Spring 2026

Programmation 2

TP 6 — *Lisez attentivement l'ensemble du document.* — *Nabil Mustafa*

Consignes

(a) *L'objectif est d'apprendre la programmation en Python.* ***Inutile de se précipiter !*** *Prenez votre temps et terminez correctement tous les TP précédents avant d'aborder celui-ci.*

(b) *Pour réussir,* ***lisez le texte attentivement et lentement****. Ne parcourez pas les consignes en diagonale. Lisez chaque phrase une fois, puis une deuxième. Si un passage reste flou après plusieurs lectures, c'est le moment idéal pour poser une question au professeur. C'est ainsi que l'on apprend efficacement.*

(c) *Avant de commencer,* ***mettez votre téléphone en mode « ne pas déranger » et rangez-le dans votre sac****. Chaque notification consultée brise votre concentration et vous oblige à reprendre le fil de votre raisonnement, ce qui allonge considérablement le temps nécessaire pour terminer l'exercice et favorise les erreurs.*

(d) *Adoptez une approche progressive : lisez une instruction, puis exécutez-la immédiatement. Évitez de lire tout le TP d'un coup.* ***Avancez étape par étape pour rester concentré sur l'action immédiate et ne pas vous sentir submergé par un long texte.***

(e) *Souvenez-vous : les étudiants qui réussissent le mieux ne sont pas ceux qui « finissent en premier », mais ceux qui* ***lisent avec soin, posent des questions lorsqu'ils bloquent et restent concentrés*** *jusqu'au bout.*

Votre objectif aujourd'hui n'est pas d'aller vite, mais de comprendre profondément.

Tâche 1: Exécuter Python

(a) Connectez-vous à `LINUX` avec votre compte utilisateur.

(b) **Téléchargez** le fichier `gasstationspuzzle.py`.

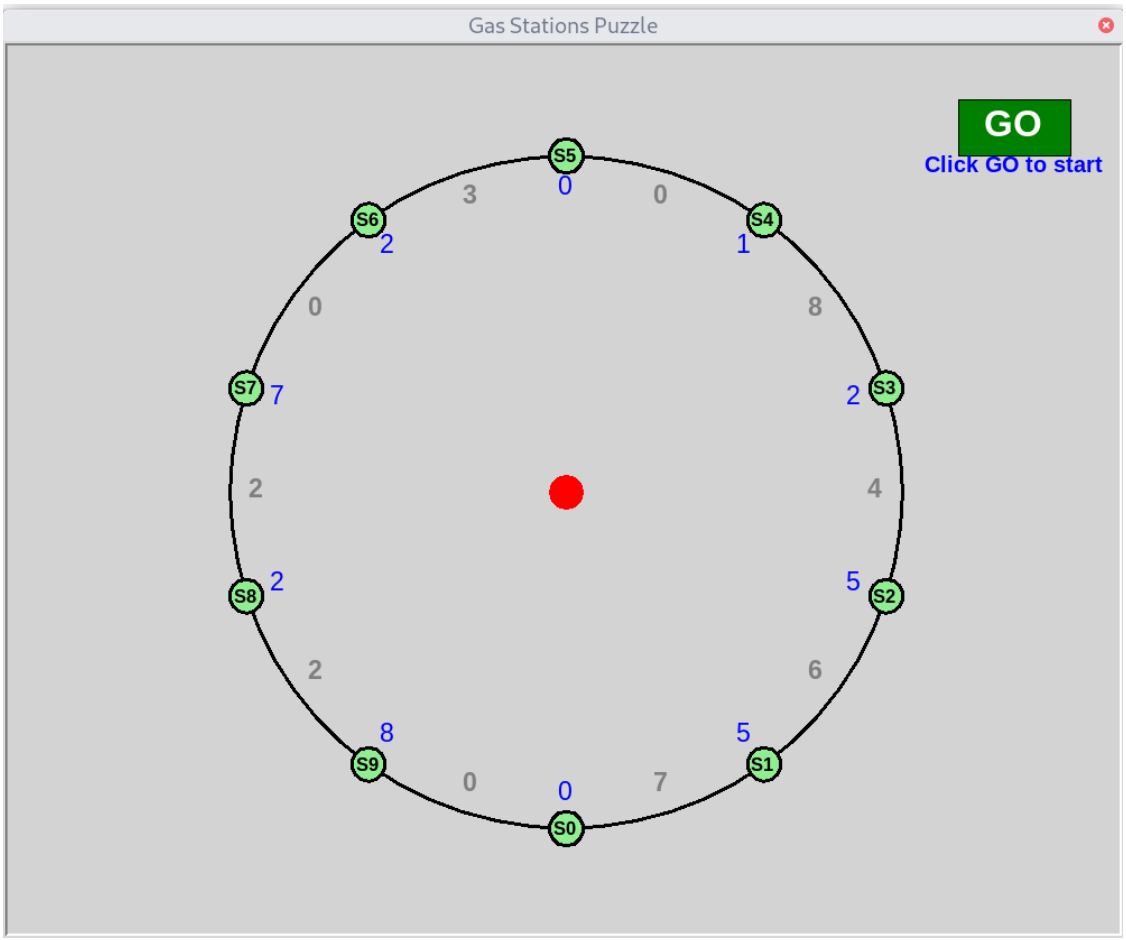
(c) **Éditer** ce fichier avec l'éditeur de votre choix. Par exemple, ouvrez un `terminal` dans le même dossier que le fichier `gasstationspuzzle.py`, et exécuter le command:

```
kate gasstationspuzzle.py
```

(d) **Exécuter** le code avec `Python`: ouvrez un `terminal` dans le *même dossier* que le fichier `gasstationspuzzle.py`, et exécuter le command :

```
python3 gasstationspuzzle.py
```

Vous verrez cette interface que j'ai créée pour vous.



Gas Stations Puzzle

On considère une piste circulaire comportant n stations-service disposées tout autour, numérotées de 0 à $n-1$ dans le sens anti-horaire.

Chaque station i, où $i \in [0, n-1]$, dispose d'une certaine quantité d'essence, notée `gas[i]`.

Cette quantité peut varier d'une station à l'autre—certaines peuvent avoir très peu, voire pas d'essence du tout, tandis que d'autres peuvent en avoir un large excédent.

Soit `cost[i]` la quantité d'essence nécessaire pour se rendre de la station i à la station $(i+1) \bmod n$.

La quantité totale d'essence disponible, toutes stations confondues, est exactement suffisante pour effectuer un tour complet de la piste.

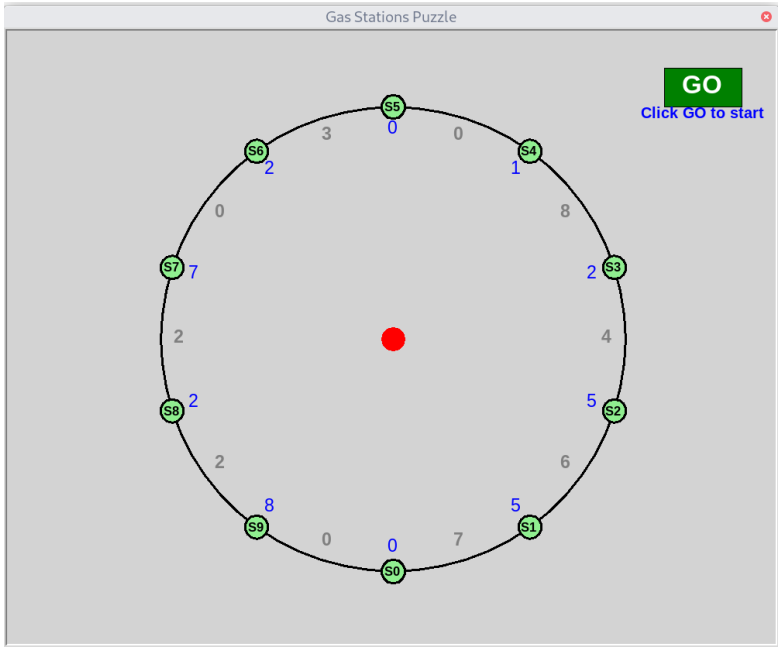
Autrement dit,

$$\sum_{i=0}^{n-1} \texttt{gas[i]} = \sum_{i=0}^{n-1} \texttt{cost[i]}.$$

Vous commencez votre trajet avec le réservoir vide. À chaque fois que vous atteignez une station, vous pouvez récupérer toute l'essence qui s'y trouve et l'ajouter à votre réservoir.

Votre objectif est de choisir une station de départ telle qu'en suivant la piste dans l'ordre, vous ne tombiez jamais en panne d'essence et puissiez ainsi effectuer un tour complet (c'est-à-dire revenir à votre point de départ).

Exemple. Considérons l'exemple suivant, avec $n = 10$ et $\sum_{i=0}^{9} \texttt{gas[i]} = \sum_{i=0}^{9} \texttt{cost[i]} = 32$.

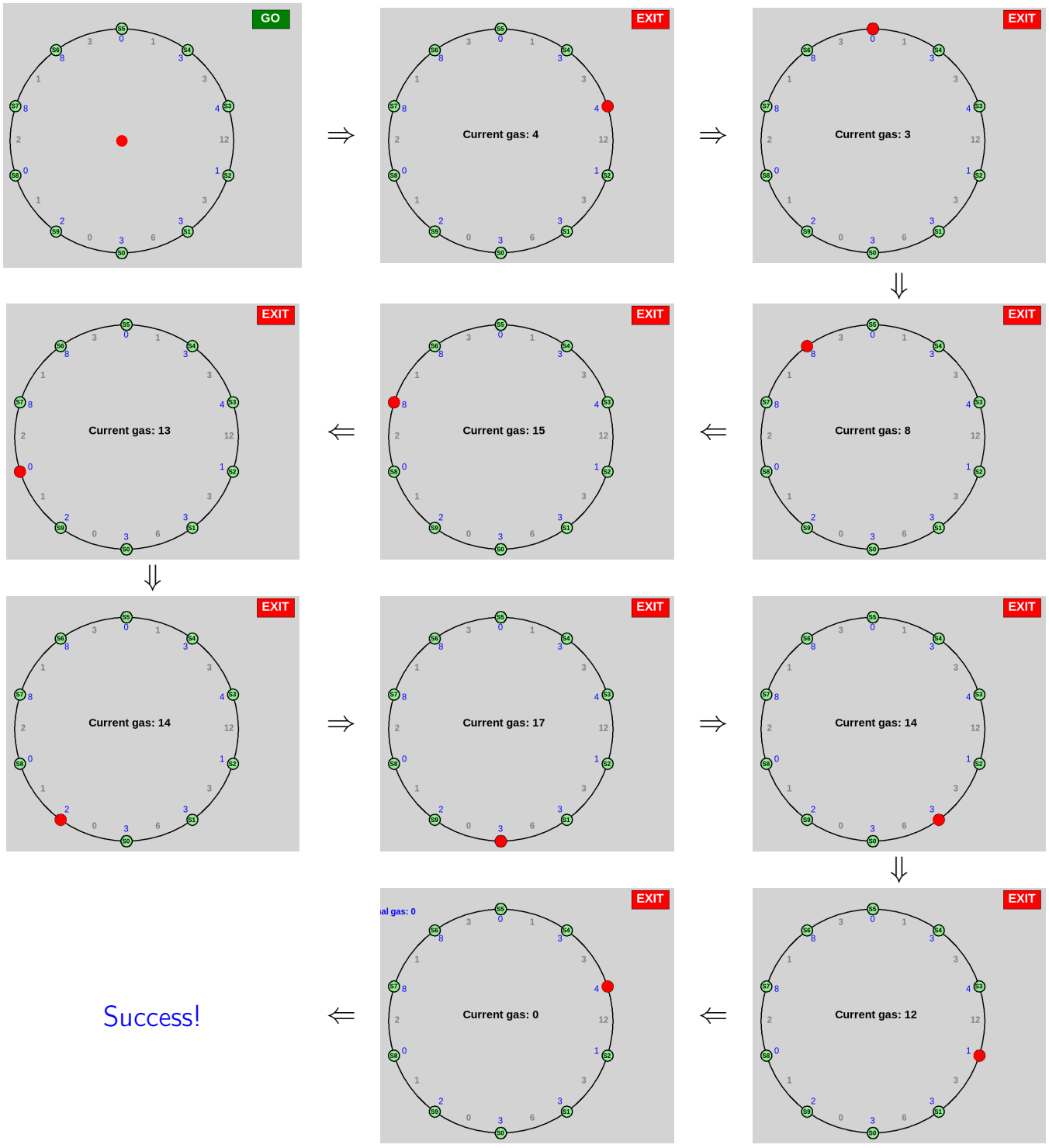


Si on part de la station `S4` : $+\underbrace{1}_{\texttt{gas[4]}} - \underbrace{0}_{\texttt{cost[4]}} + \underbrace{0}_{\texttt{gas[5]}} - \underbrace{3}_{\texttt{cost[5]}} = -2 \ldots$ panne d'essence !

Si on part de la station `S2` : $+\underbrace{5}_{\texttt{gas[2]}} - \underbrace{4}_{\texttt{cost[2]}} + \underbrace{2}_{\texttt{gas[3]}} - \underbrace{8}_{\texttt{cost[3]}} = -5 \ldots$ panne d'essence !

Exemple

Si on part de la station S3:



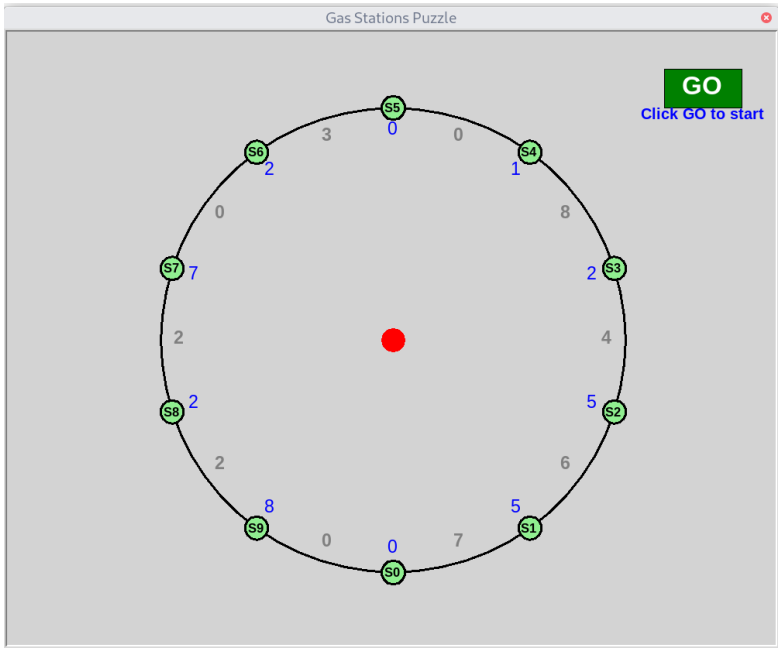
Tâche 2: Écrivez un algorithme pour Gas Stations Puzzle

Votre objectif est d'écrire le code dans la fonction `find_valid_start(gas, cost)` qui prendra en entrée une liste `gas` contenant les quantités d'essence des n stations, et une liste `cost` contenant les n coûts.

La fonction doit retourner l'indice de la station telle qu'en partant de cette station, on puisse effectuer le tour complet de la piste sans tomber en panne d'essence.

Cette fonction est appelée lorsque vous appuyez sur le bouton GO.

Pour l'exemple du problème donné dans l'image ci-dessous, voici les deux listes reçues par la fonction `find_valid_start(gas, cost)` :




```
gas  = [0, 5, 5, 2, 1, 0, 2, 7, 2, 8]
cost = [7, 6, 4, 8, 0, 3, 0, 2, 2, 0]
```

Voici la seule fonction dans `gasstationspuzzle.py` que vous devez modifier:

```
def find_valid_start(gas, cost):
    print("Gas: ", gas)
    print("Cost: ", cost)
    n = len(gas)

    #ECRIVEZ VOTRE CODE ICI

    return 0  # <--- il faut retourner la bonne indice
```