List

Points forts on peut ajouter des élements ou l'étendre garde l'ordre des éléments (si on se soucie de l'ordre)

Points faibles moins rapide : Vérifier l'appartenance d'une valeur dans une liste, il prend du temps proportionnel à la longueur de la liste dans la moyenne. les élements peuvent être toujours modifiés (moins de sécurité) Les listes ne peuvent jamais être utilisées comme des clés de dictionnaire, car les listes ne sont pas immuables

Use case : Pour répondre à la question "Quel est le numéro de téléphone de John Doe?" Avec les listes, nous stockerions les numéros de téléphone et les noms de façon séquentielle et parcourons la liste entière pour trouver le numéro de téléphone que nous avons demandé comme indiqué dans cet exemple

In [7]:

```
def find_phonenumber(phonebook, name):
    for n, p in phonebook:
        if n == name:
            return p
    return None

phonebook = [
    ("John Doe", "555-555-5555"),
    ("Albert Einstein", "212-555-5555"),

]

print ("John Doe's phone number is", find_phonenumber(phonebook, "John Doe"))
```

```
John Doe's phone number is 555-555-5555
```

Tuple

Points forts Les tuples sont plus rapides que les listes. Il rend notre code plus sûr si nous protégeons en écriture les données qui n'ont pas besoin d'être modifiées Certains tuples peuvent être utilisés comme des clés de dictionnaire (spécifiquement, des tuples qui contiennent des valeurs immuables comme des chaînes, des nombres et d'autres tuples).

Points faibles Immuable : On ne peut pas ajouter d'éléments à un tuple. Les tuples n'ont aucune méthode d'append ou extend.(ça depend) On ne peut pas supprimer des éléments d'un tuple. Les tuples n'ont pas de méthode de suppression ou de pop.

Use case Les tuples sont généralement utilisés là où l'ordre et la position sont significatifs et cohérents. Par exemple, en créant une structure de données pour choisir votre propre jeu d'aventure, j'ai choisi d'utiliser des tuples au lieu des listes car la position dans le tuple était significative. Voici un exemple de cette structure de données:

Source : http://stackoverflow.com/questions/1708510/python-list-vs-tuple-when-to-use-each (http://stackoverflow.com/questions/1708510/python-list-vs-tuple-when-to-use-each)

In [ ]:

```
pages = {'foyer': {'text' : "some text",
          'choices' : [('open the door', 'rainbow'),
                     ('go left into the kitchen', 'bottomless pit'),
                     ('stay put','foyer2')]},}
```

Dictionnaire
Points forts les élements sont référenciés par des clés la recherche est très rapide (utilsé pour intérroger des informations) accepte un ensemble vide Vous pouvez accéder à la liste des clés ou des valeurs indépendamment.

Points faibles utilise beaucoup de memoire

Use case Avec un dictionnaire, nous pouvons simplement avoir «index» les noms et les «valeurs» les numéros de téléphone comme indiqué dans l'exemple 4-2. Cela nous permet de simplement rechercher la valeur dont nous avons besoin et obtenir une référence directe à elle au lieu d'avoir à lire toutes les valeurs dans notre jeu de données.

Source : https://www.safaribooksonline.com/library/view/high-performance-python/9781449361747/ch04.html (https://www.safaribooksonline.com/library/view/high-performance-python/9781449361747/ch04.html)

In [6]:

```
phonebook = {
    "John Doe": "555-555-5555",
    "Albert Einstein" : "212-555-5555",
}
print ("John Doe's phone number is", phonebook["John Doe"])
```

```
John Doe's phone number is 555-555-5555
```

set

Points forts Set requiert que les éléments soient hashables rapide

Points faibles n'accepte pas les doublons

Use case : Rechercher si des élements appatiennent à un ensemble

Source : https://www.dotnetperls.com/set-python (https://www.dotnetperls.com/set-python)

In [8]:

```
# Create a set.
items = {"arrow", "spear", "arrow", "arrow", "rock"}

# Print set.
print(items)
print(len(items))

# Use in-keyword.
if "rock" in items:
    print("Rock exists")

# Use not-in keywords.
if "clock" not in items:
    print("Cloak not found")
```

```
{'spear', 'arrow', 'rock'}
3
Rock exists
Cloak not found
```