Cours - Serialization

La bibliothèque `json` en Python

Python fournit la bibliothèque `json` pour travailler avec JSON. Cette bibliothèque permet de convertir des objets Python en chaînes JSON et vice versa.

```
import json
```

Sérialisation (Objet Python vers JSON)

La sérialisation convertit un objet Python en une chaîne JSON. Pour ce faire, on utilise la fonction `json.dumps()`.

Exemple de Sérialisation

```
import json

# Exemple d'un dictionnaire Python
data = {
    "nom": "Alice",
    "âge": 25,
    "ville": "Paris",
    "compétences": ["Python", "Machine Learning"]
}

# Sérialiser le dictionnaire en une chaîne JSON
json_data = json.dumps(data)
print(json_data)  # {"nom": "Alice", "âge": 25, "ville": "Par
is", "compétences": ["Python", "Machine Learning"]}
```

Sérialisation dans un fichier

Pour écrire un objet Python dans un fichier JSON, on utilise `json.dump()`.

```
import json

data = {
    "nom": "Alice",
    "âge": 25,
    "ville": "Paris",
    "compétences": ["Python", "Machine Learning"]
}

# Sérialiser le dictionnaire et l'écrire dans un fichier
with open('data.json', 'w') as file:
    json.dump(data, file)
```

Désérialisation (JSON vers Objet Python)

La désérialisation convertit une chaîne JSON en un objet Python. Pour ce faire, on utilise la fonction `json.loads()`.

Exemple de Désérialisation

```
import json

# Exemple d'une chaîne JSON
json_data = '{"nom": "Alice", "âge": 25, "ville": "Paris", "c
ompétences": ["Python", "Machine Learning"]}'

# Désérialiser la chaîne JSON en un dictionnaire Python
data = json.loads(json_data)
print(data)  # {'nom': 'Alice', 'âge': 25, 'ville': 'Paris',
'compétences': ['Python', 'Machine Learning']}
```

Désérialisation à partir d'un fichier

Pour lire un objet JSON à partir d'un fichier et le convertir en objet Python, on utilise `json.load()`.

```
import json

# Lire le fichier JSON et désérialiser le contenu en un dicti
onnaire Python
with open('data.json', 'r') as file:
    data = json.load(file)
    print(data)  # {'nom': 'Alice', 'âge': 25, 'ville': 'Pari
s', 'compétences': ['Python', 'Machine Learning']}
```

Gestion des Types Complexes

Les types de données de base de Python (int, float, str, list, dict, etc.) sont automatiquement pris en charge par `json`. Pour les types complexes, vous devez fournir des méthodes de sérialisation et de désérialisation personnalisées.

Sérialisation de Types Complexes

Pour sérialiser un type d'objet non standard, vous pouvez fournir une fonction de conversion personnalisée via le paramètre `default` de `json.dumps()`.

```
import json
from datetime import datetime

# Exemple de classe complexe
class Personne:
    def __init__(self, nom, âge, date_naissance):
        self.nom = nom
        self.âge = âge
        self.date_naissance = date_naissance

# Exemple de fonction de sérialisation personnalisée
```

```
def serialize_personne(obj):
    if isinstance(obj, Personne):
        return {
            "nom": obj.nom,
            "âge": obj.âge,
            "date_naissance": obj.date_naissance.isoformat()
        }
    raise TypeError("Type non sérialisable")

# Exemple d'utilisation
personne = Personne("Alice", 25, datetime(1998, 5, 17))

# Sérialiser l'objet complexe en une chaîne JSON
json_data = json.dumps(personne, default=serialize_personne)
print(json_data)  # {"nom": "Alice", "âge": 25, "date_naissan
ce": "1998-05-17T00:00:00"}
```

Désérialisation de Types Complexes

Pour désérialiser un JSON en un type d'objet non standard, vous pouvez fournir une fonction de conversion via le paramètre `object_hook` de `json.loads()`.

```
import json
from datetime import datetime

# Exemple de fonction de désérialisation personnalisée
def deserialize_personne(dct):
    if "date_naissance" in dct:
        dct["date_naissance"] = datetime.fromisoformat(dct["d
ate_naissance"])
    return dct

# Exemple d'utilisation
json_data = '{"nom": "Alice", "âge": 25, "date_naissance": "1
998-05-17T00:00:00"}'
```

```
# Désérialiser la chaîne JSON en un dictionnaire Python
personne_dict = json.loads(json_data, object_hook=deserialize
_personne)
print(personne_dict)  # {'nom': 'Alice', 'âge': 25, 'date_nai
ssance': datetime.datetime(1998, 5, 17, 0, 0)}

# Créer un objet Personne à partir du dictionnaire
personne = Personne(**personne_dict)
print(personne.nom, personne.âge, personne.date_naissance)  #
Alice 25 1998-05-17 00:00:00
```

Exemple pratique

1. **Sérialiser et désérialiser un dictionnaire :**
   Écrire un script qui prend un dictionnaire, le sérialise dans un fichier JSON, puis lit ce fichier et désérialise le contenu en un dictionnaire Python.

   ```
   import json

   data = {
       "nom": "Bob",
       "âge": 30,
       "ville": "Lyon",
       "compétences": ["Java", "SQL"]
   }

   # Sérialiser le dictionnaire dans un fichier JSON
   with open('data_ex1.json', 'w') as file:
       json.dump(data, file)

   # Lire le fichier JSON et désérialiser le contenu en un di
   ctionnaire Python
   ```

```
with open('data_ex1.json', 'r') as file:
    data_loaded = json.load(file)
    print(data_loaded)  # {'nom': 'Bob', 'âge': 30, 'vill
e': 'Lyon', 'compétences': ['Java', 'SQL']}
```