Traitement des fichiers du jour-19

Manutention de fichiers

Jusqu'à présent, nous avons vu différents types de données Python. Nous stockons généralem ent nos données dans différents formats de fichiers. En plus de gérer les fichiers, nous verron s également différents formats de fichiers (.txt, .json, .xml, .csv, .tsv, .excel) dans cette sectio n. Tout d'abord, nous familiariser avec la gestion des fichiers avec le format de fichier comm un (.txt).

La gestion des fichiers est une partie d'importation de la programmation qui nous permet de créer, lire, mettre à jour et supprimer des fichiers. Dans Python pour gérer les données, nous utilisons la fonction intégrée *open()*.

```
# Syntaxe
Open («nom de fichier», mode) # mode (r, a, w, x, t, b) pourrait être pour lire, écrire, mettr e à jour
```

- "R" - Lire - Valeur par défaut. Ouvre un fichier pour la lecture, il renvoie une erreur si le fichier n'existe pas
- "A" - APPEND - ouvre un fichier pour ajouter, crée le fichier s'il n'existe pas
- "W" - Write - ouvre un fichier pour l'écriture, crée le fichier s'il n'existe pas
- "x" - Create - crée le fichier spécifié, renvoie une erreur si le fichier existe • "t" - t ext - valeur par défaut. Mode du texte • "B" - Binaire - Mode binaire (par exemple)

Fichiers d'ouverture pour la lecture

Le mode par défaut de *open* est la lecture, nous n'avons donc pas à spécifier «R» ou «RT». J'a i créé et enregistré un fichier nommé lecture_file_example.txt dans le répertoire des fichiers. Voyons comment cela se fait:

```
f = ouvrir ('./ files / read_file_example.txt') print (f) # < _io.textiowmarap
per name = './ files / read_file_example.txt' mode = 'r' Encoding = 'utf-8' >
```

Comme vous pouvez le voir dans l'exemple ci-dessus, j'ai imprimé le fichier ouvert et il a donné quelques informations à ce sujet. Le fichier ouvert a des méthodes de lecture différ entes: *read()*, *readline*, *readlines*. Un fichier ouvert doit être fermé avec la méthode *close()*.

- *read()*: Lisez tout le texte en tant que chaîne. Si nous voulons limiter le nombre de caractères que nous voulons lire, nous pouvons le limiter en passant la valeur int à la méthode *read(number)*.

```
f = open('./files/reading_file_example.txt')
txt = f.read()
```

```
imprimer (type (txt))
imprimer (txt) f.
close ()
# output < class 'str' > Ceci est un exemple pour montrer comment ouvrir un fichier
et lire.  Ceci est la deuxième ligne du texte.
```

Au lieu d'imprimer tout le texte, imprimons les 10 premiers caractères du fichier texte.

```
f = ouvert ('./ fichiers / lise_file_example.txt')
txt = f.read (10) print (typ
e (txt)) print (txt) f.close (
)
```

```
# output
<class 'str'>
This is an
```

- *readline()*: ne lisez que la première ligne

```
f = ouvert ('./ fichiers / lise_file_example.txt')
line = f.readline () print (type
(ligne)) print (ligne) f.close ()



# output < class 'str' > Ceci est un exemple pour montrer comment ouvrir un fichier
et lire.
```

- *readlines()*: Lisez tout le texte en ligne par ligne et renvoie une liste de lignes

```
f = ouvert ('./ fichiers / lise_file_example.txt')
lignes = f.readlines () print (type
(lignes)) imprimer (lignes) f.clos
e ()


# output < class 'list' > ['Ceci est un exemple pour montrer comment ouvrir un fichier et lire. \
n', 'Ceci est la deuxième ligne du texte.']
```

Another way to get all the lines as a list is using *splitlines()*:

```
f = ouvrir ('./ files / read_file_example.txt')
lignes = f.read (). Splitlines () print (type (lig
nes)) imprimer (lignes) f.close ()
```

```
# output < class 'list' > ['Ceci est un exemple pour montrer comment ouvrir un fichier et lir
e.', 'Ceci est la deuxième ligne du texte.']
```

Après avoir ouvert un fichier, nous devons le fermer. Il y a une forte tendance à oublier à les f ermer. Il existe une nouvelle façon d'ouvrir des fichiers à l'aide de *with* - ferme les fichiers seu ls. Reprécions l'exemple précédent avec la méthode *with*:

```
avec Open ('./ Files / Reading_File_Example.txt') comme f:
lignes = f.read (). Splitlines () print (type (lignes)) i
mprimer (lignes)


# sortir
<classe 'liste' > ['Ceci est un exemple pour montrer comment ouvrir un fichier et lire.', 'Ce
ci est la deuxième ligne du texte.']
```

Fichiers d'ouverture pour l'écriture et la mise à jour

Pour écrire dans un fichier existant, nous devons ajouter un mode en tant que paramètre à la fonction *open()*:

- "A" - Ajouter - Ajoutera à la fin du fichier, si le fichier ne crée pas un nouveau fichier.
- "W" - WRITE - RÉQUIRERA TOUT CONTENU EXISTANT, si le fichier n'existe pas, i l crée.

Ajoutez un texte au fichier que nous avons lu:

```
avec Open ('./ Files / Reading_File_Example.txt', 'A') As F:
F.Write («Ce texte doit être ajouté à la fin»)
```

La méthode ci-dessous crée un nouveau fichier, si le fichier n'existe pas:

```
avec open ('./ files / write_file_example.txt', 'w') comme f: f.write ('ce texte sera écrit da
ns un fichier nouvellement créé')
```

Suppression de fichiers

Nous avons vu dans la section précédente, comment créer et supprimer un répertoire à l'aide du module *os*. Encore une fois, si nous voulons supprimer un fichier, nous u tilisons le module *os*.

```
Importer un système d'exploitation
os.remove ('./ fichiers / example.txt')
```

Si le fichier n'existe pas, la méthode de suppression augmentera une erreur, il est donc bon d'u tiliser une condition comme celle-ci:

```
Importer OS si os.path.exists ('./ fichiers / example.txt'): os.rem
ove ('./ fichiers / example.txt') else: print ('le fichier n'existe pas'
))
```

Types de fichiers

Fichier avec l'extension TXT

Le fichier avec l'extension *txt* est une forme très courante de données et nous l'avons couvert e dans la section précédente. Passons au fichier JSON

Fichier avec l'extension JSON

JSON représente la notation d'objet JavaScript. En fait, il s'agit d'un objet JavaScript String ified ou d'un dictionnaire Python.

Example:

```
# Dictionary Person_dct = {"name": "Asabeneh", "country": "Finland", "ville": "Helsin
ki", "compétences": ["javascrip", "react", "python"]} # json: une chaîne forme un dictio
nnaire personne_json = "{'name': " Asabneh ',' Country ':': '}" {' name ": ' 'City': 'Helsin
ki', 'Skills': ['Javascrip', 'React', 'Python']} "









# Nous utilisons trois citations et faisons de plusieurs lignes pour la rendre plus lisible

Person_json = " '{"name": "asabeneh", "country": "Finlande", "ville
": "Helsinki", "compétences": ["javascrip", "react", "python"]}' " '
```

Changer JSON en dictionnaire

Pour changer un JSON en un dictionnaire, nous importons d'abord le module JSON, pui
s nous utilisons la méthode *loads*.

```
Importer Json # JSON Person_json = " '{"Name": "Asabeneh", "Count
ry": "Finland", "City": "Helsinki", "Skills": ["Javascrip", "React", "Pyt
hon"]}' " # Let's Changes Json en Dictionary Person_dct = JSON.Load
s (Person_jon) print (type (Person_dct)) print (Person_dct) print (Perso
n_dct ['name'])









# output < class 'dict' > {'name': 'asabeneh', 'country': 'Finland', 'City': 'Helsinki', 'compétences':
['javascrip', 'react', 'python']} Asabeneh
```

Changer le dictionnaire en JSON

Pour changer un dictionnaire en une méthode JSON, nous utilisons la méthode *dumps* du module JSON.

```
Importer JSON # Python Dictionary Person = {"Name": "Asabeneh", "
Country": "Finlande", "City": "Helsinki", "Skills": ["Javascrip", "React
", "Python"]}









# Convertissons-le en JSON Person_json = JSON.Dumps (personne, indent = 4) # L'endent
pourrait être 2, 4, 8. Il embellit l'impression JSON (Type (Person_json)) Print (Person_json)
```

```
# sortir
# Lorsque vous l'imprimez, il n'a pas la citation, mais en fait c'est une chaîne

# JSON n'a pas de type, c'est un type de chaîne.  < classe 'str' > {"name"
: "asabeneh",
```

```
"pays": "Finlande",
"City": "Helsinki", "Compétences":
["javascrip", "réagir", "python"]}
```

Enregistrer en tant que fichier JSON

Nous pouvons également enregistrer nos données en tant que fichier JSON. Laissez-le enre gistrer en tant que fichier JSON en utilisant les étapes suivantes. Pour écrire un fichier JSO N, nous utilisons la méthode JSON.Dump (), il peut prendre le dictionnaire, le fichier de so rtie, l'assurance_ascii et le retrait.

```
Importer JSON # Python Dictionary Person = {"name": "Asabeneh", "country": "Finland", "C
ity": "Helsinki", "Skills": ["Javascrip", "react", "Python"]} avec Open ('./ Files / JSON_Exam
ple.json', 'W', Encoding = json.dump (personne, f, assure_ascii = false, indent = 4)
```

Dans le code ci-dessus, nous utilisons le codage et l'indentation. L'indentation rend le fichier J SON facile à lire.

Fichier avec l'extension CSV

CSV signifie des valeurs séparées par les virgules. CSV est un format de fichier simple utilisé pour stocker des données tabulaires, telles qu'une feuille de calcul ou une base de données. Le CSV est un format de données très courant en science des données.

Exemple:

```
"Nom", "Country", "City", "Skills" "Asabeneh", "Finlande", "Helsi
nki", "Javascript"
```

Exemple:

```
Importez CSV avec Open ('./ Files / CSV_Example.csv') en tant
que F:
```

```
    csv_reader = csv.reader(f, delimiter=',') # w use, reader
method to read csv
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are :{", ".join(row)}')
            line_count += 1
        else:
            print(
                f'\t{row[0]} is a teachers. He lives in
{row[1]}, {row[2]}.')
            line_count += 1
    print(f'Number of lines:  {line_count}')

# output:
Column names are :name, country, city, skills
        Asabeneh is a teacher. He lives in Finland, Helsinki.
Number of lines:  2
```

Fichier avec l'extension XLSX

Pour lire les fichiers Excel, nous devons installer le package *xlrd*. Nous le couvrirons après a voir couvert l'installation du package à l'aide de PIP.

```
importer xlrd excel_book = xlrd.open_workbook ('samptans.xls) pri
nt (excel_book.nsheets) print (excel_book.sheet_names)
```

Fichier avec l'extension XML

XML est un autre format de données structuré qui ressemble à du HTML. Dans XML, les bal ises ne sont pas prédéfinies. La première ligne est une déclaration XML. La balise de personn e est la racine du XML. La personne a un attribut de genre. Exemple: XML

```
<?xml version="1.0"?>
<person gender="female">
  <name>Asabeneh</name>
  <country>Finland</country>
  <city>Helsinki</city>
  <skills>
    <skill>JavaScrip</skill>
    <skill>React</skill>
    <skill>Python</skill>
  </skills>
</person>
```

Pour plus d'informations sur la façon de lire un fichier XML, consultez la documentation

```
Importer xml.etree.elementTree en tant qu'Et Tree = et.parse ('./ f
iles / xml_example.xml') root = arbre.getroot () print ('root tag:',
root.tag) print ('attribute:', root.attrib) pour l'enfant: print ('champ
:', enfant.tag)
```

```
# SORTIE ROOT TAG: Attribut de personne
: {'Gender': 'Male'} Field: Nom Field: Countr
y Field: City Field: Compétences
```

🎯Vous faites de grands progrès. Maintenez votre élan, gardez le bon travail. Faites mainte
nant quelques exercices pour votre cerveau et vos muscles.

Exercices: Jour 19

Exercices: niveau 1

1. Écrivez une fonction qui compte le nombre de lignes et le nombre de mots dans un texte. Tous les fichiers sont dans les données du dossier: a) Lisez le fichier obama_speech.txt et c omptent le nombre de lignes et de mots b) Lisez le fichier michelle_obama_speech.txt et le nombre de lignes et de mots c) lire le fichier Donald_speech.txt et le nombre de lignes et d e mots d) Lire Melina_Trump_Speech.tx

2. Lisez le fichier de données Pays_data.json dans le répertoire de données, créez une fonc tion qui trouve les dix langues les plus parlées

```
# Votre sortie devrait ressembler à cette impression (Most_spoken_languages (nom de fichier = './ data / pays_data.js sur', 10)) [(91, `` anglais "), (45, `` français "), (25, `` Arabic "), (24, `` espagnol "), (9, «russe»), (9, «Portugèse»), (8, ». «Allemand»), (5, «chinois»), (4, «Swahili»), (4, «serbe»)]
```

```
# Votre sortie devrait ressembler à ceci
Print (Most_spoken_languages (nom de fichier = './ data / PAYS_DATA.js sur', 3))

[(91, «anglais»), (45, «fran
çais»), (25, «arabe»)]
```

3. Lisez le fichier de données Pays_data.json dans le répertoire de données, créez une fonc tion qui crée une liste des dix pays les plus peuplés

```
# Votre sortie doit ressembler à cette impression (Most_populaled_Countries (nom de fichier = './ data / pays_data .json', 10))


[  {'country': 'China', 'population': 1377422166},  {'country': 'India', 'population': 1295210000},  {'country': 'United States of America', 'population':  323947000},  {'country': 'Indonesia', 'population': 258705000},
```

{'Country': 'Brésil', 'Population': 206135893}, {'country': 'Pakistan', 'Population': 1941250 62}, {'Country': 'Nigeria', 'Population': 186988000}, {'country': 'Bangladesh', 'Population': 161006790}, ':' Bangladesh '. Fédération ',' Population ': 146599183}, {' country ':' Japon ',' Population ': 126960000}]

Votre sortie devrait ressembler à ceci

Imprimer (Most_populaled_Countries (nom de fichier = './ data / PAYS_DATA .JSON', 3)) [{'country': 'China', 'Population': 1377422166}, {'country': 'India', 'Population': 1295210000}, {Country ':' États-Unis d'Amérique ',': ': 323947

Exercices: niveau 2

4. Extraire toutes les adresses e-mail entrantes en tant que liste du fichier email_exchange _big.txt. 5. Trouvez les mots les plus courants en anglais. Appelez le nom de votre foncti on find_most_common_words, il faudra deux paramètres - une chaîne ou un fichier et un entier positif, indiquant le nombre de mots. Votre fonction renverra un tableau de tuples d ans l'ordre descendant. Vérifiez la sortie

```
# Votre sortie devrait ressembler à ceci
print (find_most_common_words ('sample.txt', 10))
[(10, «le»),
(8, «être»),
(6, «à»),
(6, «de»),
(5, 'et'),
(4, «a»),
(4, «in»),
(3, «that»),
(2, «avoir»),
(2, «i»)]
# Votre sortie doit ressembler à cette impression (find_most_common_words
('samptère.txt', 5))
```

[(10, «le»),
(8, «être»),
(6, «à»),
(6, «de»),

```
    (5, 'and')]
```

Félicitations!