TD 4 - Matrices

Ferdinand Mom

May 2019

Figure 1: Meme de qualité

1 Classiques

Exercice 1.1: createMat

Créez la fonction **createMat(n, m)** qui retourne une matrice de taille n * m remplit de 0.

```
res = createMat(1, 1)
print(res)
>>>
[
    [0]
]

res = createMat(2, 3)
print(res)
>>>
[
    [0, 0, 0],
    [0, 0, 0]
]
```

Exercice 1.2: dispMat

Créez la fonction **dispMat(M)** qui retourne une string d'une matrice M correctement affiché.

```
M = [
[1, 2, 3],
[4, 5, 6]
]

res = dispMat(M)
print(res)
>>>
1|2|3|
------
4|5|6|
------
```

Exercice 1.3: addMat

Créez la fonction **addMat(M1, M2)** qui retourne la somme des matrices *M1* et *M2*. Levez une exception si nécessaire.

```
M1 = [
[1, 2, 3],
[4, 5, 6]
]

M2 = [
[1, 2, 3],
[4, 5, 6]
]
res = addMat(M1, M2)
print(res)
>>>
[
    [2, 4, 6],
    [8, 10, 12]
]
```

```
M1 = [
[1, 2],
[4, 5]
]

M2 = [
[1, 2, 3],
[4, 5, 6]
]

res = addMat(M1, M2)
print(res)
>>> "The two input matrices don't have the same size."
```

Exercice 1.4: transMat

Créez la fonction **transMat(M)** qui retourne la transposée de la matrice *M*.

```
M = [
[1, 2, 3],
[4, 5, 6]
]

res = transMat(M)
print(res)
>>>
[
    [1, 4],
    [2, 5],
    [3, 6]
]
```

Exercice 1.5: searchMat

Créez la fonction **searchMat(M, x)** qui retourne la position *(i, j)* de la première valeur x trouvée dans la matrice M. Si x n'est pas présent, retournez *(-1, -1)*.

```
M = [
[1, 2, 3],
[4, 5, 6]
]

res = searchMat(M, 3)
print(res)
>>> (0, 2)

res = searchMat(M, 42)
print(res)
>>> (-1, -1)
```

Exercice 1.6: maxGapMat

Créez la fonction **maxGapMat(M)** qui retourne le gap maximum des lignes de la matrice M (que l'on supposera non vide). Dans l'exemple ci-dessous, le gap maximum est celui de la dernière ligne (27 = 20 - (-7)).

```
M = [
[1, 2, 3],
[4, 5, 6],
[-7, 8, 20]
]

res = maxGapMat(M)
print(res)
>>> 27
```

Exercice 1.7: rotateRightMat

Créez la fonction **rotateRightMat(M)** qui retourne à 90° vers la droite. On supposera que les matrices en entrée sont carrés.

```
M = [
[1, 2, 3],
[4, 5, 6],
[7, 8, 9]
]

res = rotateRightMat(M)
print(res)
>>>
[
    [7, 4, 1],
    [8, 5, 2],
    [9, 6, 3]
]
```

Exercice 1.8: rotateLeftMat

Créez la fonction **rotateLeftMat(M)** qui retourne à 90° vers la gauche. On supposera que la matrice M est carré.

```
M = [
[1, 2, 3],
[4, 5, 6],
[7, 8, 9]
]

res = rotateLeftMat(M)
print(res)
>>>
[
    [3, 6, 9],
    [2, 5, 8],
    [1, 4, 7]
]
```

Exercice 1.9: maxPathSum

Créez la fonction **maxPathSum(M)** qui retourne le chemin ayant la plus grande somme de M.


```
M =[
    [3],
  [7, 4],
 [2, 4, 6],
[8, 5, 9, 3]
]

res = maxPathSum(M)
print(res)
>>> 23
```

Exercice 1.10: Harry Potter

Créez la fonction **harryPotter(M)** qui retourne le chemin ayant la plus grande somme de M. Dans l'exemple ci-dessous, le chemin ayant la plus grande somme est $(3 + 8 + 4 = 15)$.

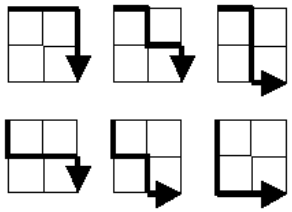
```
M = [
[3, 2, 2],
[8, 0, 1],
[7, 4, 9]
]

res = harryPotter(M)
print(res)
>>> 18
```

Exercice 1.11: Lattice Path

1) Créez la fonction **factorial(x)** qui retourne le factorielle de x.

2) Créez la fonction **latticePath(n, m)** qui retourne le nombre de chemins possibles en partant de la bordure haute-gauche jusqu'à la bordure bas-droite d'un grillage de taille n * m. Voici l'exemple avec un grillage de taille 2*2. Il y a exactement 6 chemins possibles.



```
res = latticePath(2, 2)
print(res)
>>> 6
```

True knowledge exists in knowing that you know nothing.

- Socrates