Les chaines de caractères – String

Les chaines de caractères sont des objets python.
Ces objets ont des fonctions attitrées leur permettant de réaliser des actions bien précises.

Les chaines de caractères utilisent la *class str*. Cette classe fait partie de la librairie python nommée ***string***.

Consulter la librairie strings : https://docs.python.org/fr/3/library/string.html

Dans cette librairie, il y a toutes les méthodes et fonctions utilisables par les objets str.

1. Visite des méthodes les plus importantes.

Pour démarrer correctement, il n'est pas nécessaire de connaître toute la librairie. Cependant certaines fonctions faciliteront les développements. Une fois familiarisé avec les éléments de base, le développeur pourra améliorer et compléter ses connaissances en parcourant et en testant les autres fonctionnalités fournies par la librairie.

Dans un premier temps, consultez la documentation et consacrez-vous à comprendre, tester et maîtriser les fonctions suivantes :

- upper()
- lower()
- join()
- format()
- split()
- find()

Test :

Réaliser les tests dans le terminal de votre choix :

Créer une chaine avec le contenu suivant avec python je progresse vite en programmation.

Afficher toute la chaine en lettre capitale.

Afficher le résultat en lettre normale.

La fonction ***split()*** peut servir à mettre chaque mot dans une liste en utilisant l'espace entre les mots comme séparateur .

*Créer une variable **tab** contenant un mot par index de la chaine de caractères précédente avec la fonction **split()***.

La fonction join() permet de récupérer les éléments d'un tableau et de reconstituer une chaine de caractères.

Créer une chaine de caractères en récupérant les éléments du tableau et en mettant un tiret à la place des espaces habituels.

Créer une chaine en mettant des underscores à la place des espaces habituels.

2. indice, slice, concaténation

Les chaines de caractères peuvent être indicées dans un sens comme dans l'autre (indices positifs ou négatifs). Elles peuvent être découpées en tranches (slices), de plus elles peuvent être concaténées (+) ou multipliées(*). Elles sont également itérables.

Indice +	0	1	2	3	4	5
	P	Y	T	H	O	N
Indice -	-6	-5	-4	-3	-2	-1

Les indices :

Ouvrez un terminal de votre choix

Créer la variable

```
>>> chaine = "python"
```

Afficher le contenu de la chaine
Afficher uniquement le caractère ***p***
Afficher uniquement le caractère ***o***

Le slicing :

Le slicing permet de sélectionner des portions à l'intérieur d'une chaine de caractères. Pour cela au lieu d'indiquer entre crochets l'indice du caractère souhaité, il est possible d'indiquer la position du caractère de départ, celle de fin et parfois un pas. Ceci ressemble fortement à la fonction ***range()*** mis à part que les paramètres seront séparés par des **:** .

Syntaxe :

```
>>> chaine = "abcdefghijklmnopqrstuvwxyz"
```

On peut donc utiliser les paramètres de différentes façons :

Utilisation de la position de départ

```
>>> chaine[pos_debut : ]      → ira jusqu'à la fin de la chaine
```

Utilisation de la position de départ et de la fin

```
>>> chaine[pos_debut : pos_fin]
>>> chaine[: pos_fin]          → démarre de 0 jusqu'à pos_fin.
```

Utilisation de la position de départ, de la fin et d'un pas.

```
>>> chaine[pos_debut : pos_fin : pas]
```

Exemple pour récupérer uniquement les lettres ***fghi***, il est possible de faire l'action suivante :

```
>>> chaine[5 : 9]
```

Exercice :

>>> *chaine = "**abcdefghijklmnopqrstuvwxyz**"*

*Afficher uniquement les lettres **abcdef**.*
*Afficher uniquement les lettres **mnopqr**.*
Afficher les 10 premières lettres.
Afficher les 10 dernières lettres.
*Afficher **acegikmoqsuwy.***
*Afficher **bdfhjlnprtvx.***
*Afficher **acegi.***
*Afficher tout le contenu de la chaine sans aucun indice de position**.***
Afficher la chaine à l'envers.

La concaténation :

La concaténation sert à regrouper plusieurs variables contenant des chaines de caractères en une seule à l'aide d'un opérateur arithmétique.

Ex :
```
>>> chaine_a = "abcdef"
>>> chaine_b = "hijklm"
>>> chaine_c = chaine_a + " - " + chaine_b
>>> chaine_c
'abcdef - hijklm'
```

Exercice : *1.reconstitution_boucle.py*

tab_car = ['p', 'y', 't', 'h', 'o', 'n', ' ', 'e', 's', 't', ' ', 'u', 'n', ' ', 's', 'u', 'p', 'e', 'r', ' ', 'l', 'a', 'n', 'g', 'a', 'g', 'e']

Tâches à réaliser :

Reconstituer la phrase à l'aide d'une boucle

Exercice : *2.reconstitution_join.py*

Tâches à réaliser :

Reprendre la variable tab_car
Reconstituer la phrase à l'aide de la fonction join() vue plus haut.

Astuce : Dans certaines situations, il est intéressant de forcer une chaine de caractères à devenir une ***list()***.

Tester les instructions suivant :

*chaine = **"abcdefghijklmnopqrstuvwxyz"***
tab_car = list(chaine)

3. Utilisation de la librairie string

Une librairie est aussi appelée un module, elle regroupe du code et des fonctions et méthodes d'utilisation.

Documentation officielle :
https://docs.python.org/fr/3/library/string.html

Consultez la documentation et faites des tests pour comprendre comment utiliser la librairie.

Dans un script, l'instruction ***import*** permet d'importer tout le module.
L'instruction ***from… import …*** permet de n'importer que ce dont nous avons besoin (variables, fonctions, classes du module)

Exemple :

```
# en haut de votre progamme
import string
```

ou

```
from string import upper
```

Exercice : *3.decode_le_flag.py*

Tâches à réaliser :

Avec la séquence d'index de string.printable, retrouvez le flag :

```
[15, 21, 10, 16, 90, 25, 34, 29, 17, 0, 23, 88, 1, 28, 88, 16, 27, 3, 10, 29, 92]
```

Indice le flag ressemble au format suivant :

```
flag{xxx_xx_xx}
```