Numérique et Sciences Informatiques

Première

- Recueil d'exercices -

Table des matières

Chapitre 1

Algorithmique

Langage naturel

Dans cette partie, on écrira les algorithmes en utilisant les formulations suivantes :

- *Affectation*

On donne une valeur à une variable. Cela permet de la créer ou de modifier sa valeur. Par exemple :

```
a prend la valeur 8
```

```
# En python
a = 8
```

- *Affichage*

On affiche une valeur ou un message (avec des guillemets). Par exemple :

```
# Affichage de la valeur d'une variable
Afficher a
# Affichage d'un message
Afficher "Un message"
# Affichage combiné
Afficher "a vaut ", a
```

```
# En python
print(a)
print("Un message")
print("a vaut",a)
```

- *Entrées*

On demande à l'utilisateur de saisir une valeur dont on spécifie le type et on la stocke dans une variable. Par exemple :

```
Lire un texte et le stocker dans prenom
Lire un entier et le stocker en nb
Lire un nombre décimal et le stocker dans nb_virgule
```

```python
# En python
prenom = input("Saisir le prénom")
nb = int(input("Saisir un entier"))
nb_virgule = float(input("Saisir un nombre décimal"))
```

- *Tests*

On effectue des actions en fonction du résultat d'un ou plusieurs tests. Par exemple :

```
a prend la valeur 10

Si a < 10 :
    Afficher "Trop petit"
Sinon si a > 10 :
    Afficher "Trop grand"
Sinon :
    Afficher "Gagné !"
```

```python
# En python
a = 10

if a < 10 :
    print("Trop petit")
elif a > 10 :
    print("Trop grand")
else :
    print("Gagné !")
```

- *Boucles bornées*

On répète des actions un nombre prédéfini de fois. Par exemple :

```
Pour i allant de 0 à 101 (exclus) :
    Afficher i
```

```python
# En python
for i in range(0,101) :
    print(i)
```

- *Boucles non bornées*

On répète des actions un nombre inconnu de fois : les répétitions continuent tant que la condition est remplie Par exemple :

```
a = 10
Tant que a != 0 :
    Afficher a
    a = a - 1
```

```python
# En python
a = 10
while a != 0 :
    print(a)
    a = a - 1
```

Exercice 1.1 :

Écrire un algorithme qui demande à l'utilisateur de taper la largeur et la longueur d'un champ et qui en affiche la surface.

Exercice 1.2 :

Écrire un algorithme qui demande à l'utilisateur de saisir son prénom et qui affiche le message "`Bonjour <prénom>`" en remplaçant `<prénom>` par le texte saisi.

Exercice 1.3 :

Écrire un algorithme qui demande à l'utilisateur de taper le prix au kg de pommes, la masse achetée et qui affiche le prix à payer.

Exercice 1.4 :

Écrire un algorithme qui demande à l'utilisateur de taper un nombre et qui affiche "`Positif`" ou "`Négatif`" selon le signe de ce nombre.

Exercice 1.5 :

Écrire un algorithme qui demande à l'utilisateur de taper un nombre et qui teste s'il est solution de l'équation $2x + 7 = 16$. L'algorithme affichera "`Oui`" ou "`Non`" selon la réponse.

Exercice 1.6 :

Écrire un algorithme qui demande à l'utilisateur de taper sa dernière note de Français et qui affiche "`Pas terrible`" si elle est strictement en-dessous de 10, "`Pas mal`" si elle est entre 10 et 14 (exclus) et "`Bon travail`" dans les autres cas.

Exercice 1.7 :

Afficher la table de 5 pour les nombres de 0 à 20.

Exercice 1.8 :

Afficher 53 fois le message "`J'utilise une boucle Pour`".

Exercice 1.9 :

Afficher successivement les messages "`Le double de 0 est 0`", "`Le double de 1 est 2`", ... jusqu'à "`Le double de 523 est 1046`".

Exercice 1.10 :

Demander à l'utilisateur de saisir un nombre et afficher la table de ce nombre.

Exercice 1.11 :

Demander à l'utilisateur de saisir un nombre et afficher la table de 7 jusqu'à ce nombre. Si le nombre tapé est strictement négatif, on affichera "`Il faut saisir un nombre positif`".

Exercice 1.12 :

Afficher la table de 5 jusqu'à dépasser 1 234 567.

Exercice 1.13 :

Calculer $3n - 8$ pour les valeurs de n entières $(0, 1, 2, \dots)$ jusqu'à ce que le résultat soit supérieur à 10 000. Afficher la dernière valeur de n.

Exercice 1.14 :

Demander à l'utilisateur de taper un nombre entier entre 1 et 10 jusqu'à ce qu'il saisisse 9.

Exercice 1.15 :

Demander à l'utilisateur de taper un nombre entier entre 1 et 100 jusqu'à ce qu'il saisisse 49. A chaque réponse on affichera `"Trop grand"`, `"Trop petit"` ou `"Gagné"` selon le nombre saisi.

Coût et complexité

Exercice 1.16 :

Un programme traite des données dont la taille peut être mesurée à l'aide d'une variable n.

Si $n = 100$, le programme retourne un résultat en 8 ns.

On admet que le temps d'exécution de ce programme évolue proportionnellement à une certaine puissance de n. Par exemple si le temps évolue proportionnellement à n^2, lorsque l'on triple la valeur de n, le temps est multiplié par $3^2 = 9$.

Compléter le tableau de durées approximatives ci-dessous :

Valeur de n	n^1	n^2	n^3
$n = 100$	8 ns	8 ns	8 ns
$n = 200$	16 ns		
$n = 300$		72 ns	
$n = 400$			512 ns
$n = 500$			
$n = 1000$			
$n = 10000$			
$n = 1000000$			

Exercice 1.17 :

Le temps T de réalisation d'un programme informatique n'est en réalité pas exactement égal au nombre d'opérations C. Il faut tenir compte du temps mis par la machine pour réaliser une opération.

Par exemple, si un programme s'exécute en 100 opérations et que chacune d'entre elle met 12 ns à se faire, il faut environ 1 200 ns pour mener ce programme. On définit ci-dessous des coûts (nombre d'opérations) en fonction de la taille des données n.

Estimer le temps de réalisation du programme en considérant que la machine met en moyenne 12 ns à réaliser une opération (affectation, test, calcul ...).

Compléter le tableau :

Nombre d'opérations $C(n)$	Taille des données n	Nombre d'opérations réel	Temps d'exécution
n^2+3n-1	100	10 299	$123,588\ ms$
n^2+3n-1	1 000		
n^3-n^2+n	100		
n^3-n^2+n	1 000		
2^n	100		
2^n	1 000		
$n!$	10		
$n!$	100		

Exercice 1.18 :

On cherche à déterminer si un élément est dans une liste.

Un algorithme simple est de lire les valeurs successivement et retourner `"VRAI"` si l'on rencontre l'élément cherché. Si l'on atteint la fin de la liste sans avoir rencontré l'élément, on retourne `"FAUX"`.

```
Pour toutes les valeurs de la liste :
    Si la valeur est égale à l'élément cherché :
        Retourner VRAI
Retourner FAUX
```

1. On considère la liste suivante `[1, 5, 6, 3, 9, 7, 2, 4, 8, 0]` et l'on cherche l'élément `2`. Combien de valeurs a-t-on lu avant de le trouver ?

2. Pour quel valeur le programme sera-t-il le plus long à renvoyer `"VRAI"` ?

3. On imagine désormais que la liste comporte `n` nombres et que l'élément cherché est en dernière position. Combien de valeurs lira l'algorithme avant de retourner `"VRAI"` ?

4. Même question si la liste comporte `2n` valeurs.

Exercice 1.19 :

Un "voyageur de commerce" doit passer par 3 villes A, B et C dans sa journée. Il connaît l'ensemble des distances $A \to B$, $B \to C$ et $C \to A$ (que l'on suppose symétriques : $A \to B = B \to A$). Il cherche le parcours le plus court.

Sa première approche est de lister tous les parcours différents.

1. Combien de trajets possibles existe-t-il ?

2. Et s'il doit parcourir 4 villes ?

3. On suppose qu'un ordinateur met 1 μs à calculer la longueur d'un trajet. Combien de temps met-il à calculer toutes les distances dans le cas de 4 villes ? 5 ? 10 ? 20 villes ?

4. Pourquoi cette façon de résoudre le problème du voyageur de commerce est-elle irréalisable en pratique ?

Exercice 1.20 :

On dispose d'un tableau contenant des effectifs et l'on souhaite créer un tableau d'effectifs cumulés. Par exemple `eff=[2, 7, 8, 5, 6, 4]` donnera `eff_cumul=[2, 9, 17, 22, 28, 32]`.

Une première approche consiste à additionner toutes les valeurs à chaque calcul :

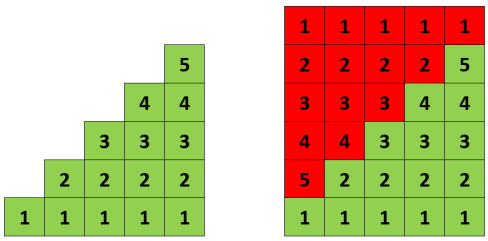
```
N prend la valeur de longueur(eff)
eff_cumul est une liste vide

Pour i allant de 0 à N (exclus) :
    cumul = 0
    Pour j allant 0 à i+1 (exclus) :
        cumul = cumul + eff[j]
    Ajouter cumul à la fin de eff_cumul
```

1. a. On applique cet algorithme à la liste `eff` définie ci-dessus. Compléter le tableau ci-dessous :

Valeur de `i`	Valeurs prises par `j`	Calcul effectué	Valeur de `eff_cumul`
0	de 0 à 0	2	[2]
1	de 0 à 1	2+7	[2,9]
2			
3			
4			
5			

b. Combien d'additions ont-été effectuées au total ?

c. Observer la figure ci-dessous. Justifier que $1+2+3+4+5=15$.



d. On souhaite désormais manipuler un tableau de 100 nombres. Combien d'additions seront nécessaires ?

e. Exprimer le nombre d'additions nécessaires en fonction de la longueur du tableau `N`.

Une nouvelle approche consiste à exploiter les calculs déjà faits à chaque calcul :

```
N prend la valeur de longueur(eff)
eff_cumul est une liste vide
cumul = 0
Pour i allant de 0 à N (exclus) :
    cumul = cumul + eff[i]
    Ajouter cumul à la fin de eff_cumul
```

2. Reprendre la question précédente avec ce nouvel algorithme.

3. Quelle façon de procéder est la plus efficace ?

Tests

Exercice 1.21 :

On propose ci-dessous le code d'une fonction permettant de compter le nombre de voyelles dans une chaîne de caractères :

```
def voyelles(chaine : str) -> int :
    compteur = 0
    for lettre in chaine :
        if lettre in "aeiouy" :
            compteur += 1
    return compteur
```

1. Que renvoie l'appel `voyelles("Jean")` ? Est-ce correct ?

2. Proposer un test permettant de vérifier que cette fonction compte correctement le nombre de voyelles en minuscules.

3. Que renvoie l'appel `voyelles("Eric")` ? Est-ce correct ?

4. Proposer un test permettant de permettant de prouver que cette fonction gère mal les voyelles en majuscules.

5. Proposer un test permettant de permettant de prouver que cette fonction gère mal les voyelles accentuées.

6. Corriger le code de cette fonction afin qu'elle réponde aux tests précédents.

Exercice 1.22 :

On propose le code `python` ci-dessous permettant de trouver l'indice de la dernière occurrence d'une valeur dans une liste :

```python
def derniere_occurrence(liste : list, valeur) -> int:
    renverse = liste[::-1] # renverse la liste
    i = len(liste) - 1
    for  elt in renverse :
        if elt == valeur :
            return i
        i -= 1
    return i
```

Indiquer ce que vérifient les tests ci-dessous :

1. `assert derniere_occurrence([3,2,1], 1) == 2`
2. `assert derniere_occurrence([3,2,1], 3) == 0`
3. `assert derniere_occurrence([3,2,1], 4) == -1`
4. `assert derniere_occurrence([3,2,1,1], 1) == 3`
5. `assert derniere_occurrence([], 1) == -1`
6. `assert derniere_occurrence([3,2,1], "a") == -1`
7. `assert derniere_occurrence([3,2,1], -1) == -1`

Exercice 1.23 :

On utilise le module `doctest` afin de tester le bon fonctionnement de la fonction ci-dessous :

```python
def supprime_suivant(liste, valeur) :
    """
    Fonction recherchant une valeur dans une liste et renvoyant la liste
    après avoir supprimée l'élément suivant la première occurrence
    de valeur trouvée
    liste : la liste dans laquelle on cherche
    valeur : la valeur à chercher
    Renvoie la liste avec ou non la valeur supprimée

    >>> supprime_suivant([4,5,6], 5)
    [4, 5]

    >>> supprime_suivant([4,5,6], 4)
    [4, 6]

    >>> supprime_suivant([4,5,6], 6)
    [4, 5, 6]

    >>> supprime_suivant([4,5,6], 7)
    [4, 5, 6]
    """

    indice = 0
    for elt in liste :
        if elt == valeur :
            l = []
            for i in range(len(liste)) :
                if i != indice + 1 :
                    l.append(liste[i])
            return l
        indice += 1
    return liste
```

1. Quels sont les objectifs de chacun de ces tests ? Que cherchent-ils à vérifier ?
2. Proposer un test permettant de vérifier le comportement de la fonction dans le cas d'une liste vide.
3. Proposer un test permettant de vérifier que la fonction supprime bien l'élément suivant la première occurrence de la valeur cherchée.

Exercice 1.24 :

On souhaite écrire en `python` une fonction `minmax` permettant d'extraire le minimum et le maximum d'une liste de nombres entiers. Par exemple `minmax([8,3,10,2,0])` renverra `(0,10)`.

Dans le cas où la liste est vide on renverra `(None,None)`.

On propose le code ci-dessous dans lequel on a écrit des tests qui seront effectués avec le module `doctest` de `python`.

```
def minmax(liste : list) -> tuple :
    """
    Fonction renvoyant le tuple composé des valeurs minimales et maximales de
↪   la liste

    >>> minmax([8,3,10,2,0])
    (0,10)

    >>> minmax([10, 50, -1, 3])
    (-1,50)

    >>> minmax([])
    (None, None)

    >>> minmax([10, 0.5])
    Traceback (most recent call last):
       ...
    AssertionError: La liste ne doit contenir que des nombres entiers

    >>> minmax([10, "a"])
    Traceback (most recent call last):
       ...
    AssertionError: La liste ne doit contenir que des nombres entiers
    """
```

Écrire le code de la fonction `minmax` de façon à ce qu'elle passe les différents tests.

Correction d'algorithmes

Exercice 1.25 : Identifier un variant de boucle

On considère l'algorithme ci-dessous :

```
a = 8
b = 20
Tant que a != b :
    a += 1
    b -= 2
```

1. Compléter le tableau ci-dessous présentant les valeurs des variables à l'issue d'un tour de boucle :

Tour de boucle	`a`	`b`	`b - a`
0	8	20	12
1	9	18	9
3			
4			
5			

2. La variable `a` est-elle un variant de cette boucle ?

3. Quelle quantité peut être choisie comme variant de boucle ? Justifier.

Exercice 1.26 :

On considère les algorithmes suivants :

Algorithme 1

```
a = 12
Tant que a est différent de 0 :
    a = a - 0.1
```

Algorithme 2

```
b = 1
Tant que b < 10 :
    b = b + 1
```

1. Quelles sont les valeurs successives prises par `a` et `b` au fil des algorithmes ?

2. La variable `a` est-elle un variant de boucle ?

3. Donner un variant de boucle pour chacun des algorithmes.

4. Expliquer pourquoi le premier algorithme ne se terminera pas en `python`.

Exercice 1.27 : Correction de la recherche d'un minimum

On considère le code `python` ci-dessous qui détermine la valeur du minimum d'une liste :

```
# lst est une liste non vide de nombres
i_mini = 0

for i in range(1, len(lst)) :
    if lst[i] < lst[i_mini] :
        i_mini = i
```

1. a. La variable `i` est-elle un variant de boucle satisfaisant ?

   b. Proposer un variant de boucle. Justifier.

On considère l'invariant de boucle suivant :

A l'issue de chaque tour de boucle i, `i_mini` *est égal à l'indice du minimum du sous − tableau* `lst[0..i]`

2. a. Justifier que cette propriété est vérifiée avant d'entrer dans la boucle (donc pour `i`=0).

   b. On suppose qu'il existe un entier k tel que l'invariant est vérifié après k tours de boucles. Prouver qu'il l'est aussi après $k+1$ tours de boucles

Exercice 1.28 : Terminaison et logarithme binaire

On considère le code `python` ci-dessous qui calcule le nombre de bits nécessaires pour écrire un nombre entier positif en binaire :

```python
def nb_bits(n : int) -> int :
    nb = 1
    reste = n
    while reste >= 1 :
        reste = reste/2
        nb = nb + 1
    return nb
```

1. a. Que renvoie l'appel `nb_bits(3)` ? Est-ce satisfaisant ?

   b. Que renvoie l'appel `nb_bits(32)` ? Est-ce satisfaisant ?

On souhaite prouver que la boucle `while` de cet algorithme se termine bien pour toute valeur positive de `n`. Pour se faire on va chercher un *variant* de boucle.

2. Rappeler les trois conditions que doit respecter un *variant* de boucle.

3. Parmi les trois propositions suivantes, laquelle est un variant de boucle correct pour cet algorithme ?

   a. la variable `n` ?

   b. la variable `nb` ?

   c. la partie entière de la variable `n` ?

   Remarque : on rappelle que la partie entière d'un nombre est son arrondi à l'entier immédiatement inférieur.

Exercice 1.29 : Correction partielle et logarithme binaire

On considère le code `python` ci-dessous qui calcule le nombre de bits nécessaires pour écrire un nombre entier positif en binaire :

```python
def nb_bits(n : int) -> int :
    assert n > 0
    i = 0
    reste = n
    while 2**i * reste < n :
        reste = reste / 2
        i = i + 1
    return i
```

1. Quel est le rôle de l'instruction `assert n > 0` ?

   Afin de prouver la correction partielle de l'algorithme, on introduit l'invariant suivant :

$$\textit{A l'issue de chaque tour de boucle } i\textit{, on a } 2^{\texttt{i}} \times \texttt{reste} \leq \texttt{n} < 2^{\texttt{i+1}} \times \texttt{reste}$$

2. a. Prouver que cette propriété est vraie lors de l'initialisation de l'algorithme.

   b. Justifier que lors du i-ième tour de boucle on a :

$$\texttt{reste} = \frac{\texttt{n}}{2^{\texttt{i}}}$$

c. On suppose qu'il existe un entier k tel que l'invariant est vérifié après k tours de boucles. Prouver qu'il l'est aussi après $k+1$ tours de boucles

Exercice 1.30 : Correction du tri à bulles

On considère l'algorithme du tri à bulles ci-dessous :

```
def tri_bulle(A) :
    # BOUCLE PRINCIPALE
    for i in range(len(A)-1, 0, -1) :
        # i parcourt les indices du grand vers le petit
        # BOUCLE SECONDAIRE
        for j in range(i) :
                # j parcourt les indices allant de 0 à i-1
                if A[j] > A[j+1] :
                    A[j], A[j+1] = A[j+1], A[j]
```

On considère la liste `[6, 5, 2, 4]`. Lors de la première itération de la boucle principale (`i=3`) et la première itération de la boucle secondaire (`j=0`), on compare les valeurs de `A[0]=6` et `A[1]=5` et on les échange.

1. Compléter le tableau ci-dessous :

i	j	A[j]	A[j+1]	A (à la fin de la boucle)
3	0	6	5	[5, 6, 2, 4]
	1	6	2	[5, 2, 6, 4]
	2	6	4	[5, 2, 4, 6]
2				
1				

L'algorithme est construit autour de deux boucles `for`. Sa terminaison est donc facile à démontrer. On s'intéresse ici à sa correction partielle.

Un invariant pour la boucle secondaire est :

A l'issue de chaque tour de boucle j, tous les éléments dela sous − liste `A[0...j]`

sont inférieurs à `A[j+1]`

2. a. Prouver que cet invariant est vérifié avant de rentrer dans la boucles secondaire.

b. On suppose que cet invariant est vérifié après j tours de la boucle secondaire. Prouver qu'il l'est aussi après `j`+1 tours.

Un invariant pour la boucle principale est :

A l'issue de chaque tour de boucle k, les k derniers éléments de `A` *sont triée*

et tous les autres éléments de `A` *leurs sont inférieurs*

3. a. Prouver que cet invariant est vérifié avant de rentrer dans la boucle principale.

b. On suppose que cet invariant est vérifié après k tours de la boucle principale. Prouver qu'il l'est aussi après `k`+1 tours.

Chapitre 2

Bases de python

Les exercices de ce chapitre nécessite de saisir du code `python`. On pourra le faire dans un environnement personnel (après avoir installé `python` sur un ordinateur) ou en ligne, par exemple sur basthon (`https://basthon.fr/`).

Affichages et variables

Exercice 2.1 : `"Hello World !"`

Afficher le message `"Hello World !"`

Exercice 2.2 : `phrase = "Hello World !"`

Créer une variable `phrase` contenant le message `"Hello World !"` puis afficher `phrase`.

Exercice 2.3 :

1. Créer une variable `bonjour` contenant le message `"Bonjour "`.
2. Créer une variable `prenom` contenant votre prénom.
3. Afficher le résultat de `bonjour + prenom`.

Exercice 2.4 :

1. Créer une variable `bonjour` contenant le message `"Bonjour "`.
2. Demander à l'utilisateur de taper son prénom et stockez le dans la variable `prenom`.
3. Afficher le résultat de `bonjour + prenom`.

Exercice 2.5 :

Créer une variable `nombre` stockant `12` puis afficher `nombre`.

Exercice 2.6 :

1. Créer une variable `a` stockant `2`.
2. Créer une variable `b` stockant $\frac{a+8}{2}$.
3. Afficher `b`. Vous devez obtenir 5 !

Exercice 2.7 :

1. Créer une variable `a` égale au tiers du carré de la somme de 5 et 22.
2. Afficher `a`. Le résultat doit tomber juste !

On pourra utiliser `n**2` pour mettre `n` au carré.

Exercice 2.8 :

1. Créer la variable `a` égale à 5.
2. Multiplier `a` par 2 et stocker le résultat dans `a` !
3. Additionner 8 à `a` et stocker le résultat dans `a`.
4. Affichez le quotient de `a` par 9.

Exercice 2.9 :

1. Demander à l'utilisateur de saisir un nombre et le stocker dans la variable `a`.
2. Afficher le résultat de `a*3`.
3. Comment expliquer ce résultat ?

Exercice 2.10 :

Écrire le code affichant le sapin comme ci-dessous :

```
     *
    ***
   *****
  *******
 *********
***********
     *
     *
```

On pourra utiliser `print(" "*4 + "*"*3)` !

Tests `if ... else ... elif ...`

Exercice 2.11 :

1. Demander à l'utilisateur de saisir un nombre et le stocker le dans la variable `a`.
2. Si `a` est positif ou nul afficher "Positif" sinon, afficher "Négatif".

Exercice 2.12 :

1. Demander à l'utilisateur de saisir un nombre et le stocker dans la variable `a`.
2. Si `a` est égal à 100 afficher "OK" sinon afficher "Réessayer".

Exercice 2.13 :

1. Demander à l'utilisateur de saisir un nombre et le stocker le dans la variable `a`.
2. Si `a` est strictement plus grand que 7 mais inférieur ou égal à 13 afficher "OK" sinon afficher "Réessayer".

Exercice 2.14 :

1. Demander à l'utilisateur de saisir un nombre et le stocker le dans la variable `a`.
2. Si `a` est plus strictement petit que 7 afficher "Trop petit". Sinon, si `a` est supérieur ou égal à 13 affichez "Trop grand". Sinon, afficher "OK".

Exercice 2.15 :

1. Demander à l'utilisateur de saisir le nom d'un fruit et le stocker dans la variable `fruit`.
2. Si le fruit est une fraise afficher "rouge". Si c'est un citron, afficher "jaune". Sinon, afficher "je ne sais pas...".

Exercice 2.16 :

Écrire le code calculant et affichant la moyenne de trois nombres saisis par l'utilisateur.

Exercice 2.17 :

Écrire le code déterminant le plus grand des trois nombres saisis par l'utilisateur.

Exercice 2.18 :

Ècrire le code demandant à l'utilisateur de saisir son année de naissance et affichant s'il est majeur, mineur ou s'il sera majeur cette année.

Boucles `for` ...

Exercice 2.19 :

Affichez 30 fois le message "Hello World".

Exercice 2.20 :

Affichez tous les nombres i entre 0 et 20 (inclus).

Exercice 2.21 :

Affichez tous les nombres n entre 50 et 100 (inclus).

Exercice 2.22 :

Affichez successivement toutes les lettres du mot "anticonstitutionnellement".

Exercice 2.23 :

1. Créer une variable `somme` valant 0
2. Additionner tous les nombres `n` entre 0 et 100 (inclus) dans la variable `somme` (on fera `somme + somme + n`) Afficher le résultat. Il vaut bien 5050 ?

Exercice 2.24 :

Combien vaut le produit de tous les entiers entre 1 et 100 ?

Exercice 2.25 :

Calculez la somme des dix nombres saisis par l'utilisateur (utiliser une variable somme que dans l'exercice 2.23)

Exercice 2.26 :

1. Demander à l'utilisateur de saisir un nombre entier `N`
2. Calculer la moyenne des `N` nombres saisis par l'utilisateur.

Exercice 2.27 :

1. Calculer la somme des 30 premiers nombres pairs.
2. Calculer la somme des 30 premiers nombres impairs.

Boucles `while` ...

Exercice 2.28 :

1. Créer une variable `n` valant 8 .
2. Diviser `n` par deux en faisant `n = n/2` jusqu'à ce que le résultat soit plus petit que $0,001$.

3. Afficher la valeur de `n`.

Exercice 2.29 :

Afficher tous les nombres de la table de 7 jusqu'à dépasser 1000. On utilisera une boucle `while`.

Exercice 2.30 :

1. Créer une variable `n` valant 8.
2. Calculer `(n+7)*2` jusqu'à dépasser 1 000 000.
3. Afficher la valeur de `n`.

Exercice 2.31 :

1. Créez une variable `n` valant 8 .
2. Calculer `(n+5)/2` jusqu'à passer sous $5,0001$.
3. Afficher la valeur de `n`.

Compréhension de code

Exercice 2.32 : Echange (1)

On considère le code `python` ci-dessous :

```
a = 10
b = 5
c = a
a = b
b = c
```

1. Que valent les variables `a`, `b` et `c` à l'issue de l'exécution ?
2. A quoi sert ce code ?

Exercice 2.33 : Echange (2)

On considère le code `python` ci-dessous :

```
a = 10
b = 5
a = a + b
b = a - b
a = a - b
```

1. Que valent les variables `a` et `b` à l'issue de l'exécution ?

2. A quoi sert ce code ?

Exercice 2.34 : Echange (3)

On considère le code python ci-dessous :

```
a = 10
b = 5
a = a * b
b = a / b
a = a / b
```

1. Que valent les variables a et b à l'issue de l'exécution ?

2. A quoi sert ce code ?

Exercice 2.35 :

1. Que permet de faire le script python ci-dessous ?

```
s=0
for i in range(6) :
    s = s + i
print(s)
```

2. Quelle la valeur de la variable s en fin d'exécution ?

3. Modifier ce code afin qu'il calcule la somme $0 + 1 + 2 + \cdots + 100$.

Exercice 2.36 :

1. Que permet de faire le script python ci-dessous ?

```
cit = """Je ne cherche pas à connaître la réponse,
         je cherche à comprendre la question"""
compteur = 0
for lettre in cit :
    if lettre == "e" :
        compteur = compteur + 1
print(compteur)
```

2. Modifier ce code afin qu'il compte les i.

Exercice 2.37 :

On considère l'algorithme suivant :

```
Lire un entier et le stocker dans K
A prend la valeur 300
T prend la valeur 6
Pour i allant de 1 à K (inclus) :
    A prend la valeur A*(1+T/100)
Afficher A
```

1. Faire fonctionner cet algorithme pour `K = 4` et compléter le tableau de valeurs ci-dessous.

i	K	T	A
1			
2			
3			
4			

2. Quel est le résultat affiché en fin d'exécution ?

3. Coder cet algorithme en `python` puis vérifier le résultat obtenu à la question précédente.

4. Alain souhaite placer 3000 € sur un compte rémunéré au taux d'intérêt de $1,5\%$. Modifier le programme précédent pour permettre à Alain de connaître somme dont il disposera sur ce compte au bout de 10 ans, sans retrait intermédiaire.

Exercice 2.38 : Suite de Syracuse

On considère l'algorithme suivant :

```
Lire un entier et le stocker dans A
Tant que A != 1 :
    Afficher A
    Si A est pair :
        A prend la valeur A/2
    Sinon :
        A prend la valeur 3*A+1
```

1. Compléter le tableau de valeur ci-dessous. On considère que `A=6`

`A` en début de boucle	`A` est-il pair ?	`A` en fin de boucle
6	oui	3
3		
		1

2. Coder cet algorithme en `python` puis vérifier le résultat obtenu à la question précédente.

3. Modifier le code afin qu'il retienne la plus grand valeur de `A` atteinte.

4. Combien vaut la plus grande valeur de `A` lorsque l'on teste tous les entiers entre 2 et 100 ?

Exercice 2.39 :

On considère l'algorithme suivant :

```
A prend la valeur 9
Tant que A < 10 :
    Afficher "Salut"
```

1. Combien de fois est affiché le message ?

2. Modifier l'algorithme afin que le message soit affiché 9 fois.

3. Coder cet algorithme en python.

Exercice 2.40 :

La fonction python ci-dessous implémente l'algorithme d'Euclide. Cet algorithme permet de calculer le PGCD de deux entiers a et b.

```
def PGCD(a,b)
    r = a % b # reste de la division de a par b
    while r != 0 :
        a = b
        b = r
        r = a%b
    return b
```

1. Compléter le tableau de valeur ci-dessous correspondant à l'appel PGCD(35,24).

a	b	r
35	24	11

2. Que renvoie la fonction ?

3. Écrire l'algorithme permettant de calculer le PGCD de deux nombres en utilisant la méthode des différences successives. La méthode des différences successives consiste à calculer la différence du plus grand nombre par le plus petit, à garder le résultat et le plus petit nombre à chaque étape pour recommencer jusqu'à l'obtention de 0. Le PGCD est alors l'avant dernière différence.

Chapitre 3

Codage de nombres

Nombres entiers positifs

Exercice 3.1 : Binaire vers décimal

Convertir les nombres suivants du binaire vers la base 10 :

a. 101_2 c. 1011_2 e. 10110_2

b. 10001_2 d. 1010101_2 f. 1010111_2

Exercice 3.2 :

1. a. Écrire en base 10 le nombre 1111_2.

   b. Calculer 2^4. Que remarque-t-on ?

2. Quel est le plus grand nombre positif que l'on puisse coder sur un octet ?

3. Un ordinateur 32 bits peut manipuler des « mots » (nombres) de 32 bits de long au maximum. Quel est le plus grand entier positif manipulable par un tel ordinateur ?

4. Même question pour un ordinateur de 64 bits.

Exercice 3.3 : Base quelconque

Si N est un entier naturel supérieur ou égal à 2, l'écriture d'un nombre en base N n'utilise que les symboles 0, 1, 2, ..., $N-1$. Par exemple en base 10, on n'utilise que les symboles de 0 à 9.

1. L'écriture 200 peut-elle correspondre à un nombre écrit en base 3 ? Et 300 ?

Le principe d'une base quelconque N est identique à celui de la base 10 : à chaque position correspond un puissance de N. Ainsi :

$$200_3 = 2 \times 3^2 + 0 \times 3^1 + 0 \times 3 \times 0 = 18_{10}$$

2. Convertir les nombre suivants vers la base 10.

a. 41_5 c. 28_9 e. 221_3

b. 41_6 d. 100_6 f. 221_9

Exercice 3.4 : Droits d'accès à un fichier

Dans les systèmes basés sur *Linux*, chaque fichier est associé à des droits de lecture (`read`), d'écriture (`write`) ou d'exécution (`execute`).

Ces informations sont codées sur 3 bits où le bit de poids fort (le plus à gauche) indique les droits de lecture, celui du milieu les droits d'écriture et celui de droite, les droits d'exécution.

Ainsi le nombre `110` signifie que le fichier est en lecture et écriture mais pas exécutable. On l'abrège en `rw-`.

1. Quelle est la valeur de 110 en base 10 ?

2. Quel nombre en base 10 doit-on utiliser pour avoir tous les droits sur un fichier ?

On change les droits d'un fichiers en utilisant la commande `chmod` et en précisant successivement la valeur décimale souhaitée pour le propriétaire du fichier puis son groupe d'utilisateurs puis pour les autres utilisateurs.

Ainsi, si on tape `chmod 720 monfichier.txt` alors le propriétaire aura les droits associés à `7` (`rwx`), le groupe à `2` (`-w-`) et les autres utilisateurs à `0` (`---`).

3. Quels sont les droits du propriétaire, du groupe et des autres dans le cas de la commande `chmod 751 monfichier.txt` ?

4. Quelle commande doit-on saisir afin de donner tous les droits au propriétaire et au groupe et les seuls droits d'exécution aux autres utilisateurs ?

Exercice 3.5 : Décimal vers binaire

Convertir les nombre suivants de la base 10 au binaire.

a. 97_{10} c. 105_{10} e. 129_{10}

b. 512_{10} d. 3000_{10} f. 123456_{10}

Exercice 3.6 : Hexadécimal vers décimal

Convertir les nombre suivants de la base 16 vers la base 10.

a. $e5_{16}$ c. $36c_{16}$

b. $ab6_{16}$ d. fff_{16}

Exercice 3.7 :

Compléter le tableau suivant :

Base 10	Base 2	Base 16
58		
	1011	
		ec3
		2d
		3af

Exercice 3.8 :

Effectuer les additions suivantes en binaire (ne pas oublier que $1+1=10$ et reporter les retenues !)

$$\begin{array}{rcccc} & & 1 & 1 & \\ + & 1 & 0 & & \\ \hline \end{array} \qquad \begin{array}{rcccc} & 1 & 0 & 1 & 0 \\ + & & 1 & 1 & 1 \\ \hline \end{array}$$

$$\begin{array}{rccccc} & 1 & 0 & 1 & 1 & \\ + & 1 & 0 & 0 & 1 & \\ \hline \end{array} \qquad \begin{array}{rccccc} & 1 & 0 & 1 & 0 & 1 \\ + & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

Exercice 3.9 :

Le complément à 1 d'un nombre binaire est le nombre s'écrivant en inversant tous ses bits. Ainsi, le complément à 1 de 101 est 010.

1. Quel est le complément à 1 du nombre 75_{10} ?
2. Quel est le complément à 1 du nombre 100_{10} ?
3. Que peut-on dire du résultat (en binaire) de l'addition d'un nombre et de son complément à 1 ?

On note $\overline{n}$ le complément à 1 du nombre n.

4. Calculer $25+\overline{25}$ puis $25+\overline{25}+1$.
5. Calculer $89+\overline{89}+1$.

Nombres entiers négatifs

Sauf indication contraire, tous les nombres négatifs seront codés à l'aide de la méthode du complément à 2.

Exercice 3.10 :

1. Parmi les nombres binaires suivants, lesquels correspondent à un nombre entier négatif codé sur 8 bits ?

a. 11011001 b. 1000000 c. 101011 d. 00001111

2. Combien valent ces nombres négatifs ?

Exercice 3.11 :

Donner l'écriture binaire sur 8 bits des nombres suivants :

a. 126 c. -47 e. -99

b. -126 d. -109 f. -128

Exercice 3.12 :

On code les nombres sur 8 bits. On rappelle que $2^8 = 256$.

1. Quel est l'écriture binaire du plus grand nombre positif que l'on puisse écrire ? Combien vaut ce nombre en base 10 ?
2. Le plus petit entier négatif que l'on puisse écrire est $1000\ 0000_2$. Quelle est sa valeur en base 10 ?
3. Quel est le plus grand entier positif (en gardant « de la place » pour les négatifs) que l'on puisse coder sur 16 bits ? En le plus petit nombre négatif ?

Exercice 3.13 : Complément à 2 et soustraction

La méthode du complément à 2 est très pratique pour effectuer des soustractions. En effet au lieu de faire $126 - 47$ on peut faire $126 + (-47)$. On additionne alors les écritures binaires de 130 et -47.

Vérifier à l'aide de la notation binaire que $126 - 47 = 83$ puis que $48 - 55 = -7$.

Exercice 3.14 :

Le module `numpy` de `python`permet de coder les entiers sur 8 bits en utilisant le complément à 2.

Expliquer les bizarreries suivantes :

```
>>> from numpy import int8

>>> int8(127) + int8(1)
-128

>>> int8(127)+int8(127)
-2
```

Remarque : en réalité `numpy` affiche aussi le message suivant

```
RuntimeWarning: overflow encountered in byte_scalars
```

De plus, en `python` "classique", la taille du codage des nombres est automatiquement adaptée à la taille des nombres. Ces bizarreries ne se produisent donc pas.

Nombres à virgule flottante

Exercice 3.15 :

On rappelle que :

$$\begin{aligned} 101,01_2 &= 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} \\ &= 4 + 1 + 0,25 \\ &= 5,25 \end{aligned}$$

Convertir en base 10 les nombres binaires suivants :

a. $0,11_2$ c. $111,111_2$ e. $0,000001_2$

b. $11,001_2$ d. $100,001_2$ f. $0,000011_2$

Exercice 3.16 :

On rappelle la méthode de conversion en binaire des nombres décimaux.

Ainsi, Pour $3,8125$, on code la partie entière : $3_{10} = 11_2$ puis on code la partie décimale :

$$\begin{aligned} 2\times 0,8125 &= \mathbf{1},625 \\ 2\times 0,625 &= \mathbf{1},25 \\ 2\times 0,25 &= \mathbf{0},5 \\ 2\times 0,5 &= \mathbf{1},0 \end{aligned}$$

On obtient donc : $3,8125_{10} = 11,1101_2$

Convertir les nombres ci-dessous en binaire :

a. $9,75_2$ b. $0,125$ c. $15,375$ d. $5,5$

Exercice 3.17 :

1. Expliquer pourquoi un ordinateur ne peut pas coder le nombre $0,1$ en binaire.

2. Expliquer le calcul suivant proposé par python :

```
>>> 0.5-0.1-0.1-0.3
5.551115123125783e-17
```

Exercice 3.18 :

En 1991, lors de la première « Guerre du Golfe », les américains utilisaient des antimissiles Patriot afin d'intercepter les missiles irakiens. Le 25 février, un missile irakien ne fut pas intercepté et entraîna la mort de 28 soldats américains. Une enquête gouvernementale mis en

évidence une erreur humaine doublée d'une erreur d'arrondi dans l'écriture binaire du nombre 0,1.

En effet, tous les dixièmes de secondes, la batterie antimissile émet un signal permettant de compter le temps écoulé. Or l'écriture de 0,1 en binaire est inexacte :

$$0,1_{10} = 0,00011001100110011001\ldots_2$$

1. La batterie antimissile code les nombres décimaux en ne gardant que 23 décimales. Quelle est la valeur exacte du nombre $0,00011001100110011001100_2$?

2. Quelle est l'erreur par rapport à 0,1 ?

Normalement la batterie devait être réinitialisée tous les jours. Après l'accident on constata qu'elle n'avait pas été éteinte depuis plus de 100 heures.

3. Combien de cycle de 0,1 secondes se déroulent en 100 heures ?

4. Quelle était l'erreur cumulée de l'horloge ?

5. Sachant qu'un missile Scud se déplace à environ 1 600 m/s, quelle fut l'erreur d'estimation sur la position du missile ?

Chapitre 4

Les booléens

Exercice 4.1 :

On considère les propositions suivantes :

- A : Tous les élèves de la classe sont nés après 1995
- B : Tous les multiples de 10 sont des multiples de 5
- C : Un ordinateur est capable de mémoriser toutes les décimales de π

1. Quelle est la valeur de vérité de chacune de ces propositions ?

2. Quelle est la valeur de vérité des propositions suivantes ?

   a. A *et* B c. A *ou* C e. A *ou* B *ou* C

   b. A *et* C d. A *et* B *et* C f. A *et* B *et* $(nonC)$

Exercice 4.2 :

On souhaite utiliser `python` afin d'automatiser la création de la table de vérité de :

$$(A \text{ et } \overline{B}) \text{ ou } (\overline{A} \text{ et } B)$$

Cela peut se faire en saisissant les lignes suivantes :

```
for A in [True, .........] :
  for B in [........., .........] :
    print("A=", A, "et B=", B)
    print((A and not B) or (...................))
```

1. Saisir le code précédent en complétant les pointillés.

2. Construire la table de vérité de $(A \text{ et } \overline{B})$ *ou* $(\overline{A} \text{ et } B)$.

3. De quel opérateur classique s'agit-il ?

Exercice 4.3 :

Utiliser la technique de l'exercice 4.2 afin de construire la table de vérité de :

$$(A \text{ et } B) \text{ ou } (A \text{ et } \overline{C}) \text{ ou non}(\ B \text{ et } C)$$

Exercice 4.4 :

On considère deux variables python définies par :

```
n = 15
phrase = "George Boole est né en 1815"
```

Quel code python doit-on saisir afin de tester la vérité des propositions suivantes :

1. `n` est supérieur ou égal à 5
2. La lettre `"z"` apparaît dans phrase
3. `n` est compris entre 10 et 16 (tous les deux exclus)
4. Le mot `"oo"` apparaît dans `phrase`
5. `n` est dans la table de 7 (penser au modulo %)
6. La longueur de `phrase` est de 20 caractères (penser à `len`)
7. `n` est plus petit que 7 ou plus grand que 20 (au sens strict)
8. `n` n'est pas compris entre 7 et 20 (au sens large)

Exercice 4.5 :

Compléter le tableau suivant :

A	B	$non\ A$	$(non\ A))\ et\ B$	$B\ ou\ A$	$((non\ A))\ et\ B)\ ou\ (B\ ou\ A)$
0	0				
0	1				
1	0				
1	1				

Exercice 4.6 :

Pour un opérateur booléen à deux opérandes (deux "variables"), il existe $2^4 = 16$ combinaisons possibles (autant que l'on peut écrire de nombres sur 4 bits).

Par exemple la table de vérité associée à 0000 est celle de l'opérateur $Faux$, celle associée à 0001 est celle de et.

A	B	$Faux$
0	0	0
0	1	0
1	0	0
1	1	0

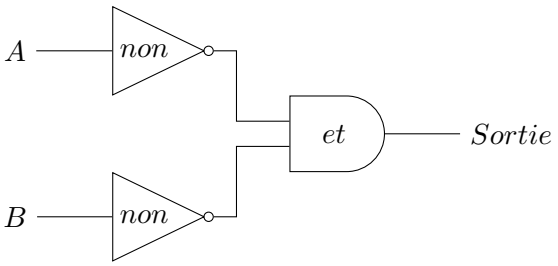
A	B	et
0	0	0
0	1	0
1	0	0
1	1	1

1. Lister les 14 autres tables de vérités possibles.

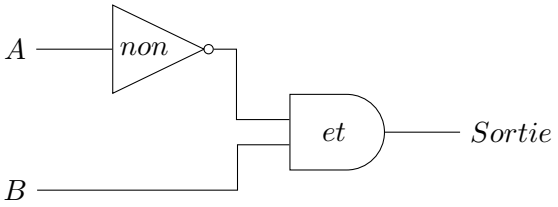
2. Chercher sur internet le nom de ces différents opérateurs (par exemple sur wikipedia)

Exercice 4.7 : Circuits logiques

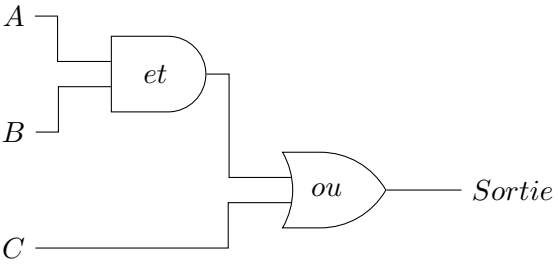
Compléter les tables de vérités des circuits logiques ci-dessous.



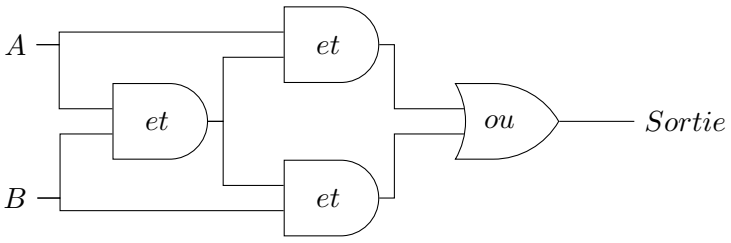
A	B	$Sortie$
0	0	
0	1	
1	0	
1	1	



A	B	$Sortie$
0	0	
0	1	
1	0	
1	1	



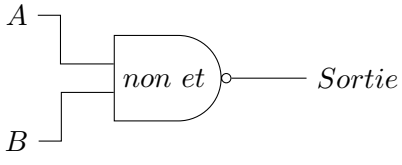
A	B	C	$Sortie$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



A	B	$Sortie$
0	0	
0	1	
1	0	
1	1	

Exercice 4.8 : Logique *non et*

On fournit ci-dessous le schéma et la table de vérite de la porte logique *non et*.



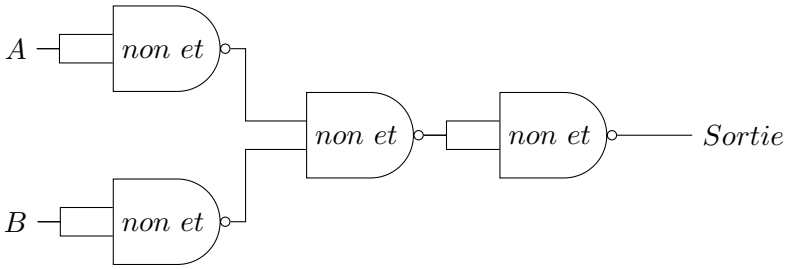
A	B	$Sortie$
0	0	1
0	1	1
1	0	1
1	1	0

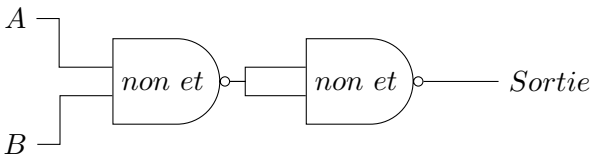
Cette porte permet à elle seule de construire les portes logiques suivantes :

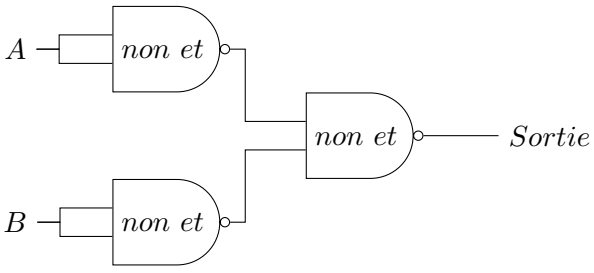
- *non*
- *et*
- *ou*
- *non ou*
- *ou exclusif* (ou XOR)
- *non ou exculsif*

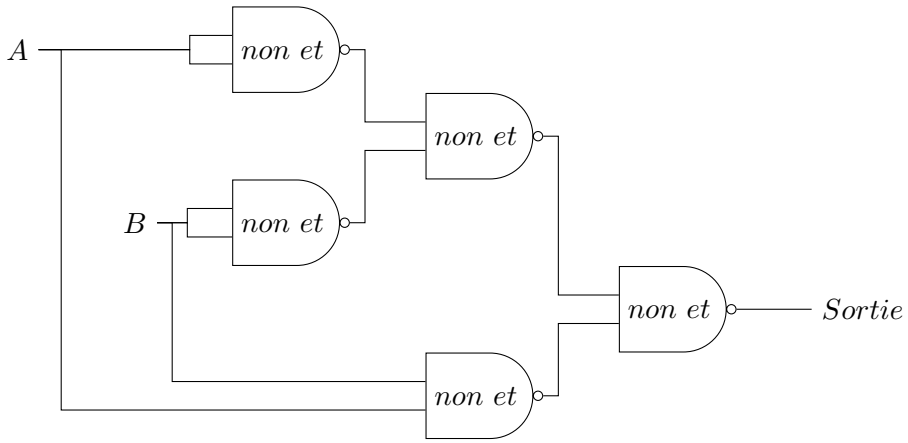
On fournit ci-dessous dans désordre les circuits permettant de construire ces portes à partir de portes *non et*.

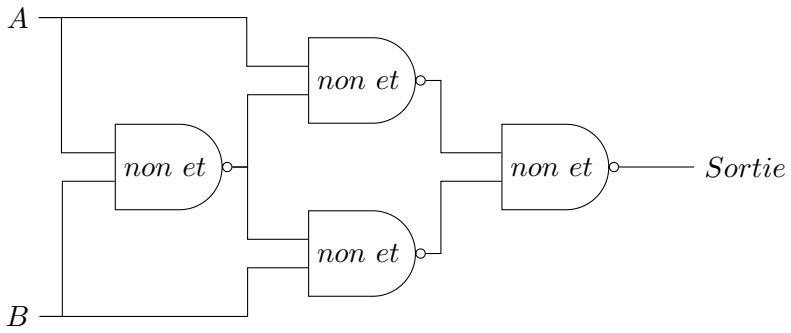
Associer chaque circuit à la bonne porte logique.








A
B
non et
non et
non et
non et
non et
Sortie


A
B
non et
non et
non et
non et
Sortie

Chapitre 5

Codage de textes

Exercice 5.1 :

Décoder les messages suivants encodés selon la table ASCII :

a. `76/97/32/116/97/98/108/101/32/65/83/67/73/73` (notation décimale)

b. `6c/61/20/62/61/73/65/20/31/36` (notation hexadécimale)

Exercice 5.2 : minuscules et MAJUSCULES en ASCII

1. Compléter le tableau à l'aide de la table ASCII.

Caractère	`A`	`a`	`M`	`m`	`Z`	`z`
Code (décimal)	65					
Code (hexadécimal)		61				
Code (binaire)			01001101			

2. Quel est l'écart (exprimé en décimal) entre les codes du `"A"` et du `"a"` ? Et entre le `"M"` et le `"m"` ?

3. De combien de bits diffèrent les codes binaires du `"B"` et du `"b"` ?

4. Expliquer le code python suivant :

```
>>> chr(ord("E") | 2**5)
'e'
```

Exercice 5.3 : Carré de Polybe

Le codage dit du carré de Polybe, remplace chaque caractère par sa position dans une grille définie à l'aide d'une clé (voir cet article de wikipedia). Voici la grille utilisée dans cet exercice (la clé est "POLYBE"). Par habitude, on omet le "W".

	0	1	2	3	4
0	P	O	L	Y	B
1	E	A	C	D	F
2	G	H	I	J	K
3	M	N	Q	R	S
4	T	U	V	X	Z

La lettre "O" est alors codée par `01` alors que le "E" est `10`.

1. Décoder le message suivant :

`0422103123014110`

2. Coder le carré de Polybe à l'aide d'une liste de liste dans Python. On pourra saisir `carre = [["P", "O", "L", "Y", "B"], ["E", ...], ...]`.

3. Coder une fonction `lettre` prenant en arguments deux nombres entiers `i` et `j` ainsi que le carré de Polybe et renvoyant la lettre du carré de Polybe située aux coordonnées `(i,j)`.

4. Recopier et compléter le code ci-dessous :

```
def decode_polybe(message : str, carre : list) -> str :
    """Décode le message crypté avec le carré de polybe"""
    clair = ""
    for indice in range(0, len(...), ...) :
        i = int(message[indice])
        j = ...
        clair = clair + lettre(i, j, carre)
    return ...
```

5. Décoder le message suivant :

`14100222122240114022013134404111343310413434 22`

Exercice 5.4 : Encodage UTF8 (1)

L'encodage UTF8 permet de réduire la taille des codes binaires des caractères courants. Son principe est le suivant :

- Les codes binaires sont à lire octet par octet
- Un octet débutant par un `0` peut s'interpréter seul. Dans le décodage, on ignore ce 0
- Un octet débutant par un `110` code le début d'un caractère qui s'étendra sur deux octets.. Dans le décodage, on ignore ce 110
- Un octet débutant par un `1110` code le début d'un caractère qui s'étendra sur trois octets.. Dans le décodage, on ignore ce 1110
- Un octet débutant par un `11110` code le début d'un caractère qui s'étendra sur quatre octets.. Dans le décodage, on ignore ce 11110
- Dans les trois cas précédents, les octets supplémentaires débutent tous par `10`. On ignore ces bits lors du décodage

Ainsi :

- `01000001` code un caractère sur un octet dont le code est $0100\ 0001_2 = 65_{10}$. C'est le `"A"`
- `11000011 10101001` code un caractère sur deux octets dont le code est $000\ 1110\ 1001_2 = 233_{10}$. C'est le `"é"`
- `11100010 10000010 10101100` code un caractère sur trois octets dont le code est :

$$0010\ 0000\ 1010\ 1100_2 = 8\ 364_{10}$$

C'est le "€".

1. Décoder le "caractère" suivant :

11110000 10011101 10000100 10011110

2. L'émoticône souriant a pour code unicode le 128 512.
   a. Convertir ce nombre en binaire.
   b. Combien de bits comporte ce nombre ?
   c. Ce caractère sera codé en UTF8 sur quatre octets. Déterminer son code binaire. On pourra ajouter des 0 au début du nombre afin de combler les octets.

Exercice 5.5 : Encodage UTF8 (2)

L'exercice 5.4 présente les bases de l'encodage UTF8. Le tableau ci-dessous présente le nombre d'octets sur lequel chaque caractère est codé en fonction de son code hexadécimal :

Premier code	Dernier code	Nombre d'octets du codage
`0000`	`007F`	1
`0080`	`07FF`	2
`0800`	`FFFF`	3
`10000`	`10FFFF`	4

Par exemple, le caractère ⋆ a pour code hexadécimal `2605` : il est donc codé sur 3 octets. En effet, en hexadécimal on a `0800` $<$ `2605` $<$ `FFFF`.

Comme on a $2605_{16} = 9733_{10} = 10011000000101_2$, on peut en déduire l'encodage UTF8 de ce caractère :

`11100010 10011000 10000101`

1. a. Le trèfle ♣ a pour code `2663`. Sur combien d'octets est-il codé ?
   b. Convertir le nombre hexadécimal `2663` en binaire.
   c. Écrire l'encodage UTF8 de ce caractère.
2. Quel est l'encodage UTF8 du symbole de paragraphe § dont le code en **décimal** est 167 ?

Exercice 5.6 : Chriffre de Vigenère

Dans le code de Vigenère (lien wikipedia), une lettre est remplacée par un nombre à l'aide d'une addition :

- On associe à chaque lettre son rang en débutant à 0 (`A` vaut 0, `B` vaut 1, ..., `Z` vaut 25)
- La "somme" de deux lettres est en fait le résultat de l'addition des deux rangs : `D + E = H` car $3 + 4 = 7$
- Si la somme dépasse le range 25, on boucle sur les premiers rangs à l'aide d'un modulo 26. `Z + B = A` car $25 + 1 = 26$ qui est égal à 1 "modulo" 26

Les deux lettres concernées par la somme sont la i-ième lettre du message et la i-ième lettre de la clé. Par exemple (en prenant `"MOT"` comme clé) :

Clé	M	O	T	M	O	T
Message en clair	C	R	Y	P	T	O
Message crypté	O	F	R	B	H	H

Comme on peut le voir, la clé est répétée autant de fois que nécessaire afin d'être aussi longue que le message.

Pour décoder on procède de la même façon en "soustrayant" les lettres.

Le but de cet exercice est de construire un programme décodant un message codé à l'aide de cette méthode.

1. Écrire en `python` une fonction `rang` prenant en argument une lettre (en majuscule) et renvoyant son rang dans l'alphabet. L'appel `rang("A")` doit ainsi renvoyer 0.

2. Écrire en `python` une fonction `lettre` prenant en argument un nombre entier (entre 0 et 25) et renvoyant la lettre de l'alphabet correspondante. L'appel `lettre(0)` doit ainsi renvoyer `"A"`.

3. Écrire en `python` une fonction `difference` prenant en arguments deux lettres (en majuscule) et renvoyant la lettre correspondant à leur difference. L'appel `difference("D", "M")` doit ainsi renvoyer "R".

4. Écrire en `python` une fonction `decode` prenant en arguments deux chaînes de caractères (la clé et le message crypté) et renvoyant le texte décrypté. L'appel `decode("MOT", "OFRBHH")` doit ainsi renvoyer `"CRYPTO"`.

5. Décoder le message suivant dont la clé est `"VIGENERE"` :

`XMIMRWKYIMVLEEJILCORRWVVOIXMRR`

Coup de pouce :

- Quel est le numéro de `"A"` dans la table ASCII ?
- En `python` La fonction `ord` renvoie le code ASCII d'un caractère
- En `python` La fonction `chr` renvoie le caractère associé à un code ASCII d'un caractère
- Saisir `(nombre % 26)` permet de calculer un modulo 26 et de "boucler" pour revenir à 0
- On pourra utiliser le code suivant afin d'ajuster la taille de la clé :

```
while len(cle) < len(message) :
    cle += cle
```

Chapitre 6

Les listes python

Listes "simples"

On rappelle les structures python suivantes :

```
# Création d'un liste vide
liste = []
# liste vaut []

# Création de la liste [3, 12, 7]
liste_bis = [3, 12, 7]

# Ajout de la valeur 5 en fin de liste
liste.append(5)
# liste vaut [5]

# Ajout de la valeur 8 en fin de liste
liste.append(8)
# liste vaut [5, 8]

# Ajout de la valeur 7 en fin de liste
liste.append(7)
# liste vaut [5, 8, 7]

# Création de la liste des 100 premiers entiers (0 à 99)
entiers = []
for i in range(100) :
    entiers.append(i)

# Accès la la valeur d'indice 0
val_0 = liste[0]
# val_0 vaut 5

# Récupération de la longueur de la liste ?
longueur = len(liste)
# longueur vaut 3 (3 éléments numérotés de 0 à 2)
```

```python
# La valeur 10 est-elle dans la liste ?
print(10 in liste)
# Affiche False

# Suppression de la dernière valeur
liste.pop()
# liste vaut désormais [5, 8]
# Cette instruction supprime ET renvoie valeur supprimée -> 7

# Suppression d'une valeur en fournissant son indice
liste.pop(0)
# liste vaut désormais [8]

# Parcours de listes...
liste = [8, 10, 12, 14]
# ...En itérant sur les indices
for indice in range(len(liste)) :
    print(liste[indice])
# ... En itérant directement sur les valeurs
for valeur in range(liste) :
    print(valeur)
```

Exercice 6.1 :

1. Créer la liste `[7, 8, 12, 25]`
2. Ajouter la valeur 30 en fin de liste.
3. Ajouter la valeur 10 à l'indice 2.
4. Ajouter la valeur 40 en fin de liste.
5. Supprimer le dernier élément de la liste.
6. Supprimer l'élément d'indice 1 de la liste.
7. Afficher la liste. Vous devez obtenir `[7, 10, 12, 25, 30]`.

Exercice 6.2 :

1. Créer la liste `[7, 8, 12]`.
2. Afficher tous les éléments à l'aide d'une boucle itérant sur les indices.
3. Afficher tous les éléments à l'aide d'une boucle itérant directement sur les valeurs.

Exercice 6.3 :

1. Créer la liste de tous les entiers entre 7 et 53.
2. Parcourir les éléments de cette liste et n'afficher que ceux qui sont dans la table de 7. On pourra utiliser un modulo `element % 7 == 0`.

Exercice 6.4 :

Écrire en `python` la fonction `fusion_liste` prenant en argument deux listes et renvoyant la liste créée en mettant la seconde liste à la suite de la première.

Par exemple `fusion([1, 2, 3], [20, 21])` renverra `[1, 2, 3, 20, 21]`.

Exercice 6.5 :

Il est possible de traiter une chaîne de caractères `python` comme une liste.

Ainsi si `texte = "bonjour le monde !"`, l'appel `texte[3]` renvoie `j`.

1. Créer la variable `texte` contenant la phrase "Je travaille sur les chaînes".

2. Afficher la $2^{ème}$ lettre de cette phrase.

3. Afficher successivement toutes les lettres de la phrase. On pourra utiliser une boucle `for`.

4. Compléter le code suivant qui copie les voyelles de la phrase dans une nouvelle liste.

```
voyelles = []
for lettre in ..... :
    if ..... in "aeiouy" :
        voyelles.append(.....)
```

Exercice 6.6 : Minimum et Maximum d'une liste

Déterminer l'élément minimum d'une liste de nombres nécessite :

- de créer une variable `minimum` initialisée à la première valeur de la liste
- parcourir tous les éléments de la liste à partir du $2^{ème}$ et :
  - comparer la valeur lue avec celle du minimum : si elle est plus strictement plus petite on met à jour la valeur du minimum
- En fin de boucle, on renvoie le minimum

1. Compléter la fonction `minimum_liste` ci-dessous :

```
def minimum_liste(liste : list) -> int :
    """Détermine et retourne le minimum de la liste"""
    # Cas où la liste est vide :
    if len(liste) == 0 :
        raise Exception("La liste est vide")

    # Initialisation
    minimum = liste[...]
    for indice un range(1, .....) :
        if liste[indice] ..... :
            minimum = .....

    return .....
```

2. Vérifier avec la liste `[3, 7, 1, 6]` que le code fonctionne.

3. Coder la fonction `maximum_liste` qui renvoie l'élément maximal d'une liste.

Exercice 6.7 : Filtrer les positifs

1. Écrire en `python` une fonction `compte_positifs` prenant en argument une liste de nombres et renvoyant le nombre de valeurs positives ou nulles de cette liste.
2. Écrire en `python` une fonction `nb_positifs` prenant en argument une liste de nombres et renvoyant une nouvelle liste formée des valeurs positives ou nulles de cette liste.

Exercice 6.8 : Filtrer selon une valeur

1. Écrire en `python` une fonction `compte_sup_n` prenant en argument une liste de nombres ainsi qu'un nombre `n` et renvoyant le nombre de valeurs de cette liste supérieures ou égales à `n`.
2. Écrire en `python` une fonction `nb_sup_n` prenant en argument une liste de nombres ainsi qu'un nombre `n` et renvoyant le valeurs de cette liste supérieures ou égales à `n`.

Exercice 6.9 :

Écrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. On pourra utiliser une boucle `for`.

Le programme doit ensuite afficher le nombre d'entiers supérieurs ou égaux à 10.

Exercice 6.10 :

Écrire en `python` la fonction `ecrase` prenant en argument une liste et un nombre `n`. Cette fonction remplacera toutes les valeurs d'indices supérieurs ou égaux à `n` par 0.

Par exemple `ecrase([3, 7, 8, 7, 9], 2)` renverra `[3, 7, 0, 0, 0]` alors que `ecrase[3, 7, 8, 7, 9], 5)` renverra `[3, 7, 8, 7, 9]`.

Exercice 6.11 :

Écrire en `python` la fonction `supprime` prenant en argument une liste et un nombre `n`. Cette fonction supprimera la première occurrence de `n` dans la liste en décalant toutes les valeurs vers le début et renverra la liste ainsi formée.

Par exemple `supprime([3, 7, 8, 7, 9], 7)` renverra `[3, 8, 7, 9]` alors que `supprime[3, 7, 8, 7, 9], 2)` renverra `[3, 7, 8, 7, 9]`.

Exercice 6.12 :

Écrire les fonctions `liste_croissante` et `liste_decroissante` qui prennent chacune en argument une liste et renvoie `True` si la liste est croissante (resp. décroissante) ou `False` dans le cas contraire.

Exercice 6.13 : Liste en 2D = liste de listes

Un tableau deux dimensions peut être représenté en `python` par une liste de listes. Par exemple :

3	2	5
7	-1	2

```
tableau = [[3, 2, 5], [7, -1, 1]]
```

Ainsi, pour accéder à la valeur `2`, on fait `tableau[0][1]` (première ligne, deuxième élément)

1. Comment accéder à la valeur `7` ? Et à `2` ?

2. Comment extraire la première ligne sous forme d'une liste ? Et la deuxième ligne ?

3. Compléter le code ci-dessous permettant de créer une liste reprenant les valeurs de la première colonne :

```
tableau = [[3, 2, 5], [7, -1, 1]]
colonne = []

for i in range(.................) :
    colonne.append( tableau[i][])
```

4. Écrire en `python` la fonction `nieme_colonne` qui prend en argument une liste des listes et un entier `n` et renvoie la `n`-ième colonne de la liste.

Exercice 6.14 :

On considère une liste de points du plan représentés par leurs coordonnées. Par exemple le point A (3 ; 5) est représenté par le tuple `A = (3,5)`. On accéder à son abscisse (sa première coordonnée) en faisant `A[0]` et à son ordonnée (sa seconde coordonnée) avec `A[1]`.

Une liste de points sera par exemple `points = [[3, 5], [1, -2], [-7, 0]]`.

1. Écrire en `python` la fonction `abscisses` prenant en argument une liste de points et renvoyant la liste de leurs abscisses.

   Par exemple, `abscisses([[3, 5], [1, -2], [-7, 0]])` renverra `[3, 1, -7]`.

2. Écrire la fonction `coordonnees` prenant en argument une liste de points et renvoyant deux listes : la première contenant les abscisses des points et la seconde leurs ordonnées.

   Par exemple, `coordonnees([[3, 5], [1, -2], [-7, 0]])` renverra `[3, 1, -7], [5, -2, 0]`.

Exercice 6.15 : Morpion

On souhaite coder une fonction vérifiant si une grille de morpion est gagnante ou non.

La grille sera codée sous la forme d'une liste de listes `python`. Par exemple :

X	O	X
O	X	O
␣	O	X

```
grille = [["X", "O", "X"],
          ["O", "X", "O"],
          [" ", "O", "X"]]
```

La grille aura toujours pour dimensions 3×3

1. a. Quel test `python` doit-on effectuer pour vérifier que la première ligne de la grille comporte trois `X` ?

   b. Quel test `python` doit-on effectuer pour vérifier que la deuxième colonne de la grille comporte trois `X` ?

   c. Quel test `python` doit-on effectuer pour vérifier que la première diagonale (démarrant en haut à gauche) de la grille comporte trois `X` ?

2. Écrire en `python` la fonction `victoire` prenant en argument une grille et un jeton à tester (`X` ou `O`) et renvoyant `True` si la grille comporte un alignement victorieux (ligne, colonne ou diagonale) pour ce jeton, `False` dans le cas contraire.

Listes par compréhension

On rappelle les structures `python` suivantes :

```
# Création d'une liste comprenant 10 fois la valeur 0
zeros = [0 for i in range(10)]

# Création de la liste des entiers de 1 à 10
entiers = [i for i in range(1, 11)]

# Création de la liste des entiers pairs de 2 à 100...
pairs = [i for i in range(2, 101, 2)]
#... ou
pairs_bis = [2*i for i in range(1, 51)]
# car 2*1=2, 2*2 = 4, ..., 2*50=100

# Liste par compréhension et tests
# Tous les entiers entre 0 et 1000
entiers = [i for i in range(0, 1001)]
# On ne garde que les entiers de la table de 77
table_77 = [nombre for nombre in entiers if nombre % 77 == 0]

# On peut aussi parcourir une autre liste...
liste = [3, 8, 9, 5, -2]
sup_6 = [nombre for nombre in liste if nombre >= 6]

# ...Ou parcourir une chaîne de caractères...
phrase = "Vive les Listes Par Compréhension"
majuscules = [lettre for lettre in phrase if lettre.isupper()]

# ...Ou une liste de listes
liste = [[3, 5], [1, -2], [-7, 0]]
premieres = [ couple[0] for couple in liste]
deuxiemes = [ couple[1] for couple in liste]
```

Exercice 6.16 :

Utiliser des listes par compréhension afin de créer les listes suivantes :

1. La liste des 20 premiers entiers (de 0 à 19)
2. La liste des 20 premiers entiers pairs (de 0 à 38)
3. La liste des 20 premiers entiers impairs (de 1 à 39)
4. La liste des voyelles du mot `ornithorynque`

5. La liste des consonnes du mot `ornithorynque`

6. La liste des tous les entiers inférieurs à 100 (inclus) dans la divisibles par 5 et 7

Exercice 6.17 : Filtre

Écrire en `python` la fonction `filtre` prenant en argument une liste de nombre et un nombre `n` et renvoyant la liste des nombres de la listes strictement supérieurs à `n`.

Exercice 6.18 : Un `map` à la main (1)

1. On considère une liste `python` de nombres. Comment créer une liste contenant les doubles de ces nombres en utilisant une liste par compréhension ?

2. Écrire en `python` la fonction `fois_n` prenant en argument une liste de nombre et un nombre `n` et renvoyant la liste des mêmes nombres multipliés par `n`.

Exercice 6.19 : Un `map` à la main (2)

Le code ci-dessous affiche `[10, 14]`.

```
def double(x : int) -> int :
    return 2*x

def applique_f(liste : list, f : callable) -> list
    """ Applique la fonction f à tous les éléments de la liste"""
    return [f(element) for element in liste]

print(applique([5, 7], double))
```

1. a. Que renvoie l'appel `applique([0, 6, 9], double)` ?

   b. Que renvoie l'appel `applique(applique([0, 6, 9], double), double)` ?

2. Modifier ce code afin qu'il multiple tous les nombres par 8.

Chapitre 7

Les dictionnaires python

On rappelle les structures python suivantes :

```
# Création d'un dictionnaire vide
dico = {}

# Création d'un dictionnaire en passant les clés : valeurs
dico = {"clé1" : "valeur1", "clé2" : "valeur2"}

# Accès à une valeur
val_1 = dico["clé1"]

# Ajout (ou modification) d'un valeur
dico["clé3"] = "valeur3"

# Parcours des clés :
for cle in dico : # ou "for cle in dico.keys()"
    print(cle)
    print(dico[cle])

# Parcours des valeurs
for valeur in dico.values() :
    print(valeur)

# Parcours des couples clé,valeur
for cle,valeur in dico.items() :
    print(cle)
    print(valeur)
```

Exercice 7.1 :

1. Créer le dictionnaire {0 : "a", 1 : "b", 2 : "c"}.
2. Afficher la valeur associée à la clé 0.
3. Ajouter la clé 3 associée à la valeur "d".
4. Afficher toutes les clés à l'aide d'une boucle for.
5. Afficher toutes les valeur à l'aide d'une boucle for.

6. Afficher tous les couples `clés : valeur` à l'aide d'une boucle `for`.

Exercice 7.2 :

Créer le dictionnaire de la table de 5 : `{0 : 0, 1 : 5, 2 : 10, 3 : 15, ..., 10 : 50}`.

Exercice 7.3 :

Créer le dictionnaire des 100 premiers carrés : `{1 : 1, 2 : 4, 3 : 9, ..., 100 : 10 000}`.

Exercice 7.4 :

On considère un dictionnaire `dico` dont toutes les valeurs sont des nombres.

Écrire le code soustrayant 8 à chacune de ces valeurs. On passera ainsi de `{"a" : 7, "b" : 0}` à `{"a" : -1, "b" : -8}`.

Exercice 7.5 :

Créer le dictionnaires `dizaines = {0 : [0,1,2,3,4,6,7,8,9], 10 : [10,11,...,19], 90 : [90, 91,...,99]}`.

Exercice 7.6 :

Créer le dictionnaire représentant un échiquier en début de partie :

- les lignes sont numérotées de 1 à 8, les colonnes de A à G
- les blancs sont sur la ligne 1
- les pièces sont stockées sous forme de tuples, par exemple `("dame", 0)` pour la dame blanche et `("roi", 1)` pour le roi noir

On aura ainsi `echiquier = {(1,"A") : ("tour", 0), ..., (8,"G") : ("tour", 1)}`.

Exercice 7.7 :

Écrire en `python` la fonction `table` prenant en argument un entier positif et renvoyant le dictionnaire correspondant à la table de cet entier.

Par exemple `table(5)` renvoie `{0 : 0, 1 : 5, 2 : 10, 3 : 15, ..., 10 : 50}`.

Exercice 7.8 :

Écrire en `python` la fonction `occurrences` prenant en argument une chaîne de caractères et renvoyant le dictionnaire associant à chaque caractère son nombre d'apparitions dans la chaîne.

Par exemple `occurrences("Ceci est une phrase !")` renvoie `{"C" : 1, "e" : 4, "c" : 1, ..., "!" : 1}`.

Exercice 7.9 :

Écrire en `python` la fonction `somme_valeurs` prenant en argument un dictionnaire dont les valeurs sont toutes "sommables" et renvoyant la somme de ses valeurs.

Par exemple `somme_valeurs({"pommes" : 200, "fraises" : 100})` renvoie 300.

Exercice 7.10 :

Écrire en `python` la fonction `mini_dico` prenant en argument un dictionnaire et renvoyant le couple `(cle, valeur)` correspondant à la valeur minimale dans le dictionnaire.

Par exemple `somme_valeurs({"pommes" : 200, "fraises" : 100})` renvoie `("fraise", 100)`.

Exercice 7.11 :

Une association stocke les commune de résidence de ses adhérents sous forme d'un dictionnaire.

Par exemple `{"Pierre" : "Paris", "Manon" : "Marseille", "Marc" : "Marseille"}`.

Écrire en `python` la fonction `valeurs` prenant en argument un dictionnaire et renvoyant la liste des valeurs **sans doublons** de ce dictionnaire.

Par exemple l'appel :

```
valeurs({"Pierre" : "Paris", "Manon" : "Marseille", "Marc" : "Marseille"})
```

renvoie `["Paris", "Marseille"]`.

Exercice 7.12 :

Écrire en `python` la fonction `concatene` qui prend en argument les deux dictionnaires et renvoie le dictionnaire formé en regroupant leurs clés. Si une clé est présente dans les deux dictionnaires la valeur associée sera celle du second.

Par exemple `concatene({1 : "a", 2 : "b"}, {3 : "c"})` renverra `{1 : "a", 2 : "b", 3 : "c"}`. De même, `concatene({1 : "a", 2 : "b"}, {2 : "c"})` renverra `{1 : "a", 2 : "c"}`.

Exercice 7.13 :

On décrit le contenu du stock d'un entrepôt à l'aide d'un dictionnaire.

Par exemple `{"pommes" : 200, "fraises" : 100}` signifie que l'on stocke 200 kg de pommes et 100 de fraises.

On souhaite "regrouper" deux entrepôts/dictionnaires en cumulant les valeurs associées à des clés identiques. Ainsi, si les deux entrepôts stockent des pommes, le dictionnaire créé aura pour valeur associée à `"pomme"` la somme des deux valeurs.

Écrire en `python` la fonction `regroupe` qui prend en argument les deux dictionnaires et répond à la question.

Par exemple l'appel :

```
regroupe({"pommes" : 200, "fraises" : 100}, {"pommes" : 100, "oranges" : 150})
```

renverra `{"pommes" : 300, "fraises" : 100, "oranges" : 150}`.

Exercice 7.14 : Dictionnaire de dictionnaires

Un lycée stocke les informations de ses élèves dans un dictionnaire dont les clés sont les noms des élèves (on suppose qu'il n'y a pas de doublons) et les valeurs leurs informations.

Par exemple :

```
lycee = {"Jean Dupont" : {"classe" : "1e1", "LVA" : "anglais", ...},
         "Romane Simon": {"classe" : "1e3", "LVA" : "allemand", ...}
        }
```

1. Comment afficher la liste des noms des élèves du lycée ?

2. Comment afficher la classe de `"Jean Dupont"` ?

3. a. Écrire le code comptant le nombre d'élèves au lycée.

   b. Écrire le code comptant le nombre d'élèves en Première 1.

Exercice 7.15 : Dictionnaire vers *JSON*

Le format *JSON* est un format de données textuelles s'apparentant à la notation des données en *Javascript*. A ce titre, son acronyme signifie *JavaScript Object Notation*.

Par exemple, pour les données décrites dans l'exercice 7.14 on obtient :

```
"eleves": [
            {"nom" : "Jean Dupont", "classe" : "1e1", "LVA" : "anglais"},
            {"nom" : "Romane Simon", "classe" : "1e3", "LVA" : "allemand"}
          ]
```

Ecrire le code transformant le dictionnaire en un texte au format *JSON*.

Exercice 7.16 : Dictionnaire vers *YAML*

Le format *YAML* est encore un format de données textuelles. Son acronyme signifie *YAML Ain't Markup Language*.

Dans ce format, c'est l'indentation qui organise l'information.

Par exemple, pour les données décrites dans l'exercice 7.14 on obtient :

```
eleves:
    - nom : Jean Dupont
      classe : 1e1
      LVA : anglais
    - nom : Romane Simon
      classe : 1e3
      LVA : allemand
```

Ecrire le code transformant le dictionnaire en un texte au format *YAML*.

Chapitre 8

Les tris

Exercice 8.1 :

Une implémentation de l'algorithme de tri par sélection met 600 ns pour trier un tableau de 100 nombres entiers.

1. Quelle sera la durée du tri d'un tableau de 200 nombres ?

2. Compléter le tableau ci-dessous.

Taille des données	Durée approchée du tri
100	600 ns
200	
400	
1 000	
2 000	
100 000	
1 000 000	

Exercice 8.2 :

Une implémentation de l'algorithme de tri par insertion met 900 ns pour trier un tableau de 100 nombres entiers.

1. Quelle sera la durée du tri d'un tableau de 200 nombres ?

2. Compléter le tableau ci-dessous.

Taille des données	Durée approchée du tri
100	900 ns
200	
500	
1 000	
5 000	
10 000	
1 000 000	

Exercice 8.3 :

On a débuté le tri d'un tableau de nombres. La composition de la liste initiale et après les premier et second tours de boucle est donné ci-dessous.

Tour de boucle	État de la liste
0 (liste à trier)	`[7, 2, 9, 3, 5]`
1	`[2, 7, 9, 3, 5]`
2	`[2, 7, 9, 3, 5]`

1. S'agit-il d'un tri par insertion ou sélection ?

2. Compléter la fin du tableau jusqu'à obtenir un liste triée.

Exercice 8.4 :

On considère les listes ci-dessous :

```
A = [3, 9, 8, 7, 2]
B = [8, 7, 9, 2, 3]
C = [9, 8, 7, 3, 2]
```

1. Trier ces listes à l'aide du tri par insertion puis du tri par sélection la main en notant l'état de la liste après chaque itération. Par exemple, après une itération du tri par sélection on a `A = [2, 9, 8, 7, 3]`.

2. Compter le nombre de comparaisons et d'échanges de valeurs effectués dans chaque cas.

Exercice 8.5 :

On considère le tri par sélection d'un liste de 5 nombres. Proposer une liste telle que :

a. on n'effectue aucun échange de valeurs

b. on effectue seulement un échange de valeurs

c. on effectue exactement deux échanges de valeurs

d. on effectue un maximum d'échanges. Combien en réalise-t-on ?

Exercice 8.6 :

Reprendre l'exercice 8.5 dans le cas du tri par insertion.

Exercice 8.7 :

Écrire le code `python` d'une fonction triant une liste de nombres dans l'ordre **décroissant**. On utilisera le tri par insertion.

Exercice 8.8 :

Écrire le code `python` d'une fonction triant une liste de nombres dans l'ordre **décroissant**. On utilisera le tri par sélection.

Exercice 8.9 : Tris de chaînes de caractères (1)

En `python` la comparaison de caractères se fait à l'aide du code *Unicode* de chaque caractère. Ainsi `"a"` est supérieur à `"A"` car leurs "numéros" respectifs dans la table unicode sont 97 et 65 (en décimal). De façon générale, les lettres en minuscule ont un code supérieur aux lettres majuscules. Au sein des lettres de même "casse" (majuscule ou minuscule), les caractères sont classés dans l'ordre alphabétique.

Lorsqu'il compare deux chaînes de caractères, `python`compare les premiers caractères. Si ceux-ci sont égaux, il passe aux deuxièmes caractères *etc.*...

Le mot `"tris"` peut s'écrire de 16 façons différentes en utilisant des caractères en majuscule ou minuscule. Par exemple `"tris"` est différent de `"triS"`.

Lister les 16 écritures différentes de `"tris"` et les trier dans l'ordre croissant comme le ferait `python`.

Exercice 8.10 : Tris de chaînes de caractères (2)

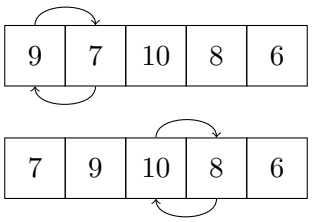
Écrire en `python` la fonction `compare_chaines` prenant en arguments deux chaînes de caractères `c1` et `c2` et renvoyant le tuple formé des deux chaînes classées dans l'ordre croissant (la plus petite des chaînes sera la première valeur du tuple).

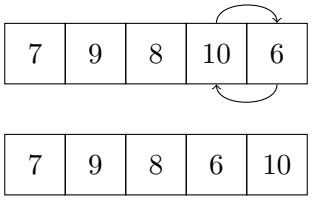
On pourra s'inspirer du code incomplet suivant :

```
Fonction compare_chaines(c1, c2) :
    taille1 est égal à la longueur de c1
    taille2 est égal à la longueur de c2
    taille_mini est égal au minimum de taille1 et taille2
    i prend la valeur 0 # indice étudié
    Tant que i est strictement inférieur à taille_mini :
        Si c1[i] < c2[i] :
            Renvoyer c1,c2
        Sinon si c1[i] > c2[i] :
            Renvoyer c2,c1
        Sinon :
            i prend la valeur i+1

    # On a atteint la fin d'une des deux chaînes
    Si i > taille1 - 1 :
        Renvoyer c1, c2
    Sinon :
        Renvoyer c2, c1
```

Exercice 8.11 : Tri à bulles

Dans le tri à bulle, à chaque itération on fait "remonter" les maximums de chaque paire de nombre :





Son pseudo-code est le suivant (la liste à trier est `A`) :

```
echange prend la valeur Vrai
Tant que echange est Vrai :
    echange prend la valeur Faux
    Pour i allant de 1 à (longueur de A - 1) (inclus) :
        Si A[i-1] > A[i] :
            Echanger les valeurs A[i-1] et A[i]
            echange prend la valeur Vrai
```

Ainsi la figure précédente illustre la première itération de la boucle "Tant que". On y compare les paires `(9, 7)` (échange), `(9, 10)` (pas d'échange), `(10, 8)` (échange) et `(10, 6)` (échange).

A l'issue de ce tour de boucle, la valeur maximale est en dernière position.

1. Écrire l'état de la liste après le deuxième tour de boucle.

2. a. Écrire l'état de la liste après chaque tour de boucle.

   b. Combien d'itération ont été nécessaires ?

3. Coder cette méthode de tri en `python`.

4. On se rend compte que chaque itération place un élément maximal à sa place définitive. Il est dès lors inutile de l'étudier à nouveau lors des "échanges". Proposer une amélioration du tri à bulles.

Exercice 8.12 : Plusieurs valeurs à trier

Il arrive régulièrement que l'on souhaite trier des éléments possédant plusieurs valeurs. Par exemple des triplets de nombres : si les éléments d'indice `0` sont égaux, on compare ceux d'indice `1` et si ceux-ci sont aussi égaux on passe aux éléments d'indice `3`.

1. Trier la liste de triplets suivante : `[(3,1,2), (7,1,2), (7,1,0), (3,5,2)]`.

2. Adapter le tri par insertion afin qu'il trie des tuples de trois nombres comme ci-dessus.

Exercice 8.13 : Tri stable

Un algorithme de tri est dit **stable** si deux éléments égaux classé dans un certain ordre avant le tri se retrouvent dans le même ordre à la fin du tri.

Afin de visualiser ce concept on considère le tri de la liste `[(2,"B"), (1, "A"), (2, "C")]`. un tri stable renverra `[(1, "A"), (2,"B"), (2, "C")]` alors qu'un trie instable renverra `[(1, "A"), (2,"C"), (2, "D")]`.

1. Les tris par insertion et sélection sont-ils stables ? Quel aspect du code nous assure de cette qualité ?

Exercice 8.14 : Dans d'autres langages

On fournit ci-dessous le code en `javascript` et en `C` de quatre fonctions `tri_A`, `tri_B`, `tri_C` et `tri_D`.

Deux de ces fonctions correspondent au tri par insertion, les deux autres au tri par sélection.

Identifier les fonctions correspondantes.

Les codes en `javascript` :

```javascript
function tri_A(A){
    for(let i = 1; i < A.length; i++){
        for(let j = i - 1; j > -1; j--){
            if(A[j + 1] < A[j]){
                [A[j+1], A[j]] = [A[j], A[j + 1]]
            }
        }
    }
    return A;
}

function tri_B(A) {
let min

for (let i = 0; i < A.length; i++) {
    min = i
    for (let j = i + 1; j < A.length; j++) {
        if (A[j] < A[min]) {
            min = j
        }
    }
    if (min !== i) {
        [A[i], A[min]] = [A[min], A[i]]
        }
    }
    return A;
}
```

Les codes en `C` (qui ne retournent pas le tableau mais le modifient "en place") :

```c
void tri_C(int A[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = A[i];
        j = i - 1;
        while (j >= 0 && A[j] > key) {
            A[j + 1] = A[j];
            j = j - 1;
        }
        A[j + 1] = key;
    }
}

void tri_D(int A[], int n)
{
```

```
    int i, j, min_idx;
    for (i = 0; i < n-1; i++)
    {
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (A[j] < arr[min_idx])
            min_idx = j;
        int temp = A[min_idx];
        A[min_idx] = A[i];
        A[i] = temp;
    }
}
```

Exercice 8.15 : Tri indirect

Il peut arriver que lors d'un trie on ne modifie pas la liste initiale mais que l'on se contente de trier les indices des éléments. Cette situation peut se rencontrer si les données sont trop lourdes à déplacer ou si l'on ne souhaite pas modifier le tableau initial.

Ainsi le liste `liste = [5,3,6]` a pour tableaux d'indices associé **avant le tri** `indices = [0,1,2]`. Après le tri, la liste est inchangée mais son tableau d'indices associé devient `indices = [1,0,2]`. Cela signifie que l'élément de `liste` d'indice `1` est le minimum de la liste et que celui d'indice `2` est le maximum.

On aura donc toujours (pour `i` compris entre `0` et `len(liste)-2`) : `liste[indice[i]] <= liste[indice[i+1]]`.

Adapter le tri par sélection pour en faire un tri indirect se fait simplement : on effectue toujours les comparaisons sur les éléments de la liste à trier mais lors du déplacement de la valeur minimale, on travaille sur le tableau des indices.

Ecrire le code `python` du tri par sélection indirect.

Chapitre 9

Données en table

Exercice 9.1 :

On ouvre le fichier `broadway.csv` contenant des informations sur les spectacles de Broadway. Les premières lignes du fichiers sont données ci-dessous (les effectifs sont ceux d'une semaine entière de représentations) :

```
Jour,Date,Mois,Année,Nom_Spectacle,Salle,Type,Effectif_Public,Remplissage_%
26,8/26/1990,8,1990,Tru,Booth,Pièce,5500,88
24,3/24/1991,3,1991,Miss Saigon,Broadway,Comédie Musicale,1737,99.5
31,3/31/1991,3,1991,Miss Saigon,Broadway,Comédie Musicale,12160,100
```

1. Rappeler le sens de l'acronyme *csv*.

2. Quel est le séparateur dans ce fichier.

3. a. Que contient la ligne 1 ?

   b. Combien ce fichier contient-il de champs ?

4. a. A partir de quelle ligne commencent les données ?

   b. Le fichier contient en tout 31 197 lignes. Combien de données comporte-t-il au total ?

5. a. Comment s'appelle le spectacle de la ligne 2 ?

   b. Combien de personnes ont assisté à ce spectacle durant une semaine ?

Exercice 9.2 : Import dans une liste de listes

On souhaite importer dans `python` les données du fichier `broadway.csv` décrit dans l'exercice 9.1.

Les données seront codées sous forme d'une liste de listes `python`.

Un utilisateur a codé son import comme ci-dessous. Le fichier `python` est enregistré dans le même dossier que le fichier *csv*.

```
spectacles = []
with open("broadway", "read") as fichier :
    for ligne in fichier :
        valeurs = lignes.split(;)
        spectacles.append(ligne)
```

Il obtient plusieurs messages d'erreurs...

1. Recopier et corriger ce code.
2. La première ligne du fichier *csv* contient les noms des champs. Comment peut-on la supprimer de la liste `spectacles` ?
3. Quel type `python` doit-on donner à chaque champs ?
4. Compléter le code ci-dessous permettant de typer chaque champ.

```
for s in spectacles :
    for i in range(9) :
        if i in [0, 2, 3 ,7] :
            s[i] = int(..............)
        elif i == ........ :
            s[i] = ........................
```

5. Lors de l'affichage des données importées, l'utilisateur observe la sortie `ComÃ©die Musicale`. Que peut-on lui suggérer ?

Exercice 9.3 : Import dans une liste de dictionnaires

On souhaite importer dans `python` les données du fichier `broadway.csv` décrit dans l'exercice 9.1.

Les données seront codées sous forme d'une liste de dictionnaires `python`. Chaque clé seront le nom d'un champ et la valeur associée sera celle de ce champ.

Compléter le code ci-dessous permettant d'importer et de typer les données :

```
spectacles = []
with open("broadway.csv", "r") as fichier :
    champs = fichier.readline()..................
    for ligne in fichier :
        valeurs = lignes.split(............)
        dico = {}
        for i in range(len(champs)) :
            if i in [0, 2, 3 ,7] :
                dico[..............[i]] = int( valeurs[........])
            elif i == 8 :
                dico[..............[i]] = .........( valeurs[........])
            else :
                ....................................................
        spectacles.append(...................)
```

Exercice 9.4 :

On dispose d'une liste `python` nommée `dicos` contenant des dictionnaires. Tous ces dictionnaires ont tous les mêmes clés.

1. Quelle instruction `python` permet de déterminer le nombre de dictionnaires présents dans `dicos` ?
2. Quelle instruction `python` permet de récupérer la liste des clés des dictionnaires de la liste `dicos` ?

On souhaite enregistrer cette liste sous forme d'un fichier *csv* dont la première ligne contiendra les noms des clés des dictionnaires et les lignes suivantes, les valeurs de chaque dictionnaire (dans l'ordre des clés) séparées par des virgules.

On rappelle que les instructions `python` suivantes :

```
dico = {"a":1,"b":2}

# Récupération des clés de dico (peut être parcouru avec une boucle for)
clés = dico.keys()
# Récupération des valeurs de dico (peut être parcouru avec une boucle for)
valeurs = dico.values()

liste = ["un", "deux", "trois"]
# Création d'une chaîne contenant tous les termes de liste
# séparés par des "//"
"//".join(liste)
# Renvoie : "un//deux//trois"
```

Compléter le code `python` ci-dessous permettant d'effectuer cette action :

```
with open("sortie.csv", "w") as fichier :
    # La première ligne avec les clés
    clés = dicos[0].
    texte_clés = ..... + "\n
    fichier.write(texte_clés)

    # Les données
    for dico in dicos :
        valeurs = .....
        texte_valeurs = ..... + "\n"
        ...............................................
```

Exercice 9.5 :

On ouvre le fichier `cars.csv` contenant des informations sur des voitures.

Les premières lignes du fichiers sont données ci-dessous :

```
Hauteur,Longueur,Largeur,Traction,Moteur,Hybride
140,143,202,4x4,Audi 3.2L 6 cylinder 250hp 236ft-lbs,True
172,63,87,4x4,Acura 3.7L 6 Cylinder 305 hp 273 ft-lbs,True
226,243,201,4x4,BMW 4.4L 8 cylinder 555hp 500 ft-lbs Turbo,True
24,216,133,Arrière,Toyota 4.0L 6 Cylinder 270 hp 278 ft-lbs,True
```

1. a. Quels sont les champs ?

   b. Quels sont leur type `python` ?

On importe ces données dans une liste de dictionnaires nommée `voitures`. On dispose d'une fonction `filtre` qui prend en argument la liste des données ainsi qu'une requête et renvoie la liste des véhicule satisfaisant cette requête. Par exemple `filtre(voitures, "voiture['Largeur']<200")` permet de sélectionner les véhicules de moins de 200 centimètres de large.

Les requêtes portent toujours sur l'argument `voiture`.

2. Écrire les requêtes permettant de sélectionner les véhicules ci-dessous :

a. Les véhicules de plus de 250 centimètres de long.

b. Les véhicules de moins de 2 mètres de long.

c. Les véhicules de moins de 2 mètres de long et de large.

d. Les véhicules de marque Audi.

e. Les véhicules hybrides.

f. Les véhicules non hybrides.

g. Les véhicules hybrides de marque Audi.

h. Les véhicules de marque BMW mais non 4x4.

Exercice 9.6 :

On a récupéré les données d'un lycée sous la forme des fichiers *csv* suivants :

- `eleves.csv` : contient les informations des élèves (id, nom, prénom, date de naissance, classe, ...)
- `professeurs.csv` : contient les informations des professeurs (id, nom, prénom, ...)
- `classes.csv` : contient les informations des classes (id, nom de la classe,id du professeur principal, id des professeurs, ...)

On fournit ci-dessous les premières lignes de ces différents fichiers :

- `eleves.csv`

```
id,nom,prénom,date_naissance,classe
1,Rivière,Jean,30/01/2003,1
2,Allaire,Rose,25/05/2004,2
```

- `professeur.csv`

```
id,nom,prénom
1,Hillion,Grace
2,Turinge,Alain
```

- `classes.csv`

```
id,nom,id_pp,id_français,id_eps
1,1e3,1,2,1
2,1e7,2,2,1
```

1. Observer les données et répondre aux questions ci-dessous :

a. En quelle classe est Rose Allaire ?

b. Qui est le professeur principal de la 1ère 7 ?

c. Qui est le professeur de français de la 1ère 3 ?

Les fichiers ont été importés dans trois listes de dictionnaires `python eleves`, `professeurs` et `classes`

Le CPE du lycée souhaite créer un nouveau fichier *csv* listant les élèves du lycée mais en ajoutant le nom de la classe de l'élève à la place de l'id de sa classe.

2. Compléter le code ci-dessous répondant à sa requête :

```
for eleve in eleves :
    for classe in classes :
        if eleve['..........'] == classe['..........'] :
            eleve['classe'] = classe['..........']
            break
```

Le CPE a une nouvelle requête : il souhaite compléter la liste `eleves` en ajoutant le nom du professeur principal de la classe.

Il va désormais falloir mettre en correspondance les trois listes.

3. a. Quels champs des listes `eleves` et `classes` faut-il mettre en correspondance ?

   b. Quels champs des listes `classes` et `professeurs` faut-il mettre en correspondance afin de trouver le nom du professeur principal ?

   c. Écrire le code `python` répondant à sa requête.

Chapitre 10

Systèmes d'exploitation

Généralités

Exercice 10.1 :

1. Citer quatre systèmes d'exploitation différents (pour ordinateur ou appareils mobiles par exemple)
2. Parmi ces systèmes d'exploitation, lesquels sont gratuits ?
3. Parmi ces systèmes d'exploitation, lesquels sont libres ?

Exercice 10.2 : Distributions *Linux*

Effectuer des recherches sur le net afin de répondre aux questions suivantes :

1. Quelle est la différence entre *UNIX* et *Linux* ?
2. En quelle année à été la première version de *Linux* ?
3. Qui est le créateur de ce système d'exploitation ?
4. Que représentent *Debian*, *Ubuntu* et *Linux Mint* par rapport au système d'exploitation *Linux* ?
5. Que sont *KDE* et *Gnome* ?

Commandes `linux`

On considère dans cette section le système de fichier suivant :

```
antoine
 |_____Images
 |       |_____anniversaire.jpg
 |       |_____vacances.jpg
```

```
|      |_____vacances2.jpg
|_____Musique
|      |_____Bach
|             |_____Goldberg
|                    |_____gould1.flac
|                    |_____gould2.flac
|                    |_____glenn_gould.jpg
|_____Travail
       |_____NSI
       |      |_____exo1.py
       |      |_____tp.ipynb
       |_____redaction.odt
|_____a_faire.txt
|_____essai.py
|_____exo_DM.py
|_____tp_images.ipynb
```

Exercice 10.3 : Copie de fichiers

On travaille dans le système de fichiers décrit au début de cette section. Antoine a ouvert un terminal et obtient l'affichage ci-dessous :

```
antoine@ordi-antoine:~$
```

1. Dans quel dossier de travail se trouve-t-il ?
2. Quelle commande lui permet d'afficher le nom du dossier courant ?

Antoine est en réalité dans le dossier `/home/antoine`.

3. Il souhaite se déplacer dans son dossier d'images. Quelle commande doit-il taper ?
4. Une fois dans son dossier d'images, il souhaite créer un sous-dossier `Vacances`. Quelle commande doit-il taper ?
5. Une fois le dossier `Vacances` créé il souhaite y déplacer ses photos de vacances. Quelle commande doit-il taper ?
6. Quelle commande taper afin de renommer le fichier `vacances.jpg` en `quiberon.jpg` ?

Exercice 10.4 : Création de fichier

On travaille dans le système de fichiers décrit au début de cette section. Antoine a ouvert un terminal et obtient l'affichage ci-dessous :

```
antoine@ordi-antoine:~/Travail/NSI$
```

1. Dans quel dossier de travail se trouve-t-il ?

Antoine souhaite créer un dossier `DM1` et à l'intérieur de ce dossier un fichier `dm1.py`.

2. Quelle commande permet de créer le dossier ?
3. une fois dans le dossier `DM1`, quelle commande permet de créer le fichier ?

Antoine saisit la commande ci-dessous :

```
antoine@ordi-antoine:~/Travail/NSI/DM$echo "#Premier exercice" > dm1.py
```

4. Quel effet a cette commande ?

Antoine saisit désormais la commande :

```
antoine@ordi-antoine:~/Travail/NSI/DM$cat dm1.py
```

5. Quel affichage obtient-il dans le terminal ?

Exercice 10.5 : Affichage et recherche de fichiers

On travaille dans le système de fichiers décrit au début de cette section. Antoine a ouvert un terminal et obtient l'affichage ci-dessous :

```
antoine@ordi-antoine:~/Images$
```

1. Dans quel dossier de travail se trouve-t-il ?
2. il souhaite faire ses exercices de NSI. Quelle(s) commande(s) doit-il saisir afin de se rendre dans le bon dossier ?
3. Une fois dans le dossier `NSI`, quelle commande doit-il saisir afin d'afficher le contenu du fichier `python` ?
4. Le notebook (fichier ayant l'extension `ipynb`) est assez long à afficher. Quelle commande ne permet d'afficher que le début du fichier ? Et que la fin ?
5. Quelle commande permet d'afficher le contenu de ce dossier ?
6. Quelle commande permet d'afficher le contenu de ce dossier y compris les **fichiers cachés** ?

Décidément, Antoine ne trouve pas le fichier sur lequel il a commencé ses exercices. Il décide de remonter dans le dossier `Antoine` puis de chercher tous les fichiers `python` de son dossier.

7. Quelle commande lui permet de remonter dans le dossier `Antoine` ?
8. Quelle commande lui permet de trouver tous les fichiers `python` dont le nom commence par `exo` et présents dans ce dossier et ses sous-dossiers ? On pourra utiliser `locate` et le joker `*`.

Exercice 10.6 : Gestion des dossiers

On travaille dans le système de fichiers décrit au début de cette section. Antoine a ouvert un terminal et obtient l'affichage ci-dessous :

```
antoine@ordi-antoine:~Musique/Bach$
```

Il souhaite ajouter un dossier "Concertos".

1. Quelle commande doit-il saisir ?

Il pense avoir fait une faute de frappe et affiche le contenu de ce dossier :

```
antoine@ordi-antoine:~Musique/Bach$ls
Consertos     Goldberg
```

2. Quelle commande permet de renommer le dossier `Consertos` en `Concertos` ?

Afin de libérer de la place sur son disque dur, il souhaite supprimer l'image `glenn_gould.jpg`.

3. Sachant qu'il est toujours dans le dossier `Musique/Bach/Concertos`, quelle(s) commande(s) doit-il saisir afin d'effacer l'image ?

Exercice 10.7 : Droits d'accès

On travaille dans le système de fichiers décrit au début de cette section. Antoine a ouvert un terminal et saisit la commande ci-dessous :

```
antoine@ordi-antoine:~$ls -l
-rw-r--r--   1 antoine antoine       58 2008-10-18 15:37 a_faire.txt
-rw-r--r--   1 antoine antoine       26 2008-10-18 15:37 essai.py
-rw-r--r--   1 antoine antoine      109 2008-10-18 15:37 exo_DM.py
drwxr-xr-x   6 antoine antoine     4096 2008-10-29 23:09 Images
drwxr-x---   2 antoine antoine     5165 2008-10-22 22:46 Musique
-rw-r--r--   1 antoine antoine     1544 2008-10-18 15:37 tp_images.ipynb
drwx------   1 antoine antoine     1026 2008-09-22 22:30 Travail
```

1. Que signifient les lettres `drwxr-x--` associées au dossier `Musique` ?

Antoine souhaite que tous les utilisateurs de cet ordinateur puissent lire les fichiers disponibles dans le dossier `Musique`.

2. Qui a accès au contenu de ce dossier actuellement ?

3. Comment faire en sorte que tous les utilisateurs puisse exécuter les fichiers contenus dans ce dossier ?

Exercice 10.8 : Installation

On travaille dans le système de fichiers décrit au début de cette section.

Un utilisateur souhaite installer le logiciel `VLC`.

1. Quelle commande lui permet de passer super-utilisateur ?

2. Une fois super-utilisateur, quelle commande doit-on saisir afin d'installer le logiciel. On pourra utiliser `apt-get`.

Chapitre 11

Le web

Interactivité côté Client

Exercice 11.1 : Une page de base...

Créer une page web `exercice1.html` avec un fichier css associé `styleexercice1.css` et comprenant :

- Un titre de type `<h1>` dont le texte est bleu
- Un paragraphe dont la police est en Arial et de couleur verte
- Une division `<div>` comprenant l'adresse du lycée et dont la couleur de fond est `#aaaaaa`
- Le logo du lycée avec une largeur de 200 pixels
- Un lien vers le site du lycée

Exercice 11.2 : ... avec un peu d'interactivité en CSS

1. Modifier le `CSS` de la page créée dans l'exercice 11.1 afin que le titre change de couleur de police lorsque l'on passe la souris dessus.

2. De la même façon faire en sorte que l'image soit agrandie à 300 pixels de largeur lorsque l'on passe la souris dessus.

Exercice 11.3 : ... avec un peu d'interactivité en JS

Reprendre l'exercice 11.2 afin de modifier la page en `Javascript`.

Exercice 11.4 :

Lire le code suivant :

```
<h1>Bonjour</h1>
<i>nom masculin</i>
<p>Terme de salutation dont on se sert pendant la journée quand on aborde
↪  quelqu'un ou, moins souvent, quand on prend <b>congé</b> de quelqu'un.</p>
<p><b>Exemple</b></p>
<p><i>Bonjour<i>, comment allez-vous ?<p>
```

1. S'agit-il de `HTML`, de `CSS` ou de `JS` ?
2. Combien y-a-t-il de paragraphes dans cette page web ?
3. Que représente la première ligne du code ?
4. Comment apparaîtra le mot "congé" dans un navigateur ?
5. Quels termes apparaissent en italique ?

Exercice 11.5 :

On considère la code suivant :

```
<a href="pagecible.html"> <img src="images/mon_image.png"> </a>
```

1. Qu'affiche ce code à l'écran ?
2. Que se passe-t-il lorsque l'on clique sur l'image ?
3. Dans quel répertoire est située l'image ?

Exercice 11.6 :

On souhaite faire en sorte que :

- le clic sur un bouton dans une page change la couleur du fond de la page
- le survol de l'image à la souris agrandisse celle-ci (à 500 pixels de largeur).

Compléter le code ci-dessous :

```
<button .........="clic_bouton()" > Clique ici !! </button>
<img src="ressources/mon_image.jpg" onmouseover=".........(this)" onmouseout =
↪  ".........(this)" >
<script>
    function clic_bouton() {
        corps = document. .........
        corps.style.......... = "red"
    }
    function image_over(img) {
        ..........style.width = .........
    }
    function image_out(img) {
        ......... = .........
    }
    </script>
```

Exercice 11.7 :

Créer une page web dont la seule fonctionnalité est un bouton situé au centre. A chaque clic sur le bouton, le fond de la page changera de couleur aléatoirement.

Coup de pouce :

- l'instruction `JS var alea = Math.random() * 256` permet de générer un nombre décimal aléatoire entre 0 et 255
- l'instruction `Math.floor(nom_de_variable)` permet d'arrondir un décimal à l'entier directement inférieur

- Une couleur aléatoire en HTML s'écrit `rgb(alea1, alea2, alea3)`

Interactivité côté Serveur

Exercice 11.8 :

On considère le formulaire HTML ci-dessous :

```
<form action="/action_page.php" method="POST">
    <label for="fname">Prénom :</label><br>
    <input type="text" id="input_prenom" name="prenom" value="Jean"><br>
    <label for="lname">Année de naissance :</label><br>
    <input type="number" id="input_annee" name="annee" value="2000"><br><br>
    <input type="submit" value="Envoyer">
</form>
```

1. Comment la ligne `<input type="submit" value="Envoyer">` apparaît-elle dans le navigateur ?

2. Quelle type de requête HTTP est envoyé lorsque l'utilisateur clique sur le bouton ?

Le traitement de ce formulaire se fait côté serveur en PHP à l'aide du code ci-dessous :

```
    <h1>
        Bonjour <?php echo $_POST[XXX]; ?> !
    </h1>
    <p>
        Vous avez eu ou aurez <?php YYY ?> ans cette année.
    </p>
```

3. Que doit-on saisir à la place des XXX afin d'afficher le prénom de l'utilisateur ?

4. Que doit-on saisir à la place des YYY afin que le code soit fonctionnel ?

Exercice 11.9 :

On a codé un questionnaire en HTML comme ci-dessous :

```
<form action="../traitement/corrigé.php" method="POST">
    <p>Quel est le double du tiers de 12 ?</p>
    <input type="number" id="ID_q1" name="Q1"><br>
    <p>Dans une fraction, comment s'appelle le nombre du "haut" ?</p>
    <input type="text" id="ID_Q2" name="Q2"><br><br>
    <input type="submit" value="Envoyer">
</form>
```

Le document HTML est dans le dossier `questions` du serveur.

1. a. Quel fichier est chargé lors de l'envoi de ce formulaire ?

   b. Dans quel dossier se trouve ce fichier ?

   c. Dessiner l'arborescence des fichiers côté serveur.

2. Compléter le code PHP ci-dessous permettant de corriger la première question :

```php
<?php
    if ($_...['Q1'] == ...) {
        ... "Vous avez répondu " . $_POST[...] . "... C'est une bonne
        ↪ réponse !";
    } else {
        ... "Vous avez répondu " . ... . "... C'est une mauvaise réponse
        ↪ !";
    }
?>
```

3. La correction de la seconde question est plus subtile. En effet l'utilisateur peut avoir saisi sa réponse en minuscule, en majuscule ou avec une majuscule en début de mot... Quel test doit-on effectuer afin de vérifier cette réponse ?

Chapitre 12

Les réseaux

Protocole HTTP

Exercice 12.1 :

On considère l'URL suivante :

`https://fr.wikipedia.org/wiki/Domain_Name_System`

1. Que signifie l'acronyme URL ?

2. La connexion vers ce sit web est-elle sécurisée ou non ? Justifier.

3. a. Quelle type d'adresse permet d'identifier ce site web sur le réseau ?

   b. Quel protocole, abordé en SNT, permet de convertir l'URL en une adresse réseau valide ?

On a saisi la commande suivante sur un machine *Windows* :

```
> ping fr.wikipedia.org -4

Envoi d'une requête 'ping' sur dyna.wikimedia.org [91.198.174.192] avec 32
↪  octets de données :
Réponse de 91.198.174.192 : octets=32 temps=158 ms TTL=55
Réponse de 91.198.174.192 : octets=32 temps=228 ms TTL=55
Réponse de 91.198.174.192 : octets=32 temps=40 ms TTL=55
Réponse de 91.198.174.192 : octets=32 temps=42 ms TTL=55

Statistiques Ping pour 91.198.174.192:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 40ms, Maximum = 228ms, Moyenne = 117ms
```

4. a. Quel est le rôle de l'option `-4` ?

   b. Quelle est l'adresse du site ?

   c. On a envoyé un `ping` vers ce site. Qu'a-t-il répondu ?

Exercice 12.2 :

Un internaute a chargé la page du site duckduckgo.com sur son ordinateur.

En parallèle il a utilisé les outils de développement de son navigateur et obtenu la capture ci-dessous.

État	Métho...	Domaine	Fichier	Initiateur	Type	Transfert	Taille
200	GET	duckduckgo.com	/?t=ffab	BrowserTabChild.jsm:9...	html	3,25 Ko	5,67 Ko
200	GET	duckduckgo.com	s1977.css	stylesheet	css	38,82 Ko	190,61 Ko
200	GET	duckduckgo.com	o1977.css	stylesheet	css	5,48 Ko	21,17 Ko
200	GET	duckduckgo.com	ProximaNova-Reg-webfont.woff2	font	html	19,36 Ko	17,65 Ko
200	GET	duckduckgo.com	ProximaNova-Sbold-webfont.woff2	font	html	19,44 Ko	17,73 Ko
200	GET	duckduckgo.com	ProximaNova-ExtraBold-webfont.woff2	font	html	22,24 Ko	20,54 Ko
200	GET	duckduckgo.com	tl3.js	script	js	1,79 Ko	111 o
200	GET	duckduckgo.com	l120.js	script	js	53,48 Ko	155,71 Ko
200	GET	duckduckgo.com	duckduckgo62.js	script	js	32,37 Ko	115,08 Ko
200	GET	duckduckgo.com	u544.js	script	js	30,97 Ko	92,67 Ko
200	GET	duckduckgo.com	d2939.js	script	js	127,41 Ko	602,50 Ko
200	GET	duckduckgo.com	ti2.js	script	js	2,06 Ko	672 o
200	GET	duckduckgo.com	logo_homepage.normal.v108.svg	img	svg	3,69 Ko	4,65 Ko
200	GET	duckduckgo.com	post3.html	d2939.js:1 (subdocum...	html	1,77 Ko	141 o
200	GET	duckduckgo.com	s2480.js	d2939.js:1 (script)	js	14,83 Ko	58,92 Ko
200	GET	improving.duckduc...	hi?1271228&b=firefox&atbi=false&ei=	l120.js:26 (img)	gif	1,77 Ko	43 o
200	GET	duckduckgo.com	arrow.svg	lazy-img	svg	1,92 Ko	427 o
200	GET	duckduckgo.com	p103.js	script	js	1,88 Ko	315 o
200	GET	duckduckgo.com	DDG-iOS-icon_152x152.png	FaviconLoader.jsm:191...	png	3,66 Ko	1,99 Ko
200	GET	duckduckgo.com	favicon.ico	FaviconLoader.jsm:191...	x-icon	3,34 Ko	5,30 Ko

1. a. Combien de requêtes ont été effectuées au total ?

   b. Quelle requête `HTTP` a été utilisé ?

   c. Que signifie l'état 200 indiqué en regard de chaque requête ?

2. a. Combien de fichiers `html` ont été transmis au total ?

   b. Combien de fichiers de mise en forme ont été transmis ?

3. a. Identifier dans les différentes requêtes celle correspondant au logo du site.

   b. Quel est le format de cette image ?

   c. Les différents fichiers envoyés on subit une compression avant de transiter sur le réseau ce qui explique les deux colonnes donnant des tailles. Quel est le taux de compression (Volume final / Volume initial) de l'image du logo ?

Exercice 12.3 :

Après une recherche sur le site duckduckgo.com, un utilisateur constate que l'URL de la page est la suivante :

`http://duckduckgo.com/?q=requ%C3%AAte+GET&t=ffab&df=m&ia=web`

1. La connexion est-elle sécurisée ?

2. a. Quelle type de requête `HTTP` a été effectuée ?

   b. Combien de paramètres comporte cette requête ?

   c. Chercher sur internet la signification de `%C3%AA`.

Le site duckduckgo.com nomme certains paramètres ainsi :

- `q` : texte de la recherche. Les différents termes sont séparés par des `+`
- `df` : âge du résultat. La valeur peut valoir `d` si l'on souhaite obtenir des résultats de moins d'un jour (pour `day`) ou `m` pour moins d'un mois

3. Quelle URL faut-il chercher afin d'afficher les résultats de moins d'une journée d'une recherche portant sur "NSI" ? Tester cette requête.

Exercice 12.4 :

On considère le code `html` ci-dessous :

```
<form action="/accueil.php" method="GET">
    <label for="email">Email :</label>
    <input type="email" id="email" name="email"><br><br>
    <label for="pwd">Mot de passe :</label>
    <input type="password" id="mdp" name="mdp" minlength="8">
    <input type="submit" value="Envoyer">
</form>
```

1. Combien d'entrées comporte ce formulaire ?

2. Quelle est la taille minimale du mot de passe ?

3. Qu'affiche la ligne `<input type="submit" value="Envoyer">` ?

4. Quelle page est chargée lorsque l'on clique sur le bouton ?

5. Pourquoi ce formulaire est-il à proscrire afin d'envoyer un mot de passe ?

Protocoles TCP/IP

Exercice 12.5 :

Les adresses IP (en version 4) ci-dessous sont toutes invalides. Proposer une correction.

a. `192.256.35.15`

b. `10.18.16`

c. `220.112.30.17.26`

d. `40.301.156.765`

Exercice 12.6 : title=`IPv4` *vs* `IPv6`

1. a. Sur combien de bits sont codées les adresses IP en version 4 ?

   b. Combien peut-on écrire d'adresses IPv4 au maximum ?

2. Mêmes questions avec les adresses IP en version 6.

Exercice 12.7 :

Un sous réseau possède est identifié grâce à une adresse IP dont seuls les premiers bits sont significatifs. Ainsi un sous-réseau d'adresse IP `191.168.1.0` dont seuls les 24 premiers bits sont significatifs peut accueillir 2^8 postes car $32 - 24 = 8$.

1. Indiquer dans chaque cas le nombre de postes maximal pour chacun des réseaux ci-dessous.

   a. `10.0.0.0` (8 bits significatifs)

   b. `172.16.0.0` (16 bits significatifs)

   c. `224.0.0.0` (3 bits significatifs)

2. En réalité, on s'interdit toujours d'utiliser deux adresses IP : la première et la dernière du sous-réseau. Combien de postes peut-on installer sur un réseau local d'adresse `192.168.1.0` et dont les 24 premiers bits sont significatifs ?

Exercice 12.8 :

1. Une machine a pour adresse IP `10.12.25.2`. Son adresse réseau est codée sur 8 bits.

   a. Quelle est l'adresse de ce réseau ?

   b. Quelle est l'adresse du premier hôte de ce réseau ?

   c. Quelle est l'adresse du dernier hôte de ce réseau ?

   d. Quelle est l'adresse de diffusion sur ce réseau ?

   Une notation abrégée de l'adresse IP et du réseau associé est `IP/nb_bits_réseau`.

   Ainsi, l'adresse IP et le réseau de la question précédente est `10.12.25.2/8`.

2. Reprendre la question précédente avec les adresses suivantes :

   a. `172.20.10/16`

   b. `192.168.20.185/24`

3. Un réseau contient 600 postes, combien d'octets seront nécessaires pour identifier tous les hôtes ?

Exercice 12.9 :

Pour chaque adresse en binaire ci-dessous :

- `10010011 11011000 01100111 10111110` masque `255.255.0.0`
- `01101100 10100100 10010101 11000101` masque `255.0.0.0`
- `11010110 01011100 10110100 11010001` masque `255.255.255.0`

1. L'écrire en notation décimale.

2. Déterminer son adresse réseau en binaire.

3. Écrire en notation décimale son adresse de réseau ainsi que son adresse de diffusion.

Exercice 12.10 :

Le service informatique d'une société est actuellement équipé d'un réseau local comportant un serveur et cinq postes de travail dédiés au développement. Le serveur possède les caractéristiques suivantes :

- Adresse IP : `192.168.10.10`
- Masque de sous-réseau : `255.255.255.0`

L'entreprise désire ajouter une station de numérisation avec les caractéristiques suivantes :

- Adresse IP : `192.168.20.11`
- Masque de sous-réseau : `255.255.255.0`

1. Expliquer pourquoi l'adresse IP de cette station cliente ne lui permet pas de communiquer avec le serveur.

2. Indiquer, parmi les six adresses IP suivantes, celles qui peuvent être affectées à la station de numérisation. Justifier chaque réponse.

- `192.168.10.0`
- `192.168.10.1`
- `192.168.10.10`
- `192.168.10.11`
- `192.168.10.254`
- `192.168.10.255`

Exercice 12.11 :

On rappelle qu'une notation abrégée de l'adresse IP et du réseau associé est `IP_sur_4_octets/nb_bits_rése`
Ainsi, un poste d'adresse IP `10.12.25.2` et dont l'adresse le réseau est `10.0.0.0` aura pour "adresse" `10.12.25.2/8`.

Une entreprise dispose d'un réseau local avec un accès à internet. Le plan d'adressage IP est le suivant :

- une tranche d'adresses de `172.22.0.1/16` à `172.22.2.254/16` est réservée aux équipements spéciaux qui ont reçu une adresse fixe : éléments actifs, imprimantes et serveurs.
- la plage d'adresses à partir de 172.22.3.0 est réservée aux stations clients

Voici un exemple de paramétrage IP de la dernière station connectée au réseau

```
DHCP activé.................: Oui
Autoconfiguration activée..: Oui
Adresse IP..................: 172.22.3.134
Masque de sous-réseau......: 255.255.0.0
Passerelle par défaut......: 172.22.0.10
Serveur DHCP................: 172.22.0.1
Serveurs DNS...............: 172.22.0.9
```

1. Quelle est l'adresse de réseau ?

2. Expliquer les éléments suivants extraits du paramétrage IP présenté ci-dessus :

   a. `Passerelle par défaut : 172.22.0.10`

   b. `DHCP activé : Oui`

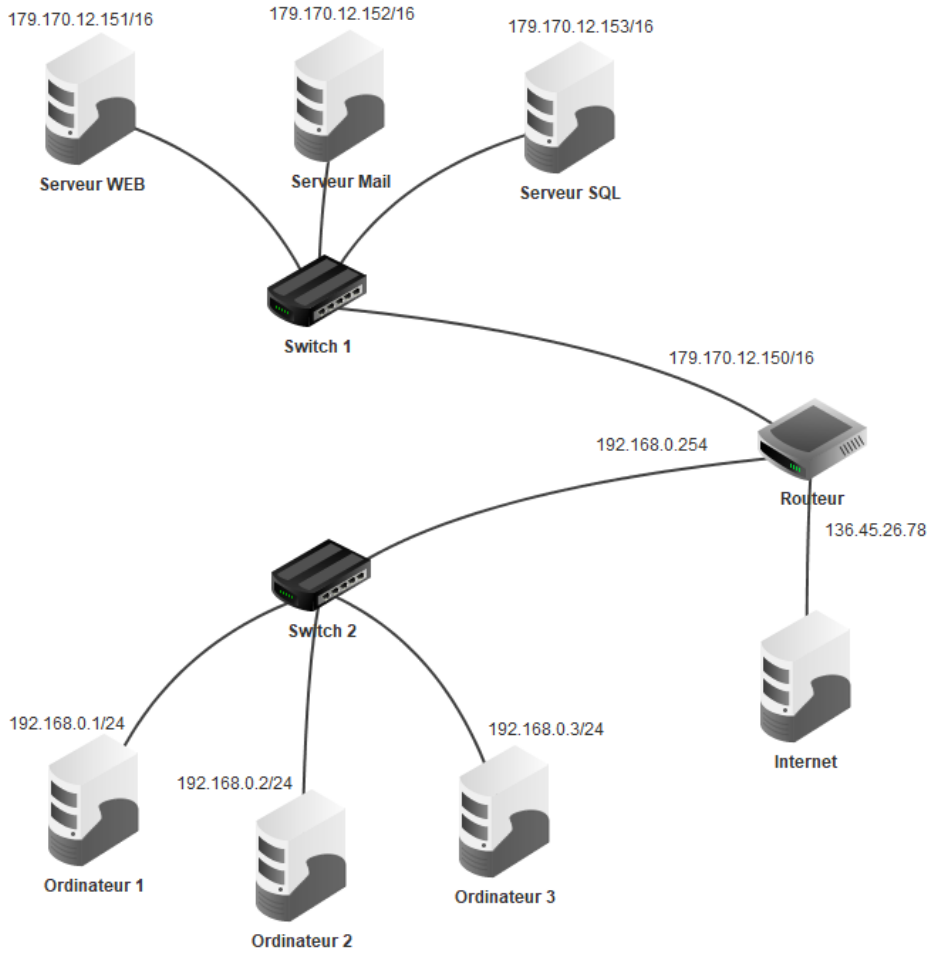
   c. `Serveur DHCP : 172.22.0.1`

3. Donner un exemple de configuration IP d'un nouveau poste connecté au réseau en DHCP :

```
Adresse IP.................: ???.???.???.???
Masque de sous-réseau......: ???.???.???.???
Passerelle par défaut......: ???.???.???.???
Serveurs DNS...............: ???.???.???.???
```

Exercice 12.12 :

Une entreprise possède le réseau informatique représenté dans la figure ci-dessous.



Les serveurs "Serveur WEB", "Serveur SQL" et "Serveur Mail" remplissent les rôles, respectivement, de serveur d'application Web, de serveur de bases de données (SGBDR) et de serveur de messagerie. Les bases de données situées sur le serveur "Serveur SQL" sont exploitées et mises à jour uniquement par le biais des applications Web.

1. Combien cette entreprise compte-t-elle de sous-réseaux connectés ?

2. Donner l'adresse de réseau des réseaux locaux et préciser si elles sont publiques ou privées.

3. Donner la configuration IP (adresse IP, masque réseau et passerelle) du serveur "Serveur WEB".

4. Parmi les trois adresses IP associées au routeur laquelle est son adresse publique ?

Exercice 12.13 :

1. Compléter le tableau suivant en complétant les conversions :

En décimal	En héxadécimal	En binaire
`135.95.183.246`		
	`A8.6C.64.BF`	
		`11001011.10100111.11000000.00010010`

En `python` si l'on souhaite extraire une sous-chaîne d'une chaîne de caractère à partir de ses indices, on utilise le *slicing* (ou saucissonage !). Par exemple :

```
>>> s = "chaîne de caractères"
>>> s[0:2]
'ch'
>>> s[1:2]
'h'
>>> s[:4]
'chaî'
>>> s[4:]
'ne de caractères'
```

2. Écrire les fonctions permettant d'effectuer les conversions précédentes (décimal vers hexadécimal, décimal vers binaire...). On utilisera des entrées toutes bien formées (en décimal par exemple, tous les nombres auront trois chiffres : `192.010.001.056` au lieu de `192.10.1.56`.

   On pourra utiliser les fonctions `int`, `bin` et `hex` de `python`.

   Par exemple :

```python
def hex_bin(ip) :
    """Fonction convertissant en binaire une adresse IP donnée en
    ↪ hexadécimal"""
    nb1 = ip[0:2]
    nb2 = ip[3:5]
    nb3 = ip[6:8]
    nb4 = ip[9:11]

    nb1_dec = int(nb1, 16)
    nb2_dec = int(nb2, 16)
    nb3_dec = int(nb3, 16)
    nb4_dec = int(nb4, 16)

    retour = ""
    retour += bin(nb1_dec)[2:] + "."     #on retire le "0b" au début du nb
    ↪ binaire
    retour += bin(nb2_dec)[2:] + "."
    retour += bin(nb3_dec)[2:] + "."
    retour += bin(nb4_dec)[2:] + "."

    return retour
```

Exercice 12.14 : _ La taille maximale des données d'un paquet TCP transitant sur un réseau Ethernet est de 1 460 octets.

1. Combien de paquets sont nécessaires à l'envoi d'une image pesant 118 ko ?

Chaque paquet est accompagné de plusieurs entêtes et données finales :

- 8 octets de préambule
- 14 octets d'entête Ethernet
- 20 octets d'entête IPv4
- 16 octets de données finales

2. Combien d'octets circulent réellement sur le réseau lors de la transmission de cette image de 118 ko ?

Exercice 12.15 :

On rappelle que dans une trame ethernet, les informations sont réparties ainsi (le premier octet a pour indice 0) :

- octets 0 à 5 (inclus) : adresse MAC du destinataire
- octets 6 à 11 (inclus) : adresse MAC de la source
- octets 26 à 29 (inclus) : adresse IP de la source
- octets 30 à 33 (inclus) : adresse IP de l'émetteur

On fournit ci-dessous une trame ethernet. Toutes les informations sont fournies en notation hexadécimale.

```
a9 26 6c d2 73 c8 b4 5c fc 9d cb 0f 08 00 45 00
00 3c a7 62 00 00 80 01 0e 4c c0 a8 01 0d ac d9
16 84 08 00 4d 45 00 01 00 16 61 62 63 64 65 66
67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76
77 61 62 63 64 65 66 67 68 69
```

Les données de cette trame débutent à l'octet 42 et vont jusqu'à la fin de la trame.

Identifier dans cette trame :

1. L'adresse MAC du destinataire
2. L'adresse MAC de la source
3. L'adresse IP du destinataire. Convertir cette adresse en décimal
4. L'adresse IP de la source. Convertir cette adresse en décimal
5. Les données. Celle-ci sont codées au format ASCII. Donner leur transcription sous forme de texte.

Chapitre 13

Interface Homme - Machine

Avec la carte Micro :Bit

Exercice 13.1 :

Afficher le classique `"Hello World"`.

Exercice 13.2 :

Afficher `"A"` lorsque l'on appuie sur le bouton `A`. On utilisera une boucle `while True` pour que le programme tourne en boucle.

Exercice 13.3 :

Afficher `"A"`, `"B"`, `"AB"` ou `"-"` selon les boutons pressés.

Exercice 13.4 :

Afficher une image lorsque l'on touche le `PIN1`.

Exercice 13.5 :

Afficher un compte à rebours.

Exercice 13.6 :

Afficher toutes les flèches à la suite.

Exercice 13.7 :

Afficher la flèche pointant vers le bouton ou le pin pressé (flèche gauche pour `A`, droite pour `B`, bas pour `PIN2`)

Exercice 13.8 :

Afficher un smiley triste lorsque la carte est secouée, un souriant lorsqu'elle est immobile.

Exercice 13.9 :

Afficher l'orientation en degré mesurée par la boussole.

Exercice 13.10 :

Afficher la température de la pièce.

Exercice 13.11 :

Créer un chronomètre (bouton `A` : départ, bouton `B` : arrêt et affichage de la durée)

Exercice 13.12 :

Allumer une diode à l'aide des boutons : le `A` indique le numéro de la colonne, le `B`, celui de la ligne. 4 pressions sur `A` et 1 sur `B` allume la diode de coordonnées (4,1).

Exercice 13.13 :

Un minuteur : Au démarrage le minuteur est à 0. On appuie sur `A` pour augmenter le temps, on appuie sur `A` pour diminuer le temps et sur le pin2 pour démarrer le minuteur. A la fin du temps écoulé, on affiche 0

Exercice 13.14 : Mastermind (version très simplifiée)

La carte est réglée (ou choisis aléatoirement) avec un code du type `AABBA`. L'utilisateur clique 5 fois sur les boutons `A` et/ou `B` : la carte sourit (ou pas) si le code est correct

Exercice 13.15 :

(Deux cartes nécessaires) Envoyer et recevoir un message d'une autre carte

Exercice 13.16 :

(Deux cartes nécessaires) Afficher `"A"` si le bouton `A` d'une autre carte est pressé

Exercice 13.17 :

(Deux cartes nécessaires) Créer un couple de carte fonctionnant comme des feux tricolores :

- La première carte affiche `"V"` pendant 5 secondes, `"O"` pendant 1 seconde et `"R"` pendant 6 secondes
- La seconde carte fait l'inverse

Exercice 13.18 :

Créer un affichage proportionnel à l'accélération : une rangée de LED allumée si pas d'accélération, deux rangées si un peu d'accélération...

Exercice 13.19 :

Créer une boussole à l'aide du compas et des flèches (il y a 8, N, NE, E, SE, S, SW, W, NW)

Chapitre 14

Architecture matérielle

Aspects matériels

Exercice 14.1 :

Placer les périphériques ci-dessous dans le tableau :

- Écran non tactile
- Souris
- Lecteur DVD/Blu-Ray d'une console
- Carte réseau Ethernet
- Écran tactile
- Lecteur/graveur de DVD
- Imprimante
- Clavier
- Clé USB

Périphériques d'entrée	Périphériques de sortie	Périphériques d'entrée/sortie

Exercice 14.2 :

Relier chaque élément à sa définition :

Système d'exploitation •	• Mémoire volatile de petite taille immédiatement accessible par le processeur
Bus •	• Programme nécessaire à la gestion des ressources matérielles et des logiciels d'un ordinateur
Mémoire Morte •	• Suite d'instructions qui seront exécutées par l'ordinateur
Registre •	• Unité de calcul de l'ordinateur. Souvent situé sur un unique circuit imprimé
Carte mère •	• Voie de transport des informations (adresses, valeurs) entre le processeur et les périphériques
Programme •	• Sorte d'interrupteur permettant de laisser passer, ou pas, du courant électrique selon qu'il est activé ou non
Périphérique de stockage •	• Ensemble de circuits gravés sur une plaque de silicium et regroupant différents éléments de l'ordinateur (processeur, RAM...)
Transistor •	• Mémoire permanente (malgré l'extinction de l'ordinateur) permettant de stocker de manière durable des informations (logiciels, données)
Processeur •	• Mémoire volatile, lisible et réinscriptible dont chaque cellule est accessible rapidement

Le processeur avec Little Man Computer

Exercice 14.3 :

On travaille avec le *Little Man Computer* disponible sur `http://peterhigginson.co.uk/LMC/`.

Initialement la mémoire contient les valeurs suivantes :

Adr.0	Adr.1	Adr.2	Adr.3	Adr.4	Adr.5	Adr.6	...	Adr.96	Adr.97	Adr.98	Adr.99
901	396	597	196	399	902	0	...	0	15	0	0

1. a. Traduire la séquence d'instructions des adresses 0 à 6. Par exemple, le code 901 dans la cellule 0 signifie `input`.

   b. Que contient la case à l'adresse 99 à la fin de l'exécution du programme ?

On considère la mémoire initiale suivante :

Adr. 0 : Input	Adr. 3 : Sub 96	Adr. 6 : Branch always 8	...
Adr. 1 : Store 96	Adr. 4 : Branch if zero 7	Adr. 7 : Load 98	Adr. 98 : 1
Adr. 2 : Input	Adr. 5 : Load 99	Adr. 8 : Output	Adr. 99 : 0

2. a. Qu'affiche ce programme si l'utilisateur saisit 12 puis 13 ? Et 8 puis 8 ?

   b. fait ce programme ?

3. Écrire le code en assembleur (au format `ADD XXX`) et non `1XX`) permettant de lire une valeur `x` dans la case 99 et de calculer le plus grand entier n tel que l'on ait $2^n < x$. Cette valeur sera stockée à l'adresse 98.

Exercice 14.4 : Simulateur d'assembleur

1. Visiter le site `http://www.peterhigginson.co.uk/AQA/`
2. Quelles différences observe-t-on avec le *Little Man Computer* ?
3. Sélectionner le programme `ADD` et observer son fonctionnement.

Chapitre 15

Recherche dichotomique

Exercice 15.1 :

On considère la liste `liste = [3, 7, 8, 2, 1, 9, 6, 5]` dans la quelle on souhaite chercher des éléments.

On utilise la fonction `recherche_lineaire` fonctionnant de la façon suivante :

- la fonction prend en argument la liste dans laquelle chercher et la valeur cherchée
- elle parcourt la liste du début à la fin
- elle compare chaque élément à la valeur cherchée :
  - s'il est égal la fonction renvoie son indice
  - sinon on passe à l'élément suivant
- si l'on a testé tous les éléments sans rencontrer la valeur cherchée la fonction renvoie `-1`

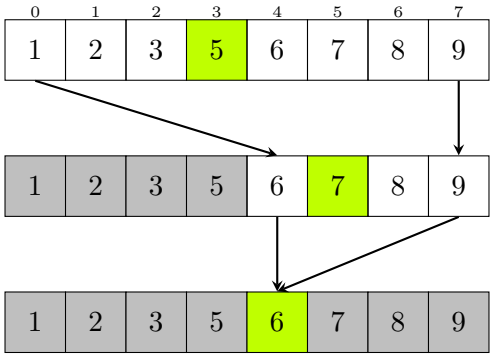
1. Que renvoie les appels suivants ?
   a. `recherche_lineaire(liste, 3)`
   b. `recherche_lineaire(liste, 5)`
   c. `recherche_lineaire(liste, 4)`
2. Combien de valeurs de la liste sont testées dans chacune des recherches précédentes ?
3. Proposer une liste et une valeur cherchée permettant :
   a. de renvoyer la valeur `3`
   b. de renvoyer la valeur `-1`
   c. d'effectuer exactement 3 comparaisons en renvoyant `2`
   d. d'effectuer exactement 3 comparaisons en renvoyant `-1`
4. On considère une liste de 100 nombres et l'on cherche le dernier nombre.
   a. Quelle valeur renvoie la fonction ?
   b. Un ordinateur met 1 milliseconde pour effectuer cette recherche. Combien de temps aurait-il mis si la liste avait compté 200 nombres ?
   c. Même question si la liste avait compté 5 000 nombres.

Exercice 15.2 :

On considère la liste `liste = [1, 2, 3, 5, 6, 7, 8, 9]` dans la quelle on souhaite chercher des éléments.

On utilise la recherche dichotomique.

On a représenté ci-dessous la recherche de la valeur 6 :



1. Réaliser une figure identique représentant la recherche de la valeur 8.

2. Réaliser une figure identique représentant la recherche de la valeur 3.

3. Réaliser une figure identique représentant la recherche de la valeur 4.

Exercice 15.3 :

On considère la liste `liste = [1, 2, 3, 5, 6, 7, 8, 9]` dans la quelle on souhaite chercher des éléments.

On utilise la recherche dichotomique.

On appelle `a` l'indice de la valeur minimale de la zone de recherche, `b` celui de la valeur maximale et `m` celui de la valeur centrale. Ainsi au début de la recherche on a `a = 0`, `b = 7` et donc `m = 3`.

Lors de la recherche de la valeur 7, les valeurs de `a` et `b` évoluent ainsi :

Tour de boucle	`a`	`m`	`b`
0	0	3	7
1			
2			
3			

1. Construire un tableau équivalent correspondant à la recherche de la valeur 8.

2. Construire un tableau équivalent correspondant à la recherche de la valeur 3.

3. Construire un tableau équivalent correspondant à la recherche de la valeur 4.

Exercice 15.4 :

On considère une liste d'entiers triée dans l'ordre croissant. On effectue une recherche dichotomique dans cette liste.

1. Dans le cas où la liste compte 4 valeurs et que l'on cherche une valeur strictement supérieure à la valeur maximale de la liste, en combien de tours de boucles s'effectuera la recherche ? On pourra "dessiner" un exemple.

2. Compléter le tableau ci-dessous liant le nombre maximal de tours de boucles à la taille de la liste.

Taille de la liste	Tours de boucle
4	
5	
8	
10	
17	
100	
256	
257	
511	
1 025	

Exercice 15.5 :

Une implémentation de la recherche dichotomique met au maximum 20 millisecondes pour trouver la dernière valeur d'une liste de 500 nombres.

1. La recherche de la dernière valeur dans une liste de 600 nombres sera-t-elle sensiblement plus longue à réaliser ? Justifier.

2. A partir de taille de liste, la recherche de la dernière valeur sera-t-elle sensiblement plus longue à réaliser ?

Chapitre 16

Algorithmes gloutons

Exercice 16.1 :

On cherche à sélectionner cinq nombres de la liste suivante en cherchant à avoir leur somme la plus grande possible (maximiser une grandeur) et en s'interdisant de choisir deux nombres voisins (contrainte).

```
liste = [15, 4, 20, 17, 11, 8, 11, 16, 7, 14, 2, 7, 5, 17, 19, 18, 4, 5, 13,
↪  8]
```

Comme on souhaite avoir le plus grand résultat final, la stratégie gloutonne consiste à choisir à chaque étape le plus grand nombre possible dans les choix restants.

1. Appliquer cet algorithme glouton sur le tableau et donner la solution trouvée.

2. Vérifiez que `[20, 18, 17, 16, 15]` est une autre solution possible.

3. Que dire de la solution gloutonne ?

4. Coder cet algorithme en `python` et vérifier le résultat trouvé à la première question.

Exercice 16.2 : Voyageur de commerce

Un voyageur de commerce doit visiter les villes de Metz, Paris, Reims et Troyes. Il peut le faire dans l'ordre qu'il souhaite mais doit impérativement se rendre dans chaque ville. Son entreprise étant basée à Nancy, il doit de plus commencer et terminer son parcours dans cette ville.

On fournit ci-dessous les distance entre les différentes villes en kilomètres.

	Nancy	Metz	Paris	Reims	Troyes
Nancy		55	303	188	183
Metz	55		306	76	203
Paris	303	306		142	153
Reims	188	176	142		123
Troyes	183	203	153	123	

1. a. Quelle approche gloutonne peut-on mettre en œuvre ?

   b. Appliquer cette démarche et proposer une solution.

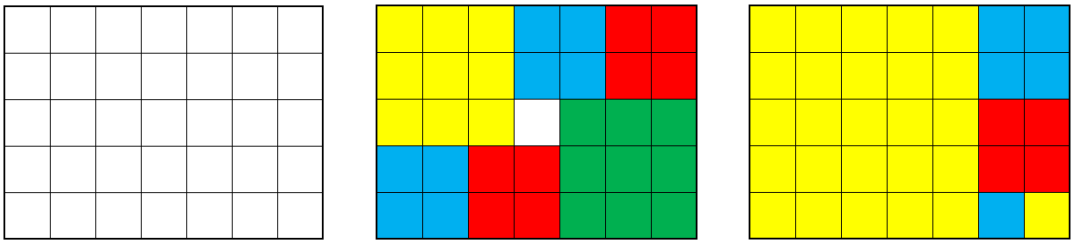
2. Calculer la distance totale pour le parcours :

$$\text{Nancy} \to \text{Metz} \to \text{Reims} \to \text{Paris} \to \text{Troyes} \to \text{Nancy}$$

3. Que dire de l'optimalité de la solution gloutonne ?

Exercice 16.3 : Pavage de rectangles

On souhaite paver un rectangle avec des carrés en en utilisant un minimum. On considère que toutes les dimensions sont des nombres entiers.



Ainsi dans la figure, on a pavé un rectangle de 7 par 5 à l'aide de 7 carrés (au centre) ou 5 carrés (à droite).

On cherche à utiliser **un minimum de carrés**.

Dans tous les codes `python` , un rectangle sera représenté à l'aide d'un tuple `(m,n)` dans lequel `m` est la longueur et `n` la largeur.

On décide d'adopter l'approche gloutonne suivante :

- on part d'un rectangle de dimensions `(m,n)`
- on pave à l'intérieur le plus grand carré possible
- on recommence la démarche dans le rectangle restant jusqu'à ce qu'il ne reste plus rien

C'est cette méthode que l'on a adopté dans la figure de droite.

1. a. On démarre avec un rectangle de dimensions `(9,4)`. Quel est le plus grand carré possible ?

   b. Quelles sont les dimensions du rectangle restant après avoir ôté ce plus grand carré ?

   c. Reprendre ces questions en partant d'un rectangle de dimensions `(9,7)`.

2. a. On démarre avec un rectangle de dimensions `(m,n)` où $m > n$. Exprimer les dimensions du rectangle restant après avoir ôté le plus grand carré possible.

   b. Reprendre la question précédente dans le cas où $m = n$.

On appelle `pavage` la donnée, sous forme d'une liste `python`, de tous les carrés pavant le rectangle de départ. Ainsi le pavage de la figure centrale est `[(3,3), (3,3), (2,2), (2,2), (2,2), (2,2), (1,1)]`.

3. Compléter le code `python` de la fonction `pavage_rectangle` prenant en argument un `rectangle` sous l'un forme d'un tuple et renvoyant le `pavage` obtenu Par l'algorithme glouton.

```
def pavage_rectangle(rectangle : tuple) -> list :
    """Détermine un pavage du rectangle"""
    pavage = []

    while rectangle != ..... : # Tant que le rectangle n'est pas
    ↪ pavé
        longueur, largeur = rectangle # les dimensions du rectangle
        if longueur == largeur :
            pavage.append( (.....,.....))
            rectangle = (.....,.....)
        else :
            dim_carre = ..... # la longueur du plus grand carré
            ↪ possible
            pavage.append( (.....,.....)) # on
            ↪ ajoute ce carré
            if longueur - dim_carre >= ..... :
                rectangle = (.....,.....)
            else :
                rectangle = (.....,.....)
    return pavage
```

4. Faire fonctionner cette fonction avec le rectangle `(7,5)`. Obtient-on la solution du début de l'exercice ?

5. On considère un rectangle `(13,11)`.

   a. Quel pavage renvoie l'algorithme glouton ? Dessiner ce pavage.

   b. Il est possible de paver ce rectangle avec le pavage `[(7,7), (6,6), (5,5), (4,4), (4,4), (1,1)]`. Dessiner ce pavage.

   c. Cette solution est-elle meilleure que la solution "gloutonne" ?

6. L'approche gloutonne est-elle optimale ?

Exercice 16.4 : Clé USB ou Sac à dos ? (source : *Eduscol*)

On dispose d'une clé USB qui est déjà bien remplie et sur laquelle il ne reste que 5 Go de libre. On souhaite copier sur cette clé des fichiers vidéos pour l'emporter en voyage. Chaque fichier a un poids et chaque vidéo a une durée. La durée n'est pas proportionnelle à la taille car les fichiers sont de format différents, certaines vidéos sont de grande qualité, d'autres sont très compressées.

Le tableau qui suit présente les 7 fichiers disponibles avec les durées données en minutes.

Fichier	Durée	Taille
Vidéo 1	114	4,57 Go
Vidéo 2	32	630 Mo
Vidéo 3	204	1,65 Go
Vidéo 4	44	85 Mo
Vidéo 5	18	2,15 Go
Vidéo 6	80	2,71 Go
Vidéo 7	5	320 Mo

On se demande donc quelles vidéos copier sur la clé USB pour que la durée des vidéos soit la plus grande possible tout en ne dépassant pas 5 Go ?

Partie 1 : Compréhension du problème

1. Quelle est la valeur que l'on cherche à maximiser/minimiser ? Quelle est la contrainte ?

2. Quel problème type peut-on reconnaître ?

Partie 2 : Résolution et implémentation

1. Représenter en `python` le tableau précédent à l'aide d'une liste de triplets nommée `table_videos` où chaque triplet représente un fichier vidéo contenant son nom, sa durée et sa taille.

On envisage dans un premier temps une approche *force brute*. Le principe est simple, il faut tester tous les cas possibles. La mise en œuvre est peu complexe :comment obtenir tous les cas sans les répéter et sans en oublier un ? Cette question pose la difficulté principale.

Une méthode est d'associer le chiffre `1` à un fichier s'il est choisi et le chiffre `0` sinon. Nous obtenons ainsi un nombre entier écrit en binaire avec 7 chiffres. Le nombre `1001100` signifie que nous avons choisi les fichiers 1, 4 et 5. Le nombre `1111111` signifie que nous avons choisi tous les fichiers. A chaque nombre correspond exactement une possibilité pour construire une partie de l'ensemble des 7 fichiers.

Il apparaît alors que le nombre total de cas est 2^7 puisqu'avec n chiffres nous pouvons écrire exactement 2^n nombres.

Il s'agit donc d'écrire une fonction qui prend en paramètres un ensemble, (dans notre problème de 7 fichiers), et renvoie l'ensemble des parties qui peuvent être constituées. Nous avons besoin de l'écriture d'un nombre en binaire.

2. Compléter la fonction `int_to_bin(n , nb)` ci-dessous qui prend en argument deux nombres entiers `n` et `nb` et qui renvoie l'écriture binaire de `n` avec un nombre de chiffres égal à `nb`.

```
def int_to_bin(n : int, nb : int) -> str :
    """n est le nombre à convertir en binaire
    nb est le nombre bits utilisés """
    ch = ""
    while n > 0 :
        r = ...................
        n = ...................
        ch = ...................
    ch = (nb - len(ch)) * "0" + ch # Ajout des 0 manquants au début de ch
    return ch
```

3. Compléter maintenant la fonction `ens_des_parties` qui prend en paramètre un ensemble d'objets et renvoie une liste dont chaque élément est une partie de l'ensemble.

```
def ens_des_parties(ensemble : list) -> list :
    """ensemble est une liste de p-uplets"""
    nb = len(ensemble) # nombre d'éléments
    n = ..... # nombre de parties
    parties = []
    for i in range(1,n) :
        ch = ..... # écriture de i sur nb bits
        partie = []
        for j in range(nb) :
            if ch[j] = ..... :
                partie.append(ensemble[j])
        ..... #ajout de la partie aux parties
    return parties
```

4. Pour chaque partie, nous devons calculer la durée totale et la taille totale des fichiers constituant la partie. Écrire deux fonctions `duree_totale` et `taille_totale` qui réalisent ces tâches.

5. Pour le programme final, nous avons besoin d'une fonction `recherche` qui prend en paramètres un ensemble de parties et la contrainte, et renvoie la partie correspondant au meilleur choix satisfaisant la contrainte après avoir parcouru toutes les parties. Compléter le code ci-dessous.

```
def recherche(ens_parties, contrainte) :
    duree_max = 0
    solution = []
    for partie in ens_parties : # un choix possible de fichiers
        duree = .....
        taile = .....
        if taille <= ..... and duree > ..... :
            duree_max = .....
            solution = partie
    return solution, duree_max
```

6. Pour terminer compléter la fonction `force_brute` et le programme final ci-dessous.

```
def force_brute(fichiers, taille_max) :
    parties = .....
    return recherche(parties, .....)

choix = force_brute(videos, 5)
duree = .....
choix_fichiers = [fichier[0] for fichier un .....]
print(choix_fichiers, duree)
```

7. Exécuter le programme obtenu et noter la solution obtenue ? Est-elle optimale ?

8. En utilisant le site Python_tutor, estimer le nombre d'opérations nécessaires pour obtenir la solution.

Partie 3 : Implémentation par algorithme glouton

1. Nous avons besoin de trois fonctions prenant en paramètre un fichier vidéo et renvoyant soit la durée, soit l'inverse de la taille (si la taille est minimale, son inverse est maximale),

soit le rapport durée/taille. Compléter ces fonctions.

```
def duree(fichier) :
    return ...................
def taille(fichier) :
    return ...................
def rapport(fichier) :
    return ...................
```

Nous définissons ensuite une fonction `sol_gloutonne` qui prend en paramètres une liste de fichiers, une taille maximale (celle que peut stocker la clé USB) et le type de choix utilisé (par durée, par taille, ou par durée/taille).

Nous commençons par trier la liste passée en paramètre suivant le type de choix utilisé avec la fonction `sorted`. Nous utilisons l'ordre décroissant (`reverse=True`).

La liste triée est parcourue et les noms des fichiers sont ajoutés un par un dans la variable `solution`, tant que la taille totale ne dépasse pas la taille maximale. La durée totale et la taille totale sont stockées dans deux variables nommées respectivement `duree_totale` et `taille_totale`.

2. Compléter la fonction `sol_gloutonne`.

```
def sol_gloutonne(liste, taille_max, choix) :
    triee = sorted(liste, key, choix, reverse=True)
    solution = []
    total_duree = 0
    taille = 0
    i = 0
    while ..................... and ..................... :
        nom, d, t = triee[i] # nom, durée et taille
        if taille + t <= taille_max :
            solution.append(nom)
            taille = ...................
            total_duree = ...................
        i += 1
    return solution, totale_duree
```

3. Exécuter le programme obtenu en choisissant un tri par durée, puis par taille et enfin par le rapport des deux. Quel est le choix le plus intéressant ? Est-il optimal ?

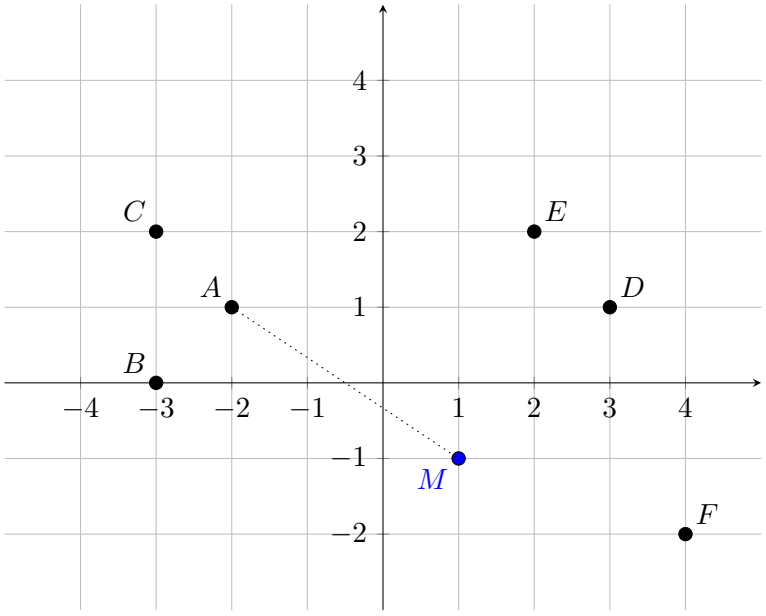
4. Estimer le nombre d'opérations nécessaires à cet algorithme puis comparer avec le nombre obtenu pour le programme par force brute.

Chapitre 17

k plus proches voisins

Exercice 17.1 : Distance de *Manhattan*

On considère la figure ci-dessous :



On utilise dans cet exercice la distance de *Manhattan* définie ainsi :

$$d(A,B) = |x_A - x_B| + |y_A - y_B|$$

Cette distance correspond à la longueur du "chemin" allant de A à B en suivant le quadrillage. Ainsi on a :

$$d(A,M) = |-2-1| + |1-(-1)| = 5$$

1. Calculer les distances entre le point M et les autres points de la figure.

2. Classer les points A à F du plus proche au plus éloigné de M.

Exercice 17.2 : Distance euclidienne

Reprendre l'exercice 17.1 en utilisant cette fois la distance euclidienne (on considère que le repère est orthonormé) :

$$d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Exercice 17.3 : Distance de *Tchebychev*

Reprendre l'exercice 17.1 en utilisant cette fois la distance de *Tchebychev*. Cette distance est égale à la distance maximale entre les coordonnées des deux points.

Ainsi, pour deux points A et M de la figure, les abscisses sont séparées de 3 et les ordonnées de 2. Leur distance de *Tchebychev* vaudra donc 3 qui est l'écart maximal.

Exercice 17.4 :

On se place dans un espace orthonormé et on considère les points suivants :

$$A(1\,;\,2\,;\,3) \qquad B(-1\,;4\,;\,3)$$

$$C(5\,;\,0\,;\,-1) \qquad D(3\,;0\,;\,-3)$$

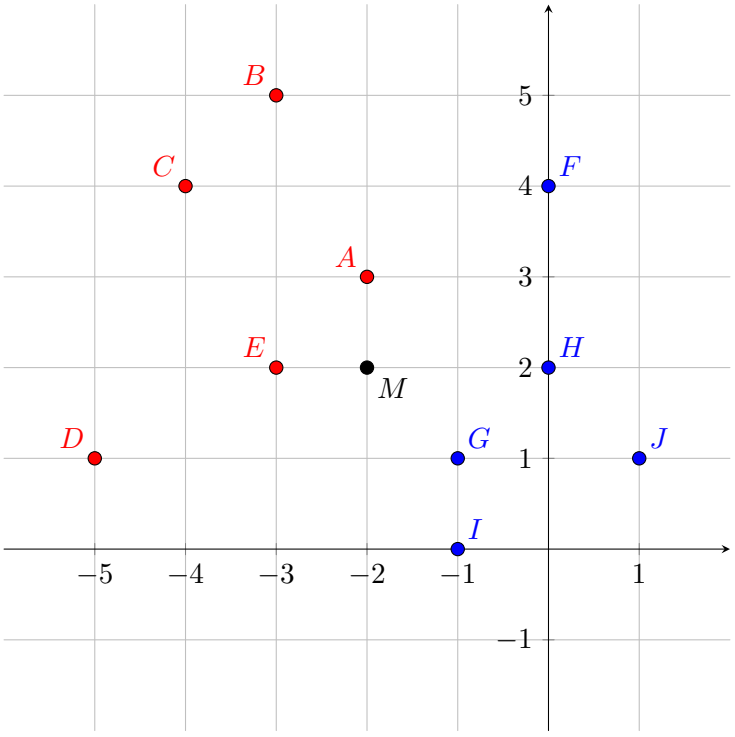
Calculer la distance euclidienne entre tous les points de la figure.

Exercice 17.5 :

Reprendre l'exercice 17.4 en utilisant la distance de *Manhattan* (voir exercice 17.1).

Exercice 17.6 :

On considère la figure ci-dessous :



Chaque point est associé à une catégorie :

- Les points **rouges** sont les A, B, C, D et E
- Les points **bleus** sont les F, G, H, I et J

On utilise l'algorithme des k plus proches voisins afin de déterminer la catégorie du point $M(-2;2)$.

On utilise la distance euclidienne (le repère est orthonormé).

1. Calculer les distances entre le point M et les autres points de la figure.

2. Classer les points du plus proche au plus éloigné de du point M.

3. On considère les 3 plus proches voisins. Quelle catégorie affecte-t-on à M ?

4. On considère les 5 plus proches voisins. Quelle catégorie affecte-t-on à M ?

5. On considère les 9 plus proches voisins. Quelle catégorie affecte-t-on à M ?