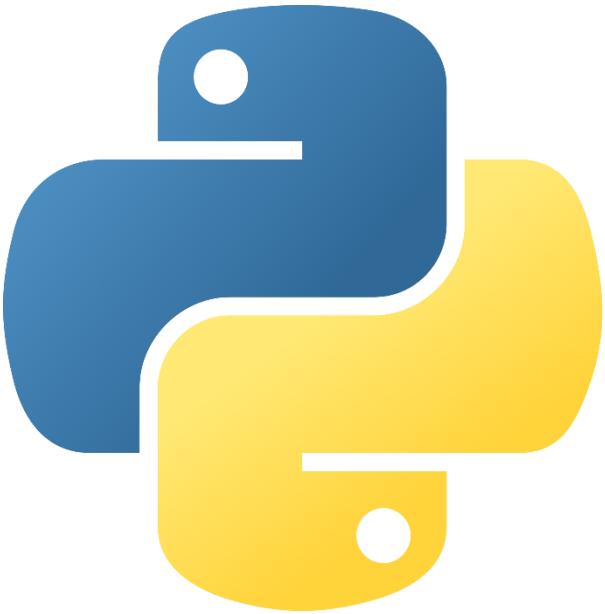
BALYO

Test : Exercice python

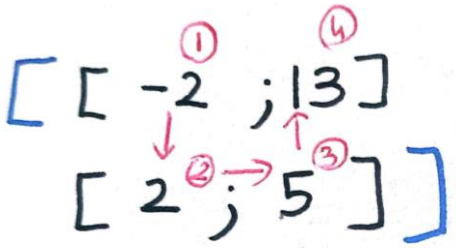


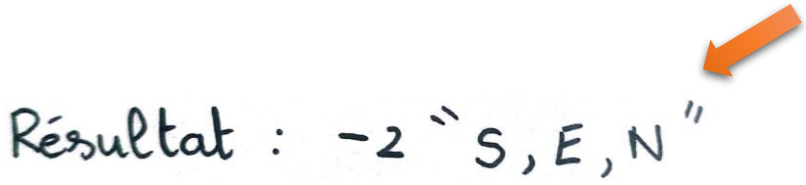
Réalisé par AKAZAF Abdellah

Sommaire :

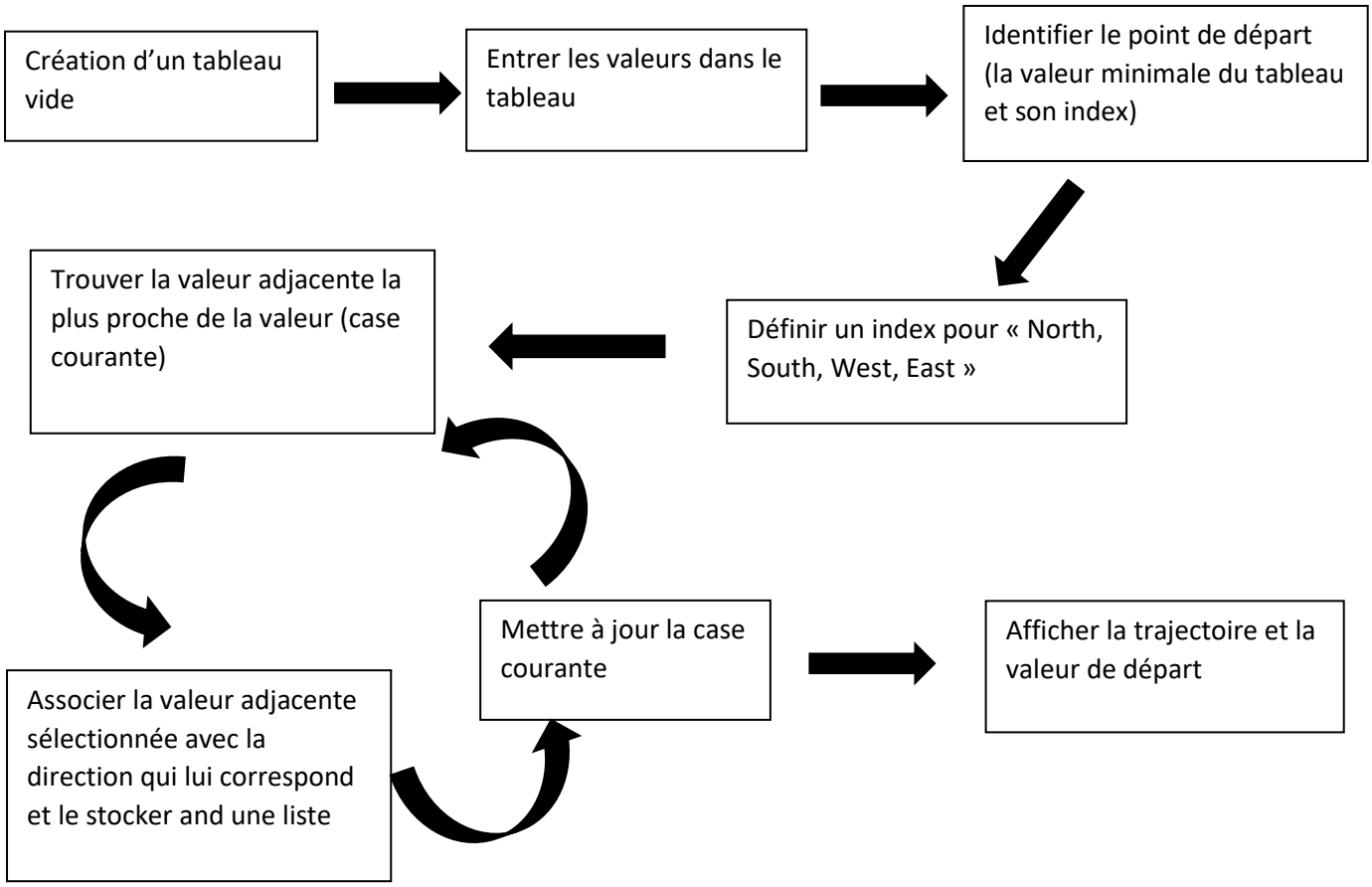
1. Introduction :

L'exercice consiste à créer un algorithme en langage Python capable de parcourir les nombres présents dans un tableau à deux dimensions. Ce parcours doit se faire dans l'ordre croissant des valeurs et surtout il doit afficher les différentes positions durant le parcours avec le nombre de départ.





Mon algorithme :



Les parties qui suivent détailleront chaque étape de mon algorithme en précisant la stratégie, les erreurs rencontrées et les conditions spécifiques

2. Création d’un tableau vide :

- **Stratégie :**

Pour créer un tableau vide à deux dimension, j’ai utilisé la fonction **np.empty()** du module Numpy

- **Conditions spécifiques :**

Il ne faut pas que le nombre de colonnes et de lignes soit respectivement (0 ; 0), (0 ; 1) ou (1 ; 0) ou des valeurs négatives. Cela signifie que la somme des colonnes et des lignes ne doit pas être inférieure à 2. Voici le code final :

```
while True:
    try:
        column_input = int(input("Enter column number: "))
        if column_input < 1:
            print("Error: The column number must be greater than 0. Please try again.")
        else:
            line_input = int(input("Enter line number: "))
            if line_input < 1:
                print("Error: The line number must be greater than 0. Please try again.")
            else:
                if (column_input + line_input) > 2:
                    break
                else:
                    print("Error: The dimension of the table must be greater than (1, 1). Please try again.")
        print()
    except ValueError:
        print("Error: Please enter valid integers for column and line numbers.")
```

- **Erreur rencontrées :**

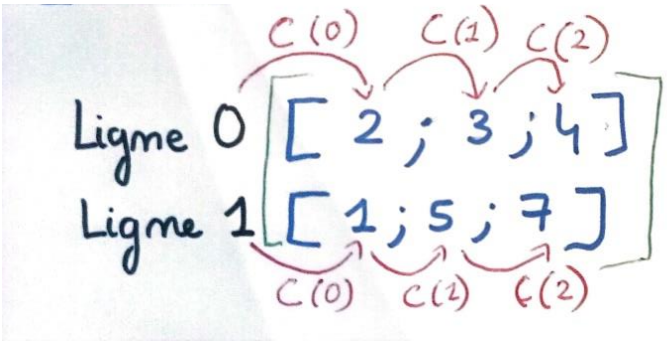
```
  File "main.py", line 8, in <module>
    tableau_zeros = np.empty((column_input, line_input))
TypeError: 'str' object cannot be interpreted as an integer
```

En effectuant des recherches sur le type d'erreur, j'ai compris que NumPy (np) interprète les valeurs comme des chaînes de caractères plutôt que des nombres. Il est donc nécessaire de préciser que les valeurs entrées doivent être interprétées comme des entiers (integer).

3. Entrer les valeurs dans le tableau :

- **Stratégie :**

Une fois qu'on a créé le tableau il faut entrer des valeurs dans le tableau. Pour pouvoir le faire on utilise 2 boucles qui : « pour chaque ligne du tableau, le programme demande une entrée pour toutes les colonnes de la ligne ». Voici le code :




```python
# Fill table with input values
input_values = [] # list used to contain all input values
for i in range(len(table)):
        for j in range(len(table[i])):
            while True:
                try:
                    value = int(input(f"Enter value at position ({i}, {j}): "))

                    if value in input_values:
                        print("Value already used! Please enter another unused value.")
                    else:
                        table[i, j] = value
                        input_values.append(value)
                        break
                except ValueError:
                    print("Invalid input! Please enter a valid number.")
```

- **Conditions spécifiques :**

Si l'utilisateur donne une lettre à la place d'un chiffre alors le programme n'aboutira pas. Donc on crée une boucle booléenne qui va repérer l'erreur et qui redemande une entrée.

Si l'utilisateur donne plus de 1 fois une même valeur, le code n'aboutira pas et va repérer l'erreur

4. Identifier le point de départ et l'index

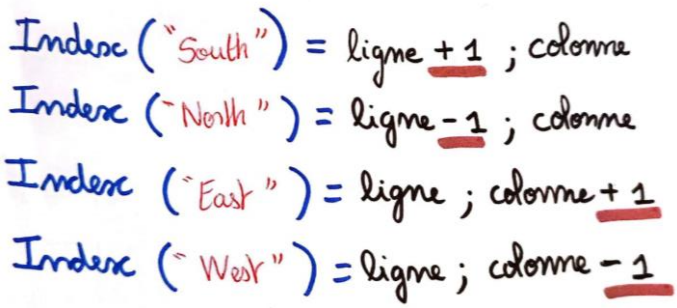
- **Stratégie :**

Pour identifier l'index de la valeur minimale du tableau, j'ai utilisé la fonction **argmin**, qui renvoie un indice unique ou plat de la valeur. Ensuite, j'ai converti cet indice en un format multidimensionnel grâce à la fonction **unravel_index**.

Définir un index pour « North, South, West, East

- **Stratégie :**

  J'ai associé un format d'index pour chaque direction. En effet, chaque direction correspond à la valeur voisine de la case courante. Cette approche est utilisée pour résoudre les prochaines étapes.



5. Différence entre la valeur courante et les valeurs adjacentes

- **Stratégie :**

Dans cette partie, on va effectuer une opération de soustraction entre la valeur de la case courante et la valeur des cases adjacentes (= associé au index des directions). Cette opération permet de trouver la valeur adjacente avec la valeur la plus proche de la case courante.

Exemple : north_v = valeur adjacente se situant au nord de la case courante

```
# Calculate the difference between the minimal value and the adjacent values (north, south, est, west)
difference_list = [north_v - minValue, south_v - minValue, east_v - minValue, west_v - minValue]
```

On va ensuite prendre le résultat le plus petit parmi cette liste, qui sera la prochaine position ou prochaine valeur minimale.

- **Conditions spécifiques :**

La valeur de la case courante doit être comparer avec toutes les valeurs des cases adjacentes. Au maximum, la valeur minimale doit se comparer à 4 valeurs et au minimum, il faut la comparer à 2 valeurs. Mon code comparera toujours la valeur minimale à 4 valeurs.

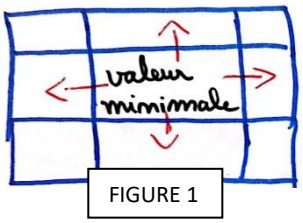


FIGURE 1

Mais, supposons qu'on se trouve dans la situation de la figure 2. Parmi 4 valeurs, 2 valeurs n'existent pas. Donc il faut crée une condition dans laquelle les 2 valeurs prendront des valeurs toujours inferieur à la valeur courante. On ne va pas arrêter la comparaison mais on va les assigner une valeur < valeur de la case courante.

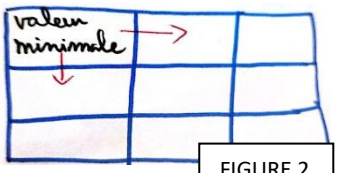
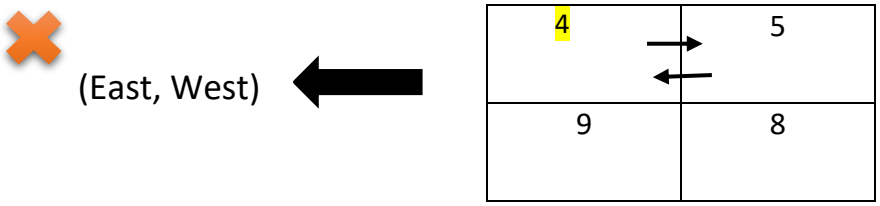
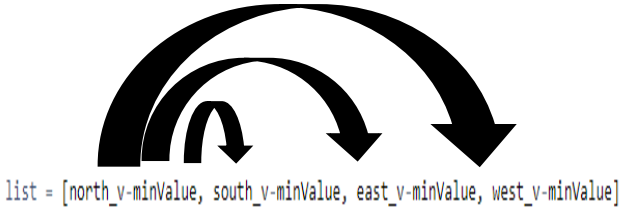


FIGURE 2

De plus, la case déjà parcourue ne doit pas être prise en compte. Dans le cas où la valeur de la case courante est comparé avec les valeurs adjacentes et que si la valeur déjà parcourue est la valeur la plus proche, alors la réponse ne sera pas bonne. Comme dans cet exemple.
C'est pour cette raison aussi qu'on ne doit pas accepter les différences négatives.



Dans notre code on commence par comparer la première valeur de la liste avec le reste. Mais supposons qu'on est dans la situation figure 2, il n'y a pas de ligne au-dessus (index de ligne <0), ce qui sous-entend qu'on va assigner la valeur -1 (idem pour West).




```
if (row_no - 1) < 0:
    north_v = minValue - 1
else:
    north_v = table[row_no - 1, column_no]
```

Vu que la valeur fictive est négative, elle sera toujours plus petite que les autres. Ainsi, la prochaine itération pointerait vers une case qui n'existe même pas. Pour éviter cela, on va créer une boucle qui prendra comme référence de valeur pour la comparaison uniquement les valeurs supérieures à 0. Cela évite de tenir compte des cases et des valeurs fictives

```
indx = 0
min = difference_list[indx]
shouldwestop = False
while((min < 0) and not(shouldwestop)):
    indx = indx + 1
    if indx > 3:
        shouldwestop = True
    else:
        min = difference_list[indx]
```

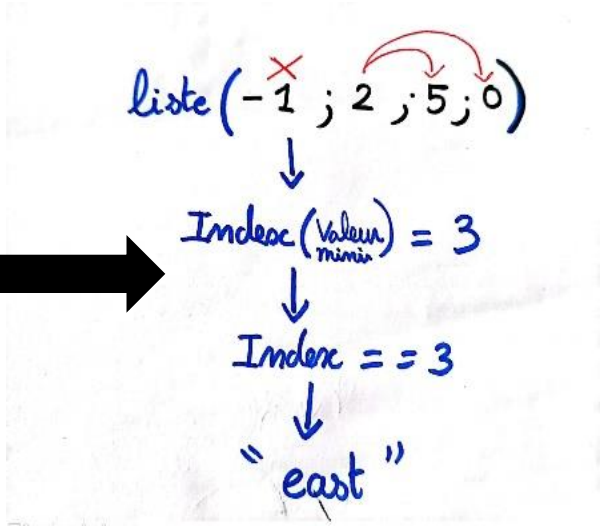
Une fois qu'on a la valeur minimale de la liste, on va ensuite trouver l'index de cette valeur dans la liste. Une fois l'index trouvé, on va associer cet index avec la direction en se basant sur l'ordre des directions dans notre liste. En respectant l'ordre, on va ajouter les directions dans la liste nommé « trajectory ». Le programme mettra à jour la case courante.

```
for i in range(indx + 1, len(difference_list)):
    if(difference_list[i] < min) and (difference_list[i] >= 0):
        min = difference_list[i]
        indx = i
```

6. Associer la valeur avec la direction qui lui correspond

Une fois qu'on a trouvé notre valeur dans la liste, il faudra l'associer avec sa direction

Exemple



7.Plan de test :

	Résultat	Explication
Test 1	[[4. 6.] [8. 7.]] Index of the starting point: (0, 0) 4.0 ['E', 'S', 'W']	Le cas nominal : le programme retourne la bonne trajectoire.
Test 2	Enter value at position (0, 0): n Invalid input! Please enter a valid number. Enter value at position (0, 0):	Une erreur est affichée si l'utilisateur saisit une valeur qui n'est pas un entier pour les éléments du tableau ainsi que pour la dimension du tableau, demandant la ressaisie de la valeur jusqu'à obtenir une valeur correcte.
Test 3	[[4. 6.] [7. 8.]] Index of the starting point: (0, 0) Wrong table input values! 1 value missed ! 4.0 ['E', 'S']	Une erreur est affichée s'il n'est pas possible de parcourir tout le tableau (si les valeurs du tableau ne sont pas correctement rangées en ordre croissant).
Test 4	Enter column number: 0 Error: The column number must be greater than 0. Please try again.	Une erreur est affichée si l'utilisateur saisit une valeur égale à 0 pour le nombre de colonnes.
Test 5	Enter column number: 2 Enter line number: 0 Error: The line number must be greater than 0. Please try again.	Une erreur est affichée si l'utilisateur saisit une valeur égale à 0 pour le nombre de lignes.
Test 6	Enter value at position (0, 0): 4 Enter value at position (0, 1): 2 Enter value at position (1, 0): 4 Value already used! Please enter another unused value.	Une erreur est affichée si l'utilisateur saisit plusieurs fois la même valeur pour les éléments du tableau
Test 7	Enter column number: 1 Enter line number: 1 Error: The dimension of the table must be greater than (1, 1). Please try again.	Une erreur est affichée si la taille du tableau saisie par l'utilisateur est égale à (1,1) et demande une nouvelle saisie.