LETTRES SORBONNE UNIVERSITÉ

Modèles de Linguistique Computationnelle

CM 4 : expressions régulières et python - un pas plus loin

M1 Langue et Informatique

Crédits: Gaël Lejeune, Karën Fort, Iana Atanassova, Djame Seddah, Eleni Kogkitsidou, Olga Seminck

Alice Millour prenom.nom@sorbonne-universite.fr
Yoann Dupont prenom.nom@sorbonne-universite.fr

2020-2021

Sorbonne-Université

Organisation

Séance d'aujourd'hui

- (une) correction du TD2
- CM 4 : plus loin dans l'utilisation de la librairie re
- présentation d'un agent conversationnel (projet d'implémentation)
- TD 3

Plan

Partie Cours

Agent conversationnel simple

Pré-requis

vous connaissez :

- la syntaxe des expressions régulières
  (voir : http://joly415.perso.math.cnrs.fr/memento_python.pdf)
- le site https://regex101.com/ qui vous permet de tester vos expressions régulières

vous savez :

- charger la librairie python permettant de travailler avec les expressions régulières
- afficher l'aide correspondante
- comprendre comment s'utilise une fonction :
  - quels sont les paramètres ?
  - quelle est la valeur de retour ?
- pour utiliser correctement les expressions régulières en python précédez les (**toujours**) de **r** : `re.findall(r'\b\w+\b', text)`

le flag `re.MULTILINE`

- `findall()` renvoie **un tableau** contenant les occurrences d'un `pattern` dans une `string`
- pour l'utiliser avec les opérateurs `^`et `$`, il faut ajouter le flag `re.MULTILINE`

```
text = '''phrase 1
phrase 2 : c'est la phrase du milieu
phrase 3'''

print(re.findall(r'phrase', text)) # toutes les occurrences de phrase
print(re.findall(r'^phrase [0-9]', text)) # une seule occurrence (^ est interprété comme le début du document)
print(re.findall(r'^phrase', text, re.MULTILINE)) # les trois occurrences en début de *ligne*
```

```
['phrase', 'phrase', 'phrase', 'phrase']
['phrase 1']
['phrase', 'phrase', 'phrase']
```

le flag `re.IGNORECASE`

- pour ignorer la casse dans vos recherches, ajoutez le flag `re.IGNORECASE`

```
#le flag re.IGNORECASE

text = "sorbonne Sorbonne SORBONNE SorBonnE"

print(re.findall(r'sorbonne',text, re.IGNORECASE))
```

```
['sorbonne', 'Sorbonne', 'SORBONNE', 'SorBonnE']
```

Fonctions principales

- `compile(pattern, flags=0)` = crée un "objet" expression régulière pouvant être réutilisé
- `search(pattern, string, flags=0)` = renvoie un "Match Object" si le pattern a été trouvé
- `sub(pattern, repl, string, count=0, flags=0)` = remplace un motif par une chaîne de caractères
- `split(pattern, string, maxsplit=0, flags=0)` = remplace un motif par une chaîne de caractères

search(pattern, string, flags=0)

cherche un `pattern` dans une `string` et renvoie

- un "`match object`" correspondant à **la première occurrence** du motif
- `None` si le motif est absent de la chaîne

```
print(re.search(r'climat\w*', "le climat climatique"))
```

```
<_sre.SRE_Match object; span=(3, 9), match='climat'>
```

```
print(re.search(r'enjeu', "le climat climatique"))
```

```
None
```

Match Object

Les `Match Objects` valent toujours True.

On peut donc tester facilement si un motif a été trouvé :

```
liste = '''pommes fraises framboises bananes kiwis mangues tomate oignon
pommes framboises bananes tomate oignon pommes framboises bananes tomate
oignon pommes framboises bananes poireaux bananes échalottes
tomate framboises kiwi bananes '''
# la liste contient-elle des aubergines ?
match = re.search(r'aubergines', liste)
if match:
    print("oui !")
else:
    print("non !")
```

```
non !
```

sub(pattern, repl, string, count=0, flags=0)

Renvoie la chaîne obtenue en **remplaçant** les occurrences (sans chevauchement) les **plus à gauche** de `pattern` dans `string` par le remplacement `repl`. Si le motif n'est pas trouvé, string est renvoyée inchangée.

```
re.sub(r'normal|normale', r'bizarre',
       "Je suis un président normal, j'ai mené une présidence normale")
```

```
"Je suis un président bizarre, j'ai mené une présidence bizarree"
```

```
re.sub(r'normale|normal', r'bizarre',
       "Je suis un président normal, j'ai mené une présidence normale")
```

```
"Je suis un président bizarre, j'ai mené une présidence bizarre"
```

split(pattern, string, maxsplit=0, flags=0)

Sépare la chaîne selon les apparitions du motif (qui agit comme un **séparateur**)

```
chaine = "fruit1 ; fruit2 ; fruit3 ; legume1 ; legume2 ; legume3"
sans_separateur = re.split(r'\W+', chaine)
print(sans_separateur)

avec_separateur = re.split(r'(\W+)', chaine)
print(avec_separateur)

avec_max = re.split(r'\W+', chaine, 3)
print(avec_max)
```

```
['fruit1', 'fruit2', 'fruit3', 'legume1', 'legume2', 'legume3']
['fruit1', ' ; ', 'fruit2', ' ; ', 'fruit3', ' ; ', 'legume1', ' ; ', 'legume2', ' ; ', 'legume3']
['fruit1', 'fruit2', 'fruit3', 'legume1 ; legume2 ; legume3']
```

Plan

Partie Cours

Agent conversationnel simple

Connaissez-vous ELIZA ?

ELIZA = “un programme informatique écrit par Joseph Weizenbaum entre 1964 et 1966, qui **simule un psychothérapeute** rogérien en reformulant la plupart des affirmations du « patient » en questions, et en les lui posant. ”

- identification mot clés et utilisation réponse à trous
- replacement des pronoms (je -> vous)
- phrases au hasard (ex : “quel temps fait-il chez vous” ?)

```
Système : "Comment allez vous ?"
Utilisateur : "Je suis déprimé."
S : "Dites-moi pourquoi vous êtes déprimé."
U : "Je passe trop de temps avec les ordinateurs."
S : "Pourquoi mentionnez-vous les ordinateurs ?"
U : "Parce que tu en es un."
S : "Quels sont vos rapports avec les ordinateurs ?"
U : "Difficiles."
S : etc
```

sources : https://fr.wikipedia.org/wiki/ELIZA et
http://li.linguist.univ-paris-diderot.fr/PagesCours/Projets10/licence3.html

ELIZA en python ?

(démo en anglais)

- tableau de règles (réplique utilisateur -> réplique ordinateur)
- fonction python qui gère l'interaction

... énoncé bientôt sur le moodle.