Context

Tous les jours vous avez accès à internet pendant votre travail.
Aujourd'hui ce n'est pas le cas, et vous n'aurez pas non plus accès à votre téléphone.
La documentation standard python vous sera fournie au format .pdf.

Vous avez interdiction d'utiliser une IDE (pas de VSCode, ni d'atom).
Vous devrez utiliser vi ou nano uniquement.

Part 0

Lisez chaque exercice et chaque question du sujet avant de commencer. Et relisez chaque partie avant de la commencer.

Créez un répertoire à votre nom, tout votre travail sera réalisé dans ce répertoire.

```
~ $ mkdir lumy ; cd lumy
~/lumy $
```

Ecrivez votre nom (ou surnom) avec un point à la fin dans un fichier nommé "rendu.txt".
ex:

```
$ cat rendu.txt
lumy.
```

En option : écrivez un autre fichier `why.txt`.
En une ou deux phrases, expliquez ce que vous attendez du domaine de l'informatique (un travail, de l'argent, des projets personnels, de la domotique, un nouveau milieu, travailler pour Google, travailler dans l'IoT…).

A partir de maintenant chaque partie de l'examen sera réalisée dans son propre répertoire.

```
$ ls
part1/ part2/ part3/ ...
$ ls part1/
part1/myfuncs.py
```

Part 1

Dans cette partie **vous n'êtes PAS autorisés à utiliser les fonctions intégrées à Python, sauf listées ci-dessous, ou d'utiliser d'autres modules (pas d'imports). Utiliser les méthodes de la classe `str` est interdit.**

Les fonctions "intégrées" sont celles dispo en Python sans avoir à faire d'imports : comme par exemple `map`, `zip`, `all`, `any`…

Vous êtes cependant autorisés à utiliser les fonctions suivantes : `len()`, `range()`, `ord()`, `chr()`.

Les opérateurs et mots clés sont **autorisés** : `and`, `or`, `not`, `>`, `for`, `while` …

Les méthodes de la classe `str` sont des fonctions que vous pouvez appeler à partir d'une chaîne de caractères : par exemple "abc".isalpha() appelle la "méthode" `isalpha` sur la chaîne "abc".

Vous réferrer à la question bonus pour savoir comment gérer les arguments incorrects.

Dans le fichier `myfuncs.py` écrivez ces 5 fonctions :

- isalpha: prend en argument une chaîne de caractères et vérifie si elle est composée uniquement de lettres
    - Retourne True si tous les symboles de la chaîne sont des lettres (majuscules ou minuscules), False sinon
- isdigit: prend en argument une chaîne de caractères et vérifier si elle est composée uniquement de nombres
    - Retourne True si tous les caractères sont des chiffres (between 0 and 9), False sinon
- isalnum: prend une chaîne en argument et vérifie si elle est alphanumérique
    - Retourne True si tous les caractères sont des lettres (majuscules ou minuscules), ou des chiffres, False sinon
- tolower : convertis une chaîne en minuscules
    - Prend une chaîne en paramètre et retourne la même chaîne avec toutes les lettres majuscules converties en minuscules (sans changer le reste)
        - Exemple : tolower("AbCd1234#") => "abcd1234#"
- lencmp: compare la longueur de deux chaînes
    - Prend deux chaînes en paramètre, retourne 0 si elles sont de longueurs égales, 1 si la première est plus longue, -1 si la première est plus courte
    - lencmp("a", "b") retourne 0
    - lencmp("aa", "z") retourne 1
    - lencmp("z", "aa") retourne -1
- strcmp: prendre 2 chaâines en paramètres, retourne 0 si elles sont égales, 1 si la première est supérieure à la seconde (dans l'ordre lexicographique), -1 si la première est inférieure
    - Cette fonction ignore la casse : strcmp("ABC", "abc") must return 0
    - "L'ordre lexicographique" est la manière dont les mots sont ordonnés dans un dictionnaire, aussi appelé "ordre alphabétique"
        - D'abord comparer chaque caractère de chaque chaîne en partant de la gauche
        - Si une chaîne possède un caractère placé plus haut dans l'ordre alphabétique que l'autre : elle est supérieure
        - Si les deux chaînes commencent par les mêmes caractères : la chaîne la plus longue est supérieure
        - Si les deux chaînes contiennent les mêmes caractères dans le même ordre : elles sont égales (évidemment)
        - Exemples :
            - "b" est supérieure à "abcd"
            - "abcd" est supérieure à "abc"
            - "z" est supérieure à "ywfrgeser"

Bonus:

- "raise" une ValueError quand les arguments ne sont pas du bon type.
- Implémenter la fonction strlen : prend une chaîne en paramètre, retourne sa longueur
    - Sans utiliser `len()`
    - Doit retourner un entier : soit la taille de la chaîne passée en paramètre, soit -1 si l'argument n'est pas de type "str"

function name	parameters	return value
isalpha	string	boolean
isdigit	string	boolean

function name	parameters	return value
isalnum	string	boolean
tolower	string	string
lencmp	string, string	int
strcmp	string, string	int

Part 2

Vous pouvez maintenant utiliser des fonctions intégrées à Python. **Pas d'imports ou d'appel à des méthodes".

Pour chaque fonction créer un fichier python du même nom, dans lequel vous déclarerez la fonction.
Un tableau est fourni avec la signature de chaque fonction, utiliser le nom de fichier indiqué dans la colonne "filename".
Vous devrez aussi définir votre fonction sous ce nom.

- Trois mots :
  - Prend une chaîne en paramètre
  - Renvoie True si la chaîne contient trois "mots" consécutifs séparés par des espaces ne contenant que des lettres
  - Exemples :
    - 'abc abc dce 1 cde' => True
    - 'abc 1 bcd 1 abc' => False

- Inverser chaque mot :
  - Prend en paramètre une chaîne composée de mots séparés par des espaces
  - 'hello world' => 'olleh dlrow'

- Mot de passe conforme:
  - Prend une chaîne en paramètre
  - Renvoie vrai si le mot de passe contient tous les éléments suivants :
    - des chiffres et des lettres
      - au moins un caractère "spécial" (qui n'est ni une lettre ni un chiffre)
      - longueur d'au moins 8 caractères

- Multiplication des chiffres
  - Prend un entier en entrée
  - Prend chaque chiffre du nombre et les multiplie entre eux
  - Si le nombre en entrée est négatif : le résultat est négatif
  - Ignorer les zéros
  - Doit retourner un `str`
  - Exemples :
    - 123045 = 1 * 2 *3 * 4 * 5 => '120'
      - -2035 = -2 * 3 * 5 => '-30'

- Suite de Syracuse :
  - Prendre un entier `N` plus grand que 1 en entrée
  - Répéter les opérations suivantes jusqu'à ce que `N` vaille 1 : si `N` est pair : diviser par 2, autrement : multiplier par 3 et ajouter 1
  - Retourner tous les nombres rencontrés dans ce procédé en tant que liste (sauf le tout premier)

    - Exemples :
        - En commençant avec 7 la séquence est : "22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1"
        - En commençant avec 10 la séquence est : "5, 16, 8, 4, 2, 1"
        - Contexte : il y a une théorie mathématique qui n'a jamais été prouvée, la "conjecture de Collatz", selon laquelle la suite de Syracuse de tout nombre entier positif ne sera pas infinie, mais des tests ont prouvé que pour tout nombre que vous pourriez entrer dans votre ordinateur la suite ne sera pas infinie

- Fizz Buzz:

    - Prend un entier en paramètre
    - Retourne une chaîne
        - `Fizz Buzz` si le nombre est divisible par 3 et par 5
        - `Fizz` si le nombre est divisible par 3
        - `Buzz` si le nombre est divisible par 5
        - Le nombre converti en chaîne dans les autres cas

    Bonus:
    Pas de question bonus, mais des points supplémentaires seront accordés si les fonctions intégrées à Python sont utilisées judicieusement dans le code.

filename	function name	parameters	return value
three_word	three_word	string	boolean
reverse_word	reverse_word	string	string
valid_password	valid_password	string	boolean
multiply_digit	multiply_digit	int	string
syracuse	syracuse	integer	list
fizzbuzz	fizzbuzz	integer	string

Part 3

Vous pouvez utiliser des imports et des méthodes.

Ecrivez cette fonction dans un fichier nommé `cpwd.py`

- check_password en utilisant l'algorithme SHA256 :
    - prend deux chaînes de caractères en paramètre
    - la première est le mot de passe en clair
    - la seconde est le hash (le mot de passe qu'on a protégé)
    - retourne True si le mot de passe en clair correspond bien au mot de passe hashé, False sinon
    - example:
        - check_password(b"test", "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08") => True
        - check_password(b"test", "915b0f00a08") => False

Hint:

hashlib can be very usefull.

- Bonus: Take a third parameters, optional with default value of `sha256` but accept also `blake` and use the right algorithm in function of this string
    - examples:

        - check_password(b"test", "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08", "sha256") => True
        - check_password(b"test", "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08", "blake") => False

Part 4

- créez un script Python `mydb`
    - prend ses arguments depuis la ligne de commande (depuis sys.argv)
        - le premier est toujours `add` or `get`
        - le deuxième est une clé
        - le troisème (pour add) est la valeur
    - créez et mettez à jour un fichier `mydb.db` qui va contenir votre base de données
    - Pour `add` la paire clé/valeur sera enregistrée
    - Pour `get` la valeur correspondant à la clé sera affichée

```
$ mydb add favorite_teacher lumy
$ mydb add favorite_day monday
$ mydb get favorite_teacher
lumy
$ mydb add favorite_teacher romain
$ mydb get favorite_day
monday
$ mydb get favorite_teacher
romain
```

Bonus: crypter (toute méthode sera acceptée, même rot13) le contenu des valeurs dans my.db.

> ROT13 ("rotate by 13 places", abbrégé ROT-13) est un simple code par substitution qui remplace chaque lettre par celle qui est 13 positions plus loin dans l'alphabet. ROT13 est un cas particulier du code de César qui était utilisé à Rome pendant l'Antiquité.

Exemple: hello => uryyb

Bonus parts

A faire uniquement si vous avez fait tout le reste

- Créez un jeu de quizz avec tkinter
- Proposez des choix multiples à l'utilisateur pour chaque question
- Votre quizz doit récupérer son contenu depuis un fichier au format de texte structuré de votre choix (.yml, .json, .xml, .csv…).
- Ce fichier contient les questions, les réponses possibles, et les bonnes réponses
- Calculez le score de votre utilisateur à la fin du quizz (combre de bonnes réponses, nombre de mauvaises réponses…)
- Bonus : imiter Kahoot et implémenter un score dépendant de la vitesse de réponse