

Les dictionnaires



Définition d'un dictionnaire:

- Un **dictionnaire** est une collection non ordonnée de paires **clé-valeur**. Chaque clé est unique et associée à une valeur.
- Il permet un accès rapide aux données via des clés, plutôt que par des indices comme dans une liste.
- Les dictionnaires sont très utiles pour structurer et organiser des données complexes.

Syntaxe



- Syntaxe de base pour créer un dictionnaire en Python.

```
appareils = {"capteur1": "température", "capteur2": "humidité", "capteur3": "pression"}
```

Les valeurs peuvent contenir tous type de données

```
capteur = {"id" : 12, "position" : "baie", "temprerature" : 17.5, "default":False}
```

Accès aux éléments d'un dictionnaire



- Accès direct par clé

```
appareils = {"capteur1": "température", "capteur2": "humidité", "capteur3": "pression"}  
print(appareils["capteur1"])
```

- Utilisation de get() pour éviter les erreurs si la clé n'existe pas

```
print(appareils.get("capteur4", "Inconnu")) # Inconnu
```



Modifier ou ajouter un élément

- Modifier un élément via sa clé.

```
appareils = {"capteur1": "température", "capteur2": "humidité", "capteur3": "pression"}  
  
appareils["capteur2"] = "humidité relative" # Modifie
```

- Ajouter un élément via sa clé

```
appareils["capteur4"] = "luminosité" # Ajoute
```

Parcourir un dictionnaire avec une boucle



- Capteur récupère la clé et type la valeur

```
appareils = {"capteur1": "température",  
             "capteur2": "humidité",  
             "capteur3": "pression"  
            }  
  
for capteur, type in appareils.items():  
    print(f"{capteur} mesure {type}")
```



Supprimer un élément

- Supprimer des éléments avec **del()** ou **pop()**.

```
appareils = {"capteur1": "température",  
            "capteur2": "humidité",  
            "capteur3": "pression"  
            }  
  
del appareils["capteur4"] # Supprime capteur4  
  
app_removed = appareils.pop("capteur3", "Inexistant") # Supprime et renvoie la valeur
```

Méthodes utiles sur les dictionnaires



- Utiliser afficher toutes les clés, valeurs ou les paires clé-valeur

```
for cle in appareils.keys():
    print(cle) # dict_keys(['capteur1', 'capteur2'])

for valeur in appareils.values():
    print(valeur)
# dict_values(['température', 'humidité relative'])

for cle, valeur in appareils.items():
    print(f"{cle}: {valeur}")
# dict_items([('capteur1', 'température'), ('capteur2', 'humidité relative')])
```

Obtenir la taille d'un dictionnaire



- Utiliser `len()` pour obtenir la taille d'un dictionnaire

```
print(len(etat_capteurs))
```


Supprimer tous les éléments



- La méthode **clear()** permet de supprimer tous les éléments d'un dictionnaire, le réinitialisant complètement sans la supprimer.

```
etat_capteurs = {"capteur1": "actif", "capteur2": "actif", "capteur3": "inactif"}  
  
# Vider le dictionnaire  
etat_capteurs.clear()  
  
print(etat_capteurs)  # {}
```

Compter les occurrences d'un élément



```
etat_capteurs = {"capteur1": "actif", "capteur2": "actif", "capteur3": "inactif"}  
  
# Compter le nombre de capteurs actifs  
nb_actifs = sum(1 for etat in etat_capteurs.values() if etat == "actif")  
print(nb_actifs) # 2
```

Fonction any()



- Retourne **True** si au moins un élément est évalué comme vrai
- Utilisé pour vérifier la présence d'un élément valide dans un dictionnaire

```
paquets = {"paquet1": 1200, "paquet2": 800, "paquet3": 1600}

# Vérifie si au moins un paquet dépasse 1500 octets
gros_paquet = any(taille > 1500 for taille in paquets.values())
print(gros_paquet) # True, car paquet3 dépasse 1500 octets
```

Fonction all()



- Retourne **True** si tous les éléments sont évalués comme vrais

```
etat_capteurs = {"capteur1": "actif", "capteur1": "actif", "capteur3": "inactif"}

# Vérifie si tous les capteurs sont actifs
tous_actifs = all(etat_capteur=="actif" for etat_capteur in etat_capteurs.values())
print(tous_actifs) # False, car capteur3 est inactif
```