Tuples et algorithmes de parcours

Largement inspiré du fabuleux site Pixees.fr

Définition : Séquences

En informatique, il est possible de « stocker » plusieurs grandeurs dans une même structure, appelée une **séquence**. De façon plus précise, une séquence est un ensemble **fini et ordonné** d'éléments, repérés par un **indice**.

Dans de très nombreux langages informatiques, mais pas dans tous, les indices **démarrent à zéro**.

Exemples

- La liste des mois de l'année est une séquence, chaque mois étant repéré par son indice (le numéro du mois), qui commence à 1.
- Les chaînes de cracatères en Python sont aussi des séquences, chaque caractère étant repéré par son *indice* commençant à 0 .

Nous étudierons deux types de séquences particulières en Python : les tuples et les tableaux (listes).

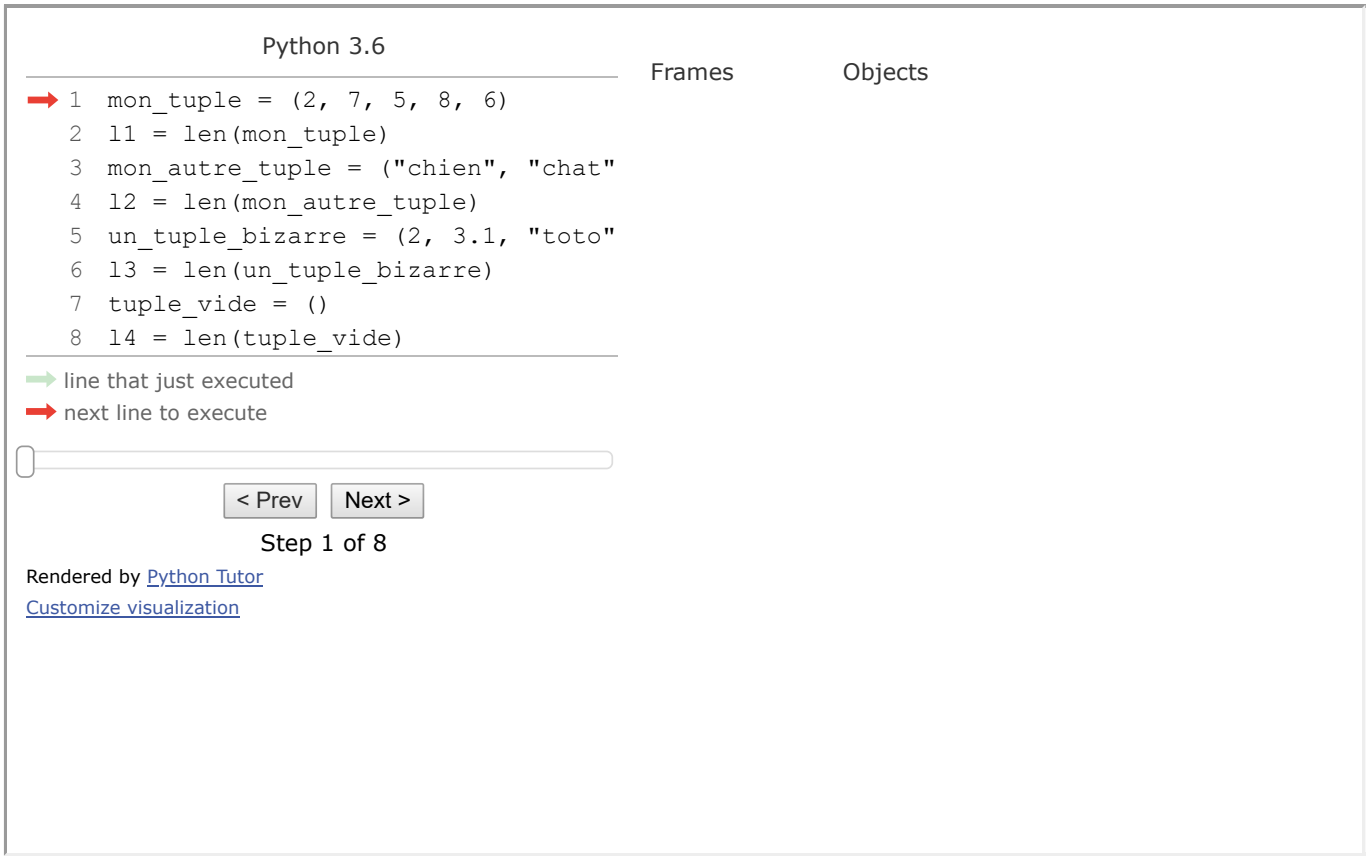
1. Premier types de séquences : les tuples

Tuples

En Python, un **tuple** est une séquence, qui est définie **entre parenthèses**, et dont les éléments sont séparés par des **virgules**, et dont les **indices commencent à** 0. Les éléments peuvent être de même nature (`int`, `float`, `str`, `bool` et même `tuple`), ou bien de natures variées.

Un tuple possède une **longueur**, qui est le nombre d'éléments le composant. Elle est obtenue grâce à la fonction *built-in* `len()` .

Par exemple, on peut voir la création d'objets de type `tuple` dans Python-tutor :



Dans la variable `mon_autre_tuple` :

- `"chien"` est l'élément d'indice 0 ;
- `"chat"` est l'élément d'indice 1 ;
- `"poisson rouge"` est l'élément d'indice 2.

Le dernier indice est donc 2, mais la **longueur** de `mon_autre_tuple` est bien 3 !

Différencier indices et longueur

Dans une séquence en Python, la longueur est égale au dernier indice plus 1 !

Le dernier indice d'un tuple `t` est donc `len(t)-1` !

Accéder aux éléments d'un tuple

Pour accéder aux élements d'un tuple, on utilise la même notation que pour accéder aux caratères d'une chaines de caractères : la notation **entre crochets**.

Le code

```
>>> mon_tuple = (2, 7, 5, 8, 6, 9, 4, 3, 1, 12)
>>> mon_tuple[2]
5
```

Visualisation dans Python-Tutor

Python 3.6

```
1 mon_tuple = (2, 7, 5, 8, 6, 9, 4,
2 a = mon_tuple[2]
```

line that just executed
next line to execute

< Prev Next >

Click to Start Visualization

Rendered by Python Tutor
Customize visualization

Frames Objects

Global frame

Exercice 1

Enoncé

1. Sachant que `un_tuple_bizarre = (2, 3.1, "toto", True)`, quelle est la valeur renvoyée par `un_tuple_bizarre[1]` ?
2. A quelle valeur est associée le nom `a` après exécution du code suivant ?

```
mon_tuple = (2, 7, 5, 8, 6, 9, 4, 3, 1, 12)
a = mon_tuple[6]
```

3. Dans le code précédent, que faut-il mettre entre les crochets pour que le nom `a` soit associé à la valeur 1 ?
4. Essayez et commentez le code suivant :

```
mon_tuple = (2, 7, 5, 8, 6, 9, 4, 3, 1, 12)
a = mon_tuple[-1]
```

5. Essayez et commentez le code suivant :

```
mon_tuple = (2, 7, 5, 8, 6, 9, 4, 3, 1, 12)
a = mon_tuple[10]
```

6. Essayez et commentez le code suivant :

```
mon_tuple = (2, 7, 5, 8, 6, 9, 4, 3, 1, 12)
mon_tuple[5] = 42
```

Solutions

A venir !

Exercice 2

Enoncé

1. Essayez et commentez le code suivant :

```
mon_tuple = ("le", "bonjour", "monde")
print(f"{mon_tuple[1].capitalize()} {mon_tuple[0]} {mon_tuple[2]} !")
```

2. Essayez et commentez le code suivant :

```
def add(a, b) :
    return (a, b, a+b)

mon_tuple = add(5,8)
print(f"{mon_tuple[0]} + {mon_tuple[1]} = {mon_tuple[2]}")
```

3. Essayez et commentez le code suivant :

```
mon_tuple = ("Luke Skywalker", "Mark Hamill", "Jedi", "Dark Vador")
personnage, acteur, metier, pere = mon_tuple
```

Solutions

A venir !

Tuple unpacking

La méthode utilisée dans l'exercice précédent s'appelle le **tuple unpacking**, soit « désempaquetage » de tuple. Elle est très souvent utilisée pour décomposer des tuples renvoyés comme valeur de retour d'une fonction.

Exercice 3

Enoncé

Essayez et commentez le code suivant :

```
def euclide(a, b) :
    return (a,b,a//b,a%b)
res = euclide(20,7)
print(type(res))
diviseur, dividende, quotient, reste = res
print(type(quotient))
```

Solution

A venir !

2. Parcourir une séquence

Parcours de séquence

Le parcours d'une séquence peut être fait de deux manières différentes :

- par les indices ;
- par les éléments.

En python ces deux types de parcours sont effectués par l'intermédiaire d'une boucle `for`.

Parcours par les indices

```
mon_tuple = (1, 3, 5, 7)
for i in range(0,len(mon_tuple)) :
    print(mon_tuple[i])
```

- Pour rappel, la fonction `range(a,b)` itère sur les entiers naturels de a inclus à b exclu. Ici, la parcours se fait donc pour i allant de 0 à `len(mon\tuple)`, soit 4.
- La valeur de départ de la fonction `range` étant 0, on aurait pu l'omettre.

Parcours par les éléments

Le parcours par indice est possible en Python, et parfois nécessaire. Mais il existe une possibilité de parcours de la séquence plus directe :

```
mon_tuple = (1, 3, 5, 7)
for element in mon_tuple :
    print(element)
```

A chaque tour de boucle, le nom `element` va être associé à une valeur du tuple.

- le nom `element` n'est qu'un choix de ma part, j'aurais tout aussi bien pu écrire `toto`.
- Ce type de boucle existe aussi dans d'autres langages, et porte souvent le nom de boucle `foreach`.
- Un inconvénient est que vous n'avez que l'élément, et que par conséquent il vous manque son indice. Heureusement, une fonction Python (`enumerate`), peut permettre de combiner les deux types de boucles :

  ```
  mon_tuple = (1, 3, 5, 7)
  for indice, element in enumerate(mon_tuple) :
      print(f"{indice}->{element}")
  ```

Exercice 4

Enoncé

1. Essayez et commentez le code suivant :

```
mon_tuple = (12, 15, 34, 23, 11, 15, 36)
for n in mon_tuple :
    if n%2 == 0 :
        print(n)
```

2. Essayez et commentez le code suivant :

```
mon_tuple = (12, 15, 34, 23, 11, 15, 36)
for i in range(len(mon_tuple)) :
    if mon_tuple[i]%2 == 0 :
        print(mon_tuple[i])
```

3. Essayez et commentez le code suivant :

```
mon_tuple = (12, 15, 34, 23, 11, 15, 36)
for i in range(len(mon_tuple)) :
    if i%2 == 0 :
        print(mon_tuple[i])
```

4. Que faut-il écrire pour obtenir les termes impairs du tuple ?
5. Que faut-il écrire pour obtenir les termes de rang impairs du tuple ?

Solutions

A venir !

Algorithmes de parcours

Enoncé

Pour chacune des questions ci-dessous, je vous demande :

- de vous mettre par 2;
- de chercher d'abord à la main, sur papier ;
- de décrire l'algorithme demandé en langage naturel;
- enfin d'en proposer une version Python.

Pour chacune des questions suivants, on suppose que les **tuples donnés sont des tuples de nombres, entiers ou flottants, et que ces tuples sont non vides**.

1. Trouver un algorithme puis écrire une fonction Python `maximum(t)` qui prend un tuple en entrée et renvoie le plus grand nombre de ce tuple, *sans utiliser la fonction built-in* `max`. 2.Trouver un algorithme puis écrire une fonction Python `minimum(t)` qui prend un tuple en entrée et renvoie le plus petit nombre de ce tuple, *sans utiliser la fonction built-in* `min`.
2. Trouver un algorithme puis écrire une fonction Python `somme(t)` qui prend un tuple en entrée et renvoie la somme des valeurs de ce tuple, *sans utiliser la fonction built-in* `sum`.
3. Trouver un algorithme puis écrire une fonction Python `moyenne(t)` qui prend un tuple en entrée et renvoie la moyenne des valeurs de ce tuple, *sans utiliser la fonction built-in* `sum`.
4. Trouver un algorithme puis écrire une fonction Python `palindrome(t)` qui prend un tuple en entrée et renvoie `True` si le tuple est un palindrome, et `False` sinon.

   Palindrome

   Un palindrome est une séquence qui peut se lire dans les deux sens sans changer ses valeurs :
   - $(6, 4, 3, 4, 6)$ est un palindrome;
   - $(2, 4, 4, 2)$ est un palindrome;
   - (12) est un palindrome ;
   - $(3, 4, 5, 3)$ n'est pas un palindrome.

   Si vous le codez suffisamment bien, votre code devrait aussi fonctionner pour les chaines de caractères comme : « été », « kayak », « Noël a trop par rapport à Léon » ou « Engage le jeu que je le gagne ».

Une citation

« Les tentatives de création de machines pensantes nous seront d'une grande aide pour découvrir comment nous pensons nous-mêmes. »

De Alan Turing, Conférence à la BBC - 15 Mai 1951