Université de Bouira
Département d'Informatique
Module : Traitement Automatique des Langues
Niveau M1-ISIL

Série TP 2

Traitement bas niveau – Encodage et Tokenization.

1 – Encodage :

Depuis sa version 3, Python utilise par défaut Unicode UTF-8 pour l'encodage.

Tout le code Python est en Unicode UTF-8, et idéalement, toutes les données doivent l'être aussi.

Le texte en Python peut être présenté en utilisant le str unicode ou le bytes.

Pratique : exécuter le code suivant sous Python 3

```
# fixer le codage des caractères dans l'entête de script
# -*- coding: utf-8 -*-

# L'encodage - afficher tous les codages qui existes
# Importer les bibliothèques sys et encodings
import sys
import encodings

# get encoding sets
codages = encodings.aliases.aliases.values()

# éliminer les doubles
codages = set(codages)

# trier les éléments de la liste
codages = sorted(codages)

# commande join permet de mettre un séparateur entre les éléments
print("\n".join(codages))

# deux fonctions de Python concernent le codage Unicode des
caractères : chr et ord
print(chr(97))
print(chr(237))
print(chr(8364))
print(ord('A'))
print(ord('é'))
print(ord('€'))

# en Python 3 il existe deux types de chaine de caractère : le type
str (unicode) et le type bytes(octets)
ch = 'Bonjour'
print(type(ch)) # str
bt = b'Bonsoir'
print(type(bt)) # bytes
```

```
# SyntaxError: bytes can only contain ASCII literal characters.
bt_2 = b'مرحبا'

# solution 1 - fonction bytes pour convertir le str en bytes
ch_2 = 'مرحبا'
bt_2 = bytes(ch_2, 'utf-8')
print(type(bt_2))

# solution 2 - encoder str en bytes - les # caractères en bytes sont
affichés comme s'ils sont des caractères # encodés en ASCII
before = "Hello €"
after = before.encode("utf-8")
print(after, type(after))

# reconvertir le bytes vers le bon encodage str utf-8 - pas d'erreur
print(after.decode("utf-8"))

# essayer de décoder bytes avec l'encodage ascii - UnicodeDecodeError
print(after.decode("ascii"))

# exemple 2 - encoder de l'ascii vers bytes en remplaçant les
# caracatères non ascii en caractères ascii aléatoires
before = "This is the euro symbol: €"
after = before.encode("ascii", errors = "replace")
print(after)
# reconvertir/décoder vers le str utf-8
print(after.decode("ascii")) # perte du caractère € dans la chaine
originale, remplacé par le ?, car non représenté dans l'ascii

# afficher des formules chimiques
print("The chemical formula of water is H\u2082O.Water dissociates
into H\u207A and OH\u207B")

# afficher les caractères arabe en unicode dans l'intervalle
for i in range(0x0600, 0x06ff):
    print(chr(i))

# afficher les caractères tifinagh en unicode dans l'intervalle
for i in range(0x2d30,0x2d6f):
    print(chr(i))

# afficher les caractères arabe en unicode avec leurs noms
import unicodedata
for i in range(0x0627,0x06ff):
    try:
        print(i, unicodedata.name(chr(i)))
    except:
        name ="no name"
```

2 – Tokenization - Segmentation:

Pratique : exécuter le code suivant sous Python 3

```
# 1 - Tester la fonction tokenize() simple - Split text into words
def tokenize(text):
    # diviser la ligne par les espaces
    list_word = text.split(" ")
    return list_word

text = "I'm Very Hungry, I want to eat something."
tokens = tokenize(text);
print(tokens)


# 2 - Plusieurs implémentation de la fonction tokenize() en
utilisant le module re (regular expression)

import re

# tokenize par des exp-reg simples
def tokenize_regex_punct(text):
    tokens = re.split("[.,:; ]+", text)
    return tokens

# tokenize par des exp-reg simples, en gardant la ponctuation
def tokenize_regex_punct_keep(text):
    tokens = re.split("([.,:; ]+)", text)
    return tokens

# tokenize par des expression régulière
def tokenize_regex(text):
    tokens = re.split("\W+", text)
    return tokens

# tokenize par des expression régulière, en gardant la ponctuation
def tokenize_regex_keep_punct(text):
    tokens = re.split("(\W+)", text)
    return tokens

text = "I'm Very Hungry, I want to eat something. United Kingdom."

tokens = tokenize_regex_punct(text);
print(tokens)
tokens = tokenize_regex_punct_keep(text);
print(tokens)
tokens = tokenize_regex(text);
print(tokens)
tokens = tokenize_regex_keep_punct(text);
print(tokens)
```

[…] : Une classe de caractères : tous les caractères énumérés dans la classe, avec possibilité de plages dont les bornes sont séparées par « -«. Exemple : [abc] équivaut à : a, b ou c

\w : Représente tout caractère de « mot » (caractère alphanumérique + tiret bas). Présence alphanumérique. Équivalent à [a-zA-Z0-9_]

\W : Représente tout caractère qui n'est pas un caractère de « mot ». Pas de présence alphanumérique. Equivalent à [^a-zA- Z0-9_]

^ : Indique un commencement de segment mais signifie aussi "contraire de"

+ : 1 ou plusieurs occurrences

(…) : Un groupe de capture : utilisée pour limiter la portée d'un masque de recherche ou d'une alternative