Devoir pratique
Chaîne de caractères

Tous documents autorisés.

1. Le conjugueur (Utiliser le fichier **conjugueur.py**)

Le 1er groupe comprend tous les verbes dits « réguliers » qui finissent en -er (sauf le verbe aller). Ils représentent près de 90 % des verbes.

Au présent les terminaisons sont les suivantes pour un verbe xxxER on remplace ER par la terminaison adaptée:

Je	xxxE
Tu	xxxES
Il/Elle	xxxE
Nous	xxxONS
Vous	xxxEZ
Ils/Elles	xxxENT

On se propose de faire une application permettant de donner la conjugaison des verbes du premier groupe au présent.

1.1. Proposez un programme qui demande un verbe à l'infinitif. Si ce dernier se termine par "er" et n'est pas le verbe aller, affichez la conjugaison de ce dernier au présent de l'indicatif.

1.2. On souhaite améliorer l'application afin que si le mot commence par une voyelle (verbe arriver), on ait "j'" et non "je" pour la première personne du singulier. Proposez une solution.

Les verbes en -ger comme manger, ranger... prennent un e intercalaire à la 1ère personne du pluriel (ex : pour le verbe manger, on aura mangEons et non mangons).

1.3. Proposez une solution pour répondre à cette problématique.

Exemple de rendu affiché dans la console :

```
votre verbe ?danser
je danse
tu danses
il/elle danse
nous dansons
vous dansez
ils/elles dansent
```

```
votre verbe ? manger
je  mange
tu  manges
il/elle  mange
nous  mangeons
vous  mangez
ils/elles  mangent
```

```
votre verbe ?aller
verbe non supporté

votre verbe ?dormir
verbe non supporté
```

2. Cryptage et décryptage de messages (Utiliser le fichier **cryptage-decryptage.py**)

On souhaite pouvoir crypter/décrypter des messages en utilisant un algorithme de substitution des lettres. Voici la table de substitution proposée :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
@	8	[	0	3	{	6	#	1	}	:	7	W	Z	*	?	O	%	$	-	V	^	M	+	/	N

Le fonctionnement de l'application doit être le suivant :

- Via un menu, l'utilisateur peut choisir s'il souhaite crypter ou décrypter un message ;
- Dans les 2 cas, il lui est demandé le message à crypter ou à décrypter ;
- Selon le choix effectué, le cryptage ou le décryptage du message qu'il vient de saisir s'affiche ;

Il est conseillé de déclarer 2 listes initialisés, l'un avec les lettres, l'autre avec les caractères de substitution correspondant.

Au niveau du traitement de la chaîne saisie, il faudra commencer par mettre toutes les lettres en majuscule. Les caractères accentués ne seront pas traités.

Pour le cryptage, il suffit de parcourir la chaîne de caractères et de tester si le caractère courant est une lettre, de trouver l'indice correspondant et de remplacer cette lettre par le code de substitution. Si le caractère courant n'est pas une lettre, on ne le remplace pas.

2.1. Proposez un programme permettant le cryptage d'un message.

Pour le décryptage, le principe est le même, mais l'indice du caractère courant ne peut pas être calculé. Il faut donc rechercher le caractère dans la liste de substitution et grâce à l'indice de ce dernier, mettre la lettre correspondante.

2.2. Complétez votre programme afin de permettre le décryptage d'un message.

Exemples de rendu affichés dans la console :

```
----------------------------------------
1 : Codage d'une phrase
2 : Décodage d'une phrase
q : quitter
----------------------------------------
votre choix ?1
votre phrase en clair ?il pleut encore
la phrase codée est 17 ?73V- 3Z[*§3
```

```
----------------------------------------
1 : Codage d'une phrase
2 : Décodage d'une phrase
q : quitter
----------------------------------------
votre choix ?2
votre phrase codée ?17 ?73V- 3Z[*§3
la phrase décodée est IL PLEUT ENCORE
```

3. Le code Morse (Utiliser le fichier **morse.py**)

Écrivez un programme qui demande à une personne une chaîne de caractères, ne contenant que des lettres, des chiffres, puis affiche la correspondance de cette dernière en code morse.

Il faudra commencer par mettre toutes les lettres en majuscule.

ASCII	Lettre	ASCII	Lettre	ASCII	Lettre	ASCII	Lettre
65	A . _	74	J . _ _ _	83	S . . .	48	0 _ _ _ _ _
66	B _ . . .	75	K _ . _	84	T _	49	1 . _ _ _ _
67	C _ . _ .	76	L . _ . .	85	U . . _	50	2 . . _ _ _
68	D _ . .	77	M _ _	86	V . . . _	51	3 . . . _ _
69	E .	78	N _ .	87	W . _ _	52	4 . . . . _
70	F . . _ .	79	O _ _ _	88	X _ . . _	53	5 . . . . .
71	G _ _ .	80	P . _ _ .	89	Y _ . _ _	54	6 _ . . . .
72	H	81	Q _ _ . _	90	Z _ _ . .	55	7 _ _ . . .
73	I . .	82	R . _ .			56	8 _ _ _ . .
						57	9 _ _ _ _ .

Remarque : utiliser la fonction « **ord** » (TP6 page 2)

Exemple de rendu affiché dans la console :

```
votre texte ?1912 sos titanic
.----
----.
.----
..---
...
---
...
-
..
-
.-
-.
..
-.-.
```

4 Trois exercices

4.1 Ecrire une fonction **`motLePlusLong`** qui prend en argument une liste de mots et renvoie le mot le plus long.

Exemple d'exécution :

```
liste=["bonjour","ici","fleur","fleuve","informatique"]
print(motLePlusLong(liste))

affiche dans la console : informatique
```

4.2 La distance de Hamming entre deux mots de même longueur est le nombre d'endroits où les lettres sont différentes.

Par exemple : JAPON - SAVON

La première lettre de JAPON est différente de la première lettre de SAVON, les troisièmes aussi sont différentes.

La distance de Hamming entre JAPON et SAVON vaut donc 2.

Écrire une fonction **`distance_hamming`** qui qui prend en argument deux mots et qui calcule la distance de Hamming.

Exemple d'exécution :

```
result=distance_hamming('lapin','satin')

print(result)

affichage dans la console : 2
```

4.3 Ecrire une fonction **`double`** qui prend en argument un mot et renvoie le mot obtenu en doublant chaque lettre.

Exemple d'exécution :

```
print(double("bon"))

affiche dans la console : bboonn
```