NSI - Bac Blanc

LGT Tani Malandi

2022 — 2023

Nom : Prénom : Classe :

Table des matières

Consignes et barême :

- Le devoir est a réaliser sur une copie
- Durée de l'épreuve : 3 heures 30
- Documents supplémentaires interdits
- Calculatrices et smartphones interdits
- **Le candidat traite au choix 3 exercices parmi les 5 exercices proposés**
- Barême :
  - Chaque exercice est sur 6 points
  - La présentation est évaluée sur 2 points
  - La note globale est sur 20 points.

EXERCICE 1 (6 points)

Cet exercice porte sur les structures de données (pile).

La notation polonaise inverse (NPI) permet d'écrire des expressions de calculs numériques sans utiliser de parenthèse. Cette manière de présenter les calculs a été utilisée dans des calculatrices de bureau dès la fin des années 1960. La NPI est une forme d'écriture d'expressions algébriques qui se distingue par la position relative que prennent les nombres et leurs opérations.

Par exemple :

Notation classique	Notation NPI
3+9	3 9 +
8×(3+5)	8 3 5 + ×
(17+5)×4	17 5 + 4 ×

L'expression est lue et évaluée de la gauche vers la droite en mettant à jour une pile.

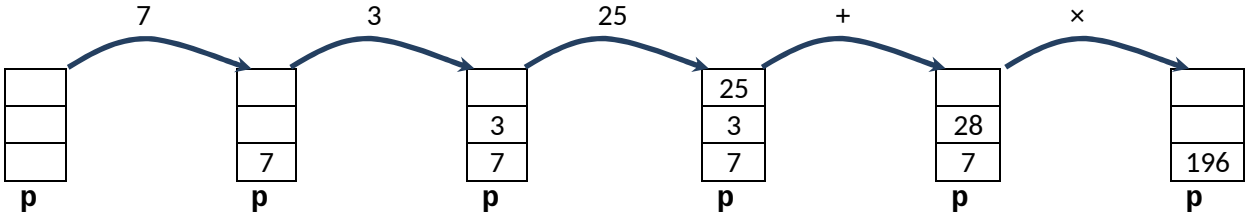
- Les nombres sont empilés dans l'ordre de la lecture.
- Dès la lecture d'un opérateur (+, -, ×, /), les deux nombres au sommet de la pile sont dépilés et remplacés par le résultat de l'opération effectuée avec ces deux nombres. Ce résultat est ensuite empilé au sommet de la pile.

A la fin de la lecture, la valeur au sommet est renvoyée.

Exemple : l'expression **7 3 25 + ×** qui correspond au calcul **7×(3+25)** s'évalue à 196 comme le montrent les états successifs de la pile créée, nommée **p** :

- On empile la valeur 7.
- On empile la valeur 3.
- On empile la valeur 25.
- On remplace les deux nombres du sommet de la pile (25 et 3) par leur somme 28.
- On remplace les deux nombres du sommet de la pile (28 et 7) par leur produit 196.

Schéma descriptif des différentes étapes d'exécution.



1. En vous inspirant de l'exemple ci-dessus, dessiner le schéma descriptif de ce que donne l'évaluation par la NPI de l'expression **12 4 5 × +** .

2. On dispose de la pile suivante nommée p1 :

On rappelle ci-dessous les primitives de la structure de pile (LIFO : Last In First out) :

Fonction	Description
pile_vide()	Créé et renvoie une nouvelle pile vide
empiler(p, e)	Place l'élément **e** au sommet de la pile **p**.
depiler(p)	Supprime et renvoie l'élément se trouvant au sommet de **p**.
est_vide(p)	Renvoie un booléen indiquant si **p** est vide ou non.

On dispose aussi de la fonction suivante, qui prend en paramètre une pile p :

```
def top(p) :
    x = depiler(p)
    empiler(p, x)
    return x
```

On exécute la ligne suivante `temp = top(p1)` :

a. Quelle valeur contient la variable `temp` après cette exécution ?

b. Représenter la pile `p1` après cette exécution.

3. En utilisant uniquement les 4 primitives d'une pile, écrire en langage Python la fonction `addition(p)` qui prend en paramètre une pile p d'au moins deux éléments et qui remplace les deux nombres du sommet d'une pile p par leur somme. Remarque : cette fonction ne renvoie rien, mais la pile p est modifiée.

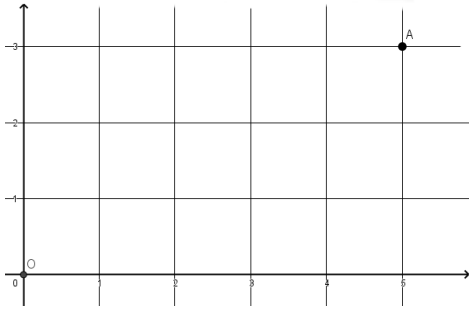
4. On considère que l'on dispose également d'une fonction `multiplication(p)` qui remplace les deux nombres du sommet d'une pile p par leur produit (on ne demande pas d'écrire cette fonction). Recopier et compléter, en n'utilisant que les primitives d'une pile et les deux fonctions `addition` et `multiplication`, la suite d'instructions (ci-dessous) qui réalise le calcul (3+5)×7 dont l'écriture en NPI est : 3 5 + 7 ×

```
p=pile_vide()
empiler(p,3)
```

Exercice 2 (6 points)

Thème abordé : programmation Python.

On considère un jeu de plateforme où un personnage se déplace dans un espace à deux dimensions. Pour cela, on autorise seulement deux déplacements élémentaires : de la gauche vers la droite ou du bas vers le haut. La longueur d'un déplacement correspond au nombre de déplacements élémentaires qui le constituent. Afin de représenter ces déplacements, on se place dans un repère où les coordonnées sont des nombres entiers positifs. Le personnage au début du jeu est situé à l'origine du repère de coordonnées (0,0) et il souhaite se rendre au point A de coordonnées (5,3).



Les lignes de code de cet exercice seront écrites en langage Python.

On décide de coder les déplacements élémentaires de la manière suivante :

- le caractère `'0'` représente un déplacement élémentaire vers la droite,
- le caractère `'1'` représente un déplacement élémentaire vers le haut.

Un déplacement de longueur **n** sera donc une chaîne de caractères composée de **n** caractères `'0'` ou `'1'`. Par exemple, un déplacement possible de O à A est `'00011001'` et sa longueur est 8.

1. On considère la fonction `mystere` ci-dessous.

```
1 | def mystere(dep):
2 |     x = 0
3 |     y = 0
4 |     for c in dep:
5 |       if c == '0':
6 |         x = x+1
7 |       else:
8 |         y = y+1
9 |     return [x, y]
```

a. Que renvoie `mystere('01110111')` ?

b. De manière plus générale, que renvoie la fonction `mystere` pour une chaîne représentant un déplacement donné ?

2. Ecrire une fonction `accessible(dep, arrivee)` de sorte qu'elle renvoie `True` si le déplacement `dep` se termine sur le point `arrivee` et `False` dans le cas contraire. Les types des paramètres sont donc :

- `dep` : une chaîne de caractères
- `arrivee` : une liste de deux entiers

Par exemple : `accessible('00110001', [5,3])` renvoie `True` alors que `accessible('01010111', [5,3])` renvoie `False`.

On rappelle que l'expression `str(randint(0,1))` utilisant la fonction `randint` de la bibliothèque `random` renvoie aléatoirement un caractère égal à `'0'` ou `'1'`.

On décide de trouver un déplacement de longueur 8 qui permette d'atteindre le point `arrivee`. Pour cela, on se propose de créer de manière aléatoire un déplacement de longueur 8, de tester si le point `arrivee` est accessible pour ce déplacement et de recommencer tant que ce n'est pas le cas.

3. Recopier et compléter la fonction **chemin** ci-dessous qui prend en paramètre les coordonnées du point `arrivee` et qui renvoie un déplacement de 8 pas qui permette d'atteindre le point `arrivee`. Quelles sont les préconditions sur le paramètre `arrivee` ?

```
1 | from random import randint
2 | def chemin(arrivee):
3 |   deplacement = '00000000'
4 |   while ..... :
5 |     .....
6 |     for k in range(8):
7 |       pas = str(randint(0,1))
8 |      ..... = deplacement + .....
9 |   return deplacement
```

4. La fonction `int(ch,2)` renvoie l'écriture décimale d'un nombre donné en binaire sous forme d'une chaîne de caractères `ch`. Par exemple `int('00000101',2)` renvoie `5`. Quelle est la plus grande valeur possible renvoyée par `int(chem,2)` si `chem` est un déplacement de longueur 8 permettant d'atteindre le point A de coordonnées (5,3) ?

EXERCICE 3 (6 points)

Cet exercice porte sur les tableaux et sur la programmation de base en Python.

On rappelle que `len` est une fonction qui prend un tableau en paramètre et renvoie sa longueur. C'est-à-dire le nombre d'éléments présents dans le tableau.
Exemple : `len([12, 54, 34, 57])` vaut 4.

Le but de cet exercice est de programmer différentes réductions pour un site de vente de vêtements en ligne.
On rappelle que si le prix d'un article avant réduction est de x euros,

- son prix vaut $0,5x$ si on lui applique une réduction de 50%,
- son prix vaut $0,6x$ si on lui applique une réduction de 40%,
- son prix vaut $0,7x$ si on lui applique une réduction de 30%,
- son prix vaut $0,8x$ si on lui applique une réduction de 20%,
- son prix vaut $0,9x$ si on lui applique une réduction de 10%.

Dans le système informatique du site de vente, l'ensemble des articles qu'un client veut acheter, appelé *panier*, est modélisé par un tableau de flottants.
Par exemple, si un client veut acheter un pantalon à 30,50 euros, un tee-shirt à 15 euros, une paire de chaussettes à 6 euros, une jupe à 20 euros, une paire de collants à 5 euros, une robe à 35 euros et un short à 10,50 euros, le système informatique aura le tableau suivant :

`tab = [30.5, 15.0, 6.0, 20.0, 5.0, 35.0, 10.5]`.

1. (a) Écrire une fonction Python `total_hors_reduction` ayant pour argument le tableau des prix des articles du panier d'un client et renvoyant le total des prix de ces articles.

   (b) Le site de vente propose la promotion suivante comme offre de bienvenue : 20% de réduction sur le premier article de la liste, 30% de réduction sur le deuxième article de la liste (s'il y a au moins deux articles) et aucune réduction sur le reste des articles (s'il y en a).

   Recopier sur la copie et compléter la fonction Python `offre_bienvenue` prenant en paramètre le tableau `tab` des prix des articles du panier d'un client et renvoyant le total à payer lorsqu'on leur applique l'offre de bienvenue.

```
def offre_bienvenue(tab):
    """ tableau -> float """
    somme=0
    longueur=len(tab)
    if longueur > 0 :
        somme=tab[0]*...
    if longueur > 1 :
        somme=somme + ...
    if longueur > 2 :
        for i in range(2,longueur):
            somme=...
    return ...
```

Pour toute la suite de l'exercice, on pourra utiliser la fonction `total_hors_reduction` même si la question 1 n'a pas été traitée.

2. Lors de la période des soldes, le site de vente propose les réductions suivantes :
   - si le panier contient 5 articles ou plus, une réduction globale de 50%,
   - si le panier contient 4 articles, une réduction globale de 40%,
   - si le panier contient 3 articles, une réduction globale de 30%,
   - si le panier contient 2 articles, une réduction globale de 20%,
   - si le panier contient 1 article, une réduction globale de 10%.

   Proposer une fonction Python `prix_solde` ayant pour argument le tableau `tab` des prix des articles du panier d'un client et renvoyant le total des prix de ces articles lorsqu'on leur applique la réduction des soldes.

3. (a) Écrire une fonction `minimum` qui prend en paramètre un tableau `tab` de nombres et renvoie la valeur minimum présente dans le tableau.

   (b) Pour ses bons clients, le site de vente propose une offre promotionnelle, à partir de 2 articles achetés, l'article le moins cher des articles commandés est offert.

   Écrire une fonction Python `offre_bon_client` ayant pour paramètre le tableau des prix des articles du panier d'un client et renvoyant le total à payer lorsqu'on leur applique l'offre bon client.

4. Afin de diminuer le stock de ses articles dans ses entrepôts, l'entreprise imagine faire l'offre suivante à ses clients : en suivant l'ordre des articles dans le panier du client, elle considère les 3 premiers articles et offre le moins cher, puis les 3 suivants et offre le moins cher et ainsi de suite jusqu'à ce qu'il reste au plus 2 articles qui n'ont alors droit à aucune réduction.

   Exemple : Si le panier du client contient un pantalon à 30,50 euros, un tee-shirt à 15 euros, une paire de chaussettes à 6 euros, une jupe à 20 euros, une paire de collants à 5 euros, une robe à 35 euros et un short à 10,50 euros, ce panier est représenté par le tableau suivant :

   ```
   tab = [30.5, 15.0, 6.0, 20.0, 5.0, 35.0, 10.5]
   ```

   Pour le premier groupe (le pantalon à 30,50 euros, le tee-shirt à 15 euros, la paire de chaussettes à 6 euros), l'article le moins cher, la paire de chaussettes à 6 euros, est offert. Pour le second groupe (la jupe à 20 euros, la paire de collants à 5 euros, la robe à 35 euros), la paire de collants à 5 euros est offerte.

   Donc le total après promotion de déstockage est 111 euros.

   On constate que le prix après promotion de déstockage dépend de l'ordre dans lequel se présentent les articles dans le panier.

   (a) Proposer un panier contenant les mêmes articles que ceux de l'exemple mais ayant un prix après promotion de déstockage différent de 111 euros.

   (b) Proposer un panier contenant les mêmes articles mais ayant le prix après promotion de déstockage le plus bas possible.

   (c) Une fois ses articles choisis, quel algorithme le client peut-il utiliser pour modifier son panier afin de s'assurer qu'il obtiendra le prix après promotion de déstockage le plus bas possible ? On ne demande pas d'écrire cet algorithme.

EXERCICE 4 (6 points)

Cet exercice porte sur les structures de données (programmation objet).

Simon souhaite créer en Python le jeu de cartes « la bataille » pour deux joueurs. Les questions qui suivent demandent de reprogrammer quelques fonctions du jeu. On rappelle ici les règles du jeu de la bataille :

Préparation

- Distribuer toutes les cartes aux deux joueurs.
- Les joueurs ne prennent pas connaissance de leurs cartes et les laissent en tas face cachée devant eux.

Déroulement

- A chaque tour, chaque joueur dévoile la carte du haut de son tas.
- Le joueur qui présente la carte ayant la plus haute valeur emporte les deux cartes qu'il place sous son tas.
- **Les valeurs des cartes sont :** dans l'ordre de la plus forte à la plus faible : As, Roi, Dame, Valet, 10, 9, 8, 7, 6, 5, 4, 3 et 2 (la plus faible)

Si deux cartes sont de même valeur, il y a "bataille".

- Chaque joueur pose alors une carte face cachée, suivie d'une carte face visible sur la carte dévoilée précédemment.
- On recommence l'opération s'il y a de nouveau une bataille sinon, le joueur ayant la valeur la plus forte emporte tout le tas.

Lorsque l'un des joueurs **possède toutes les cartes du jeu**, la partie s'arrête et ce dernier gagne.

Pour cela Simon crée une classe Python `Carte`. Chaque instance de la classe a deux attributs : un pour sa valeur et un pour sa couleur. Il donne au valet la valeur 11, à la dame la valeur 12, au roi la valeur 13 et à l'as la valeur 14. La couleur est une chaîne de caractères : `"trefle"`, `"carreau"`, `"coeur"` ou `"pique"`.

1. Simon a écrit la classe Python `Carte` suivante, ayant deux attributs `valeur` et `couleur`, et dont le constructeur prend deux arguments : `val` et `coul`.

a. Recopier et compléter les [.....] des lignes 3 et 4 ci-dessous.

```
1. class Carte:
2.       def __init__(self, val, coul):
3.           [.....].valeur = [.....]
4.           [.....] = coul
```

b. Parmi les propositions ci-dessous quelle instruction permet de créer l'objet « 7 de cœur » sous le nom `c7` ?

- ¬ `c7.__init__(self, 7, "coeur")`
- ¬ `c7 = Carte(self, 7, "coeur")`
- ¬ `c7 = Carte(7, "coeur")`
- ¬ `from Carte import 7, "coeur"`

2. On souhaite créer le jeu de cartes. Pour cela, on écrit une fonction `initialiser()` :

- ¬ sans paramètre
- ¬ qui renvoie une liste de 52 objets de la classe `Carte` représentant les 52 cartes du jeu.

  Voici une proposition de code. Recopier et compléter les lignes suivantes pour que la fonction réponde à la demande :

```
def initialiser() :
  jeu = []
  for c in ["coeur", "carreau", "trefle", "pique"] : # couleur carte
    for v in range(...) :                            # valeur carte
      carte_cree  = ...
      jeu.append(carte_cree)
  return jeu
```

3. On rappelle que dans une partie de bataille, les deux joueurs tirent chacun une carte du dessus de leur tas, et celui qui tire la carte la plus forte remporte les deux cartes et les place en dessous de son tas.

Parmi les structures linéaires de données suivantes : Tableau, File, Pile, quelle est celle qui modélise le mieux un tas de cartes dans ce jeu de la bataille ? Justifier votre choix.

4. Écrire une fonction `comparer(carte1, carte2)`qui prend en paramètres deux objets de la classe `Carte`. Cette fonction renvoie :

- ¬ `0` si la force des deux cartes est identique,
- ¬ `1` si la carte `carte1` est strictement plus forte que `carte2`
- ¬ `-1` si la carte `carte2` est strictement plus forte que `carte1`

EXERCICE 5 (6 points)

Thème : Exécution de programmes, recherche et corrections de bugs

Les questions proposées sont indépendantes les unes des autres.

1. On considère la fonction `somme(n)` qui reçoit en paramètre un entier n strictement positif et renvoie le résultat du calcul $1 + - + - + \cdots + -$.

```
1|    def somme(n) :
2|        total = 0
3|        for i in range(n) :
4|            total = total  + 1/i
5|        return total
```

Lors de l'exécution de `somme(10)`, le message d'erreur `"ZeroDivisionError: division by zero"` apparait. Identifier le problème et corriger la fonction pour qu'elle effectue le calcul demandé.

2. On considère la fonction `maxi(L)` qui prend comme paramètre une liste **L** de nombres et renvoie le plus grand nombre de cette liste :

```
1|    def maxi(L) :
2|        indice = 0
3|        maximum = 0
4|        while indice <= len(L) :
5|            if L[indice] > maximum :
6|                maximum = L[indice]
7|            indice = indice + 1
8|        return maximum
```

a. Lors de l'exécution de `maxi([2, 4, 9, 1])` une erreur est déclenchée. Identifier et corriger le problème.

b. Le bug précédent est maintenant corrigé. Que renvoie à présent l'exécution de `maxi([-2, -7, -3])` ? Modifier la fonction pour qu'elle renvoie le bon résultat.

3. On souhaite réaliser une fonction qui génère une liste de **n** joueurs identifiés par leur numéro. Par exemple on souhaite que l'appel `genere(3)` renvoie la liste `['Joueur 1', 'Joueur 2', 'Joueur 3']`.

```
1|   def genere(n) :
2|       L = []
3|       for i in range(1, n+1) :
4|           L.append('Joueur '+i)
5|       return L
```

L'appel `genere(3)` déclenche l'erreur suivante : **`TypeError: can only concatenate str (not "int") to str`**.

Expliquer ce message d'erreur et corriger la fonction afin de régler le problème.

4. On considère la fonction **`suite(n)`** qui reçoit un entier positif et renvoie un entier.

```
1|   def suite(n) :
2|       if n == 0 :
3|           return 0
4|       else :
5|           return 3+2*suite(n-2)
```

a. Quelle valeur renvoie l'appel de `suite(6)` ?

b. Que se passe-t-il si on exécute `suite(7)` ?

5. On considère le code Python ci-dessous :

```
1|   x = 4
2|   L = []
3|   def modif(x, L) :
4|       x = x + 1
5|       L.append(2*x)
6|       return x, L
7|
8|   print(modif(x, L))
9|   print(x, L)
```

a. Qu'affiche le premier `print` ?

b. Qu'affiche le second `print` ?