B73 Scripting sous Linux

python™

Dictionnaires

Aujourd'hui

1. API (*Application Programming Interface*) :
   - Programme permettant à des applications d'interagir ensembles.
   - Notre application peut donc faire des demandes d'informations à des ressources en ligne.
   - Les API "*RESTful*" ont une structure similaire au protocole HTML
   - Utilisé pour les demandes externes (sites web) et interne (sites intranet / autre application interne)
   - Retourne les données dans le format JSON ou XML habituellement

Aujourd'hui

2. Json :
   - Format pour le transfert de données.
   - N'est pas spécifique à un langage particulier.
   - Permet de transférer des données sous forme de listes et dictionnaires.

Aujourd'hui

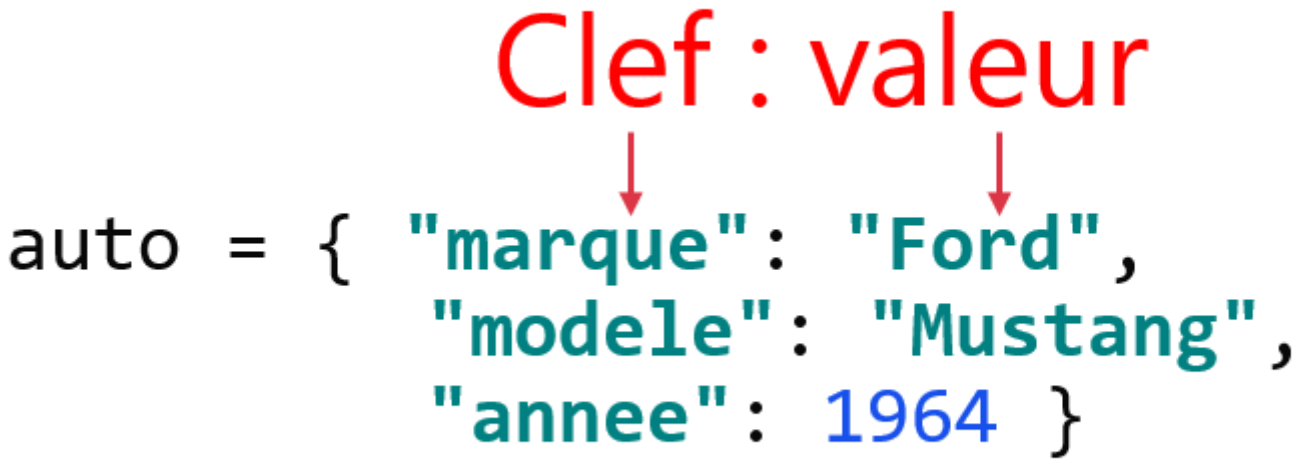
3. Dictionnaires
   - Structure de donnée dans python.
   - Les données sont sous forme de paires clef : valeur
   - Déclaré avec des accolades **{ }**

Dictionnaire

Dictionnaires

retour

- 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']) # utilisation de la clef pour obtenir sa valeur
# Ford

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

Dictionnaires – obtenir et ajouter des valeurs

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

  Ne cause pas d'erreurs si la clef n'existe pas.

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

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

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

Dictionnaires – retirer une paire clef/valeur

- 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

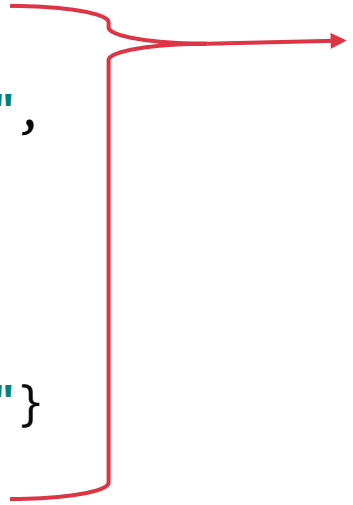
Dictionnaires - Méthodes

- 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

- 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.

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

```
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
```

Dictionnaires dans une liste

- Ici, une listes de voitures. 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
```