Programmation en Python

Anne Garcia-Sanchez

M2i cyber dev - CFA CCI Avignon
19 octobre 2023

type tuple

type séquence comme chaînes de caractères et listes

test d'appartenance avec `in`, accès aux éléments par [ ], slicing...

peut référencer des objets hétérogènes

```
mytuple = (13, True, 4.2, "hello")
```

objet immuable

conversion tuple/list

```
>>> mytuple = (1, False, "hello", 2.5, 2)
>>> mylist = list(mytuple)
>>> mylist
[1, False, 'hello', 2.5, 2]
>>> mytuple
(1, False, 'hello', 2.5, 2)
>>> mylist[0] = 42
>>> mylist
[42, False, 'hello', 2.5, 2]
>>> mytuple
(1, False, 'hello', 2.5, 2)
>>> mynewtuple = tuple(mylist)
>>> mynewtuple
(42, False, 'hello', 2.5, 2)
>>>
```

tuple unpacking

tuple unpacking: récupérer les valeurs dans des variables

```
>>> values = (1, True, "hello")
>>> a, b, c = values
>>> a
1
>>> b
True
>>> c
'hello'
>>>
```

extented tuple unpacking

```
>>> mylist = [0, 1, 2, 3, 4, 5, 6, 7]
>>> a, *b = mylist
>>> a
0
>>> b
[1, 2, 3, 4, 5, 6, 7]
>>> *a, b = mylist
>>> a
[0, 1, 2, 3, 4, 5, 6]
>>> b
7
```

extented tuple unpacking

```
>>> *a, b = "hello"
>>> a
['h', 'e', 'l', 'l']
>>> b
'o'
```

type dict: dictionnaires

les types séquences: listes, chaînes de caractères, tuples

sont optimisés pour l'accès, la modification et l'effacement en fonction d'un numéro de séquence

mais ne permettent pas d'indicer avec autre chose que des entiers

$\longrightarrow$ type **dict**: permet d'associer une valeur à une autre valeur
associe clés - valeurs

Dictionnaires

objets mutables: on peut les modifier en place

efficacité mémoire

clé: objet immuable : int, float, bool, str...

```
>>> mydict = {"un": "one", "deux": "two", "trois": "three"}
>>> mydict
{'un': 'one', 'deux': 'two', 'trois': 'three'}
>>> mydict["un"]
'one'
>>>
```

Dictionnaires

Paul	Pierre	Lucas	Hugo
L3	L3	M1	M2

```
>>> students = {
...     "Paul"  : "L3",
...     "Pierre": "L3",
...     "Lucas" : "M1",
...     "Hugo"  : "M2",
... }
>>> students
{'Paul': 'L3', 'Pierre': 'L3', 'Lucas': 'M1', 'Hugo':
>>> students["Paul"]
'L3'
>>>
```

Dictionnaires

l'itérateur sur un dictionnaire est un itérateur sur les clés

```
>>> for student in students:
...     print(student)
...
...

Paul
Pierre
Lucas
Hugo
>>>
```

Dictionnaires

```
>>> for student in students:
...     print(student, students[student])
...
...
Paul L3
Pierre L3
Lucas M1
Hugo M2
>>>
```

Dictionnaires

	prénom	section	ville
0	Paul	L3	Pertuis
1	Pierre	L3	Avignon
2	Lucas	M1	Cavaillon
3	Hugo	M2	Orange

```
students = [
    {"prenom": "Paul", "section": "L3", "ville": "Pertuis"},
    {"prenom": "Pierre", "section": "L3", "ville": "Cavaillon"},
    {"prenom": "Lucas", "section": "M1", "ville": "Avignon"},
    {"prenom": "Hugo", "section": "M2", "ville": "Orange"},
]

for student in students:
    print(student["prenom"], student["section"], student["ville"],)
```

IDLE Shell 3.10.12

File Edit Shell Debug Options Window Help

```
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linu
Type "help", "copyright", "credits" or "license()" for more infor
>>>
========== RESTART: /home/anne/Documents/PYTHON/DEMO/dicostudentsl
Paul L3 Pertuis
Pierre L3 Cavaillon
Lucas M1 Avignon
Hugo M2 Orange
>>>
```

Dictionnaires

```
>>> diconb = {
...     "one": "un",
...     "two":"deux",
...     "three": "trois",
... }
...
>>> diconb.keys()
dict_keys(['one', 'two', 'three'])
>>> diconb.values()
dict_values(['un', 'deux', 'trois'])
>>> diconb.items()
dict_items([('one', 'un'), ('two', 'deux'), ('three', 'trois')])
>>> for key, value in diconb.items():
...     print(f"{key} se traduit {value}")
...
...
one se traduit un
two se traduit deux
three se traduit trois
>>>
```