Chaînes de caractères

Une chaine de caractères est une sequence de lettres. Elle est délimité par des guillemets simples ou doubles.

Un index dans une chaîne

Chaque élement peut être accédé par un **indice** entre crochets.

In [128]:
```
fruit = 'banane'
fruit[0]
```

Out[128]: 'b'

L'indexation commence avec zéro. L'indexe 1 pointe vers la deuxième lettre.

In [6]:
```
fruit[1]
```

Out[6]: 'a'

On peut utiliser également une variable comme indice de chaîne.

In [7]:
```
i = 2
fruit[i]
```

Out[7]: 'n'

Voici comment obtenir la lettre après la i-ième lettre.

In [8]:
```
fruit[i+1]
```

Out[8]: 'a'

Longuer de chaîne - len

La fonction `len` retourne la longuer de chaîne.

In [10]:
```
len(fruit)
```

Out[10]: 6

Pour obtenir la dernière lettre de la chaine, il faut utiliser l'index `n-1` .

In [14]:
```
n = len(fruit)
fruit[n-1]
```

Out[14]:
```
'e'
```

L'index `-1` pointe vers la dernière lettre.

In [12]:
```
fruit[-1]
```

Out[12]:
```
'e'
```

Parcour avec une boucle for

On peut utiliser une boucle `while` pour accéder à chaque lettre, en utilisant un indice.

In [17]:
```
i = 0
while i < len(fruit):
    print(i, fruit[i])
    i += 1
```

```
0 b
1 a
2 n
3 a
4 n
5 e
```

Une autre façon plus courte est d'utiliser la boucle `for` . A chaque itération un autre caractère de la chaîne est affecté à `c` .

In [15]:
```
for c in fruit:
    print(c)
```

```
b
a
n
a
n
e
```

Voici un exemple ou chaque lettre de la chaîne **prefixes** est combiné avec la chaîne **suffix** pour former un nouveau mot.

In [129]:
```
prefixes = 'BLMPRST'
suffix = 'ack'

for c in prefixes:
    print(c + suffix)
```

```
Back
Lack
Mack
Pack
Rack
Sack
Tack
```

Tranches de chaînes

Un sous-ensemble de caractères s'appele une **tranche**. La séléction se fait en utilisant deux indices avec la notation `:` .

In [23]:
```
s = 'Monty Python'
s[0:5]
```

Out[23]: `'Monty'`

In [25]:
```
s[6:12]
```

Out[25]: `'Python'`

Si le premier ou dernier indice est omis, la chaine va jusqu'au début ou la fin.

In [26]:
```
s[:3]
```

Out[26]: `'Mon'`

In [27]:
```
s[3:]
```

Out[27]: `'ty Python'`

Si les deux indices sont les identiques, la sous-chaîne est vide.

In [28]:
```
s[3:3]
```

Out[28]: `''`

Les chaînes sont immuables. On ne peux pas réaffecter une tranche.

In [30]:
```
s[3] = 'k'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-30-5780f0e03450> in <module>()
----> 1 s[3] = 'k'

TypeError: 'str' object does not support item assignment
```

Si on veut modifier une lettre, il faut composer une nouvelle chaîne.

In [32]:
```
s[:2]+'k'+s[3:]
```

Out[32]:
```
'Mokty Python'
```

Rechercher une lettre

Que fait la fonction `trouver` ?

In [35]:
```
def trouver(mot, lettre):
    i = 0
    while i < len(mot):
        if mot[i] == lettre:
            return i
        i += 1
    return -1
```

In [36]:
```
trouver('Monty', 'y')
```

Out[36]:
```
4
```

In [37]:
```
trouver('Monty', 'x')
```

Out[37]:
```
-1
```

Si la lettre existe dans le mot, son index est retourné. Si la lettre n'existe pas, la valeur -1 est retournée.

Compter des lettres

La fonction suivante compte l'occurence d'une lettre dans un mot.

In [43]:
```
def count(mot, lettre):
    cnt = 0
    for c in mot:
        if c == lettre:
            cnt += 1
    return cnt

count('banana', 'a')
```

Out[43]: 3

In [41]:
```
count('banana', 'c')
```

Out[41]: 0

Méthodes de chaines de caractères

Le type `string` possède beaucoup de méthodes. Les méthodes sont ajoutées aux objets chaîne avec un point. La méthdode `upper` transformer en majuscules, la méthode `lower` transforme en minuscules.

In [44]:
```
s.upper()
```

Out[44]: 'MONTY PYTHON'

In [45]:
```
s.lower()
```

Out[45]: 'monty python'

La méthode `find(c)` trouve l'index du caractère `c` . La méthode `replace` remplace une chaîne par une autre.

In [46]:
```
s.find('y')
```

Out[46]: 4

In [47]:
```
s.find('th')
```

Out[47]: 8

In [53]:
```
s.replace('n', 'ng')
```

Out[53]: 'Mongty Pythong'

L'opérateur in

L'opérateur bouléen `in` retourne `True` si un caractère ou une séquence fait partie d'une chaîne.

In [57]:
```
'ana' in 'banana'
```

Out[57]:
```
True
```

La fonction `common` imprime les caractères en commun dans les deux chaînes.

In [62]:
```
def common(word1, word2):
    for c in word1:
        if c in word2:
            print(c)
```

In [64]:
```
common('pommes', 'oranges')
```

```
o
e
s
```

Comparaison de chaines

Les 6 opérateurs de comparaison (`>` , `<=` , `==` , `!=` , `>=` , `>`) sont aussi définis pour des chaînes.

In [65]:
```
def check(a, b):
    if a < b:
        print(a, 'is before', b)
    else:
        print(a, 'is after', b)
```

In [69]:
```
check('ananas', 'banana')
```

```
ananas is before banana
```

In [68]:
```
check('orange', 'banana')
```

```
orange is after banana
```

In [70]:
```
check('Orange', 'banana')
```

```
Orange is before banana
```

En Python les lettre majuscues viennent avant les minuscules.

Exercices

ex 1 strip et replace

La fonction `strip` enlève des espace en début et fin d'une chaîne.

In [71]:
```
'  monty    '.strip()
```

Out[71]: 'monty'

In [72]:
```
'monty'.replace('t', 'ke')
```

Out[72]: 'monkey'

ex 2 count

```
La fonction `count` compte le nombre d'occurrences d'un caractère.
```

In [73]:
```
count('banana', 'a')
```

Out[73]: 3

ex 3 slice indexing

Un troisième indexe indique les incréments. `[::2]` indexe chaque deuxième caractère.

In [74]:
```
s = 'Monty Python'
```

In [75]:
```
s[::2]
```

Out[75]: 'MnyPto'

In [76]:
```
s[::-1]
```

Out[76]: 'nohtyP ytnoM'

In [77]:
```
def is_palindrome(s):
    return s == s[::-1]
```

In [130]:
```
is_palindrome('hannah'), is_palindrome('harry')
```

Out[130]: (True, False)

ex 4 trouver des minuscules

Pour chacune des 5 fonctions, décrivez ce qu'elle fait réellement.

In [85]:
```
def minusc1(s):
    for c in s:
        if c.islower():
            return True
        else:
            return False
minusc1('Jim'), minusc1('jim')
```

Out[85]:
```
(False, True)
```

Est-ce que la première lettre est minuscule ?

In [86]:
```
def minusc2(s):
    for c in s:
        if 'c'.islower():
            return 'True'
        else:
            return 'False'
minusc2('Jim'), minusc2('jim')
```

Out[86]:
```
('True', 'True')
```

Retourne toujours la chaîne 'True'

In [89]:
```
def minusc3(s):
    for c in s:
        drapeau = c.islower()
    return drapeau
minusc3('Jim'), minusc3('jiM')
```

Out[89]:
```
(True, False)
```

Est-ce que la dernière lettre est minuscule ?

In [93]:
```
def minusc4(s):
    drapeau = False
    for c in s:
        drapeau = drapeau or c.islower()
    return drapeau
minusc4('Jim'), minusc4('JIM')
```

Out[93]:
```
(True, False)
```

Est-ce qu'il y a au moin une lettre minuscule ?

In [98]:
```
def minusc5(s):
    for c in s:
        if not c.islower():
            return False
    return True
minusc5('jim'), minusc5('Jim')
```

Out[98]:
```
(True, False)
```

Est-ce que toutes les lettres sont minuscule ?

ex 5 ciffre de César

In [126]:
```
def rotate_word(s, n):
    res = ''
    for c in s:
        m = ord(c) + n
        if m > ord('Z'):
            m -= 26
        res += chr(m)
    return res
```

In [127]:
```
rotate_word('JAPPA', 4), rotate_word('HAL', 1), rotate_word('RAVIER',
```

Out[127]:
```
('NETTE', 'IBM', 'ENIVRE', 'YES')
```