École Pascal – 1è 1	**Pour le :** 03 / 06 / 2021
Code projet : lpgcdm-0	

Dernier DM de NSI

LPGCDM (Le Plus Grand Carré Du Monde)

Le but de ce DM est de rendre un programme Python qui trouve, dans une carte donnée, **le plus grand carré possible occupant un espace vide**.

```
. o . .          . o . .
. . . o    ➔     x x . o
. . . .          x x . .
```

Le projet LPGCDM vous familiarisera avec une problème récurrent en programmation : la recherche d'une solution optimale parmi un très grand nombre de possibilités, dans un délai raisonnable.

Performance :

Vous serez évalué sur la **performance de votre algorithme**, c'est-à-dire :

- la surface du carré que vous arriverez à trouver,
- le temps que vous mettrez pour trouver ce carré.

Remarque : il est très difficile d'écrire un algo « parfait », qui trouve le meilleur carré à chaque fois dans un temps hyper-réduit : **ayez pour objectif de trouver un compromis raisonnable**.

Votre capacité à trouver le bon carré sera notée sur 5 (votre réponse doit bien sûr être un carré valide). Vos DM seront ensuite triés par performance : le premier devoir obtiendra 5/5 en vitesse, le suivant aura 4/5, etc. La performance représentera donc en tout 10 points sur 20.

Pour indication, la résolution d'une grille de 50 x 50 ne devrait pas conduire à une exécution de plus d'1 min sur mon vieux Macbook Air de 2014 avec 1.4 GHz de CPU.

Contraintes :

- Vous devrez utiliser « sys.argv » pour récupérer les paramètres passés au programme (voir cours de la séance 48). Vous devrez gérer correctement le cas où aucun paramètre n'est passé.
- Le fichier devra être lu depuis un fichier dont le nom sera passé en paramètre (attention : le nom ne pourra pas être donné pendant l'exécution avec `input()`, par exemple).

Vous devez annoter le type de TOUTES vos fonctions (cf. séance 48). Les annotations d'une fonction comprennent :

- le type de ses paramètres,
- et son éventuelle valeur de retour (`None` si la fonction ne retourne rien).

Il va de soi que vos fonctions devront respecter ces annotations : si vous retournez une `string` alors que vous annonciez retourner un `int`, cela vous coûtera des points.

- Vous devrez avoir un découpage en fonctions suffisamment détaillé. Vous devez avoir *au minimum* 1 fonction de lecture de fichier, 1 fonction d'affichage de résultat, 1 fonction de gestion des paramètres, et des fonctions de calcul du résultat. Vos fonctions devront faire 30 lignes maximum.
- Vous devrez avoir une fonction `main()` qui ne prend aucun paramètre et ne retourne rien. Votre `main` sera la seule fonction appelée depuis le scope global (à part le jeu de tests si vous en écrivez), et lancera votre programme proprement dit.

Note : vous aurez à votre disposition un programme pour générer automatiquement des maps (cf. annexe 3).
Chronométrage :

Vous devrez afficher à la fin de la résolution le temps (en h, min, s) mis pour résoudre la map. Ce temps ne mesurera que la durée de la résolution proprement dite.

Bonus : les tests (3 pts)

Vous aurez un bonus de 3 points si vous testez TOUTES vos fonctions avec `assert`.

Attention, vous ne pouvez pas vous contenter d'un test par fonction pour le principe : vous devez tester les différents cas possibles, y compris les « cas-limites » (chaîne ou liste vide, valeur nulle, nombres nuls ou négatifs…). La validité des tests pour le bonus est laissée à l'appréciation du correcteur.

Vous devrez avoir une option pour lancer votre jeu de tests (cf. annexe 1).

Rendu :

Le rendu devra se faire sur votre compte GitHub. La version corrigée sera la dernière qui aura été « pushée » avant l'heure-limite de rendu.

Vous aurez des pénalités dans les cas suivants :

- rendu en retard ou invalide,
- mauvais nom de fichier,
- options mal-gérées (voir annexe 1),
- boucle infinie ou programme qui ne termine jamais,
- programme qui plante pendant l'exécution.

Enfin, même avec les bonus, votre devoir ne peut dépasser 20. Les points bonus ne sont pas reportables à un autre devoir.

Bon courage !

Annexe 1 : exemples de map d'entrée

Les fichiers de map utilisés pour la correction seront au format texte et se termineront par l'extension « .map ».

Les cartes passées en paramètre seront toujours valides :

- toutes les lignes auront la même longueur,
- il y aura au moins une ligne d'au moins une case,
- il y aura un caractère de retour à la ligne à la fin de chaque ligne,
- il n'y aura pas de caractère étrange présent dans la carte.

Dans les fichiers de map, les vides sont notés par un « . » (code ASCII : 46), les pleins par un « o » (ASCII 111). Vous pouvez utiliser le caractère ASCII imprimable de votre choix pour représenter les caractères pleins.

Dans le cas où plusieurs solutions existent, on choisira **le carré le plus en haut puis le plus à gauche**.

Exemple 1 :

Entrée	Sortie attendue
. o . . . o . . .	en mode « -r » : . o . x x o x x . Temps : 0.3 s en mode « -c » : 0 1 1 2

Exemple 2 :

Entrée	Sortie attendue
o o . o o o . o . . . o . . . . o . . . . o o . . o . . . . . o . . . o . o . . . . . o . . . . o o . . . o . o . . o . . . . o o o . o . o o . . o . . . . o . o o . . . o . o o o o o . . . . . . o . . . o . . . o . o . . . . . o o o . . . . o . o . . . . . o . o . . . . o .	en mode « -r » : o o . o o o . o . . . o . . . . o x x x . o o . . o . . x x x o . . . o . o . x x x . o . . . . o o . . . o . o . . o . . . . o o o . o . o o . . o . . . . o . o o . . . o . o o o o o . . . . . . o . . . o . . . o . o . . . . . o o o . . . . o . o . . . . . o . o . . . . o . Temps : 1 min 2 s en mode « -c » : 6 1 8 3

Exemple 3 :

Entrée	Sortie attendue
o o o o	en mode « -r » : o o o o Temps : 0.1 s en mode « -c » : 0 0 0 0

Remarques pour le mode –r :

- Vous ne devez pas modifier la map originale : seul votre carré-solution ira remplacer la zone vide.
- Votre impression doit préserver l'affichage proposé dans la map, avec une espace entre chaque caractère.

Annexe 2 : les paramètres du programme

Votre programme doit gérer toutes les options suivantes :

`-f` ou `--file`	**Ce paramètre est obligatoire** : il permet de passer en paramètre le fichier de la map. `?> python nom_prenom.py –f file1.map` … Note : vous devez gérer le cas où aucun fichier correct n'est passé après `–f`.

`-p` ou `--project`	Affiche le code projet du DM.
	Exemple :
	`?> python nom_prenom.py --project`
	`Projet : lpgcdm-0`
	`…(suite du programme)`

`-a` ou `--authors`	Affiche les auteurs du projet rendu.
	`?> python nom_prenom.py -a`
	`Auteur1 : Julia Petitbidon`
	`Auteur2 : Sylvestre Jolitruc`
	…

`-v` ou `--verbose`	Affiche à la fois les options de `--project` et `--authors`.
`-h` ou `--help`	Affiche le texte d'aide de votre choix.

`-r` ou `--print-result`	**Mode de rendu 1 :** affiche la carte de départ avec le carré trouvé (c'est l'option par défaut).
	`?> python nom_prenom.py --print-result`
	`. o .`
	`X X o`
	`X X .`

`-c` ou `--print-coordinates`	**Mode de rendu 2 :** affiche la coordonnée du sommet en haut à gauche du carré (abscisse/ordonnée), puis la coordonnée du sommet en bas à droite.
	`?> python nom_prenom.py -c`
	`5 6 13 11`

`-t` ou `--test`	(Facultatif) Lance vos jeux de tests. Ceci annule l'exécution normale du programme, càd que vous ne devez pas résoudre la map.

Annexe 3 : générateur de maps en Python (à améliorer comme vous le souhaitez)

```
from random import random

CHR_FILL = "o"
CHR_EMPTY = "."
MAP_DENSITY = 0.3   # pour modifier le ratio pleins/vides
SHOULD_PRINT_IN_FILE = False


def gen_map(x: int, y: int, density: float) -> list[str]:
    map = []

    for i in range(y):
        s = CHR_FILL if random() < density else CHR_EMPTY
        for j in range(x - 1):
            # Notez le if... else: ... en notation raccourcie
            c = CHR_FILL if random() < density else CHR_EMPTY
            s += " " + c
        map.append(s)

    return "\n".join(map) + "\n"


if __name__ == "__main__":
    while True:
        try:
            width = int(input("Entrez la largeur de la map : "))
            height = int(input("Entrez la hauteur de la map : "))
        except ValueError:
            print("Valeurs entières plz")
        else:
            map = gen_map(width, height, MAP_DENSITY)
            break

    if SHOULD_PRINT_IN_FILE:
        while True:
            name = input("Nom de fichier ? ")
            try:
                fo = open(f"{name}.map", "x")
            except FileExistsError:
                print(f"{name}.map existe déjà, veuillez saisir un autre nom")
            else:
                fo.write(map)
                fo.close()
                break

    print(map)
    print("Done")
```