc, encapsulés dans les flux textes ou binaires

INSA ROUEN NORMANDIE

Le module io 2 / 6

Hiérarchie de classes du module io (version simplifiée)



Le module io 3 / 6

Classe IOBase

- « Classe abstraite, pas de constructeur public »
- Elle définit différentes méthodes qui sont spécialisées par les sous classes
- Elle ne déclare pas les méthodes `read()`, `readinto()`, or `write()` car leurs signatures varient trop
- Les exceptions `ValueError` et `UnsupportedOperation` peuvent être levées

Classe RawIOBase

- « Classe abstraite, pas de constructeur public »
- Elle définit les méthodes de base de lecture et d'écriture d'octets

INSA ROUEN NORMANDIE

Le module io 4 / 6

Classe FileIO

- Elle permet de manipuler des fichiers systèmes
- L'exception FileExistsError peut être levée

Classe BufferedIOBase

- « Classe abstraite, pas de constructeur public »
- Elle met en place un système de cache permettant d'optimiser les lectures ou écritures
- L'exception BlockingIOError peut être levée

Classe BytesIO

- Elle permet de gérer un flux d'octets en mémoire
- La zone mémoire utilisée est supprimée à l'appel de la méthode close

Le module io 5 / 6

Classes BufferedReader, BufferedWriter et BufferedRandom

- Elles permettent d'utiliser des flux bufferisés en lecture, écriture ou lecture/écriture

Classe BufferedRWPair

- Elle permet d'utiliser deux flux bufferisés, l'un en lecture, l'autre en écriture

Classe TextIOBase

- « Classe abstraite, pas de constructeur public »
- Elle gère la persistance de chaînes de caractères encodées

Classe TextIOWrapper

- Elle gère la persistance de chaînes de caractères encodées dans des flux bufferisés

Le module io 6 / 6

Classe StringIO

- Elle gère la persistance de chaînes de caractères encodées en mémoire

Quelques méthodes et propriétés

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

IOBase

close, closed, __enter__, __exit__, fileno, flush, isatty, __iter__, __next__, readable, readline, readlines, seek, seekable, truncate, tell, writable, writelines

RawIOBase

read, readall, readinto, write

BufferedIOBase

detach, read, readinto, read1, write

TextIOBase

detach, encoding, errors, newlines, read, readline, write

Le module sys

- Le module sys propose trois flux :
  - `stdin` qui est un `TextIOWrapper` ouvert en lecture avec un encodage dépendant du système (aujourd'hui très certainement UTF-8)
    - `input` utilise ce flux
  - `stdout` qui est un `TextIOWrapper` ouvert en écriture avec un encodage dépendant du système (aujourd'hui très certainement UTF-8)
    - `print` et `input` utilisent ce flux
  - `stderr` idem `stdout`

INSA ROUEN NORMANDIE

Les fichiers 1 / 3

Ouverture d'un fichier

- Réalisée à l'aide de la fonction `open`
  - Utilise le motif de conception *Factory* : la classe de l'objet retournée varie en fonction des paramètres (du type -texte ou binaire- du mode -lecture, écriture, ajout, etc.-), par exemple :
    - `TextIOWrapper` pour un fichier texte
    - `BufferedReader` pour un fichier binaire ouvert en lecture
    - `BufferedWriter` pour un fichier binaire ouvert en écriture
    - `BufferedRandom` pour un fichier binaire ouvert en lecture/écriture
- L'instance retournée est un *context manager*, donc utilisable avec l'instruction `with...as`

INSA ROUEN NORMANDIE

Les fichiers 2 / 3

Paramètres (version simplifiée)

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,
newline=None)
```

file	chemin du fichier (`str`, `bytes`, `os.PathLike`)
mode	chaîne de caractères composée de : `'r'` en lecture (par défaut), `'w'` en écriture, `'x'` création exclusive (exception si fichier existe déjà), `'a'` en ajout, `'b'` en binaire, `'t'` en mode texte (par défaut), `'+'` en mise à jour (lecture écriture)
buffering	pas de buffer (0 en mode binaire, 1 en mode texte), sinon taille du buffer (la valeur par défaut essaye de faire au mieux)
encoding	en mode texte uniquement
errors	en mode texte pour indiquer comment gérer les erreurs d'encodage (`'strict'`, `'ignore'`, `'replace'`, etc.)
newline	en mode texte uniquement

INSA ROUEN NORMANDIE

Les fichiers 3 / 3

points.py

```
    @classmethod
    def enregistrer_les_points(cls, flux: io.TextIOBase):
        for pt in cls._les_points:
            flux.write("%d %d\n" % (pt.x, pt.y))
```

```
In [3]: from point import Point2D

In [4]: pt1=Point2D(1,1)

In [5]: pt2=Point2D(2,2)

In [6]: with open("/tmp/points.txt", "w") as f:
    Point2D.enregistrer_les_points(f)

In [7]:
Do you really want to exit ([y]/n)? y
$ cat /tmp/points.txt
1 1
2 2
```

Les modules `pickle` et `json`

La sérialisation d'objets

- Le module `pickle` propose des fonctions permettant de sérialiser (respectivement désérialiser) des objets dans un flux binaire qui doit proposer la méthode `write` (respectivement `read`)
- Le module `json` propose des fonctions permettant de sérialiser et désérialiser des objets dans un flux texte
- Méthodes `dump` et `dumps` pour la sérialisation
- Méthodes `load` et `loads` pour la désérialisation

INSA ROUEN NORMANDIE