LINGE1225 : Programmation en économie et gestion

Point Théorique 6

Dictionnaires, lecture et écriture de fichier, et exceptions

François Fouss & Marco Saerens

Année académique 2020-2021

1

Livre de référence

- Chapitre 9 : Manipuler des fichiers
- Chapitre 10 : Approfondir les structures de données



Plan

- Dictionnaires
- Application dictionnaires
- Lecture/écriture d'un fichier « .csv »*
- Exceptions
- Application fichier et exceptions

* Comma-separated values, connu sous le sigle CSV, est un format texte ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules. https://fr.wikipedia.org/wiki/Comma-separated_values

3

Dictionnaires

Dictionnaire

- ➤ Les dictionnaires sont modifiables comme les listes mais ne sont pas des séquences.
- Les éléments enregistrés ne sont pas disposés dans un ordre immuable.
- ➤ Il est possible d'accéder à n'importe quel élément à l'aide d'un index spécifique appelé clé. Pas comme avec des indices ds les listes (≠ séquencé)
- ➤ Les éléments mémorisés dans un dictionnaire peuvent être de n'importe quel type : valeurs numériques, des chaînes, des listes, des dictionnaires et aussi des fonctions.

ex type dictionnaire : association d'infos à 1 pers

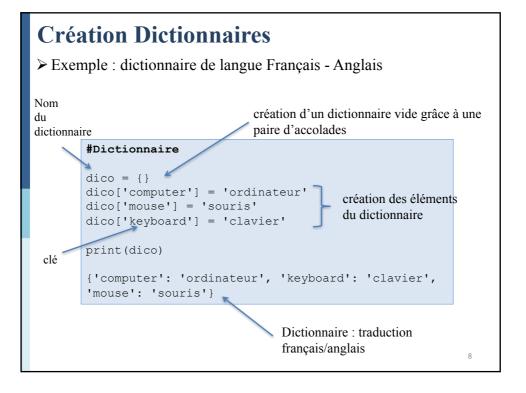
5

défini par des { }

Dictionnaire Vs Liste

 Syntaxe: <pre></pre>	Dictionnaire	Liste
 Utile pour trouver des informations associées à une clé On trouve une valeur en connaissant sa clé On trouve une valeur en connaissant son indice dans la liste Les clés sont uniques pour chaque valeur Mutable = modifiable 	• Syntaxe: <variable> = {<clé> : <valeur>,}</valeur></clé></variable>	• Syntaxe: <variable> = [<valeur>, <valeur>,]</valeur></valeur></variable>
 On trouve une valeur en connaissant sa clé On trouve une valeur en connaissant son indice dans la liste Les clés sont uniques pour chaque valeur Mutable = modifiable Mutable = modifiable Mutable = Mutable Mutable = Mutable Mutable = Mutable 	L'ordre des objets n'a pas d'importance	L'ordre a de l'importance
 clé son indice dans la liste Les clés sont uniques pour chaque valeur Mutable = modifiable Mutable = modifiable Mutable = Mutable Mutable = Mutable Mutable = Mutable Mutable = Mutable 		Très utile lorsque la liste est triée
 • Mutable = modifiable • Mutable Pour 1 clé donnée = 1 seule valeur (semblable à 1 fct) MAIS on peut changer cette valeur (cad 		
Pour 1 clé donnée = 1 seule valeur (semblable à 1 fct) MAIS on peut changer cette valeur (cad		L'indice est unique pour chaque valeur
à 1 fct) MAIS on peut changer cette valeur (cad	• Mutable = modifiable	• Mutable
	à 1 fct) MAIS on peut changer cette valeur (cad	

Dictionnaire: exemple > Voici un exemple de dictionnaire > Ce dictionnaire comprend la traduction de mots de l'anglais au français, #Dictionnaire Clé Valeur dico = {'computer': 'ordinateur', 'keyboard': 'clavier', 'mouse': 'souris'} Les clés sont uniques, pas les valeurs! > Une clé est associée une valeur MAIS une valeur peut avoir plusieurs clés associées ex: 2 mots en anglais peuvent avoir 1 même trad 1) to walk = marcher & the market = le marché -> 2 clés ≠ mais 1 seule valeur pour les 2 marché 2) nice = gentil & friendly = gentil -> 2 clés ≠ mais 1 seule et même valeur pour les 2



Même chose que ci-dessus avec une syntaxe ≠

Création Dictionnaires : clés

➤ Dans les dictionnaires, les <u>index s'appellent donc des clés</u>, et les éléments peuvent donc s'appeler des <u>paires clé-valeur</u>. Dans le dictionnaire anglais français, les clés et valeurs sont des chaînes de caractères.

Important : l'ordre dans lequel on fournit les informations au dictionnaire est différent de l'ordre dans lequel les éléments apparaissent dans le dictionnaire. C'est pourquoi on appelle les valeurs d'un dictionnaire grâce à sa clé tel que :

```
print(dico['mouse'])
souris
```

Supprimer un élément d'un dictionnaire :

```
del dico['mouse']
```

Connaître la longueur du dictionnaire :

```
print(len(dico))
```

9

Test d'appartenance : in

Pour vérifier si 1 valeur se trouve dans le dicc

```
print(Dico)
{'computer': 'ordinateur', 'keyboard': 'clavier'}
if 'mouse' in Dico:
    print('OK')
else
    print('KO')
```

Nous venons de supprimer 'mouse' du dictionnaire, il ne s'y trouve donc plus.

« in » permet de renvoyer vrai si la clé se trouve ds le dico et faux si la clé ne se trouve pas ds le dico

Parcours d'un dictionnaire

clef va énumérer tt les associations clé-valeur

1

Dictionnaires: utilité?

- ➤ Du fait qu'ils ne sont pas des séquences, les dictionnaires sont utiles pour gérer des ensembles de données où l'on est amené à effectuer fréquemment des ajouts ou suppressions, dans n'importe quel ordre.
- > Ils remplacent les listes lorsqu'il s'agit de traiter des ensembles de données numérotées dont les numéros d'indice ne se suivent pas.

=> Les clés peuvent être des strings, des entiers, des nombres décimaux

```
#dictionnaire : utilité

client = {}
client[4337] = 'Dupond'
client[256] = 'Durant'
client[743] = 'Schmidt'

lci on associe le nom du client
à chaque numéro de client
```

Application dictionnaires

Exercice

Ecrivez une fonction qui échange les clés et les valeurs d'un dictionnaire (ce qui permettra par exemple de transformer un dictionnaire anglais/français en un dictionnaire français/ anglais). On suppose que le dictionnaire ne contient pas plusieurs valeurs identiques.

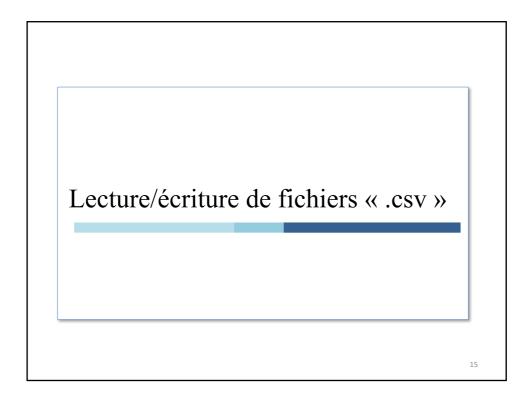
Solution

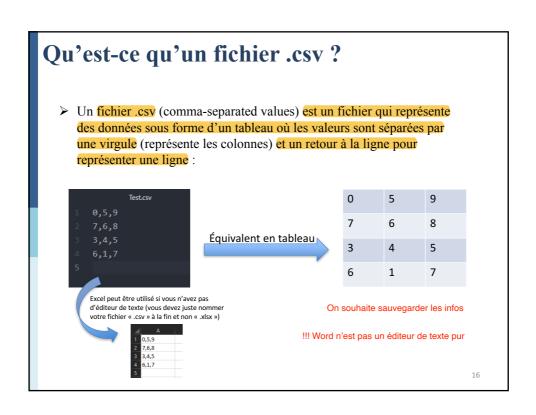
```
#Echange des clés et des valeurs dans un dictionnaire
def inverse(dico):
     "Construction d'un nouveau dico, pas à pas"
    dico_inv={}-
                                              initialise à un dico vide
    for cle in dico:
         item=dico[cle]
         item=dico[cle] item devient la clé de dico_inv & cle devient la dico_inv [item] = cle valeur de dico_inv
     return dico_inv
```

Programme test

```
#Echange des clés et des valeurs dans un dictionnaire
print (dico)
print(inverse(dico))
```

Il faut que la relation soit univoque cad par cle 1 seule valeur (car sinon on aura plusiuers clés les mêmes)





Ouvrir un fichier

- ➤ Sous Python, l'accès aux fichiers est assuré par l'intermédiaire d'un objet-interface particulier, que l'on appelle objet-fichier.
- La création de cet objet se fait à l'aide de la fonction intégrée open ().
- ➤ Cette fonction renvoie un objet ayant des méthodes (fonctionnalités) spécifiques, qui vous permettront de lire et/ou écrire dans le fichier.

1

Ecrire un fichier

fichier véritable (sur disque ou disquette)

objet-fichier

#Ecriture séquentielle

- Création de l'objet-fichier obFichier, qui fait référence à un fichier véritable dont le nom sera Monfichier.
- ➤ La fonction open () attend deux arguments de type chaînes de caractères. Le premier est le nom du fichier à ouvrir et le deuxième la méthode d'ouverture.
- > Ici le mode d'ouverture "a" = il faut ouvrir ce fichier en mode "ajout". C'est comme si on fais ait « append » : on rajoute à la fin du fighier
- Il existe un autre mode "w", qui crée toujours un nouveau fichier lorsqu'on ajoute des données (il « overwrite ») contrairement à la méthode "a" qui ajoute les données à la fin du fichier. On supprime l'ancien fichier et on remplace par les nouvelles données

18

C'est un flux qui va de l'exécution du programme vers un emplacement du disque dur

- 1er argument : nom de ma chaîne

- 2ème argument : soit « a » ajout à

la fin du fichier ou « w » supprime l'ancien fichier et le remplace par

les nouvelles données

de caractère - virgule « , »

Lire un fichier

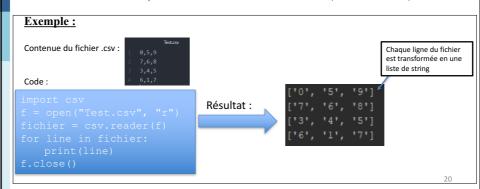
- ➤ Une fois ouvert en mode « lecture », on peut lire le contenu ligne par ligne.
- ➤ Ensuite, parcourir le contenu ligne par ligne à l'aide d'une boucle *for*.

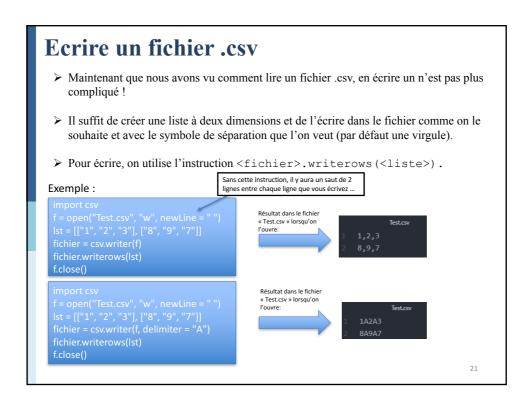
```
#Lecture séquentielle
                                  Nom du fichier à ouvrir
ofi = open('Monfichier.txt', 'r')
                                                     Ouverture du fichier
t = ofi.readlines()
                                                     en mode « lecture »
for line in t :
   print(line)
ofi.close()
                                              Affiche ligne par ligne
                                              le contenue du fichier
Bonjour, fichier !
                                              « fichier.txt »
Quel beau temps
                    aujourd'hui !
                    referme le fichier après usage
```

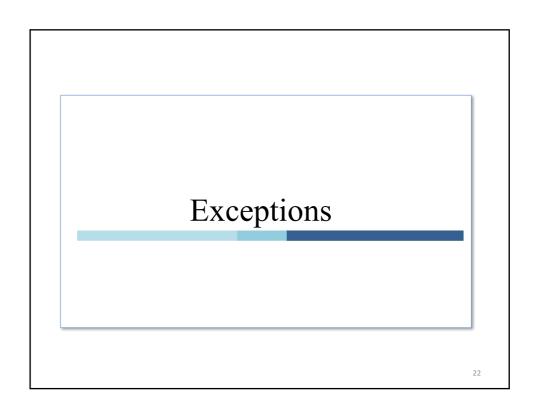
➤ Il existe aussi la méthode read () qui lit les données présentes dans le fichier et les transfère dans une variable de type chaîne de caractères (string).

Lire un fichier .csv

- > Python propose une bibliothèque pour la manipulation de fichiers .csv.
- ➤ Comme vue précédemment, pour utiliser des « modules » en python, il faut mettre en tête de page : import <module>.
- Le module pour les fichiers .csv est : csv
- Pour lire un fichier, on utilise la méthode « reader (<fichier>) ».







Exceptions

➤ Lorsqu'il y a une erreur dans un code, l'exécution est généralement interrompue et un message d'erreur plus ou moins explicite est affiché:

#Exception
print(22/0)
ZeroDivisionError: int division or modulo by zero

Type d'erreur Information spécifique sur l'erreur

2:

Exceptions

- ➤ Dans de nombreux cas, il est possible de prévoir à l'avance certaines des erreurs qui peuvent se produire.
- ➤ Il faut alors inclure dans le programme (à l'endroit où l'erreur pourrait se produire) des instructions particulières indiquant ce qu'il faut faire si l'erreur se produit.
- Ces instructions spécifiques ne seront jamais exécutées si l'erreur ne se produit pas.

Bloc try - except - else

- > Python utilise les instructions *try except else* pour gérer les erreurs.
- Le bloc d'instructions qui suit directement une instruction try est exécuté sous réserve.
- ➤ Si une erreur survient pendant l'exécution de l'une de ces instructions, alors Python annule cette instruction et exécute à sa place le code inclus dans le bloc qui suit l'instruction except.
- ➤ Si aucune erreur traitée dans les instructions **except** ne s'est produite, alors c'est le bloc qui suit l'instruction **else** qui est exécuté.
- Dans tous les cas, l'exécution du programme peut se poursuivre ensuite avec les instructions ultérieures.

25

Exemple

```
#Bloc try - except - else
def existe(fname):
                               Si une erreur arrive, on va dans
       f = open(fname,'r')
                               le « except ». Au sinon, le
                               « except » n'est pas exécuté.
       f.close()
       return True
                               Cela permet à ce que le code ne
                               s'arrête pas en cas d'erreur
   except:
       return False
filename = input("Veuillez entrer le nom du fichier : ")
if_existe(filename):
   print("Ce fichier existe bel et bien.")
   print("Le fichier", filename, "est introuvable.")
```

➤ Il est également possible de faire suivre l'instruction **try** de plusieurs blocs **except**, chacun d'entre eux traitant un type d'erreur spécifique (voir documentation Python).

Application fichiers et exceptions

2

Générer une matrice à partir d'un fichier .csv

Vous voulez créer une matrice à partir d'un fichier .csv. De plus, vous voulez des integers à la place de strings pour pouvoir faire des calculs par la suite. /! \ Vous devez gérer tous les cas d'erreur qui pourraient se produire.

Solution:

```
import csv

def get_a_list(fichier):
    try:
    f = open(fichier, "r")
    fichier_ouvert = csv.reader(f)
    lst = []
    try:
    for line in fichier_ouvert:
        lst_bis = []
        for value in line:
            value_bis = int(value)
            lst_append(value_bis)
        lst.append(lst_bis)
        f.close()
        return lst
    except:
        f.close()
        print ("Valeur non « castable » en integer")
        return 0
    except:
        print ("Votre fichier n'existe pas")
    return 0

ma_liste = get_a_list("Test.csv")
    print (ma_liste)
```