TP : fonctions simples avec tuples, NLTK et spaCy

Objectif du TP. Dans ce TP, vous allez écrire de petites fonctions Python pour manipuler :

- des **tuples**,
- des sorties simples de **NLTK**,
- des sorties simples de **spaCy**.

Le but est de pratiquer les listes, les boucles, les dictionnaires, les fonctions et les tuples.

1. Installation

Avant de commencer, installer les bibliothèques nécessaires :

```
pip install nltk spacy
python -m spacy download en_core_web_sm
```

Pour NLTK, lancer une fois dans Python :

```
import nltk
nltk.download("punkt")
nltk.download("averaged_perceptron_tagger")
nltk.download("maxent_ne_chunker")
nltk.download("words")
```

Dans ce TP, on utilisera surtout des fonctions simples. On ne demande pas de programmation avancée.

2. Rappel : tuples

Un tuple permet de regrouper plusieurs valeurs dans un ordre fixe.

```
entite = ("Paris", "LOC")
span = (10, 15, "Paris", "LOC")
```

Exercice 1 — Lire un tuple

On donne :

```
span = (10, 15, "Paris", "LOC")
```

Écrire les fonctions suivantes :

1) `texte_entite(span)` qui renvoie `"Paris"`
2) `type_entite(span)` qui renvoie `"LOC"`
3) `resume_entite(span)` qui renvoie `"Paris/LOC"`

Exercice 2 — Décomposer un tuple

Écrire une fonction `infos_span(span)` qui renvoie la phrase :

L'entité Paris commence à 10 et finit à 15.

On peut décomposer un tuple ainsi :

```
debut, fin, texte, typ = span
```

3. Premières fonctions avec NLTK

On utilisera ce texte :

```
texte = "Anna studies in Inalco and lives in France."
```

Exercice 3 — Tokenisation avec NLTK

Importer `word_tokenize` depuis NLTK, puis écrire une fonction `tokens_nltk(texte)` qui renvoie la liste des tokens.

Résultat attendu :

```
['Anna', 'studies', 'in', 'Inalco', 'and', 'lives', 'in', 'France', '.']
```

Exercice 4 — Compter les tokens

Écrire une fonction `nb_tokens_nltk(texte)` qui :

— tokenise le texte avec NLTK,

— renvoie le nombre de tokens.

Exercice 5 — POS-tagging

Importer `pos_tag` depuis NLTK, puis écrire une fonction `tags_nltk(texte)` qui renvoie une liste de tuples :

```
[('Anna', 'NNP'), ('studies', 'VBZ'), ...]
```

Exercice 6 — Garder seulement les noms propres

Écrire une fonction `noms_propres_nltk(texte)` qui :

— fait le `pos_tag`,

— garde seulement les mots dont le tag est `"NNP"`,

— renvoie une liste de mots.

4. Fonctions simples de NER avec NLTK

Exercice 7 — Extraire des entités

Écrire une fonction `entites_nltk(texte)` qui :

1) tokenise le texte,

2) fait le POS-tagging,

3) applique `ne_chunk`,
4) renvoie une liste de tuples `(texte, type)`.

Exemple de résultat possible :

```
[('Anna', 'PERSON'), ('Inalco', 'ORGANIZATION'), ('France', 'GPE')]
```

Pour extraire les entités, il faut parcourir la sortie de `ne_chunk` et vérifier si un élément possède un `label()`.

Exercice 8 — Filtrer par type

Écrire une fonction `garder_type(ents, typ)` qui :

— prend une liste de tuples `(texte, type)`,
— renvoie seulement les entités du type demandé.

Exemple :

```
ents = [('Anna', 'PERSON'), ('France', 'GPE'), ('Inalco', 'ORGANIZATION')]
print(garder_type(ents, "GPE"))
```

Résultat attendu :

```
[('France', 'GPE')]
```

5. Premières fonctions avec spaCy

On utilisera le même texte :

```
texte = "Anna studies in Inalco and lives in France."
```

Exercice 9 — Charger spaCy

Écrire les lignes suivantes au début de votre fichier :

```
import spacy
nlp = spacy.load("en_core_web_sm")
```

Puis écrire une fonction `doc_spacy(texte)` qui renvoie :

```
doc = nlp(texte)
```

Exercice 10 — Récupérer les tokens avec spaCy

Écrire une fonction `tokens_spacy(texte)` qui renvoie une liste de chaînes.
Exemple :

```
['Anna', 'studies', 'in', 'Inalco', 'and', 'lives', 'in', 'France', '.']
```

Exercice 11 — POS avec spaCy

Écrire une fonction `tags_spacy(texte)` qui renvoie une liste de tuples :

```
[('Anna', 'PROPN'), ('studies', 'VERB'), ...]
```

Exercice 12 — Entités avec spaCy

Écrire une fonction `entites_spacy(texte)` qui renvoie une liste de tuples :

```
[('Anna', 'PERSON'), ('Inalco', 'ORG'), ('France', 'GPE')]
```

Exercice 13 — Ajouter les positions

Écrire une fonction `spans_spacy(texte)` qui renvoie une liste de tuples :

```
[(0, 4, 'Anna', 'PERSON'), ...]
```

Chaque tuple doit contenir :

— la position de début,

— la position de fin,

— le texte de l'entité,

— le type.

6. Comparer NLTK et spaCy

Exercice 14 — Deux sorties différentes

Écrire une fonction `compare_ner(texte)` qui :

— affiche le résultat de `entites_nltk(texte)`,

— affiche le résultat de `entites_spacy(texte)`.

> Les deux bibliothèques peuvent donner des sorties légèrement différentes. Ce n'est pas forcément une erreur.

Exercice 15 — Compter les types d'entités

Écrire une fonction `compter_types(liste_tuples)` qui prend une liste de tuples et renvoie un dictionnaire :

```
{'PERSON': 1, 'ORG': 1, 'GPE': 1}
```

7. Petit exercice de synthèse

Exercice 16 — Pipeline simple

Écrire un petit programme qui :

1) choisit un texte,
2) récupère les entités avec spaCy,
3) affiche la liste des tuples,
4) affiche le dictionnaire des types,
5) affiche seulement les lieux.

8. Bonus

Bonus 1 — Transformer une liste d'entités en phrases

À partir de :

```
[('Anna', 'PERSON'), ('France', 'GPE')]
```

écrire une fonction qui affiche :

```
Anna est de type PERSON
France est de type GPE
```

Bonus 2 — Liste de tuples vers dictionnaire

Écrire une fonction qui transforme :

```
[('Anna', 'PERSON'), ('France', 'GPE')]
```

en :

```
{'Anna': 'PERSON', 'France': 'GPE'}
```

9. Conseils

— Tester chaque fonction séparément.
— Commencer par les tuples, puis NLTK, puis spaCy.
— Réutiliser les fonctions déjà écrites.
— Afficher souvent les résultats avec `print()`.

Le plus important dans ce TP n'est pas d'écrire beaucoup de code, mais d'écrire des fonctions courtes, simples et correctes.