

Exercice 1

```
# Créer une fonction square_root prenant un nombre en paramètre, et retournant la
racine carrée de ce nombre (vous pouvez utiliser le module math)
# Printer l'appel de la fonction avec un nombre donné

import math
def square_root(number):
    return math.sqrt(number)

print(square_root(4))
```

Exercice 2

```
# Créer une fonction extendator ne prenant aucun paramètre, et qui ne retourne
rien.
# Faire en sorte qu'elle modifie la liste suivante définie dans l'espace global en
l'étendant de la liste suivante [4, 5, 6]

liste_depart = [1, 2, 3]

def extendator():
    liste_depart.extend([4, 5, 6])

extendator()

print(liste_depart)
```

Exercice 3

```
# Créer une fonction add prenant en paramètres deux nombres et qui retourne
l'addition de ses deux nombres
# Créer une fonction carre qui prend un nombre en paramètre et retourne ce dernier
élevé au carré
# A l'aide de ses deux fonction, retourner la somme des carrés de 2 et 3 ( $2^2 + 3^2$ )
# Printer le résultat

def add(x, y):
    return x + y

def carre(x):
    return x * x

print(add(carre(2), carre(3)))
```

Exercice 4

```
# Le labyrinthe
# Partant de maze[0][0] (8) dans le labyrinthe, créer une fonction récursive
capable de trouver la sortie (9)
# Les 0 présents dans le labyrinthe sont les murs infranchissables
# Les 1 sont des chemins possibles
# Parcours en diagonale interdit
# L'entrée du labyrinthe sera toujours à [0][0]
# La fonction fonctionne quelque soit la longueur et la largeur du labyrinthe
# La fonction ne prévoit pas le cas de figure suivant où quatre 1 sont côte à côte
:
# [[8, 1, 1*, 1*]
# [0, 0, 1*, 1*] -> *quatre 1 en carré interdit boucle infinie probable
# [0, 0, 0, 9]]
```

```

#   x = 0  1  2  3  4  5      y
maze = [[8, 1, 1, 1, 1, 0], # 0
        [1, 0, 1, 0, 1, 0], # 1
        [1, 0, 1, 1, 1, 1], # 2
        [1, 0, 0, 0, 1, 0], # 3
        [1, 1, 1, 0, 1, 0], # 4
        [0, 1, 0, 0, 1, 9]] # 5

def find_maze_exit(m: list[list[int]], x: int, y: int):
    print("Exploration aux indexes:", x, y)

    if m[y][x] == 9:
        print(x, y)
        exit()

    m[y][x] = -1

    # on explore à gauche
    if x > 0:
        # On s'assure de ne pas sortir des limites du labyrinthe
        if m[y][x - 1] >= 1:
            find_maze_exit(m, x - 1, y)

    # on explore à droite
    if x < len(m[0]) - 1:
        if m[y][x + 1] >= 1:
            find_maze_exit(m, x + 1, y)

    # on explore en haut
    if y > 0:
        if m[y - 1][x] >= 1:
            find_maze_exit(m, x, y - 1)

    # on explore en bas
    if y < len(m) - 1:
        if m[y + 1][x] >= 1:
            find_maze_exit(m, x, y + 1)

find_maze_exit(maze, 0, 0)

```