À partir de maintenant, il n'y aura plus d'exemples pour tester vos fonctions dans les sujets de TP. Vous devez être capables de créer vos propres tests qui soient suffisamment nombreux et pertinents pour vous assurer que votre programme fonctionne. Merci d'écrire les tests dans le fichier source pour en garder une trace.

Rappel : enregistrez votre fichier avant de commencer à coder et sauvegardez le régulièrement pour ne pas perdre votre travail en cas de problème. Au lycée, l'idéal est d'enregistrer votre travail sur une clé usb. Si vous n'avez pas de clé usb, enregistrez vos fichiers sur le disque « `U:` » et pas sur le disque « `C:` » ni sur le bureau.

Sur les ordinateurs du lycée, ouvrez les fichiers `.txt` et `.csv` avec `Notepad++` qui se trouve dans le dossier `Bureau/Outils`.

Exercice 1. Navigation dans le système de fichiers

Dans cet exercice :

- Lorsqu'on vous demande d'utiliser "l'interface de votre système d'exploitation", cela signifie que vous ne devez pas utiliser Python, vous devez utiliser votre souris comme vous le feriez habituellement.
- Contrairement aux autres exercices, vous devez écrire les commandes directement dans la console et pas dans le fichier source. Pensez à noter vos solutions quelque part, par exemple dans le fichier sous forme de commentaires. La console possède un historique : pour retrouver les commandes exécutées précédemment, utilisez les flèches haut/bas de votre clavier.

Avant de commencer l'exercice, écrivez « `import os` » dans le fichier source, enregistrez le sur une clé usb ou sur le disque « `U:` », puis exécutez le. Dans la console, la commande `os.getcwd()` devrait vous renvoyer le chemin du dossier dans lequel est enregistré votre fichier.

1. À l'aide de l'interface de votre système d'exploitation, ouvrir le dossier où se trouve votre fichier source et créer un nouveau dossier nommé `TP05`.
2. Dans la console et à l'aide de la fonction `os.chdir`, faire en sorte que le dossier courant soit le dossier `TP05`. Vérifier que la valeur de `os.getcwd()` est bien égale à `"<...>/TP05"` (où le `<...>` est remplacé par le chemin du dossier où se trouve votre fichier source).
3. Dans la console, créer à l'aide de la fonction `open` de Python un fichier `test.txt` dans le dossier `TP05` contenant la chaîne de caractères `"Hello World !"`. Avec l'interface du système d'exploitation, aller dans le dossier `TP05` et vérifier que ça a fonctionné.
4. Grâce au mode ajout de la fonction `open`, ajouter une deuxième ligne `"Bonjour tout le monde !"` au fichier `test.txt`. Avec l'interface du système d'exploitation, vérifier que le fichier a été modifié et qu'il contient maintenant deux lignes.
5. Pour cette question, le dossier courant doit rester `TP05` (c'est à dire que `os.getcwd()` doit valoir `"<...>/TP05"`). À l'aide de la fonction `mkdir` du module `os`, créez deux dossiers `dossier_test1` et `dossier_test2` dans le dossier `TP05`, puis un sous-dossier `sous_dossier_test` dans `dossier_test1`. Avec l'interface du système d'exploitation, vérifier que les trois dossiers ont été créés.
6. Dans la console, faire en sorte que le dossier courant devienne `TP05/dossier_test1/sous_dossier_test`. Vérifier que ça a fonctionné avec la commande `os.getcwd()`.
7. Pour cette question, le dossier courant doit rester `TP05/dossier_test1/sous_dossier_test`. Dans la console, et à l'aide d'un chemin relatif, créer dans le dossier `dossier_test2` un fichier `nombres.txt` qui contient les nombres de 0 à 100 séparés par des espaces. Avec l'interface du système d'exploitation, vérifier que le fichier a été créé.
8. À l'aide de la fonction `os.chdir`, faire en sorte que le dossier courant soit le dossier `TP05`. Vérifier que ça a fonctionné avec la commande `os.getcwd()`.

Exercice 2. Tables de multiplications

Écrire une fonction `table_mult` qui prend en entrée un entier `n` et qui enregistre dans un fichier le tableau à `n+1` lignes et `n+1` colonnes représentant les tables de multiplications comme dans l'exemple ci-contre où `n` vaut 5. Utilisez des tabulations (`"\t"`) pour faire les espaces. Lors des tests, vérifiez que Python n'affiche pas `None` dans la console

x	|	1	2	3	4	5
1	|	1	2	3	4	5
2	|	2	4	6	8	10
3	|	3	6	9	12	15
4	|	4	8	12	16	20
5	|	5	10	15	20	25

Exercice 3. Méthode `split` pour les chaînes de caractères

Dans les exercices qui suivent, on aura besoin d'utiliser la méthode `split` pour les chaînes des caractères. Grâce aux tests ci-dessous, expliquer ce que fait la méthode `split` :

```
s = "Ceci est,un;test Ceci est,un,test;Ceci;est un;test"
L1 = s.split(' ')
L2 = s.split(',')
L3 = s.split(';')
print(L1)
print(L2)
print(L3)
```

Exercice 4. Population française

Le fichier `population.csv` disponible sur la page du cours énumère toutes les communes de France ainsi que leurs populations en 2020.

`https://informatique-lhp.fr/itc-mpsi.html`

Remarque : un fichier `.csv` est en fait un fichier `.txt`, ouvrez le avec `Notepad++` ou bloc-note, pas avec Excel. Ces données ont été récupérées sur le site de l'INSEE :

`https://www.insee.fr/fr/statistiques/7632446`

Déterminer la population totale de la France en 2020 (Réponse attendue : 67 162 154).

Exercice 5. Le jeu du plus ou moins

Le but de cet exercice est de gérer un tableau des scores pour un jeu. Lorsque l'énoncé demande d'afficher une phrase contenant `<b>` ou `<n>`, cela signifie que vous devez afficher la valeur de `b` ou `n`. Par exemple, si `n` vaut `100` et que l'on vous demande d'afficher `"Entrez un nombre entre 0 et <n>: "`, alors vous devez afficher `"Entrez un nombre entre 0 et 100: "` dans la console.

Dans le "jeu du plus ou moins", le but du joueur est de deviner un nombre `a` tiré aléatoirement entre `0` et `n` par l'ordinateur. Plus précisément, l'ordinateur doit afficher `"Entrez un nombre entre 0 et <n>: "`, attendre que le joueur donne un nombre `b` puis :

- Si `b` est strictement négatif, l'ordinateur doit afficher `"Le nombre <b> n'est pas valide car il est négatif"`.
- Sinon, si `b` est plus grand que `n`, l'ordinateur doit afficher `"Le nombre <b> n'est pas valide car il est strictement plus grand que <n>"`.
- Sinon, si `b` est strictement plus petit que `a`, l'ordinateur doit afficher `"Le nombre recherché est strictement plus grand que <b>"`.
- Sinon, si `b` est strictement plus grand que `a`, l'ordinateur doit afficher `"Le nombre recherché est strictement plus petit que <b>"`.
- Enfin, si `a` et `b` sont égaux, l'ordinateur doit afficher `"Bravo, vous avez trouvé le nombre recherché"`.

Tant que le joueur n'a pas trouvé `a`, l'ordinateur lui demande d'essayer à nouveau en affichant `"Entrez un nombre entre 0 et <n>: "`. À la fin d'une partie, l'ordinateur affiche `"Votre score: "` suivi du nombre d'essais utilisés pour trouver `a` (un nombre invalide n'est pas compté comme un essai).

Afin d'interagir avec Python, utilisez la fonction `input` qui prend en argument une chaîne de caractères `s`, affiche `s` en console puis renvoie la saisie du joueur. Le joueur fait sa saisie dans la console et la termine grâce à la touche entrée. Utilisez également la fonction `randint` du module `random` pour générer l'entier aléatoire `a`.

1. Écrire une fonction `partie` qui prend en argument l'entier `n`, lance le jeu du plus ou moins et renvoie le score du joueur. Testez votre fonction.

Dans la suite, on souhaite enregistrer le tableau des scores dans un fichier, comme dans l'exemple ci-contre. Les lignes du fichier sont triées par score et chacune d'entre elles contient le nom du joueur, une tabulation (`"\t"`) puis le meilleur score du joueur.

```
Omar	7
Fred	12
Jamel	20
Eric	24
Ramzy	28
```

2. Écrire une fonction `nv_score` qui prend en entrée deux chaînes de caractères `nom_fichier` et `nom_joueur` ainsi qu'un entier `score`, et qui met à jour le fichier `nom_fichier` avec ce nouveau score. Dans le cas où `nom_joueur` n'apparaissait pas dans le fichier, une nouvelle ligne est créée. Dans le cas où `nom_joueur` apparaissait dans le

fichier, son score est mis à jour. Enfin, votre fonction doit renvoyer le meilleur score du joueur en question.
Vous pouvez supposer que les lignes du fichier sont initialement triées par score et qu'un joueur apparaît au plus une fois dans le fichier, de plus ces deux conditions doivent être vérifiées après l'appel à votre fonction.
Vous pouvez utiliser la fonction `exists` du module `os.path` pour tester si un fichier ou dossier existe. Pensez également à utiliser la méthode `split` de l'exercice 3. Si besoin, pensez à écrire des fonctions intermédiaires.
Proposez des tests pour votre fonction `nv_score` puis vérifier que vous obtenez le fichier ci-dessus lorsque vous appelez successivement :

```
nouveau_score("fichier_test.res","Jamel",40)
nouveau_score("fichier_test.res","Fred", 12)
nouveau_score("fichier_test.res","Omar", 14)
nouveau_score("fichier_test.res","Eric", 30)
nouveau_score("fichier_test.res","Ramzy",28)
nouveau_score("fichier_test.res","Ramzy",30)
nouveau_score("fichier_test.res","Omar", 13)
nouveau_score("fichier_test.res","Omar", 8)
nouveau_score("fichier_test.res","Omar", 7)
nouveau_score("fichier_test.res","Eric", 24)
nouveau_score("fichier_test.res","Jamel",20)
nouveau_score("fichier_test.res","Jamel",30)
```

3. Écrire une fonction (sans argument) `lancer_jeu` qui dans l'ordre :
   - Souhaite la bienvenue au joueur et lui demande de rentrer son nom en console.
   - Demande au joueur une valeur pour `n`.
   - Lance le jeu du plus ou moins puis enregistre son score dans le fichier `"tableau_scores_<n>.res"` et affiche le meilleur score du joueur en console.
   - Demande une nouvelle valeur pour `n` et retourne au point précédent. On passe au point suivant lorsque la valeur donnée pour `n` est négative ou nulle.
   - Remercie le joueur d'avoir jouer.

   Testez votre fonction. En particulier, vérifiez que plusieurs fichiers sont créés pour différentes valeurs de `n`.

Exercices à rendre au plus tard le 26/11/2023 à 20h

Exercice 6.

Revoir le cours sur les dictionnaires et la complexité pour le prochain TP.

Exercice 7. Chiffrement par ROT13

Dans cet exercice, on s'intéresse au ROT13 qui est une technique de chiffrement simple (et peu sécurisée) pour les chaînes de caractères. Le principe est de remplacer chaque lettre par la lettre se trouvant 13 positions plus loin dans l'alphabet. Par exemple, `a` est remplacée par `n`, `b` est remplacée par `o` et `z` est remplacée par `m` (on considère que l'alphabet est cyclique). Les autres caractères comme l'espace, les lettres accentuées ou les majuscules ne sont pas modifiées par ROT13.

Afin d'implémenter ROT13, on numérote la lettre `a` par 0, le lettre `b` par 1 et ainsi de suite jusqu'à la lettre `z` qui est numérotée par 25.

1. À l'aide de la fonction `chr` de Python, écrire une fonction `num_to_lettre` qui prend en entrée un numéro entre 0 et 25 et renvoie la lettre correspondante. Faites des tests ou regardez l'aide pour comprendre comment utiliser la fonction `chr`.
2. À l'aide de la fonction `ord` de Python, écrire une fonction `lettre_to_num` qui soit la réciproque de la fonction `num_to_lettre` sur $[\![0, 25]\!]$. Faites des tests ou regardez l'aide pour comprendre comment utiliser la fonction `ord`.
3. Écrire une fonction `ROT13` qui prend en entrée une chaîne de caractères et renvoie la chaîne obtenue en utilisant le chiffrement ROT13. Vérifiez que `"bateau"` devient `"ongrnh"`, que `"rot13"` devient `"ebg13"` et que `"Mangez des pommes"` devient `"Mnatrm qrf cbzzrf"`.
4. Essayez d'appliquer le chiffrement ROT13 deux fois de suite à des chaînes de caractères de votre choix. Que constatez vous ? Le justifier.

Exercice 8. Dessins dans la console

Dans cet exercice, on utilisera des étoiles séparées par des espaces pour créer les chaînes de caractères demandées. On pourra supposer sans le vérifier que les arguments des différentes fonctions sont strictement positifs.

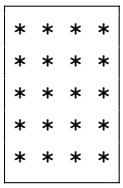

Figure 1

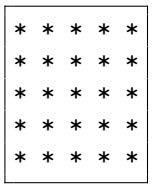

Figure 2

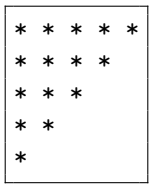

Figure 3

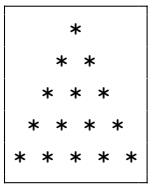

Figure 4

1. Écrire une fonction de signature « `ligne(n: int) -> str` » qui renvoie une chaîne de caractères composée de `n` étoiles séparées par des espaces. Par exemple `ligne(4)` vaut `"* * * *"` (sans espace au début, ni à la fin).

Dans la suite, les lignes des différents dessins devront être générées à l'aide de la fonction `ligne`.

2. Écrire une fonction de signature « `rectangle(l: int, h: int) -> str` » qui renvoie une chaîne de caractères contenant un rectangle dont le côté horizontal a pour largeur `l` et dont le côté vertical a pour hauteur `h`. Par exemple, l'appel à `print(rectangle(4,5))` doit afficher le dessin de la figure 1. Testez pour différentes valeurs de `l` et `h`.
3. Écrire une fonction de signature « `carre(c: int) -> str` » qui renvoie une chaîne de caractères contenant un carré de côté `c`. Par exemple, l'appel à `print(carre(5))` doit afficher le dessin de la figure 2. Testez pour différentes valeurs de `c`.
4. Écrire une fonction de signature « `triangle1(c: int) -> str` » qui renvoie une chaîne de caractères contenant un triangle comme dans l'exemple de la figure 3 où `c` vaut `5`. Testez pour différentes valeurs de `c`.
5. Écrire une fonction de signature « `triangle2(c: int) -> str` » qui renvoie une chaîne de caractères contenant un triangle comme dans l'exemple de la figure 4 où `c` vaut `5`. Testez pour différentes valeurs de `c`.

Exercice 9. Voies de Nancy (exercice facultatif)

Le fichier `20190828-nancy-215403957.csv` disponible sur le site :

`https://www.data.gouv.fr/fr/datasets/base-adresse-locale-ville-de-nancy/`

contient toutes les adresses de la ville de Nancy. On pourra utiliser la méthode `split` de l'exercice 3 pour le décomposer.
Remarque : un fichier `.csv` est en fait un fichier `.txt`, ouvrez le avec `Notepad++` ou bloc-note, pas avec Excel.
En utilisant ce fichier et Python, répondre aux questions suivantes :

1. Quelle est la plus petite latitude à Nancy ?
   Réponse attendue : 48.667339.
2. À quelle voie appartient l'adresse avec le plus grand numéro et quel est ce numéro ?
   Réponse attendue : 4710 sur le Quai Sainte-Catherine.
3. Quel est le nombre de voies à Nancy ?
   Réponse attendue : 626
4. Quelle est la voie avec le plus d'adresses différentes et combien y a-t-il d'adresses dans cette voie ? Dans cette question, on prend en compte les bis, ter et quater. Par exemple, le 1 et le 1 bis ne sont pas les mêmes adresses.
   Réponse attendue : Avenue de la Libération avec 267 adresses.
5. Quelle est la voie avec le plus de numéros différents et combien y a-t-il de numéros dans cette voie ? Dans cette question, on ne prend pas en compte les bis, ter et quater. Par exemple, le 1 et le 1 bis sont le même numéro.
   Réponse attendue : Avenue de la Libération avec 258 numéros.