Exercice 1

Questions à reponses courtes

1.1 fonctions et méthodes

Les **fonctions** que l'on peut utiliser avec des listes, des tuples ou des chaines str sont : `len` et `list`.

1. Avec la liste `L = ["a","b","c","d"]`, comment obtenir la longueur de la liste (le nombre d'éléments) ? Quelle sera la valeur retournée ?
2. Comment transformer la chaine de caractères "abcd" en une liste `["a","b","c","d"]` ?
3. Transformer cette liste en `["b","b","c","d"]`

Les **méthodes** associées aux listes sont : append, remove, pop, index, extend

Avec la liste `L = ["a","b","c","d"]` :

3. Quelle instruction permet d'accéder au **rang** de l'élément "c" ?
4. Quelle instruction permet de supprimer le dernier élément, "d" ?
5. Quelle instruction permet d'ajouter `"e"` à la fin de la liste ?

1.2 Découpage d'une Liste (slicing)

Le découpage de liste (*slicing*) permet d'extraire une séquence d'une liste. Soit `L` une liste. La syntaxe `L[i:j]` permet d'extraire tous les éléments consécutifs compris entre l'élément de rang `i` inclus et l'élément de rang `j` exclus (jusqu'à `j-1`).

On donne :

```
mois = ["Janvier", "Fevrier", "Mars", "Avril",
        "Mai", "Juin", "Juillet", "Aout", "Septembre"
        "Octobre", "Novembre", "Decembre"]
```

1. En utilisant la technique du *slicing* écrire une instruction permettant :

- d'extraire de la liste`mois` uniquement les 6 premiers mois.
- d'extraire les 2 mois d'été

2. Les objets créés, sont ils copiés de la liste d'origine par *valeur* ou bien par *référence* ?

1.3 Compréhension de liste

Quelle-s expression-s Python a-ont pour valeur la liste [2, 5, 8, 11] ?

- `[3 * i + 2 for i in range(3)]`
- `[3 * i + 2 for i in range(4)]`
- `[3 * i - 1 for i in range(1,5)]`
- `[3 * i - 1 for i in range(4)]`

Exercice 2

Traitement des éléments de liste

2.1 Additionner les valeurs

Le programme suivant ajoute les valeurs d'une liste dans une variable.

1. Compléter le script :

```
L = [1,2,3,4,5,6,7,8,9,10]
accumulateur = 0
for ... in ...
    accumulateur += ...
print(...
```

2. Ecrire une fonction `additionne` à partir de ce script.
3. Adapter le script de cette fonction pour qu'elle additionne les chiffres dans un mot, contenant uniquement des caractères compris entre "0" et "9" Exemple :

```
>>> additionne("01234")
10
```

2.2 Compteur d'espaces

Dans cet exercice, on considère des phrases composées de mots.

- On appelle « mot » une chaîne de caractères composée avec des caractères choisis parmi les 26 lettres minuscules ou majuscules de l'alphabet,
- On appelle *phrase* une chaîne de caractères :
  - composée avec un ou plusieurs *mots* séparés entre eux par un seul caractère espace `' '`,
  - se finissant :
    - soit par un point `'.'` qui est alors collé au dernier mot,
    - soit par un point d'exclamation `'!'` ou d'interrogation `'?'` qui est alors séparé du dernier mot par un seul caractère espace `' '`.

Voici deux exemples de phrases :

- 'Cet exercice est simple.'
- 'Le point d exclamation est separe !'

Après avoir remarqué le lien entre le nombre de mots et le nombres de caractères espace dans une phrase, programmer une **fonction** `nombre_de_mots` qui prend en paramètre une phrase et renvoie le nombre de mots présents dans cette phrase.

```
>>> nombre_de_mots('Cet exercice est simple.')
4
>>> nombre_de_mots('Le point d exclamation est séparé !')
6
>>> nombre_de_mots('Combien de mots y a t il dans cette phrase ?')
10
>>> nombre_de_mots('Fin.')
1
```

2.3 parcours de chaine de caractères, mot à trou

On considère des chaînes de caractères contenant uniquement des majuscules et des carac- tères * appelées *mots à trous*.

Par exemple `INFO*MA*IQUE`, `***I***E**` et `*S*` sont des mots à trous.

Programmer une fonction `correspond` qui :

- prend en paramètres deux chaînes de caractères `mot` et `mot_a_trous` où `mot_a_trous` est un mot à trous comme indiqué ci-dessus,
- renvoie :
  - `True` si on peut obtenir `mot` en remplaçant convenablement les caractères `'*'` de `mot_a_trous`.
  - `False` sinon.

Exemple :

```
>>> correspond('INFORMATIQUE', 'INFO*MA*IQUE')
True
>>> correspond('AUTOMATIQUE', 'INFO*MA*IQUE')
False
>>> correspond('STOP', 'S*')
False
>>> correspond('AUTO', '*UT*')
True
```

Exercice 3

Dictionnaires

3.1 Capitales du monde

```
capitales = {'France':'Paris', 'Italie':'Rome', 'Angleterre':'Londres'}
```

Quel est l'affichage dans la console pour chacune des instructions suvantes ?

```
# instruction 1
list(capitales.keys())
```

```
# instruction 2
list(capitales.values())
```

```
# instruction 3
list(capitales.items())
```

Dans chaque cas, choisir parmi :

- `['France':'Paris', 'Italie':'Rome', 'Angleterre':'Londres']`
- `[('France', 'Paris'), ('Italie', 'Rome'), ('Angleterre', 'Londres')]`
- `['Paris', Rome', 'Londres']`
- `['France', 'Italie', 'Angleterre']`

3.2 Index telephonique

On définit :

```
contacts = {'Toto': 'toto@nsi.fr', 'Chloé': 'chloe@nsi.com','Paul': '
   paul@nsi.net', 'Clémence': 'clemence@nsi.org' }
```

a. Parmi les propositions suivantes, laquelle est exacte ?

- 'Chloé' est une valeur de la variable contacts
- 'Chloé' est une clé de la variable contacts
- 'Chloé' est un attribut de la variable contacts
- 'Chloé' est un champ de la variable contacts

b. pour le dictionnaire `contacts` de la question précédente :

- Comment accède t-on à la valeur `'clemence@nsi.org'` ?
- Comment créé-t-on une nouvelle entrée pour Léa, `lea@nsi.org` dans ce dictionnaire ?
- Comment obtenir à l'aide d'une instruction, la liste de tous les contact (et pas leur mail) ?
- Comment obtenir la liste de tous les mails ?

c. On définit la liste de dictionnaires :

```
repertoire = [{'nom': 'Francette', 'poste': 412},{'nom': 'Jeanne', '
   poste': 222},{'nom': 'Martine', 'poste': 231}]
```

Quelle expression permet d'accéder au poste Martine ?

Exercice 4

Tableau

Un tableau contenant les prenoms et ages est représenté par la liste `L` suivante :

```
L = [('Theo', 12), ('Lidia', 15), ('Emilie', 15), ('Vincent', 17)]
```

1. QCM : Quelle est la valeur de l'expression `[element[0] for element in L if element[1] >=15]`

- `['Lidia']`
- `['Vincent']`
- `['Lidia', 'Emilie', 'Vincent']`
- `('Lidia', 'Emilie', 'Vincent')`

2. Ecrire le script python utilisant une boucle `for` correspondant à cette écriture en compréhension. (écriture en plusieurs lignes).
3. Ecrire l'instruction qui permettra d'ajouter `('Richard',3)` à la fin de la liste `L`
4. Après ces modifications, on écrit : `L.pop()`. Qu'est ce qui est retourné dans la console Python ? Quel est alors le contenu de la liste `L` ?

Exercice 5

Echiquier et compréhension de Liste

Chaque case d'un échiquier peut être représentée par le tuple (colonne, ligne). Par exemple, la première case noire en bas à gauche est ('A', '1').

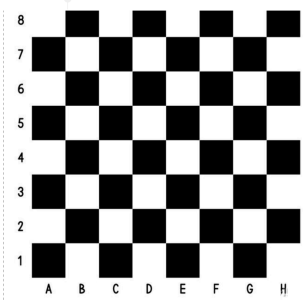

FIGURE 1 – Echiquier 64 cases

Les instructions suivantes permettent de construire les listes à partir des chaines de caractères.

```
>>> lign = list('ABCDEFGH')
>>> col = list('12345678')
>>> lign
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
>>> col
['1', '2', '3', '4', '5', '6', '7', '8']
```

On souhaite créer une liste avec les 64 tuples possibles, par combinaison de (lettre, chiffre).

1. Ecrire une série d'instructions utilisant 2 boucles bornées pour créer cette liste `T1`

2. Quelle instruction python permet cette fois de générer `T1`en compréhension de liste ?

```
>>> T1 = [(lign[i],col[i]) for i in range(len(col))]
>>> T1 = [(l,c) for l in lign for c in col]
```

3. Pour représenter l'échiquier on peut utiliser un dictionnaire : les clés seront les tuples représentant chacune des cases, les valeurs, la piece qui occupe l'echiquier. L'absence de piece sur une case sera représentée par la valeur `None`.

`{('A',1): None, ('B',1): None,...}`

Ecrire l'-les instructions qui génère-nt le dictionnaire `Echiquier`, vide de toute pièce.

5.1 Découpage d'une Liste (slicing)

Le découpage de liste (*slicing*) permet d'extraire une séquence d'une liste. Soit `L` une liste. La syntaxe `L[i:j]` permet d'extraire tous les éléments consécutifs compris entre l'élément de rang `i` inclus et l'élément de rang `j` exclus (jusqu'à `j-1`).

On donne :

```
mois = ["Janvier", "Fevrier", "Mars", "Avril",
        "Mai", "Juin", "Juillet", "Aout", "Septembre"
        "Octobre", "Novembre", "Decembre"]
```

1. On fait une copie de `mois` de la manière suivante :

```
months_in_french = mois
```

On modifie `months_in_french` avec l'instruction suivante :

```
months_in_french[10] = "Decembre"
```

2. Que vaut la liste 'months_in_french ?

- `["Janvier", ..,"Octobre", "Novembre", "Decembre"]`
- `["Janvier", ..,"Octobre", "Decembre", "Decembre"]`

3. Que vaut la liste `mois` ? (Justifier)

- `["Janvier", ..,"Octobre", "Novembre", "Decembre"]`
- `["Janvier", ..,"Octobre", "Decembre", "Decembre"]`

Exercice 6

Parcours d'un tableau

On dispose d'une table `tab` constituée d'une liste de trois sous-listes (un tableau). Chacune des sous-listes contient quatre caractères.

```
tab=[['A', 'B', 'C', 'D'],
     ['E', 'F', 'G', 'H'],
     ['I', 'J', 'K', 'L'] ]
```

a. Parmi les propositions suivantes, laquelle permet de convertir cette table en une liste L contenant dans l'ordre, ligne par ligne, les 12 caractères de tab ?

script A :

```
L = []
for i in range(3):
  for j in range(4):
    L.append(tab[i][j])
```

script B :

```
L = []
for i in range(4):
  for j in range(3):
    L.append(tab[j][i])
```

Exercice 7

Copie de Listes

a. Que vaut L1 après le script suivant ?

```
L1 = [1,2,3,4]
L2 = L1
L2.pop()
```

b. Que vaut L1 après le script suivant ?

```
L1 = [1,2,3,4]
L2 = list(L1)
L2.append(5)
```

c. Que vaut L1 après le script suivant ?

```
L1 = [1,2,3,4]
L2 = L1.copy()
L2.append(5)
```