Les String en Python

Semaine 7

Ali Assi, PhD

Un caractère est un symbole unique, par exemple une lettre minuscule " `a` ", une lettre majuscule " `B` ", un symbole spécial " `&` ", un symbole représentant un chiffre " `7` ", une espace " " que l'on notera aussi " ".

Une chaîne de caractères est une suite de caractères, comme un mot " `Bonjour` ", une phrase ' `Il fait beau.` ', un mot de passe " `N[w5ms}e!` ". Une chaîne de caractères est un objet séquentiel immuable où chaque caractère représente un élément de la séquence.

In [1]:
```
day = "sunday"
```

Immuable dans le cas d'une chaîne de caractères signifie qu'une fois qu'un objet chaîne de caractères est initialisé en mémoire, nous ne pouvons pas le modifier, mais nous pouvons quand même le manipuler avec de nombreuses méthodes intégrées.

Par **séquence**, nous signifions qu'un objet chaîne de caractères est indexable et itérable comme une liste (nous en parlerons plus tard), ce qui signifie que nous pouvons accéder à un élément de l'objet chaîne de caractères (qui est fondamentalement un caractère) par son index et itérer à travers les valeurs individuelles dans une boucle.

In [2]:
```
# Exemple

##Immuable
day = "sunday"

day[0] = "S"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[2], line 6
      1 # Exemple
      2
      3 ##Immuable
      4 day = "sunday"
----> 6 day[0] = "S"

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

In [3]:
```
# Exemple

day = "sunday"

#we can also take a slice of the string due to its indexing properties
print(day[0:3])

#Print the full string
day2 = day[::]
print(day2)

#Print the string, jump two elements from whole string
print(day[::2]) #Output : sna

# Print the string, jump two elements from start index 0 to stop index 9
# but don't include the element at index at 9
print(day[0:9:2]) #Output : sna

#Print the string using backwards index, jump one element (Reverse the string)
print(day[::-1]) #Output : yadnus
```

```
sun
sunday
sna
sna
yadnus
```

Opérations sur les chaînes

La **concaténation**, c'est-à-dire la mise bout à bout de deux chaînes, s'effectue à l'aide de l'opérateur `+` . Par exemple " `para` "+" `pluie` " donne la chaîne "parapluie".

In [4]:
```
# Exemple

str1 = "para"
str2 = "pluie"

#Concatenate Hello & World
str3 = str1 + str2

print(str3)
```

```
parapluie
```

La chaîne vide `""` est utile lorsque l'on veut initialiser une chaîne avant d'y ajouter d'autres caractères.

La longueur d'une chaîne est le nombre de caractères qu'elle contient. Elle s'obtient par la fonction `len()` . Par exemple `len("Hello World")` renvoie 11 (l'espace compte comme un caractère).

In [5]:
```
len("Hello World")
```

Out[5]:
```
11
```

Si mot est une chaîne alors on peut récupérer chaque caractère par mot[i]. Par exemple:

Lettre	a	v	i	o	n
Rang	0	1	2	3	4

alors

In [6]:
```
mot = "avion"
print(mot[0])
print(mot[1])
print(mot[2])
print(mot[3])
print(mot[4])
```

```
a
v
i
o
n
```

De façon plus générale, si mot est une chaîne, les caractères s'obtiennent par `mot[i]` pour `i` variant de `0` à `len(mot)-1`

On peut extraire plusieurs caractères d'une chaîne à l'aide de la syntaxe `mot[i:j]` qui renvoie une chaîne formée des caractères numéro `i` à `j - 1` (attention le caractère numéro `j` n'est pas inclus !)

Lettre	v	e	n	d	r	e	d	i
Rang	0	1	2	3	4	5	6	7

In [7]:
```
mot = "vendredi"
print(mot[0: 4])#renvoie la sous-chaîne "vend" formée des caractères de rang 0
```

```
vend
```

Opérateur d'appartenance dans les chaînes de caractères

On peut tester si un caractère appartient à une certaine liste de caractères. Par exemple : `if carac in ["a", "A", "b", "B", "c", "C"]:` permet d'exécuter des instructions si le caractère carac est l'une des lettres a, A, b, B, c, C.

Pour éviter certaines lettres, on utiliserait : `if carac not in ["X", "Y", "Z"]:`

In [8]:
```
#Exemple
carac = 'a'
if carac in ["a", "A", "b", "B", "c", "C"]:
    print(f"{carac} est dans la liste")
else:
    print(f"{carac} n'est pas dans la liste")
```

```
a est dans la liste
```

Une boucle `for ... in ...` permet de parcourir une chaîne, caractère par caractère :

In [9]:
```
mot = "vendredi"
for carac in mot:
    print(carac)
```

```
v
e
n
d
r
e
d
i
```

Opérateur Relationnel dans les chaînes de caractères

Tous les opérateurs de comparaison sont également applicables aux chaînes de caractères. Les chaînes de caractères sont comparées sur la base de la valeur ASCII ou Unicode (c'est-à-dire Dictonary Order).

In [10]:
```
#Exemple

str1 = "Hello"
str2 = "Sunday morning"
str3 = "Hi"

print(str3 < str2) #Output : True
print(str1 != str2) #Output : True
print(str1 > str3) #Output : False
```

```
True
True
False
```

Quelques méthodes utiles pour les chaînes de caractères

capitalize()

Elle met la première lettre en majuscule.

In [11]:
```
# Exemple

day = "sunday morning"
d = day.capitalize()
print(d)# Output : 'Sunday morning'
# captalizes the first character of the string
```

```
Sunday morning
```

.title()

Elle met la première lettre en majuscule de chaque mot de la chaîne de caractères.

In [12]:
```
day = "sunday morning"
d = day.title()
print(day)
print(d) # Output :  'Sunday Morning'
# captalizes the first character of each word in the string
```

```
sunday morning
Sunday Morning
```

upper ()

Elle met toutes les lettres en majuscules.

In [13]:
```
day = "sunday morning"
d = day.upper()
print(d) # Output :  'SUNDAY MORNING'
```

```
SUNDAY MORNING
```

lower()

Elle met toutes les lettres en minuscules.

In [14]:
```
day = "SUNDAY morning"
d = day.lower()
print(d) # Output :  'sunday morning'
```

```
sunday morning
```

swapcase()

Inverse la casse de chaque caractère d'une chaîne de caractères

In [15]:
```
day = "SUndAy morning"
d = day.swapcase()
print(d) # Output :  'suNDaY MORNING'
```

```
suNDaY MORNING
```

isupper()

Il vérifie si toutes les lettres sont en majuscules.

In [16]:
```
day = "SUNDAY MORNING"
d = day.isupper()
print(d) # Output : True

day =  "Sunday morning"
d = day.isupper()
print(d) # Output : False
```

```
True
False
```

islower()

Il vérifie si toutes les lettres sont en minuscules.

In [17]:
```
day = "sunday morning"
d = day.islower()
print(d) # Output : True

day = "SUNDAY MORNING"
d = day.islower()
print(d) # Output : False
```

```
True
False
```

split()

In [18]:
```
day = "sunday morning"
list_from_day = day.split()
print(list_from_day)

day = "sunday,morning"
list_from_day = day.split(",")
print(list_from_day)

day = "sunday-morning"
list_from_day = day.split("-")
print(list_from_day)
```

```
['sunday', 'morning']
['sunday', 'morning']
['sunday', 'morning']
```

join()

Elle combine les chaînes d'une liste en une seule chaîne de caractères séparée en fonction de(s) caractère(s) indiqué(s) en paramètre.

In [19]:
```
day = ["sunday",  "morning"]
day_name = "".join(day)
print(day_name)

day_name = "-".join(day)
print(day_name)

day_name = "&&&".join(day)
print(day_name)
```

```
sundaymorning
sunday-morning
sunday&&&morning
```

strip()

Elle supprime l'espace vide de deux cotes d'une chaîne de caractères.

In [20]:
```
day = "   sunday morning    "
print(len(day))

day = day.strip()

print(len(day))
```

```
21
14
```

In [ ]: