Fonctions prédéfinies démo en classe

November 19, 2019

1 Fonctions prédéfinies (chaînes de caractères)

On utilise une fois de plus la série de variables suivantes pour faire les manipulations.

[7]:
```
# Liste des décimales de pi
liste_dec = [1,4,1,5,9,2,6,5,3,5,8,9,7,9,3,2,3,8,4,6,2,6,4,\
             3,3,8,3,2,7,9,5,0,2,8,8,4,1,9,7,1,6,9,3,9,9,3,\
             7,5,1,0,5,8,2,0,9,7,4,9,4,4,5,9,2,3,0,7,8,1,6,\
             4,0,6,2,8,6,2,0,8,9,9,8,6,2,8,0,3,4,8,2,5,3,4,2]

no_dept_aquitaine = [16, 17, 19, 23, 24, 33, 40, 46, 64, 79, 86, 87]

noms_dept_aquitaine = ["charente", "charente-maritime", "correze", "creuse", \
                       "dordogne", "gironde", "landes", "lot-et-garonne", \
                       "pyrenees-atlantiques", "deux-sevres", "vienne",␣
 ↪"haute-vienne"]

texte1 = "Le narrateur rencontre dans un bus un jeune homme au long cou, coiffé␣
 ↪d'un chapeau mou orné d'une tresse tenant lieu de ruban. Ce quidam échange␣
 ↪quelques mots assez vifs avec un autre voyageur, puis va s'asseoir à une␣
 ↪autre place. Un peu plus tard, le narrateur revoit le même jeune homme cour␣
 ↪de Rome devant la gare Saint-Lazare en train de discuter avec un ami qui lui␣
 ↪conseille d'ajuster (ou d'ajouter) un bouton de son pardessus."

stop_words = ["le", "la", "un", "une", "un"]
```

Notation “pointée”: la fonction *upper()* qui met une chaîne en majuscules s’applique à un argument. Mais au lieu de la notation *upper(s)* que l’on rencontre fréqumment, en python on emploie une notation pointée (“objet”) qui peut s’appliquer sur des variables (comme *machine* ci-dessous) comme à des constantes (comme “palais” ci-dessous). N.B. le fait s’appliquer la méthode *upper()* à la variable “machine” ne change pas le contenu de celle-ci.

[1]:
```
machine = "maison"
print(machine.upper())
print(machine)
print("palais".upper())
```

```
MAISON
```

```
maison
PALAIS
```

1.1 Fonctions de recherche d'une (sous-)chaîne dans une chaîne

startwidth() et *endswith()* sont des fonctions booléennes.

[5]:
```
print("mon beau navire".startswith("le"))
```

```
False
```

[6]:
```
print("mon beau navire".endswith("ire"))
```

```
True
```

Les fonctions principales de recherche sont *find* et *index*. On leur donne uns sous-chaîne à trouver dans la chaîne, et elles renvoient l'indice de la première occurrence de la sous-chaîne dans la chaîne. La différence entre les deux tient à leur comportement quand la sous-chaîne recherchée n'apparaît pas.

[11]:
```
print(texte1.find("cou"))
print(texte1[58:62])
print(texte1.find("xkcd"))
print(texte1.index("xkcd"))
```

```
58
cou,
-1


      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-11-dce92de896da> in <module>
         2 print(texte1[58:62])
         3 print(texte1.find("xkcd"))
   ----> 4 print(texte1.index("xkcd"))


       ValueError: substring not found
```

La fonction *count* permet de compter les occurrences d'une sous-chaîne.

[13]:
```
print(texte1.count('un '))
```

```
6
```

La fonction *split()* est très utile, elle découpe une chaîne en sous-chaîne séparées par ce qu'elle considère comme un séparateur.

1.2 Fonctions de découpage de chaînes

- strip() très utile pour ôter des caractères en fin de chaîne (ou en début)
- partition() coupe une chaîne en deux par rapport à un séparateur
- split() coupe une chaine en morceaux selon des séparateurs donnés en paramètre
- join() concatène des strings en insérant un séparateur It is always true that string.join(string.split(s, sep), sep) equals s.

[14]:
```
print(texte1.split())
```

```
['Le', 'narrateur', 'rencontre', 'dans', 'un', 'bus', 'un', 'jeune', 'homme',
'au', 'long', 'cou,', 'coiffé', "d'un", 'chapeau', 'mou', 'orné', "d'une",
'tresse', 'tenant', 'lieu', 'de', 'ruban.', 'Ce', 'quidam', 'échange',
'quelques', 'mots', 'assez', 'vifs', 'avec', 'un', 'autre', 'voyageur,', 'puis',
'va', "s'asseoir", 'à', 'une', 'autre', 'place.', 'Un', 'peu', 'plus', 'tard,',
'le', 'narrateur', 'revoit', 'le', 'même', 'jeune', 'homme', 'cour', 'de',
'Rome', 'devant', 'la', 'gare', 'Saint-Lazare', 'en', 'train', 'de', 'discuter',
'avec', 'un', 'ami', 'qui', 'lui', 'conseille', "d'ajuster", '(ou',
"d'ajouter)", 'un', 'bouton', 'de', 'son', 'pardessus.']
```

On peut proposer un séparateur spécifique.

[15]:
```
print(texte1.split('a'))
```

```
['Le n', 'rr', 'teur rencontre d', 'ns un bus un jeune homme ', "u long cou,
coiffé d'un ch", 'pe', "u mou orné d'une tresse ten", 'nt lieu de rub', 'n. Ce
quid', 'm éch', 'nge quelques mots ', 'ssez vifs ', 'vec un ', 'utre voy',
'geur, puis v', " s'", 'sseoir à une ', 'utre pl', 'ce. Un peu plus t', 'rd, le
n', 'rr', 'teur revoit le même jeune homme cour de Rome dev', 'nt l', ' g', 're
S', 'int-L', 'z', 're en tr', 'in de discuter ', 'vec un ', "mi qui lui
conseille d'", "juster (ou d'", 'jouter) un bouton de son p', 'rdessus.']
```

[17]:
```
print("charente-maritime".partition('-'))
```

```
('charente', '-', 'maritime')
```

1.3 Fonctions d'analyse d'une chaîne

- str.isalnum()
- str.isalpha()
- str.isascii()
- str.isdecimal()
- str.isdigit()
- str.isidentifier()
- str.islower()
- str.isnumeric()
- str.isprintable()

- str.isspace()

[18]:
```
print('MAISON'.islower())
```

```
False
```

1.4 Fonctions de transformation de chaîne

- upper()
- lower()
- capitalize()
- swapcase()