

# Première NSI - types construits

Travaux dirigés

qkzk

2022/07/05

## Se tester

### 1. Tuples et listes

1. Parmi les affirmations suivantes lesquelles sont vraies ?
  - a. Une liste peut contenir plusieurs éléments.
  - b. On peut ajouter des éléments à un tuple.
  - c. On peut modifier les éléments d'un tuple.
  - d. Un tuple peut contenir à la fois des nombres et des chaînes de caractères.
2. Si `liste` est la liste `[1, 3, 5]`, quelles sont les opérations valides ?
  - a. `liste.append(4)`
  - b. `liste[0]`
  - c. `liste[0] = 4`
  - d. `liste[4] = 7`
  - e. `liste = [1, 3, 10, 7, 3]`
3. Si `triplet` est le tuple `(1, 3, 5)`, quelles sont les opérations valides ?
  - a. `triplet.append(4)`
  - b. `triplet[0]`
  - c. `triplet[0] = 4`
4. Associer correctement les collections à leur description :

Collection	Description
<code>list</code>	tableau mutable d'associations de paires de clé et valeur
<code>dict</code>	collection ordonnée et non mutable d'éléments de tout type
<code>tuple</code>	collection ordonnée et non mutable de caractères
<code>str</code>	collection ordonnée mutable d'éléments de tout type

### 2. Dictionnaires

```
dico = {"a": True, "b": False, "c": False}
```

1. Quelle est la valeur de `dico[1]` ?
2. Quelle est la valeur de `dico["a"]` ?
3. Quelle instruction permet de modifier le dictionnaire de façon à ce que sa nouvelle valeur soit :

```
{"a": True, "b": False, "c": False, "e": True}
```

4. Qu'affichent les instructions suivantes ?

```
for cle in dico.keys():
    print(cle, end=" ")
```

**Remarque:** Par défaut la fonction `print` termine l'affichage par un retour à la ligne (le caractère `\n`). On peut modifier ce comportement en ajoutant `end=" "`, cela remplace le retour à la ligne par un espace.

5. Quels sont les affichages possibles lors de l'exécution du code suivant ?

```
for truc in dico.items():
    print(truc, end=" ")
```

### 3. Compréhension et structures imbriquées

1. Si `liste` désigne la liste `[1, [2, 3], [4, 5], 6, 7]` que faut `len(liste)` ?
2. Que vaut `[2 * n for n in range(5)]` ?
3. Supposons que `liste = [-5, 2, 3, -7, 42, 7]`  
Que vaut `[n for n in liste if n > 0]` ?

## S'entraîner

### 4. Comprendre les tuples et les listes

On donne le script python :

```
premiers = [2, 3, 5, 7]
couple = (7, 4)
i = 3
a = premiers[i]
```

1. Quelle est la valeur de `premiers[2]`
2. Après l'exécution de `couple.append(1)`, quelle sera la valeur de `couple` ?
3. Après l'instruction `premiers[3] = 11` quelle sera la valeur de `premiers`.

### 5. Détecter les erreurs dans les boucles et les listes.

Tous ces codes sont faux. Décrire les problèmes et les rectifier afin qu'ils réalisent ce qu'on attend.

1. Afficher les 10 premiers nombres pairs.

```
for i in range(10):
    if i % 2 == 0:
        print(i)
```

2. Ajouter les éléments 3, 4 et 5 à `ma_liste`.

```
ma_liste = (1, 2)
ma_liste.append(3)
ma_liste.append(4)
ma_liste.append(5)
```

3. Construire la liste des voyelles de la phrase. Chaque voyelle ne doit figurer qu'une fois au plus dans la liste finale.

```

voyelles = "aeiouy"
ma_phrase = "salut les jeunes, aujourd'hui je vais marcher et manger neuf pizzas."
voyelles_presentes = []
for lettre in ma_phrase:
    if lettre in voyelles:
        voyelles_presentes.append(lettre)

```

4. Calculer un capital après 10 années. Henri a déposé 1000€ sur son compte. Les intérêts sont annuels et composés, on augmente de 5% chaque année le montant du capital. Ensuite on affiche le montant du capital atteint.

```

capital = 1000
for annee in range(10):
    capital = 1.05 * capital
print(capital)

```

5. Compter le tours de boucles jusqu'à avoir atteint une valeur donnée et l'afficher.

```

seuil = 200
nombre_de_tours = 0

while valeur < seuil:
    nombre_de_tours += 1
    valeur = 1.05 * valeur + 3

print("Il a fallu", nombre_de_tours, "pour atteindre", seuil)

```

## 6. Manipulation de dictionnaires

1. Construire le dictionnaire correspondant au tableau suivant qui décrit certaines caractéristiques d'une ville. On utilisera la variable `ville`.

clé	valeur
nom	Granville
population (hab)	30.000
superficie (km <sup>2</sup> )	15
altitude (m)	252
préfecture	non

Quel choix semble pertinent pour enregistrer le fait que Granville n'est pas une préfecture ?

On proposera deux syntaxes différentes :

- en créant directement le dictionnaire,
- en ajoutant un par un les éléments à partir d'un dictionnaire vide.

2. On suppose disposer de la variable `ville` en mémoire. Écrire une expression booléenne pour tester les affirmations suivantes. Évaluer l'affirmation.

Par exemple pour tester l'affirmation : "le nom de la ville est 'Paris'", on peut écrire : `ville["nom"] == "Paris"` dont la valeur est `False`.

1. Le nom de la ville ne comporte pas la lettre "e".
2. La population de la ville dépasse 20.000 habitants.
3. La ville est une préfecture.
4. La densité (habitants divisé par superficie) est supérieure à 400 habitants au kilomètre carré.

3. Boucles et dictionnaires.

1. À l'aide d'une boucle, afficher toutes les clés du dictionnaire.

2. À l'aide d'une boucle, construire la liste des valeurs du dictionnaire.
3. À l'aide d'une boucle construire l'affichage suivant :

```
Pour la clé : nom , la valeur correspondante est : Granville
Pour la clé : population , la valeur correspondante est : 30000
Pour la clé : superficie , la valeur correspondante est : 15
Pour la clé : altitude , la valeur correspondante est : 252
Pour la clé : préfecture , la valeur correspondante est : False
```

## 7. Modéliser des Pokémons avec des dictionnaires et des tuples

On modélise des informations (nom, taille et poids) sur des Pokémons de la façon suivante :

```
exemple_pokemons = {'Bulbizarre': (70, 7), 'Herbizarre': (100, 13),
                    'Ago': (200, 7), 'Jungko': (170, 52)}
```

Par exemple, Bulbizarre est un Pokémon qui mesure 70 cm et pèse 7 kg.

1. Quel est le type de `exemple_pokemons` ?
2. Quelle instruction permet d'ajouter à cette structure de donnée le Pokémon Goupix qui mesure 60 cm et qui pèse 10 kg ?
3. On donne le code suivant :

```
def le_plus_grand(pokemons):
    grand = None
    taille_max = None
    for (nom, (tailles, poids)) in pokemons.items():
        if taille_max is None or taille > taille_max:
            taille_max = taille
            grand = nom
    return (grand, taille_max)
```

- a. Quelle est la valeur de `le_plus_grand(pokemons)` ?
- b. Écrire le code d'une fonction `le_plus_léger` qui prend des Pokémon en paramètre et qui renvoie un tuple dont la première composante est le nom du Pokémon le plus léger et la deuxième est son poids.

```
assert le_plus_léger(pokemons) == ('Bulbizarre', 7)
```

4. Écrire le code d'une fonction `taille` qui prend en paramètre un dictionnaire de Pokémon ainsi que le nom d'un Pokémon, et qui renvoie la taille de ce Pokémon.

```
assert taille(exemple_pokemons, 'Abo') == 200
assert taille(exemple_pokemons, 'Jungko') == 170
assert taille(exemple_pokemons, 'Dracaufeu') == None
```

## 8. Construire des listes en compréhension

1. Parmi les extraits de programme suivants, lesquels permettent de construire la liste des cinq premiers nombres impairs ?

```
impairs1 = [1, 3, 5, 7, 9]
```

```
impairs2 = []
for n in range(5):
    impairs.append(2 * n + 1)
```

```
impairs3 = [2 * n + 1 for n in range(5)]
```

```
impairs4 = [n for n in range(1, 11, 2)]
```

```
impairs5 = []
n = 0
while len(impairs) != 5:
    if n % 2 == 1:
        impairs.append(n)
    n = n + 1
```

2. Donner plusieurs programmes permettant de construire la liste des 25 premiers multiples de 7.
3. Construire par compréhension la liste  $[31 ** 2, 41 ** 2, 51 ** 2, \dots, 1311 ** 2, 1321 ** 2]$
4. a. Quelle est la valeur de `couples` à la fin de l'exécution du programme suivant ?

```
lettres = ["a", "b", "c"]
nombres = [1, 5]
couples = [(c, n) for c in lettres for n in nombres]
```

- b. Proposer un programme qui permet de construire `couples` en utilisant des boucles bornées.

## 9. Vérifier un carré magique

Un **carré d'ordre**  $n$  est un tableau carré contenant  $n^2$  entiers strictement positifs.

On dit qu'un carré d'ordre  $n$  est **magique** si :

- il contient tous les nombres entiers de 1 à  $n^2$
- les sommes des nombres de chaque rangée, de chaque colonne et de chaque diagonale principale sont égales.

On modélise un carré par une liste de listes de nombres. Exemples :

```
carre3 = [
    [2, 7, 6],
    [9, 5, 1],
    [4, 3, 8]
]
```

```
carre4 = [
    [ 4,  5, 11, 14],
    [15, 10,  8,  1],
    [ 6,  3, 13, 12],
    [ 9, 16,  2,  7]
]
```

1. a. Quelle est la valeur de `len(carre4)` ?  
b. Quelle est la valeur de `carre3[1]` ?  
c. Quelle est la valeur de `carre3[0][2]` ?  
d. Quelle instruction permet de récupérer la valeur 3 de `carre4` ?

2. a. On propose le code suivant :

```
def somme_ligne(carre: list, n: int) -> int:
    """
    carre un tableau carré de nombres
    n est un nombre entier
    """
    somme = 0
    for nombre in carre[n]:
        somme = somme + nombre
    return somme
```

Que vaut `somme_ligne(carre4, 2)` ? À quoi sert cette fonction ?

- b. Écrire le code d'une fonction qui prend un carré en paramètre et qui vérifie que les sommes de nombres de chaque ligne sont égales.
- c. Proposer le code d'une fonction d'une fonction qui prend un carré en paramètre ainsi que le numéro d'une colonne, et qui renvoie la somme des nombres de cette colonne.

**Pour aller plus loin :** écrivez une fonction `est_magique` qui prend un carré en paramètre et renvoie `True` si le carré est magique et `False` sinon.

## 10. Manipuler des listes de nombres

On donne le code de la fonction suivante :

```
def mystere(liste: list) -> bool:
    """
    Paramètre : une liste d'éléments comparables
    """
    for i in range(1, len(liste)):
        # ICI
        if liste[i - 1] > liste[i]:
            return False
    return True

assert mystere([1, 2, 6, 9])
assert not mystere([1, 6, 2, 9])
```

1. Compléter le tableau avec les valeurs des expressions à chaque passage par la ligne indiquée par le commentaire # ICI lors de l'appel à `mystere([1, 2, 6, 9])`

i	i-1	liste[i - 1]	liste[i]

2. Compléter le tabeauu avec les valeurs des expressions à chaque passage par la ligne indiquée par le commentaire # ICI lors de l'appel à `mystere([1, 6, 2, 9])`

i	i-1	liste[i - 1]	liste[i]

3. `a = mystere(['kiwi', 'pomme', 'ananas', 'framboise'])`  
`b = mystere(['ananas', 'framboise', 'kiwi', 'pomme'])`

Quelle est la valeur de a ? De b ?

Que fait la fonction `mystere` ?

## 11. Itérer sur les éléments d'un dictionnaire

Au zoo de Beauval, il y a 5 éléphants d'Asie, 17 écureuils d'Asie, 2 pandas d'Asie...

On représente cet inventaire à l'aide d'un dictionnaire, de la façon suivante :

```
zoo_beauval = {
    'elephant': ('Asie', 5),
    'écureuil': ('Asie', 17),
    'panda': ('Asie', 2)
    'hippopotame': ('Afrique', 7),
    'girafe': ('Afrique', 4)}
```

On représente de la même façon le zoo de la Flèche :

```
zoo_la_fleche = {
    'ours': ('Europe', 4),
    'tigre': ('Asie', 7),
    'girafe': ('Afrique', 11),
    'hippopotame': ('Afrique', 3)}
```

1. On souhaite se doter d'une fonction `plus_grand_nombre` qui prend un zoo en paramètre et qui renvoie le nom de l'animal le plus représenté dans ce zoo.

Par exemple :

```
assert plus_grand_nombre(zoo_la_fleche) == 'girafe'
assert plus_grand_nombre(zoo_beauval) == 'écureuil'
```

- a. Quel type de boucle peut-on envisager pour le code de cette fonction ?

```
for cle in dico.keys()
```

```
for valeur in dico.values()
```

```
for (cle, valeur) in dico.items()
```

aucune boucle.

- b. Écrire le code de cette fonction.

2. On souhaite se doter d'une fonction `nombre_total` qui prend un zoo en paramètre ainsi que le nom d'un continent, et qui renvoie le nombre d'animaux originaires de ce continent dans le zoo. Par exemple :

```
assert nombre_total(zoo_beauval, 'Asie') == 24
assert nombre_total(zoo_la_fleche, 'Afrique') == 14
```

- a. Quel type de boucle peut-on envisager pour le code de cette fonction ?

- b. Écrire le code de cette fonction.

3. On souhaite se doter d'une fonction `nombre` qui prend un zoo en paramètre ainsi que le nom d'un animal, et qui renvoie le nombre de représentants de cet animal dans le zoo. Par exemple :

```
assert nombre(zoo_la_fleche, 'panda') == 0
assert nombre(zoo_beauval, 'panda') == 2
```

- Quel type de boucle peut-on envisager pour le code de cette fonction ?
- Écrire le code de cette fonction.

## Objectif BAC

Cet exercice propose plusieurs modélisations pour représenter les notes des élèves d'une classe.

### Le sujet

#### Partie 1 : Modélisation simpliste

On modélise les notes d'une élève de la façon suivante :

```
notes_de_lea = [12, 14, 3, 16, 17, 2, 13, 19]
```

- Quel est le type de `notes_de_lea` ?
- Que vaut l'expression `notes_de_lea[2]` ?
- Quelle instruction permet d'ajouter une note de 15 à cette structure de données ?
- On propose le code suivant :

```
def fonction(liste_de_notes):
    """
    `liste_de_notes` est une liste de nombres qui modélise les notes d'un
    élève
    Cette fonction renvoie ???
    """
    compteur1 = 0
    compteur2 = 0
    for note in liste_de_notes:
        if note >= 10:
            compteur1 = compteur1 + 1
        else:
            compteur2 = compteur2 + 1
    return (compteur1, compteur2)

notes_de_lea = [12, 14, 3, 16, 17, 2, 13, 19]
assert fonction(notes_de_lea) == ???
```

- Quel est le type de retour de cette fonction ?
- Recopier et compléter la dernière de ce code.
- Recopier et compléter la dernière ligne de la documentation. On demande ici d'expliquer en quelques mots ce que fait la fonction.

#### Partie 2 : Modélisation avec une structure de données imbriquées

La modélisation précédente n'est pas satisfaisante si l'on veut conserver les notes de plusieurs élèves dans une même structure de données.

On propose, dans cette partie, de modéliser les notes des élèves de la façon suivante :

```
notes_de_la_classe = [('Enzo', 3), ('Emma', 16), ('Lucas', 14), ('Manon', 13)]
```

- Quel est le type de `notes_de_la_classe` ?

2. Que vaut l'expression `notes_de_la_classe[2]` ?
3. Quelle instruction permet d'ajouter à cette structure de données, une note de 15 obtenue par Farid ?
4. On veut écrire une fonction `nom_du_genie` qui prend une telle structure de données en paramètre et qui renvoie le nom de l'élève qui a eu la meilleure note.
  - a. Proposer un test pour cette fonction.
  - b. On donne le code mélangé de cette fonction. À vous de le remettre dans l'ordre.

```

        note_max = note
    note_max = None
def nom_du_genie(les_notes):
    return genie
        genie = nom
    genie = None
        if note_max == None or note > note_max:
    for (nom, note) in les_notes:

```

- c. Que vaut l'expression `nom_du_genie([])` ?

### Partie 3 : Une modélisation plus complète.

Dans cette partie, on souhaite modéliser dans une même structure de données les notes des élèves d'une classe en précisant le nom de la matière concernée par la note. On propose la modélisation suivante :

```

notes = {'Enzo': ('Math', 3), 'Emma': ('Math', 16),
         'Lucas': ('NSI', 14), 'Manon': ('NSI', 13)}

```

1. Quel est le type de `notes` ?
2. Que vaut l'expression `notes[2]` ?
  - a. 14
  - b. Lucas
  - c. ('NSI', 14)
  - d. 3
  - e. Cette expression génère une erreur.
3. Quelle instruction permet d'ajouter la note de 15 obtenue par Farid en NSI ?
4. Quel est l'affichage généré par l'exécution du code suivant ?
 

```
python for (nom, (matiere, note)) in notes.items():
    if note < 15:
        print(nom)
```
5. On veut écrire une fonction qui prend une telle structure de données en paramètre et qui renvoie le nom de l'élève qui a eu la moins bonne note, toutes matières confondues.
  - a. Proposer un test pour cette fonction.
  - b. Écrire le code de cette fonction.
6. On veut écrire une fonction `tri_par_matiere` qui prend une telle structure de données en paramètre et qui renvoie un dictionnaire dont les clés sont les noms des matières et les valeurs les notes obtenues par les élèves dans chaque matière.

*Exemple :*

```

>>> notes = {'Emma': ('Math', 16), 'Lucas': ('NSI', 14), 'Manon': ('NSI', 13)}
>>> tri_par_matiere(notes)
{'Math': [16], 'NSI': [14, 13]}

```

Écrire le code de cette fonction.

## La feuille de route

### Partie 1 : Modélisation simpliste

1. Les symboles utilisés (crochets [], parenthèses (), accolades { }) devraient vous aider.
2. Souvenez-vous que l'indexation commence à 0.
3. Lire le code d'une fonction.
  - b. Il s'agit ici de compléter un test pour cette fonction.
  - c. Il s'agit ici de compléter la documentation de cette fonction.

### Partie 2 : Modélisation avec une structure de données imbriquées

4. b. **Remettre en ordre un code mélangé**  
Le code est donné. L'indentation devrait vous aider à remettre les lignes dans l'ordre.

### Partie 3 : Une modélisation plus complète

5. **Écrire le code d'une fonction**  
Vous pouvez proposer comme test une instruction qui commence par **assert**.  
Inspirez-vous des codes proposés dans les autres questions.