2N6 Programmation 2

python™

Dictionnaire

Dictionnaires

- Similaire aux listes, les dictionnaires sont des structures de données qui peuvent stocker plusieurs valeurs.
- Les valeurs sont associées à une "clef" qui permet de récupérer les valeurs.

Dictionnaires


```
auto = { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1964 }


print(auto)
# {'marque': 'Ford', 'modele': 'Mustang', 'annee': 1964}

print(auto['marque'])
# Ford

print(f"{auto['marque']} {auto['modele']} {auto['annee']}")
# Ford Mustang 1964
```

Dictionnaires - Méthodes

- dict.get("clef") retourne la valeur de la clef dans le dictionnaire
  - auto.get("modele") ➔ "Mustang"

```
auto = { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1964 }
```

- dict["clef"] = valeur Change la valeur à correspondant à la clef. Si elle n'existe pas, ajoute la paire clef:valeur
  - auto["annee"] = 1968
  - auto["couleur"] = "rouge"

```
auto ➔ { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1968 ,
         "couleur": "rouge"}
```

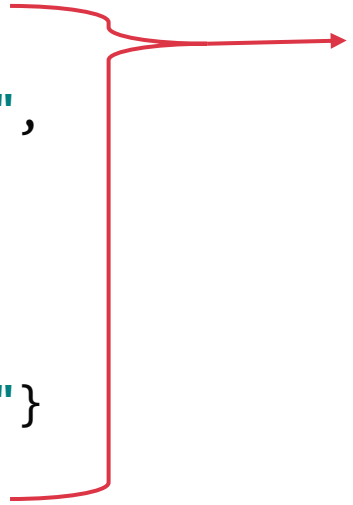
Dictionnaires - Méthodes

- dict.update({dictionnaire}) update peut ajouter des paires clef:valeur ou changer la valeur de clefs existantes ou être utilisé pour concaténer des dictionnaires

  - auto.update(ajout)

```
auto = { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1964 }


ajout = { "puissance": 7800,
          "couleur": "rouge"}
```



```
auto ➔ { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1964 ,
         "puissance": 7800,
         "couleur": "rouge"}
```

Dictionnaires - Méthodes

- del dict["clef"] retire la paire clef:valeur du dictionnaire.
  - del auto["annee"]

```
auto = { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1964 }
```



```
auto ➔ { "marque": "Ford",
         "modele": "Mustang" }
```

- dict.pop["clef"] retire la paire clef:valeur du dictionnaire et retourne la valeur uniquement.
  - annee_fabrication = auto.pop("annee")

```
auto = { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1964 }
```



```
auto ➔ { "marque": "Ford",
         "modele": "Mustang" }

annee_fabrication ➔ 1964
```

Dictionnaires - Méthodes

- len(dict) la fonction len nous retourne le nombre total de paires clef:valeur dans le dictionnaire.
  - nb_auto = len(auto)
  - nb_auto ➔ 3

```
auto = { "marque": "Ford",
         "modele": "Mustang",
         "annee": 1964 }
```

- dict.keys() retourne toutes les clefs dans le dictionnaire.
  - clef = auto.keys()
  - clef ➔ dict_keys([‘marque', ‘modele', ‘annee'])

- dict.values() retourne toutes les valeurs dans le dictionnaire.
  - valeurs = auto.values()
  - valeurs ➔ dict_values([‘Ford‘, ‘Mustang‘, ‘1964‘])

Dictionnaires - Méthodes

- Les méthodes keys() et values() nous redonne des objets itérables. On peut donc passer au travers avec une boucle for.

```
for clef in auto.keys():
    print(clef)

# marque
# modele
# annee
```

```
for valeur in auto.values():
    print(valeur)

# Ford
# Mustang
# 1964
```

Dictionnaires - Méthodes

- dict.items() permet d'obtenir toutes les paires clef:valeur dans un objet itérable

```
# On peut obtenir toutes les clés:valeurs dans notre dictionnaire avec la méthode .items()
print(etudiant.items())
#dict_items([('nom', 'Lucie'), ('cours', ['Reseau 1', 'Prog 2 en Python']), ('Tel', '514-321-1234')])


# Pour passer à travers toutes les paires clés:valeurs de notre dictionnaire
for key, value in etudiant.items():
    print(key,value)
#nom Lucie
#cours ['Reseau 1', 'Prog 2 en Python']
#Tel 514-321-1234
```

Liste dans un dictionnaire

- Les dictionnaires et les listes sont souvent utilisés ensemble pour permettre de stocker de nombreuses données de façon flexible.
- Ici, un dictionnaire représentant une auto et contenant une liste d'accessoires.

```
auto = {
    "marque": "Reliant",
    "modele": "Robin",
    "annee": 1988,
    "accessoires": [
        "Marchepied chromé",
        "Moteur V8",
        "Dés en minou sur le rétroviseur"
    ]
}

print(f"Il y a {len(auto['accessoires'])} accessoires:")

for item in auto['accessoires']:
    print("- " + item)

# Il y a 3 accessoires:
# - Marche-pied chromé
# - Moteur V8
# - Dés en minou sur le rétroviseur
```

Dictionnaires dans une liste

- Ici, une listes de autos. Chaque voiture est représentée par un dictionnaire.
- Il n'y a pas de limites aux « niveau de profondeur » des dictionnaires et listes.

```
autos = [
    {"marque": "Ford", "modele": "Mustang", "annee": 1964},
    {"marque": "Reliant", "modele": "Robin", "annee": 1988},
    {"marque": "Toyota", "modele": "Tercel", "annee": 1991}
]


print(f"Il y a {len(autos)} autos:")


for auto in autos:
    print(f"- {auto['marque']} {auto['modele']} {auto['annee']}")


# Il y a 3 autos:
# - Ford Mustang 1964
# - Reliant Robin 1988
# - Toyota Tercel 1991
```

JSON

Formats de sérialisation

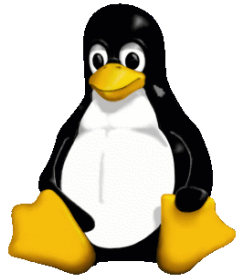
JSON

```json
{
    "first_name": "John",
    "last_name": "Smith",
    "age": 25,
    "address": {
        "street_address": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postal_code": "10021"
    },
    "phone_numbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ],
    "sex": {
        "type": "male"
    }
}
```

XML

```xml
<person>
 <firstName>John</firstName>
 <lastName>Smith</lastName>
 <age>25</age>
 <address>
   <streetAddress>21 2nd Street</streetAddress>
   <city>New York</city>
   <state>NY</state>
   <postalCode>10021</postalCode>
 </address>
 <phoneNumbers>
   <phoneNumber>
     <type>home</type>
     <number>212 555-1234</number>
   </phoneNumber>
   <phoneNumber>
     <type>fax</type>
     <number>646 555-4567</number>
   </phoneNumber>
 </phoneNumbers>
 <sex>
   <type>male</type>
 </sex>
</person>
```

YAML



```yaml
first_name: John
last_name: Smith
age: 25
address:
  street_address: 21 2nd Street
  city: New York
  state: NY
  postal code: "10021"
phone_numbers:
  - type: home
    number: 212 555-1234
  - type: fax
    number: 646 555-4567
sex:
  type: male
```

Souvent utilisé dans les fichiers de configuration Linux

JSON

- JSON (JavaScript Object Notation) est un format standard de sérialisation d'objets sous forme de données textuelles.
- Permet de stocker et transmettre des objets dans un format standard, indépendant du langage.
- Format indépendant et ouvert, créé au début des années 2000 et standardisé en 2017.
- Présentement le standard le plus répandu pour les échanges de données entre les applications.

JSON

```
[
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou"
        ]
    },
    {
        "marque": "Toyota",
        "modele": "Tercel",
        "annee": 1991,
        "accessoires": []
    }
]
```

Les **listes** sont entre **crochets [ ]**

- Une liste peut contenir des valeurs brutes, des dictionnaires, ou d'autres tableaux.

Les **dictionnaires** sont entre **accolades { }**

- Un dictionnaire est composé d'un ou plusieurs champs composés d'une clé et une valeur.
- La valeur d'un champ peut être une chaîne de caractères, un nombre, un tableau ou un dictionnaire.

Chaque élément est séparé des autres par des virgules