Boucle “for” =>

Lors de l’utilisation de la syntaxe ‘for ... in …’, chaque élément de la liste spécifiée après le mot « in » (via la fonction ‘range’ par exemple) sera attribué tour à tour à la variable spécifiée après le for.
Seront également exécutées à chaque attribution les instructions indentées.
La boucle “for” s’arrête lorsque tous les éléments de la liste ont été attribués une fois.

Cela permet donc de contrôler le nombre d’itérations d’une boucle. L’utilisation la plus courante est la forme « for i in range(x) », où x est un entier.
La fonction range() génère une liste comprenant chaque valeur de l’intervalle entre les deux valeurs spécifiées, ou entre 0 et la valeur spécifiée si elle est unique.
Le pas étant de 1 par défaut, les instructions indentées seront répétées x fois.

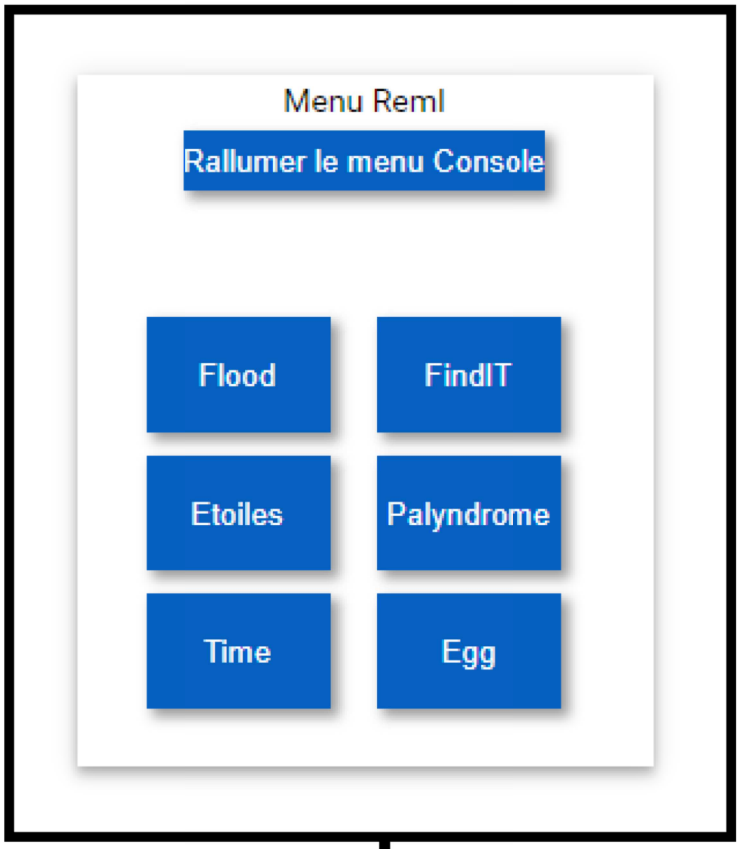
Chaînes de caractère =>

Une chaîne de caractères prend la forme d’une variable appelée ‘string’.
Elle contient plusieurs caractères isolés les uns des autres.

Il est possible d’appeler cette variable en précisant un entier relatif pour récupérer un caractère spécifique contenu dans celle-ci.
Le premier caractère de la string est désigné par 0, le deuxième par 1, le troisième par 2…
La valeur désignant le dernier caractère de la variable est égale à (nombreDeCaractères – 1).
Il est possible d’utiliser des entiers négatifs, le dernier caractère de la variable est désigné par -1, l’avant dernier par -2…
Le premier caractère de la variable peut ainsi être désigné par la valeur (-1*nombreDeCaractères).

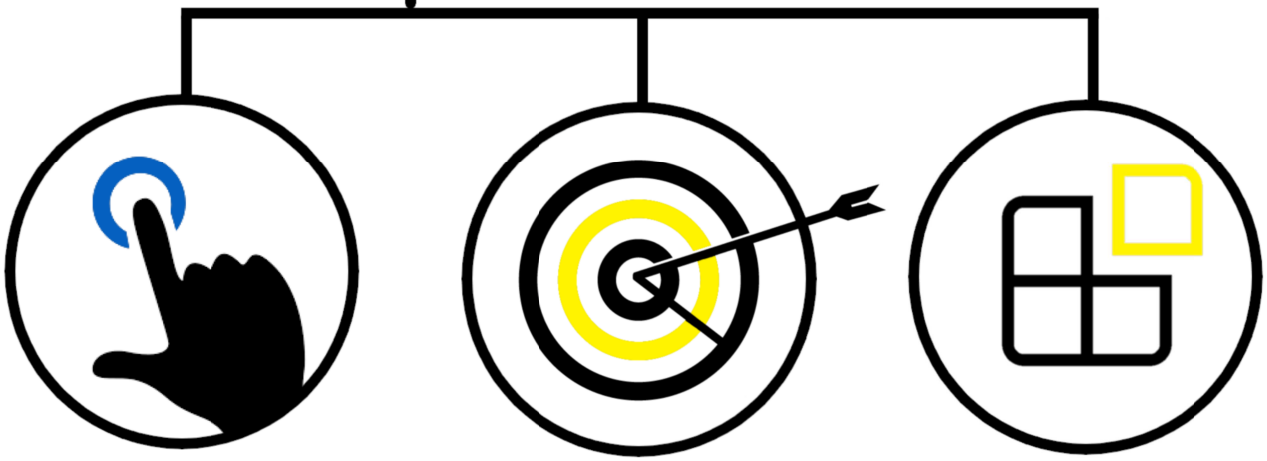
(exemple : « variable_string[0] » récupérera le premier caractère contenu dans variable_string)

Présentation du Programme :



MADE WITH REM

REMote interface



ERGONOMIQUE ET SIMPLE A PRENDRE EN MAIN

EFFICACE ET COMPLET

ETRÊMEMENT MODULAIRE

LISTE DES FONCTIONNALITES

Flood

Emerveillez vous devant votre console remplie de chiffres !

Entrez votre mot et recherchez des trucs dedans !

FindIT

Etoiles

Etoilez vos mots et dévoilez leur véritable beauté !

Retrouvez l'envers de vos mots, et sachez enfin si ce sont des palindromes !

Palyndrome

Time

Connaissez l'heure !
(pas la votre, celle des serveurs de Repl.It)

Descriptif spécifique de chaque fonction programmée et lien vers le projet (déjà indiqué ci dessus, mais répété plus bas pour respecter l'ordre de cahier des charges. Parce que… on sait jamais.

`findTDT(objet, chr2Find, Interface=False)`
Cette fonction évaluera chaque caractère de la string « objet » et donnera en sortie un message composé comprenant le nombre de fois que le contenu de « chr2Find » correspond à un des caractères de la string, ainsi le rang de chaque caractère repéré.
Le paramètre Interface lorsque spécifié « True » empêchera l'intégration des codes couleurs en sortie.

`starry(objet, Interface=False)`
Renvoie le contenu de « objet » avec des étoiles entre chaque lettre.
Le paramètre Interface lorsque spécifié « True » empêchera l'intégration des codes couleurs en sortie.

`invert(objet, Interface=False)`
Renvoie une string contenant chaque caractère de « objet » dans l'ordre inverse, ainsi qu'un message True ou False (ou un boléen équivalent lorsque Interface = False) selon que l'inverse de « objet » corresponde à « objet » (True) ou non (False)
Le paramètre Interface lorsque spécifié « True » empêchera l'intégration des codes couleurs en sortie.

`flood()`
Affiche la suite 2**n+1 accompagné de « UNEXPECTED ERROR » jusqu'à la 16e valeur, de façon à rendre illisible la console.

`date()`
Renvoie l'heure des serveurs hébergeant le programme.

`ConsoleMenu(RemI2Exit=0)`
Affiche le menu console dans la console, attend un Input et exécute la fonction correspondante, ou génère ses propres messages d'erreurs non invasifs pour informer l'utilisateur que sa saisie est incorrecte.
(RemI2Exit sert à changer un peu le message du menu lorsqu'il est réactivé depuis l'interface RemI)

> `menuTxT(RemI2Exit_menuTxT, Mset=-1)`
> Affiche le texte qui résume le menu et demande un input.

[Les fonctions suivantes ne sont utilisées que dans l'interface RemI]

`main(self)`
Contient les données de la fenêtre du menu à proprement parler. Est exécuté au démarrage du serveur. c'est elle qui contient les boutons qui activent les sous-programmes (fonctions précédentes)
Toutes les autres fenêtres ne sont en réalité que des boites de dialogue.

`BTaction(self, widget, action)`
Tous les listener des boutons du Menu activent cette fonction, en changeant simplement le paramètre action (qui permet d'identifier quel bouton l'a activée). Cette fonction recueillera une saisie utilisateur via des boites de dialogue et activera la fonction pour donner le résultat attendu.

> `vrai_dialog(self, Titre, Message, Callback)`
> génère la boite de dialogue.

```
def dialogue_confirmed(widget, sortie)
```
s’active en tant que callback lorsque le bouton OK est pressé dans la boite de dialogue. Active la fonction spécifiée dans vrai_dialog.
(le paramètre Callback est transféré grâce à une variable global)

```
chr2FindDialog(findTDT_objet, Interface)
```
Génère une seconde boîte de dialogue car la fonction findTDT() nécessite deux saisies distinctes.

```
dialogue_confirmed(widget, sortie)
```
s’active en tant que callback lorsque le bouton OK est pressé dans la boite de dialogue. Active la fonction findTDT(). (récupère la première saisie utilisateur via une variable global)

```
eggFunction(self)
```
gère la boite de dialogue et le bouton principal ‘egg’

```
egg_action(self)
```
s’active en tant que callback lorsque le bouton ‘egg’ est pressé.
Modifie la position et le message du bouton, et en rajoute même un à un moment.

```
egg_label_edit_bt2(self)
```
s’active en tant que callback lorsqu’un bouton généré par egg_action est pressé. fait apparaître un texte dans la boîte de dialogue.

Retrouvez tout cela sur : https://repl.it/@RaphaelNguyen/Django-avec-les-chaines
(lien vers mon projet)