NSI 1ère - Données

Types construits

qkzk

Types construits

Qu'est-ce ?

Par type construit, on entend, tout objet composé de plusieurs objets simples

Tuples

Qu'est-ce qu'un tuple ?

Un *tuple* est une série de valeurs séparées par des virgules.

Exemple : `tup = (1, 2, 3)` ou `tup = ('a', 'b', 'c')`

En python, les tuples peuvent être constitués de valeurs de type différent.

Manipuler les tuples

Les tuples ne sont pas mutables : on ne peut en changer le contenu

On accède à un élément par son indice :

```
>>> tup = ('a', 'b', 'c')
>>> tup[2]
'c'
```

Fonction qui retourne un tuple

Une fonction peut retourner un tuple !

```
def oppose_vecteur(x, y):
    return -x, -y
```

et cela donne :

```
>>> oppose_vecteur(1, 3)
(-1, -3)
```

Tableaux

Qu'est-ce qu'un tableau ?

Un **tableau** est une collection *mutable* d'objets.

Contrairement aux tuples, on peut en changer le contenu. On peut aussi ajouter ou retirer des éléments à un tableau.

En python, tous les tableaux ont le type `list`

Pourquoi cette distinction ? Pour éviter les confusions ultérieures !

Tableaux construits à la main

On peut créer, de plusieurs manières un tableau :

```
>>> tab = ["pierre", "paul", "jacques"]
>>> tab[1]
"paul"
```

À l'aide d'une boucle :

```
>>> tab = [] # tableau vide
>>> for i in range(5): # i de 0 à 4
...   tab.append(i ** 2) # ajouter un élément à la fin de tab
>>> tab
[0, 1, 4, 9, 16]
```

Tableaux construits par compréhension

Il existe une manière beaucoup plus simple d'écrire les tableaux : par compréhension

`[carres des entiers de 0 à 4] = [0, 1, 4, 9, 16]`

En python :

```
>>> tab = [i ** 2 for i in range(5)]
>>> tab
[0, 1, 4, 9, 16]
```

Liste par compréhension complexe

On peut imbriquer plusieurs boucles ou ajouter des conditions :

`[carres des entiers inférieurs à 10 et multiples de 3] = [0, 9, 81]`

En python :

```
>>> tab = [i ** 2 for i in range(10) if i % 3 == 0]
>>> tab
[0, 9, 81]
```

`i % 3` est le reste de la division de i par 3 (se lit `i` *modulo* `3`).

Dictionnaire

Qu'est-ce qu'un dictionnaire ?

Un dictionnaire est un enregistrement de valeurs associées à des clés (parfois appelées champs).

Exemple : répertoire téléphonique

Nom	Téléphone
Marcel	0320666666
Robert	0320123456
Amandine	0320987654

Dictionnaire par clés et valeurs

En python cela donne :

```
tel = {
  "Marcel":  "0320666666",
  "Robert":  "0320123456",
  "Amandine":  "0320987654",
}
```

Accéder à une valeur

On accède à une **valeur** par sa **clé**

`dictionnaire[cle] ----------> valeur`

```
>>> tel["Amandine"]
"0320987654"
```

Dictionnaire : mutable

Les dictionnaires sont mutables.

Si Robert change de numéro :

```
tel["Robert"] = "0320445566"
```

Remarquez bien la différence de syntaxe : on utilise `:` pour déclarer le dictionnaire et `=` pour changer une valeur

Itérer

Collections

En python (mais aussi dans beaucoup de langages), les éléments cités plus haut sont des collections. Cela signifie qu'on peut itérer dessus.

On peut écrire des boucles `for element in objet_construit:`

Cas simple

Pour les :

- chaînes de caractères,
- tuples,
- listes Python (=tableau)

La syntaxe est la même et `element` désigne l'objet contenu dans `objet_construit`

```
>>> chaine = "aZe"
>>> for lettre in chaine:
...   print(lettre)
a
Z
e
```

```
>>> tuple = (6, 4, 2)
>>> for t in tuple:
...   t ** 2
36
16
4
```

```
>>> liste = [a-1 for a in range(3)]
>>> for x in liste:
...    x + 2
1
2
3
```

Cas particulier propres aux dictionnaires

Il existe plusieurs manières d'itérer sur un dictionnaire.

Mais ATTENTION dans **Python < 3.6** il n'y a pas d'ordre particulier.

Itération simple :

```
tel = {
  "Marcel":  "0320666666",
  "Robert":  "0320123456",
  "Amandine":  "0320987654",
}
```

```
>>> for personne in tel:
...  tel[personne]
"0320666666"
"0320123456"
"0320987654"
```

Itération avec `.keys()`

`keys()` : collection des clés (les noms dans l'exemple plus haut.)

```
>>> for personne in tel.keys():
...  tel[personne]
"0320666666"
"0320123456"
"0320987654"
```

C'est identique à l'itération normale !

Itération avec `.keys()`

`items()` : collection des TUPLES (clé, valeur)

```
>>> for personne, tel in tel.items():
...  print("le numéro de ", personne, " est ", tel)
```

```
Le numéro de Marcel est 0320666666
Le numéro de Robert est 0320123456
Le numéro de Amandine est 0320987654
```

Itération avec `.values()`

Cette fois on ne récupère que les *valeurs*. Cela ne sert pas à grand chose dans le cadre d'une boucle

Dictionnaire par compréhension. *Hors programme.*

On peut créer des dictionnaires par compréhension :

```
>>> carres = {a: a ** 2 for a in range(4)}
>>> carres
{
  0: 0,
  1: 1,
  2: 4,
  3: 9
}
```