

☞ Terminale Spé NSI : Évaluation en classe n°1 ☞

EXERCICE 1

On rappelle l'interface d'une structure de File.

Fonction	Description
<code>creer_file()</code>	Renvoie une file vide Exemple : <code>maFile = creer_file()</code>
<code>est_vide(f)</code>	Renvoie <code>True</code> si la valeur de la variable <code>f</code> est une file vide. Elle renvoie <code>False</code> si la valeur de <code>f</code> n'est pas vide.
<code>nb_elements(f)</code>	Renvoie le nombre d'éléments de la file. Exemple : <code>n = nb_elements(maFile)</code>
<code>enfiler(e, f)</code>	Enfile la valeur de <code>e</code> dans la file nommée <code>f</code> . Exemple : <code>enfiler("vendredi", maFile)</code>
<code>consulter(f)</code>	Renvoie la valeur de la tête de la file nommée <code>f</code> si celle-ci n'est pas vide. Produit une erreur d'assertion sinon. Exemple : <code>s = consulter(maFile)</code>
<code>defiler(f)</code>	Renvoie la tête de la file nommée <code>f</code> et la supprime, dans le cas où la file n'est pas vide. Produit une erreur d'assertion sinon. Exemple : <code>defiler(maFile)</code>
<code>afficher(f)</code>	Renvoie une chaîne de caractères composée des éléments de la file nommée <code>f</code> écrits de gauche à droite depuis la queue jusqu'à la tête. La tête est donc l'élément affiché le plus à droite. Les éléments sont séparés par une virgule. Si la file est vide la fonction renvoie <code>'file vide'</code> .

1. Questions relatives à cette interface de File.

- Écrivez une série d'instructions permettant de créer une File nommée `mf` et d'y enfiler successivement les valeurs 5, 7 et 11.
- Que renvoie alors l'instruction `afficher(mf)` ?
- Que se passe-t-il si on exécute ensuite l'instruction `t = defiler(mf)` ?

2. Questions portant sur l'implémentation.

Pour réaliser l'implémentation de cette structure de File, on a choisi un tableau dynamique Python dans lequel la tête de la File est toujours le dernier élément du tableau.

- Complétez le code du constructeur ci-dessous :

```
1 def creer_file() -> list:
2     """Renvoie une file vide"""
3     return .....
```

- Complétez les codes des deux accesseurs ci-dessous :

```
1 def est_vide(f : list) -> bool:
2     """Renvoie True si la file f est vide, false sinon."""
3     return .....
```

```

1 def consulter(f : list):
2     """Renvoie la valeur de la tête de la file f."""
3     # pré-condition
4     assert not est_vide(f), "La file est vide."
5     # traitement
6     return .....

```

(c) Complétez enfin les codes des deux opérateurs ci-dessous :

```

1 def defiler(f : list) -> None:
2     """Renvoie la valeur de la tête de la file f et la supprime...
3     ....
4     """
5     # pré-condition
6     assert not est_vide(f), "La file est vide."
7     # traitement
8     return .....

```

```

1 def enfiler(e, f : list) -> None:
2     """Enfile e dans la file f.
3     La file f est modifiée en place.
4     """
5     # ajout en fin du tableau
6     .....
7     # décalage d'un rang vers la gauche
8     for .....:
9         .....
10        .....
11        .....
12    return None

```

EXERCICE 2

On dispose d'un fichier nommé `structure_pile.py` qui contient les fonctions suivantes.

Fonction	Description
<code>creer_pile()</code>	Renvoie une pile vide Exemple : <code>maPile = creer_pile()</code>
<code>est_vide(pile)</code>	Renvoie <code>True</code> si la valeur de la variable <code>pile</code> est une pile vide. Elle renvoie <code>False</code> si la valeur de <code>pile</code> n'est pas vide.
<code>nb_elements(pile)</code>	Renvoie le nombre d'éléments de la pile. Exemple : <code>n = nb_elements(maPile)</code>
<code>empiler(e, pile)</code>	Empile la valeur de <code>e</code> dans la pile nommée <code>pile</code> et renvoie une nouvelle pile. Exemple : <code>maPile = empiler("vendredi", maPile)</code>
<code>consulter(pile)</code>	Renvoie la valeur du sommet de la pile nommée <code>pile</code> si celle-ci n'est pas vide. Produit une erreur d'assertion sinon. Exemple : <code>s = consulter(maPile)</code>

Fonction	Description
<code>depiler(pile)</code>	Renvoie une nouvelle pile après avoir supprimé la valeur du sommet de la pile nommée <code>pile</code> , dans le cas où la pile n'est pas vide. Produit une erreur d'assertion sinon. Exemple : <code>maPile = depiler(maPile)</code>
<code>afficher(pile)</code>	Renvoie une chaîne de caractères composée des éléments de la pile écrits de gauche à droite du fond de la pile vers son sommet. Le sommet est donc l'élément affiché le plus à droite. Les éléments sont séparés par une virgule. Si la pile est vide la fonction renvoie ' <code>pile vide</code> '.

Pour toutes ces fonctions autres que `creer_pile`, si la valeur de la pile passée en argument n'a pas été initialisée par cette fonction alors une erreur d'assertion est produite.

Enfin, la valeur `pile` passée en argument de ces fonctions n'est pas modifiée en place.

1. Questions relatives à l'interface.

- Écrivez une suite d'instructions permettant d'affecter à une variable nommée `pile1` les valeurs 7, 5 et 2 insérées dans cet ordre.
- Donnez l'affichage produit ensuite à l'issue de l'exécution des instructions suivantes.

```

element1 = consulter(pile1)
pile1 = depiler(pile1)
pile1 = empiler(5, pile1)
pile1 = empiler(element1, pile1)
afficher(pile1)

```

2. On donne maintenant le programme suivant.

```

1 tab = [2, 4, 3, 6, 8, 5, 7, 10, 1]
2 p = creer_pile()
3 for v in tab:
4     if v % 2 == 0: # si la valeur de v est paire
5         p = empiler(v, p)
6     else:
7         p = depiler(p)

```

- Quel est l'état de la variable nommée `p` à chaque étape de ce programme?
- Donnez deux valeurs du tableau `tab` qui conduisent à une erreur d'assertion, une avec un seul élément dans le tableau et l'autre avec trois éléments.

3. On donne ensuite la fonction `mystere` suivante :

```

1 def mystere(pile, element):
2     p = creer_pile()
3     n = nb_elements(pile)
4     for _ in range(n):
5         elem = consulter(pile)
6         pile = depiler(pile)
7         p = empiler(elem, p)
8         if elem == element:
9             return p
10    return p

```

(a) Dans chacun des quatre cas suivants, quel est l'affichage obtenu dans la console après la dernière instruction?

- Cas n° 1 :

```
>>>afficher(pile)
'7, 5, 2, 3'
>>>pile = mystere(pile, 2)
>>>afficher(pile)
```

- Cas n° 2 :

```
>>>afficher(pile)
'7, 5, 2, 3'
>>>pile = mystere(pile, 9)
>>>afficher(pile)
```

- Cas n° 3 :

```
>>>afficher(pile)
'7, 5, 2, 3'
>>>pile = mystere(pile, 3)
>>>afficher(pile)
```

- Cas n° 4 :

```
>>>est_vide(pile)
True
>>>pile = mystere(pile, 3)
>>>afficher(pile)
```

(b) Expliquez ce que permet d'obtenir la fonction `mystere`.

4. Complétez le code de la fonction `etendre(pile1, pile2)` qui prend en arguments deux piles et qui renvoie une nouvelle pile en empilant sur les éléments de `pile1` ceux de `pile2` rangés dans l'ordre inverse.

```
1 def etendre(pile1, pile2):
2     """Renvoie une nouvelle pile en empilant sur les
3         éléments de pile1 ceux de pile2 rangés dans
4         l'ordre inverse.
5     """
6     p = pile1 # copie de pile1
7     while .....:
8         .....
9         .....
10    return .....
```

On donne ci-dessous les résultats attendus pour certaines instructions.

```
>>>afficher(pile1)
'7, 5, 2, 3'
>>>afficher(pile2)
'1, 3, 4'
>>>pile1 = etendre(pile1, pile2)
>>>afficher(pile1)
'7, 5, 2, 3, 4, 3, 1'
```

5. Complétez le code de la fonction `supprimer_tout(element, pile)` qui renvoie une nouvelle pile en ayant supprimé toutes les occurrences de la valeur de `element` parmi les éléments de la variable `pile`.

```

1 def supprimer_tout(element, pile):
2     """Renvoie une nouvelle pile après avoir supprimé toutes
3         les occurrences de element dans pile.
4     """
5     p = creer_pile() # une pile temporaire
6     while .....:
7         .....
8         .....
9         if .....:
10            .....
11     return .....

```

On donne ci-dessous les résultats attendus pour certaines instructions.

```






>>>afficher(pile)
'7, 5, 2, 3, 5'
>>>pile = supprimer_tout(5, pile)
>>>afficher(pile)
'7, 2, 3'

```

Nom :

Prénom :

22/09/2023 - Structures de données, Pile et File

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	S'exprimer avec clarté et précision	