INF1 : Algorithmique et Programmation

Cours 8 :
Structures de données et Dictionnaires

Domitile Lourdeaux

Université de technologie de Compiègne

Printemps 2024

Types simples et types composites

Types simples

- Entier : `int`
- Réel : `float`
- Booléen : `bool`
- Caractère : *`char`*

Types composites

- Chaine de caractères : `string`
- Liste et Tableau : `list`, `tuple`
- Structure : `dict` (dictionnaires), `class` (classe)
- Ensemble : `set`

Remarques

- La correspondance n'est pas toujours exacte (e.g. une liste est plus générale qu'un tableau)
- Les types `set` et `class` seront étudiés dans un prochain cours

Types composites en python

Séquences : chaines, listes, tuples

- Suites ordonnées d'éléments
- Accessibles par un index (nombre entier)

Dictionnaires

- Eléments non ordonnés
- Accessibles à l'aide d'une « clé »

Ensembles

- Eléments non ordonnés, uniques
- Les ensembles seront vus dans un prochain cours

Tuples

Un tuple

- est semblable à une liste
- mais n'est pas modifiable (mutable) - comme les chaines de caractères

Collection d'éléments

- séparés par des virgules item encadrés de préférence par des parenthèses

Exemples :

```
>>> coord = (3, 5, 1)
... axes = ('x', 'y', 'z')
```

Les parenthèses ne sont pas indispensables mais conseillées pour la lisibilité du code

Opérations sur les tuples (1)

Similaires aux opérations sur les listes

```
>>> t = (4, -5, 1, 0, -3)
>>> len(t)
5
>>> print(t[1:4])
(-5, 1, 0)
```

Mais les tuples ne sont pas modifiables

```
>>> t[2:] = (-5, 4)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Opérations sur les tuples (2)

L'ajout (`append`) et le retrait (`del`) sont donc impossibles

Pour modifier, il faut créer un nouveau tuple

```
>>> t = (-5, 4) + t[2:]
>>> print(t)
(-5, 4, 1, 0, -3)
```

Motivations

Problème

- Comment modéliser une entité ayant plusieurs caractéristiques de types différents ?

Exemples

Personne

- Nom
- Prénom
- Age
- ...

```
>>> personne1 = ['Anna', 'Paddington', '20']
>>> personne2 = ['Bob', 'Hemingway', 19]
```

Voiture

- Marque
- Type
- Cylindrée
- ...

```
>>> voiture1 = ['Fiat', '500', 1]
>>> voiture2 = ['Ferrari', 'V12', 6.5]
```

Définitions

Plusieurs possibilités

- Si les éléments sont ordonnés, on peut dans certains cas utiliser des listes ou des tuples
- En Python, on peut utiliser des structures comme les **dictionnaires** ou les objets (plus tard)

Définition

Les dictionnaires sont des collections d'objets non-ordonnées. Un dictionnaire est composé d'éléments et chaque élément se compose d'une paire clé : valeur.

Dans d'autres langages de programmation, on parle de tableaux associatifs ou de hashs.

Le type dictionnaire

Ensemble de paires clé-valeur

- Clés
  - Type accepté : tout type non modifiable (chaine de caractères, entier, réel, tuple)
- Valeurs
  - Type accepté : tout type (valeurs numériques, chaines, listes, tuples, dictionnaires, fonctions, classes, instances)

Les dictionnaires sont modifiables (mutables)

Syntaxe

- Un dictionnaire se note entre accolades
- Les éléments sont séparés par des virgules
- Chaque élément est une paire d'objets (clé : valeur) séparés par « : »

Création d'un dictionnaire (1)

Création d'un dictionnaire vide

- d = {}

Exemple avec des clés de type chaine de caractères

```
>>> d = {"marque":"Fiat", "modèle":"500", "cv":4}
```

Remarque

Il existe aussi un constructeur `dict()` lorsque les clés sont des chaines

```
>>> d = dict(marque = "Fiat", modèle = "500", cylindrée = 1)
```

Création d'un dictionnaire (2)

Exemple avec des clés de type entier

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

- Les clés sont : 0, 1, 2, 3 , 4. Les valeurs sont les carrés des clés
- On peut définir le dictionnaire ainsi :

```
>>> d = {x : x*x for x in range(5)}
>>> print(d)
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

Opérations sur les dictionnaires (1)

```
>>> d = {"marque":"Fiat", "modèle":"500", "cv":4}
```

Accès à la valeur associée à une clé par : `d[clé]`

```
>>> print(d["marque"])
Fiat
```

Ajout : `d[nouvelleclé] = valeur`

```
>>> d["chevaux"]=4
>>> print(d)
{'marque': 'Fiat', 'modèle': '500', 'cylindrée': 1, 'chevaux': 4}
```

Suppression : `del d[clé]`

```
>>> del d["chevaux"]
>>> print(d)
{'marque': 'Fiat', 'modèle': '500', 'cylindrée': 1}
```

Opérations sur les dictionnaires (2)

Accès à une clé qui n'existe pas

```
>>> print(d["couleur"])
Traceback (most recent call last):
  File "<input>", line 1, in <module>
KeyError: 'couleur'
```

Tentative de destruction d'une clé-valeur qui n'existe pas

```
>>> del d["couleur"]
Traceback (most recent call last):
  File "<input>", line 1, in <module>
KeyError: 'couleur'
```

Opérations sur les dictionnaires (3)

Solution 1 : tester l'existence de la clé avec `in`

```
>>> if "couleur" in d : print(d["couleur"])
...
>>> if "marque" in d : print(d["marque"])
...
Fiat
```

Solution 2 : Méthode `get(clé, default)`

```
>>> print(d.get("couleur", "Couleur non précisée"))
Couleur non précisée
>>> print(d.get("marque", "Marque non précisée"))
Fiat
```

Exemple

```
>>> d = {"nom" : "Da Vinci", "naissance":1452, "mort":1519}
...
>>> type(d)
<class 'dict'>
>>> print(d["naissance"])
1452
>>> if not "prenom" in d :
...     d["prenom"] = "Leonardo"
...
>>> print(d)
{'nom': 'Da Vinci', 'naissance': 1452, 'mort': 1519, 'prenom': 'Leonardo'}
>>> for cle in d :
...     print(f"{cle} : {d[cle]}")
...
nom : Da Vinci
naissance : 1452
mort : 1519
prenom : Leonardo
```

Clés, valeurs éléments : méthodes avancées (1)

Liste des clés : méthode `keys()`

```
>>> list(d.keys())
['nom', 'naissance', 'mort', 'prenom']
```

Liste des valeurs : méthode `values()`

```
>>> list(d.values())
['Da Vinci', 1452, 1519, 'Leonardo']
```

Liste des éléments : méthode `items()`

```
>>> list(d.items())
[('nom', 'Da Vinci'), ('naissance', 1452), ('mort', 1519), ('prenom', 'Leonardo')]
```

Clés, valeurs éléments : méthodes avancées (2)

Parcours d'un dictionnaire

```
>>> for cle in d :
...     print(f"{cle} : {d[cle]}")
...
nom : Da Vinci
naissance : 1452
mort : 1519
prenom : Leonardo
```

```
>>> for cle, valeur in d.items() :
...     print(f"{cle} : {valeur}")
...
nom : Da Vinci
naissance : 1452
mort : 1519
prenom : Leonardo
```

Exercice

Compter toutes les occurrences des caractères d'une chaine de caractères donnée

Algorithme ?

Exercice (1)

Compter toutes les occurrences des caractères d'une chaine de caractères donnée

Algorithme

- On construit un dictionnaire `d` dont les clés sont les lettres et les valeurs le nombre d'occurrences
- Pour chaque caractère `c` dans `chaine`
  - Si `c` est une clé de `d`, incrémenter sa valeur (nombre d'occurrences)
  - Sinon l'initialiser à 1

Exercice (2)

```
def occurence(ch) :
    "nombre d'occurences de caract res dans ch"
    d = {}
    for c in ch :
        if c in d :
            d[c] += 1
        else :
            d[c] = 1
    return d

chaine1 = "anticonstitutionnellement"
chaine2 = "portez ce vieux whisky au juge blond qui fume"
print(occurence(chaine1))
print(occurence(chaine2))
{'a': 1, 'n': 5, 't': 5, 'i': 3, 'c': 1, 'o': 2, 's': 1, 'u
{'p': 1, 'o': 2, 'r': 1, 't': 1, 'e': 5, 'z': 1, ' ': 8, 'c
```

Tableau de dictionnaires

Création

- Le tableau peut être réparti sur plusieurs lignes. Un dictionnaire par ligne par exemple
- Attention à la syntaxe : les éléments (dictionnaires) du tableau doivent être séparés par des virgules

Exemple

```
>>> peintres = [{"nom" : "Da Vinci", "naissance":1452, "mort":1519},
...             {"nom" : "Michelangelo", "naissance":1475, "mort":1564},
...             {"nom" : "Raphaël", "naissance":1483, "mort":1520},
...             {"nom" : "Bellini", "naissance":1430, "mort":1516}]
...
```

Accès au champ du i^{eme} élément du tableau : `d[i][cle]`

```
>>> print(peintres[2]["nom"])
Raphaël
```

Dictionnaires et listes

Les dictionnaires et les listes sont modifiables

Les clés sont analogues aux indices d'un tableau mais :

- Ajout de `elt` à un tableau : `tab.append(elt)`
- Ajout de `cle:valeur` à un dictionnaire : `dico[cle] = valeur`

Les dictionnaires ne sont pas des séquences : Eléments non ordonnés

- Slice impossible : `print(dico[1:3])` provoque une erreur
- Intéressant pour l'ajout non consécutif de données

```
>>> td = {}
>>> td[25] = "Bob"
>>> td[3] = "Anna"
>>> td[10] = "Alice"
>>> print(td)
{25: 'Bob', 3: 'Anna', 10: 'Alice'}
```

Copie de dictionnaires

Attention

Comme pour les listes, l'instruction :

```
d = dico # d étant un dictionnaire
```

ne crée pas une copie de dico, mais seulement de sa référence

`d` et `dico` référencent alors le même dictionnaire

Pour faire une copie effective utiliser la méthode : `copy()`

```
>>> dico = {'a':1, 'b':2, 'd':4}
>>> d1 = dico
>>> d2 = dico.copy()
>>> d1['c'] = 3
>>> d2['e'] = 5
>>> dico
{'a': 1, 'b': 2, 'd': 4, 'c': 3}
>>> d1
{'a': 1, 'b': 2, 'd': 4, 'c': 3}
>>> d2
{'a': 1, 'b': 2, 'd': 4, 'e': 5}
```

Gestion d'un restaurant (1)

(inspiré d'un examen)
On souhaite gérer un menu du type suivant :

```
menu = [{"nom":"asperges", "genre":"E", "prix":10, "suggestion":"O"},
        {"nom":"tofu mariné", "genre":"P", "prix":12, "suggestion":"N"},
        {"nom":"tarte aux fraises", "genre":"D", "prix":8, "suggestion":"O"}]
```

1. Ecrire l'instruction permettant d'afficher le prix du dernier plat du menu
2. Ecrire la fonction `saisiePlat()` permettant de saisir un nouveau plat
3. Ecrire la procédure `affichageSuggestions()` permettant d'afficher les suggestions du jour
4. Ecrire un fonction `ajoutPlats()` permettant de saisir de nouveaux plats

Gestion d'un restaurant (2)

```
def saisiePlat() :
    "Saisie des plats"
    print('--- Saisie nouveau plat au menu ---')
    plat = {}
    plat["nom"] = input('Nom : ')
    plat["genre"] = input('Genre (E/P/D): ')
    plat["prix"] = int(input('Prix : '))
    plat["suggestion"] = input('Suggestion (O/N) : ')
    return plat

menu = saisiePlat()
```

Gestion d'un restaurant (3)

```
menu = [{"nom":"asperges", "genre":"E", "prix":10, "suggest
        {"nom":"tofu marin ", "genre":"P", "prix":12, "sug
        {"nom":"tarte aux fraises", "genre":"D", "prix":8,

print(f'prix du dernier plat : {menu[-1]["prix"]}')

def ajoutPlats(menu) :
    "Ajout de plats"
    print('--- Saisie de nouveaux plats au menu ---')
    reponse = input("Voulez-vous saisir un nouveau plat (O/
    while reponse :
        nouveau_plat = saisiePlat()
        menu.append(nouveau_plat)
        reponse = input("Voulez-vous saisir un nouveau plat
    return menu

menu = ajoutPlats(menu)
```

Gestion d'un restaurant (4)

```
def afficheSuggestions(menu) :
    "Affiche les suggestions"
    print('—— Affichage des suggestions ——')
    for plat in menu :
        if plat["suggestion"] == "O" :
            print(f'{plat["nom"]} : {plat["prix"]}')

afficheSuggestions(menu)
```

Questions...