

## ☞ Terminale Spé NSI : Évaluation en classe n°2 ☞

### EXERCICE 1

On donne ci-dessous le code d'une fonction en langage Python

```
1 def f(t):  
2     for i in range(len(t), 0, -1):  
3         print(t[i])  
4     return None
```

1. Quelle propriété doit vérifier la valeur de la variable nommée `t` pour que la boucle `for` fonctionne?

Donnez trois exemples de types construits Python possibles pour cette valeur.

2. Écrivez une spécification pour cette fonction.
3. Voici la copie d'une console Python :

```
>>> help(isinstance)  
Help on built-in function isinstance in module builtins:  
  
isinstance(obj, class_or_tuple, /)  
    Return whether an object is an instance of a class or of a subclass thereof.  
  
    A tuple, as in `isinstance(x, (A, B, ...))`, may be given as the target to  
    check against. This is equivalent to `isinstance(x, A) or isinstance(x, B)  
    or ...` etc.  
  
>>> |
```

En vous appuyant sur cette documentation et en utilisant soit l'opération `raise` soit un bloc `try` : puis `except`, écrivez un test de conformité de la valeur passée par une personne utilisatrice de la fonction nommée `f`.

Précisez où insérer ce test dans le code de cette fonction.

4. Analyse du code.
  - (a) Donnez un exemple de valeur passée en argument pour laquelle le code de l'énoncé va déclencher une erreur `IndexError`.  
Expliquez la raison de cette erreur.
  - (b) Ce code est-il correct? Autrement dit, réalise-t-il ce qui est attendu? Expliquez.
  - (c) Proposez une solution pour coder correctement cette fonction.

### EXERCICE 2

Dans cet exercice, on s'intéresse à une fonction dont la signature est

```
partageTableau(tab : list) -> (list, list):
```

qui partage le tableau nommé `tab` en deux sous-tableaux, celui des éléments de `tab` d'indices pairs et celui de ceux d'indices impairs. Cette fonction renvoie ces deux sous-tableaux sous forme d'un tuple.

1. À partir de cette seule spécification, écrivez un jeu cohérent de valeurs qui permet de tester cette fonction.
2. Le code qui suit prétend être une implémentation de cette fonction.

```

1 def partageTableau(tab : list) -> (list, list):
2     assert isinstance(tab, list),\
3         "La valeur passée en argument doit être de type list."
4     pairs = []
5     impairs = []
6     i = 0
7     while i != len(tab):
8         pairs.append(tab[i])
9         impairs.append(tab[i + 1])
10        i = i + 2
11    return pairs, impairs

```

- Expliquez quels sont vos tests qui ne passent pas et pourquoi.
- Proposez une correction en adaptant le code de l'énoncé sans en modifier fondamentalement l'algorithme.

### EXERCICE 3

On donne ci-dessous une interface minimale pour une structure de dictionnaire.

Fonction	Description
creeDico()	Crée et renvoie un dictionnaire vide.
estClef(k, d)	Renvoie <code>True</code> si et seulement si la valeur de <code>k</code> est une clef du dictionnaire <code>d</code> . Renvoie <code>False</code> sinon.
litDansDico(k, d)	Renvoie la valeur associée à la clef <code>k</code> dans le dictionnaire <code>d</code> si cette clef est présente. Renvoie <code>None</code> dans le cas contraire.
ecritDansDico(k, v, d)	Ajoute dans le dictionnaire <code>d</code> l'association entre la clef <code>k</code> et la valeur <code>v</code> en remplaçant éventuellement une association existante si la clef <code>k</code> est déjà présente.

Il s'agit, dans un premier temps, de réaliser en partie cette interface de dictionnaire sans utiliser le type `dict` de Python mais avec un tableau dynamique, donc une valeur de type `list`, dont les éléments sont des tuples (`k`, `v`) et en faisant en sorte qu'aucune clef n'apparaisse deux fois.

#### 1. Un peu de réalisation.

- Complétez le code de la fonction `creeDico()` ci-dessous.

```

1 def creeDico():
2     """Cette fonction crée et renvoie un dictionnaire vide."""
3     return .....

```

- Complétez le code de la fonction `ecritDansDico(k, v, d)` ci-dessous.

Ajoutez des commentaires pour renseigner les étapes de l'algorithme mis en œuvre.

```

1 def ecritDansDico(k, v, d) -> None:
2     """Cette fonction ajoute l'association (k, v) dans le dictionnaire
3         d si la clef k n'y est pas présente, sinon elle remplace la
4         valeur associée à cette clef.
5     """

```

```

6   for i in range(len(d)):
7       if d[i][0] == k:
8           .....
9           .....
10          .....
11  return None

```

2. On suppose maintenant qu'un module intitulé dictionnaire et offrant l'interface présentée en introduction, existe dans un répertoire de fichiers en tant que fichier nommé `dictionnaire.py`.

L'objectif de cette question est d'implémenter dans un deuxième module l'algorithme du tri par comptage en utilisant les fonctions offertes par l'interface du module `dictionnaire`.

Le principe du tri par comptage d'un tableau de nombres entiers est le suivant.

- ① On recherche la valeur minimale et la valeur maximale dans le tableau.  
Par exemple avec `[98, 27, 12, 27]` on trouve 12 et 98.
- ② On complète un dictionnaire dont les clefs sont les valeurs entières présentes dans le tableau et les valeurs en sont les nombres d'occurrences.  
Par exemple avec `[98, 27, 12, 27]` on trouve `{98:1, 27:2, 12:1}`.
- ③ On parcourt toutes les valeurs comprises entre le minimum et le maximum en exploitant le dictionnaire pour produire et renvoyer un tableau comportant toutes les valeurs du tableau initial triées en ordre croissant.  
Par exemple avec `[98, 27, 12, 27]` on trouve `[12, 27, 27, 98]`.

On se propose de créer ce module en l'enregistrant sous le nom `triParComptage.py` dans le même répertoire de fichiers que le fichier `dictionnaire.py`.

L'interface de ce module `triParComptage` ne comportera qu'une seule fonction, les autres seront donc encapsulées.

- (a) Écrivez la *docstring* de ce module ainsi que l'importation des fonctions de l'interface du module `dictionnaire`.
- (b) Écrivez l'interface de ce module.
- (c) On admet que ce module contient déjà une fonction qui renvoie un tuple dont les deux valeurs sont le minimum et le maximum d'un tableau, de type `list`, passé en argument.  
Proposez un nom et une signature pour cette fonction.
- (d) Complétez la signature et le code de la fonction de ce module qui implémente le point ② de l'algorithme du tri par comptage.

```

1  def .....:
2      """Cette fonction renvoie un dictionnaire dont les clefs sont les
3          valeurs des entiers présents dans tab et dont les valeurs sont
4          les occurrences de ces entiers dans tab.
5      """
6      dico = .....
7      for .....:
8          if .....:
9              occ = .....
10         .....

```

```
11     else:
12         .....
13     return dico
```

(e) Complétez la signature et le code de la fonction de ce module qui implémente le point ③ de l’algorithme du tri par comptage.  
Commentez le code.







```
14 def .....:
15     """Cette fonction renvoie un nouveau tableau formé des
16         valeurs du tableau tab triées par ordre croissant.
17         Les valeurs de tab sont des nombres entiers.
18     """
19     result = .....
20     if .....: # si tableau non vide
21         dico_occ = .....
22         mini, maxi = .....
23         for .....:
24             .....
25             .....
26             .....
27             .....
28     return result
```

(f) Proposez cinq assertions bien choisies permettant de contrôler l’implémentation de la fonction de l’interface.

Nom :Prénom :

\*\*\*\*\*

17/10/2023 - Modularité, tests, gestion de bugs

Réf	Intitulé	Code
Données	Implémenter et exploiter	
Algorithmique et programmation	Décomposer et recomposer	
Algorithmique et programmation	Anticiper et tester	
Algorithmique et programmation	Évaluer un script	
Algorithmique et programmation	Décrire et spécifier	
Communiquer à l’écrit	Développer une argumentation	
Communiquer à l’écrit	S’exprimer avec clarté et précision	