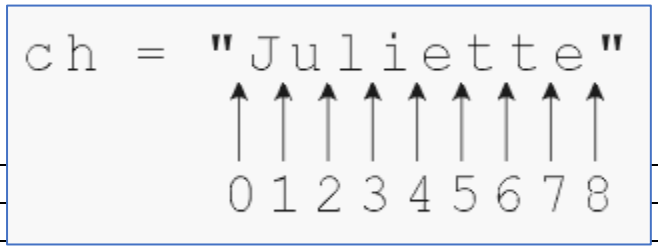
1. Introduction et rappels

Les **chaînes de caractères** font partie d'une catégorie d'objets Python que l'on appelle des **séquences**, et dont font partie **aussi les listes et les tuples**. On peut effectuer sur les séquences tout un ensemble d'opérations similaires.

Les chaînes de caractères peuvent être exprimées de différentes manières. Elles peuvent être écrites entre guillemets simples (`'...'`) ou entre guillemets (`"..."`) sans distinction le ***backslash*** **** peut être utilisé pour protéger un guillemet. En jargon informatique cela s'appelle échapper le caractère à afficher correctement.



2. Les caractères particuliers

`"`	guillemet
`'`	apostrophe
`\n`	passage à la ligne
`\\`	insertion du symbole `\`
`\%`	pourcentage, ce symbole est aussi un caractère spécial
`\t`	tabulation
`\r`	retour à la ligne, peu usité, il a surtout son importance lorsqu'on passe d'un système *Windows* à *Linux* car *Windows* l'ajoute automatiquement à tous ses fichiers textes

`print("J\'ai vingt ans ! \nC\'est la fête\t Youpi !")`	J'ai vingt ans ! C'est la fête Youpi !
`print("C:\\Users\\Documents")`	C:\Users\Documents

3. Les opérateurs

Opérateur	Effet	Exemple	Sortie
`+`	Concaténation de chaine de caractères	`vehicule ="grosse"+" " +"voiture"`	grosse voiture
`+=`	Concaténation plus affectation	`vehicule += " rouge"` `print(vehicule)`	grosse voiture rouge
`in/not in`	Une chaîne en contient-elle une autre ?	`print( "rouge" in vehicule)` `print( "rouge" not in vehicule)`	True False
`[0]`	Obtention du énième caractère, le premier caractère a pour indice 0	`ch = "abc"` `print(t[0])`	a
`[i:j]`	Obtention des caractères compris entre les indices `i` et `j-1` inclus, le premier caractère a pour indice 0	`ch = "Morganne"` `print(t[2:5])`	rga r indice2 Premier n indice 5

4. Parcours d'une chaine de caractères

while ……. :	for in :	Sortie

`nom = 'Jacqueline'`
`index = 0`
`while index < len(nom):`
`  print(nom[index] + ' *'),`
`  index = index + 1`
	`nom = 'Jacqueline'`
	`for character in nom:`
	`  print(character + ' *')`
		J *
		a *
		c *
		q *
		u *
		e *
		l *
		i *
		n *
		e *

Le for est plus simple à écrire il évite l'utilisation obligatoire de la condition d'évolution du while (un compteur) index = index +1

5. Recherche dans une chaine de caractères

`car = "e"`
`voyelles = "aeiouyAEIOUY"`
`if car in voyelles:`
`  print (car, "est une voyelle")`
	`e est une voyelle`

6. Longueur d'une chaine de caractères fonction len()

`txt = "apple#banana#cherry#orange"` `print("Longueur de la chaîne : ",len(txt))`	Longueur de la chaîne : 26

7. Les principales méthodes

- Compte le nombre d'occurrence d'une sous-chaine dans la chaine testée

string.count(value) Any type (string, number, list, tuple, etc.)	
`txt = "apple#banana#cherry#orange"` `print("Nombre de a : ",txt.count("a"))`	Nombre de a : 5

- Retourne la première occurrence de la recherche dans une chaine sinon retourne -1

string.find(value, start, end)

txt = "Hello, welcome to my world." x = txt.find("welcome") print(x)	7

Il existe la méthode ***string.index(value, start, end)*** *qui fait la même chose mais qui lève une exception en cas de recherche non aboutie.*

- Retourner une liste des différentes parties de la chaine par rapport à un séparateur

***string*.split(*separator, max*)**

```
txt = "apple#banana#cherry#orange"
x = txt.split("#")
print(x)
# setting the max parameter to 1, will re-
turn a list with 2 elements!
x = txt.split("#", 1)
print(x)
```

```
['apple', 'banana', 'cherry', 'orange']
['apple', 'banana#cherry#orange']
```

- Regrouper les éléments d'une liste dans une chaine de caractères

string.join(objet itératif) (tout objet conteneur constitué de chaine de caractères tuple liste)

```
myTuple = ("John", "Peter", "Vicky")
x = "#".join(myTuple)
print(x)
myList = ["Anna", "Maria", "Lisbeth"]
x = "#".join(myList)
print(x)
```

```
John#Peter#Vicky

Anna#Maria#Lisbeth
```

```
myDict = {"name": "John", "coun-
try": "Norway"}
mySeparator = "TEST"
x = mySeparator.join(myDict)
```

```
nameTESTcountry
```

8. Autre méthodes remarquables :

format()	Formats specified values in a string – voir cours précédent
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
lower()	Convertie la chaine en minuscules
upper()	Convertie la chaine en majuscules