Cours : tuples, NLTK, spaCy et NER

Python : tuples et introduction à la NER

présenté par

Revekka Kyriakoglou

le

15 février 2026

Plan

- Rappel : tuples
- Qu'est-ce que la NER ?
- Installation des outils
- NLTK : étapes du NLP
- spaCy : pipeline intégré
- Comparaison NLTK / spaCy
- Représenter les résultats avec des tuples et des dictionnaires
- Exercices

Pourquoi des tuples ?

Idée

Un tuple est un objet Python qui permet de regrouper plusieurs valeurs dans un ordre fixé.

```
personne = ("Alice", 20)
coord = (48.85, 2.35)
```

Un tuple ressemble à une liste, mais on l'utilise souvent pour des données qui vont ensemble et qu'on ne veut pas modifier.

Tuple ou liste ?

- Liste : `[10, 20, 30]`
- Tuple : `(10, 20, 30)`

```
L = [10, 20, 30]
T = (10, 20, 30)

L[0] = 99       # possible
# T[0] = 99     # erreur
```

Une liste est modifiable. Un tuple ne l'est pas.

Accéder aux éléments d'un tuple

```python
point = (4, 7)

print(point[0])   # 4
print(point[1])   # 7
print(len(point)) # 2
```

Comme pour les listes, les indices commencent à 0.

Décomposer un tuple

```
point = (4, 7)
x, y = point

print(x)
print(y)
```

C'est pratique quand un tuple représente plusieurs informations liées.

Pourquoi des tuples en NER ?

Idée

En NER, on veut souvent stocker ensemble : le texte, le type, et parfois les positions.

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

Les tuples sont simples, ordonnés, et faciles à manipuler.

Exercice 0 : manipuler des tuples

On vous donne :

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

Questions :

1. Afficher uniquement le texte de l'entité.
2. Afficher `"Paris/LOC"`.
3. Décomposer le tuple en variables `debut, fin, txt, typ`.

Exercice 0 : manipuler des tuples

On vous donne :

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

Questions :

1. Afficher uniquement le texte de l'entité.
2. Afficher `"Paris/LOC"`.
3. Décomposer le tuple en variables `debut, fin, txt, typ`.

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

print(span[2])
print(span[2] + "/" + span[3])

debut, fin, txt, typ = span
print(debut, fin, txt, typ)
```

Qu'est-ce que la NER ?

Définition

La **Named Entity Recognition (NER)** consiste à repérer dans un texte des entités nommées et à leur attribuer un type.

- Personne
- Lieu
- Organisation
- Date

Exemple : dans `"Anna studies in Inalco in France."`, on peut reconnaître `Anna`, `Inalco` et `France`.

Pourquoi c'est utile ?

- indexer des documents
- extraire des informations
- préparer d'autres tâches de NLP
- résumer ou annoter un texte

La NER = détection + catégorisation.

Installation : bibliothèques Python

À installer

Pour ce cours, il faut Python, NLTK et spaCy.

```
pip install nltk spacy
```

Si `pip` ne fonctionne pas, essayer : `python -m pip install nltk spacy`

Installation : ressources NLTK

```
import nltk

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

Ces ressources servent à la tokenisation, au POS-tagging et à la reconnaissance d'entités avec NLTK. On les télécharge une seule fois.

Installation : modèle spaCy

```
python -m spacy download en_core_web_sm
```

Ce modèle anglais est utilisé dans les exemples. Ensuite, dans Python : `nlp = spacy.load("en_core_web_sm")`

Exemple de texte

```
texte = "Anna is born in France and studies in Inalco."
```

Nous allons appliquer plusieurs étapes du NLP sur ce même texte.

NLTK : 1. tokenisation

```
from nltk import word_tokenize

tokens = word_tokenize(texte)
print(tokens)
```

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

La tokenisation découpe le texte en mots.

NLTK : 2. POS-tagging

```python
from nltk import pos_tag

tags = pos_tag(tokens)
print(tags)
```

```
[('Anna', 'NNP'), ('is', 'VBZ'), ('born', 'VBN'),
 ('in', 'IN'), ('France', 'NNP'), ('and', 'CC'),
 ('studies', 'VBZ'), ('in', 'IN'), ('Inalco', 'NNP'),
 ('.', '.')]
```

Chaque mot reçoit une catégorie grammaticale.

NLTK : 3. NER

```
from nltk import ne_chunk

entities = ne_chunk(tags)
print(entities)
```

NLTK permet ensuite de détecter des entités nommées à partir des mots et des tags.

NLTK : extraction d'un format simple

```
res = []
for node in entities:
    if hasattr(node, "label"):
        txt = " ".join(w for w, _ in node.leaves())
        res.append((txt, node.label()))

print(res)
```

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

On transforme la sortie de NLTK en liste de tuples.

spaCy : chargement du modèle

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp(texte)
```

spaCy applique automatiquement plusieurs analyses dans un seul pipeline.

spaCy : tokens et catégories

```
for token in doc:
    print(token.text, token.pos_)
```

Les tokens et les catégories grammaticales sont directement accessibles.

spaCy : NER

```python
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Anna PERSON
France GPE
Inalco ORG
```

Avec spaCy, les entités sont déjà prêtes à être utilisées.

spaCy : format exploitable

```
res = [(ent.text, ent.label_) for ent in doc.ents]
print(res)
```

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

On récupère très facilement une liste de tuples.

NLTK vs spaCy

NLTK	**spaCy**
Étapes séparées	Pipeline intégré
Plus de code	Moins de code
Très pédagogique	Très pratique
Extraction manuelle	Résultat direct

À retenir

Conclusion

NLTK

- permet de comprendre les étapes du NLP
- fonctionne bien pour apprendre
- demande plusieurs fonctions

spaCy

- propose un pipeline complet
- donne des résultats directement exploitables
- est souvent plus pratique pour des scripts réels

Compter les entités par type

```python
def compter_par_type(liste_tuples):
    d = {}
    for t in liste_tuples:
        typ = t[-1]
        if typ in d:
            d[typ] += 1
        else:
            d[typ] = 1
    return d
```

Ici, on transforme une liste de tuples en dictionnaire de comptage.

Exercice 1

À partir de :

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

1. Afficher seulement les types.
2. Compter combien il y a d'entités de chaque type.
3. Ajouter une nouvelle entité dans une nouvelle liste.

Résumé

- Les tuples servent à stocker plusieurs informations ensemble.
- La NER détecte des entités comme des personnes, lieux et organisations.
- NLTK montre les étapes du NLP.
- spaCy fournit un pipeline plus direct.
- Les résultats peuvent être représentés avec des tuples et des dictionnaires.