2N6 Programmation 2

python™

Approfondissement des strings

Ex1 – string.py

Ex de f-string

retour

ex_f-strings.py

```python
salutation = "Bonjour et bienvenue"
nom = "gallant"
prenom = "Pierre-Paul"

print(f"\n{salutation} Mr.{prenom} {nom.capitalize()} au cours 2N6 pour réseautique.\n")
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL .NET INTERACTIVE JUPYTER AZURE COMMENTS

```
PS C:\> & "C:/Program Files/Python310/python.exe" "c:/Users/pierre-paul.gallant/OneDrive - Cégep Édouard-Montpeti

Bonjour et bienvenue Mr.Pierre-Paul Gallant au cours 2N6 pour réseautique.

PS C:\>
```

DEMO

Les strings

- Les objets « str » peuvent être considérés comme des listes de caractères.
- Mais ils sont immuables (ne peuvent pas être changés). Les sous strings générés sont de nouveaux objets.
- Toutes les opérations de « slicing » peuvent être effectuées sur des « strings ».

```
>>> liste1 = "Bonjour le monde"
>>> print(liste1)
Bonjour le monde
```

```
>>> print(liste1[0:5])
Bonjo
>>> print(liste1[:5])
Bonjo
```

```
>>> print(liste1[2:])
njour le monde
```

```
>>> print(liste1[-1])
e
>>> print(liste1[-5:])
monde
```

Méthodes courantes des strings

- split ("séparateur") : retourne liste les portions de strings en séparant le string initial selon un caractère passé en paramètre
- .strip() retire les espaces au début et à la fin
  - lstrip() retire les espaces au début du string (left-strip)
  - rstrip() retire les espaces à la fin du string (right-strip)
- .find("text") et index() retournent l'index du premier caractère de la séquence trouvée
  - Si on veut uniquement une valeur booléenne : utilisez le mot-clef "in". Ex : "chou" in "chou-fleur" -> True
- .lower() : retourne un str où toutes les lettres sont minuscules.
- .upper() : retourne un str où toutes les lettres sont majuscule.
- Méthodes is… (isnumeric(), islower(), usascii()… etc.) : retournent une valeur booléenne indiquant si le string correspond à ce que la méthode vérifie.
- .zfill() ajoute des zéros à gauche d'un string pour obtenir un string d'une taille prédéfinie
- .encode() change l'encodage des caractères du string

! Un string est immutable. Toutes les méthodes retournent un nouveau string ou autre valeur.

Les méthodes des strings

- encode() change l'encodage des caractères du string

```
>>> cours = "réseau 1"
>>> cours.encode("utf-8")
```

- find() et index() retournent l'index du premier caractère de la séquence trouvée

```
>>> print(cours.find("eau"))
3
```

- Les méthodes is... retournent une valeur booléenne indiquant si le string correspond à ce que la méthode vérifie.

```
>>> "3".isdigit()
True
```

Les méthodes des strings

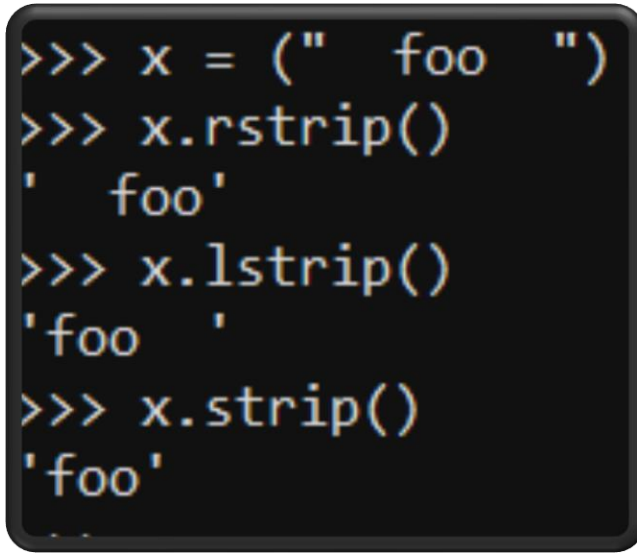
- split() / rsplit() retournent les portions de strings en séparant le string initial selon un caractère passé en paramètre.
  - Peut-être capturé dans une seule variable sous forme de liste.
  - Peut aussi être capturé dans plusieurs variables distinctes

```
>>> cours = "réseau1/prog1/prog2/prog3"
>>> liste_cours = cours.split("/")
>>> print(liste_cours)
['réseau1', 'prog1', 'prog2', 'prog3']
>>>
```

```
>>> cours = "réseau1/prog1"
>>> cours1, cours2 = cours.split("/")
>>> print(cours1)
réseau1
>>> print(cours2)
prog1
>>>
```

Les méthodes des strings

- lstrip() retire les espaces au début du string (left-strip)
- rstrip() retire les espaces à la fin du string (right-strip)
- strip() retire les espaces au début et à la fin



Les méthodes des strings

- zfill() ajoute des zéros à gauche d'un string pour obtenir un string d'une taille prédéfinie

```
>>> y = ["1-foo","2-bar","10-byr"]
>>> print(y.sort())
None
>>> y = ["1-foo","2-bar","10-byr"]
>>> y.sort()
>>> print(y)
['1-foo', '10-byr', '2-bar']
```

```
>>> for i in y : x.append(i.zfill(6))
...
>>> print(x)
['01-foo', '10-byr', '02-bar']
>>> x.sort()
>>> print(x)
['01-foo', '02-bar', '10-byr']
>>>
```

*

*ce n'est qu'un exemple de boucle. zfill() peut être utilisé sur n'importe quel string

Introduction aux modules

Ex-2 os.py

Le module os

- os.chdir(`"chemin"`) : change le répertoire dans lequel l'interprétateur « agit »
- os.getcwd() : affiche le répertoire courant
- os.listdir() :liste les répertoires et fichiers
- os.mkdir(`"nom_répertoire"`) : crée un répertoire
- os.makedirs(`"repertoire/repertoire"`) : crée répertoire(s) et sous-répertoire(s)
- os.rmdir() : supprime un répertoire VIDE

Le module os

- os.environ : dictionnaire des variables de système
  - getenv() et putenv() pour obtenir ou modifier ces variables
- os.remove() : supprime un fichier
- os.rename(`"nom_répertoire"`, `"nouveau nom"`) : renomme un fichier ou répertoire
- os.path : sous-module pour travailler avec les paths
- os.stat(`"nom_répertoire"`) : donne des informations sur le fichier / répertoire passé en argument

Le module OS

- Fait partie de la librairie standard
- Permet d'accéder aux données et d'interagir avec l'os

os_exemple.py > ...

```
1   import os
2   info_env = os.environ
3   home = info_env.get("HOME")
4   os.mkdir(f"{home}/scripts2")
5   os.chdir(f"{home}/scripts2")
6   os.makedirs("resultats/recursif")
7   print(os.getcwd())
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
pierre-paul@pp-vm:~/scripts$ /bin/python3 /home
/home/pierre-paul/scripts2
pierre-paul@pp-vm:~/scripts$ tree ../scripts2
../scripts2
└── resultats
    └── recursif

2 directories, 0 files
```

- os. chdir() : change le répertoire courant
- ~~os. mkdir() : créer **un** répertoire~~
- os. makedirs() : crée **un ou plusieurs** répertoires de façon récursive
- os. getcwd() : retourne un str indiquant le répertoire courant

python.org/3/library/os.html

Le sous-module path

- Contient de nombreuses fonctions spécifiques à la manipulations de path

```
# différentes fonctions du sous-module path. Le fichier dem
print(os.path.basename(f'{chemin}/demo4.txt'))  # 'demo4.txt'
print(os.path.dirname(f'{chemin}/demo4.txt'))   # 'c:\\Users\\......\\R04_demo'
print(os.path.split(f'{chemin}/demo4.txt'))     # ('c:\\Users\\......\\R04_demo', 'demo4.txt')
print(os.path.exists(f'{chemin}/demo4.txt'))    # True
print(os.path.isfile(f'{chemin}/demo4.txt'))    # True
print(os.path.isdir(f'{chemin}/demo4.txt'))     # False
print(os.path.splitext(f'{chemin}/demo4.txt'))  #('c:\\Users\\......\\R04_demo', '.txt')#
```

Fichiers et Répertoires

Comment interagir avec les fichiers avec python

La fonction open()

- Permet facilement d'interagir avec des fichiers
- Peut créer des fichiers ou ouvrir des fichiers existants
- Deux paramètres essentiels :

```
with open("status.conf", "w") as fichier_lu :
```

Fichier à ouvrir

Mode d'ouverture

Variable contenant un objet correspondant au fichier créé. Permet d'interagir facilement avec ce fichier.

La fonction open()

- Les modes d'ouverture :

```
with open("status.conf", "w") as fichier_lu :
```

r Ouvre pour lecture seulement.

w Ouvre pour écriture, créé le fichier si nécessaire.
Écrase fichier existant. (Ne garde pas ce qui est déjà dans le fichier.)

x Créé un nouveau fichier et l'ouvre pour écriture.
Échoue si le fichier existe déjà.

a Ouvre pour écriture MAIS concatène à la fin si le fichier existe.

L’instruction “with”

- À moins de circonstances extraordinaires, on utilise tout le temps l’instruction « with » lorsqu’on utilise la fonction « open() »

- Ici, on crée un nouveau fichier vide avec open() et on le **ferme automatiquement en sortant** de l’instruction "with"

Excellent, sécuritaire

```
2    with open("test.txt","w") as file:
3        pass
```

Interopérabilité

Interopérabilité

*« **L'interopérabilité** est la capacité que possède un produit ou un système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs et ce sans restriction d'accès ou de mise en œuvre. »*

Source: Wikipédia

Travailler avec les CSVs

- CSV : un fichier texte dont les données sont délimitées par un caractère spécial (une virgule (**,**) par défaut).

- Exemple où le 'délimiter' est un point-virgule (**;**)
- La première ligne contient les en-têtes de colonnes.

```
No étudiant;Groupe;Nom de l'étudiant;Prénom de l'étudiant;Prog.
2273383;1010;Carrier;Alexandre;420.BU
2222119;1010;Dion;Paul;420.BB
2229304;1010;Douida;Wissale;420.BA
2218593;1010;El Ammari;Amar;420.BA
```

Formats de sérialisation

JSON

```
[
    {
        "nom": "Anna",
        "id": 543564,
        "programme": "420-INFO"
    },
    {
        "nom": "Greg",
        "id": 987123,
        "programme": "420-INFO"
    },
 {
        "nom": "Bob",
        "id": 369852,
        "programme": "238-FRAN"
    },
 {
        "nom": "Joseph",
        "id": S753869,
        "programme": "135-PHYS"
    },
 {
        "nom": "Hubert",
        "id": 125478,
        "programme": "238-FRAN"
    },
]
```

Sous forme de tableau :

	A	B	C
1	nom	id	programme
2	Anna	543564	420-INFO
3	Greg	987123	420-INFO
4	Bob	369852	238-FRAN
5	Joseph	753869	135-PHYS
6	Hubert	125478	238-FRAN
7	Zeus	659327	135-PHYS
8	Joel	583649	420-INFO
9			

CSV

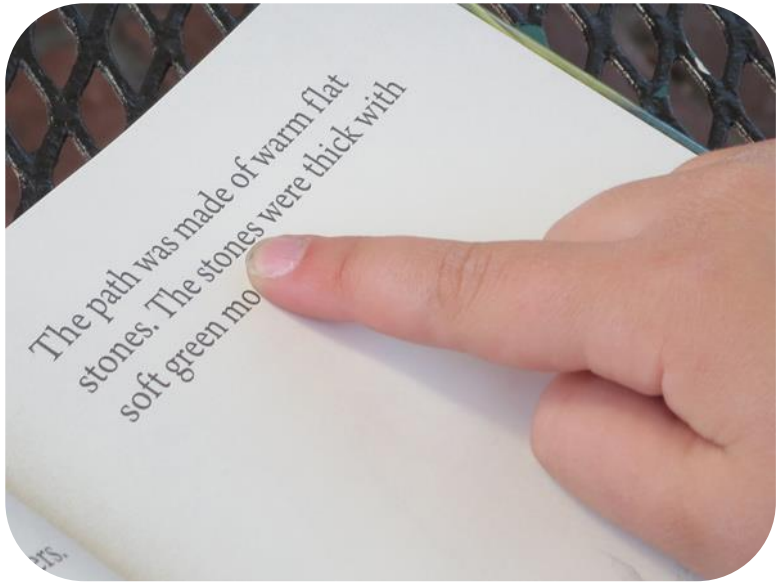
```
nom,id,programme
Anna,543564,420-INFO
Greg,987123,420-INFO
Bob,369852,238-FRAN
Joseph,753869,135-PHYS
Hubert,125478,238-FRAN
Zeus,659327,135-PHYS
Joel,583649,420-INFO
```

Travailler avec les CSVs

- On utilise le module csv, conçu spécifiquement pour travailler de façon efficace avec des fichiers CSVs

```
3   import csv
4
5   with open('etudiants.csv', 'r',encoding='utf-8') as csv_file:
6       csv_reader = csv.reader(csv_file)
7       # saute la première ligne #
8       next(csv_reader)
9       for line in csv_reader:
10          print(line)
11
```

Lire les CSVs



- Pour lire un csv. On commence par instancier un objet à partir de la classe **Reader** avec le fichier qu'on désire lire en paramètre.
- Cet objet est itérable et peut donc être utilisé avec une boucle for.

```
3   import csv
4
5   with open('etudiants.csv', 'r',encoding='utf-8') as csv_file:
6       csv_reader = csv.reader(csv_file)
7       # saute la première ligne #
8       next(csv_reader)
9       for line in csv_reader:
10          print(line)
11
```

- On peut le comparer à un pointeur qui indique où nous sommes rendus dans le fichier qu'on lit

Lire les CSVs

```
import csv 1.

with open('etudiants.csv',"r") as csv_file: 2.
    csv_reader = csv.reader(csv_file) 3.
    # saute la première ligne #
    next(csv_reader)  4.
    for line in csv_reader : 5.
        print(line)
```

```
csv_reader = [
                ['nom', 'id', 'programme']
                ['Anna', '543564', '420-INFO']
                ['Greg', '987123', '420-INFO']
                ['Bob', '369852', '238-FRAN']
                ['Joseph', '753869', '135-PHYS']
                ['Hubert', '125478', '238-FRAN']
                ['Zeus', '659327', '135-PHYS']
                ['Joel', '583649', '420-INFO']
              ]
```

1. Importation du module csv.
2. Ouverture du fichier .csv qu'on veut lire.
3. Création d'un objet **Reader** (nécessaire à la lecture du fichier)
4. (Optionnel) Saute la première ligne du fichier. Généralement celle avec les titres.
5. L'objet Reader est itérable, il se comporte un peu comme une liste de listes.

Écrire un CSV

- Le module csv permet aussi de créer facilement des CSVs à partir de nos données
- On crée un objet à partir de la classe **Writer** avec le fichier dans lequel on veut écrire en paramètre.
- Ici, on itère sur la structure de données contenant l'information à écrire.

```
1   import csv
2   liste_employers = [["steve","E211"],["Ana","A809"],["Jon","B32"],["Sophie","N\\A"]]
3
4   with open('listeEmployes.csv', 'w',encoding='utf-8') as csv_file:
5       csv_writer = csv.writer(csv_file, delimiter = ';', lineterminator="\n")
6       csv_writer.writerow(["Employé", "Bureau"])
7       for line in liste_employers:
8           csv_writer.writerow(line)
```

Écrire un CSV

```
with open("listeEmployes.csv","w") as csv_file :
    csv_writer = csv.writer(csv_file,
                            delimiter=";",
                            lineterminator="\n")
    csv_writer.writerow("Employe","Bureau")
    for line in liste_employers:
        csv_writer.writerow(line)
```

1. Ouverture / création du fichier .csv en mode écriture.
2. Création d'un objet de la classe **Writer** pour écrire un CSV. Les paramètres optionnels :
   - *delimiter* : le charactère qui sépare les valeurs
   - *lineterminator* : charactère de fin de ligne
3. (Optionnel) la méthode .writerow() Écrit une ligne. Dans ce cas : les titres des colonnes.
4. On écrit toutes les informations dans le fichier csv. Dans ce cas en une seule ligne avec une liste