

 **** 

**Projet python :**

**Classe : Ingenierie informatique et technologie**

**émergente 1.**

**Module : Algorithmique avancée et Python.**

**Groupe : 1**

**Réalisé par :**

**Zakaria el harchi -Ilyas Bennane**

**Ayman Cherkani -Safiya Daoudi -Hajar Machmoum**

 **2022-2023**



**Sommaire :**

1. **Introduction :**

* **Objectif du projet**

1. **Pseudocode :**

* **Le pseudocode associé au projet**

1. **Organnigramme :**

* **Représentation du projet complet en utilisant un organnigramme simplifié**

1. **Explication des principaux étapes du projet :**

* **Extraction des données**
* **Génération d’un fichier csv**
* **Construction de deux matrices :**

**Matrice de temps – Matrice de distance**

* **Partie du plus court chemin**

1. **Explication de GUI :**

* **Première fenetre**
* **Deuxième fenetre**
* **Code python de GUI**
* **Explication des principaux parties du code**

1. **Explication des codes pythons du projet :**

* **Code sous python de la partie d’API**
* **Code sous python des matrices**



* **Code sous python de la partie du plus court chemin**
* **Explication de Travelling Salesman Problem**
* **Code sous python de traçage du chemin dans Map**

1. **Conclusion :**



**INTRODUCTION :**

En raison des nombreuses étapes et des parties

impliquées , la distribution de gaz aux diverses

stations de service peut être une procédure difficile

et compliquée . Le transport du gazoil peut être l'un

des facteurs qui rend difficile cette distribution.En

effet, le gazoil doit être transporté de raffineries ou

de terminaux de stockage jusqu'aux stations de gaz,

ce qui peut être coûteux en raison des frais de

transport et de logistique.En outre, Il y a plusieurs

moyens de transporter le gazole, tels que par

camion, par train ou par pipeline. Mais, ce qui nous

intéresse dans notre mini-projet, c’est la

distribution du gazoil par camion en utilisant une

bonne stratégie en minimisant le cout du

transport,c-à-d on va travailler à écrire un code

python qui va permettre le passage d’un camion

d’une station de départ choisie vers tous les autres

stations (shell) en utilisant le plus court chemin



**Pseudocode :**



**Algorithme :Plus\_court\_chemin**

**Variables :matrix,S**

**Début :**

**Ecrire(« Bienvenue,veuillez choisir la station de départ »)**

**Lire( S )**

**Si S est dans (liste des stations) :**

**Solve\_salesman\_problem\_function(matrix,S)**

**Retourner liste**

**Afficher map qui contient le plus court chemin**

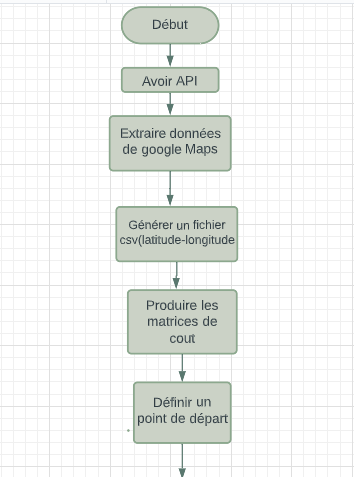
**Sinon**

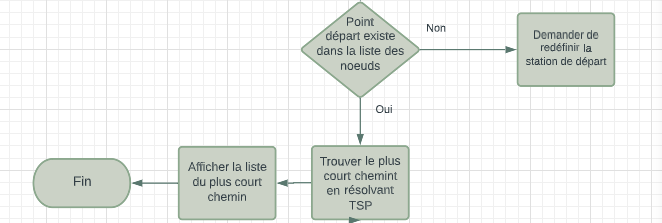
**Ecrire(« la station que vous avez entré est invalide,veuiller saisir le point de départ une autre fois »)**

**Fin**



**Flowchart :**







**Explications des principaux étapes de notre projet :**

* **Extraction des données :**

L'extraction de données de **Google Maps** peut être très utile dans de nombreux domaines, tels que la cartographie, l'analyse de données, la géolocalisation et la planification de trajets. Elle peut permettre de récupérer des informations précieuses sur les lieux, les itinéraires et les distances, ainsi que sur les caractéristiques des régions et des villes.

Il existe plusieurs façons d'extraire des données de Google Maps. L'une des façons les plus simples est d'utiliser l'API de Google Maps, qui offre un accès programmatique aux données de Google Maps. Cela permet de récupérer des informations telles que les coordonnées géographiques, les itinéraires et les distances, ainsi que des données sur les points d'intérêt et les événements.

Il est également possible d'extraire des données de Google Maps en utilisant des outils de scraping de données Web. Ces outils permettent de récupérer automatiquement des données à partir de sites Web en utilisant des scripts ou des programmes. Cependant, il est important de se rappeler que l'utilisation de ces outils peut être soumise à des restrictions et à des conditions d'utilisation strictes, et il est important de respecter les termes d'utilisation de Google Maps.

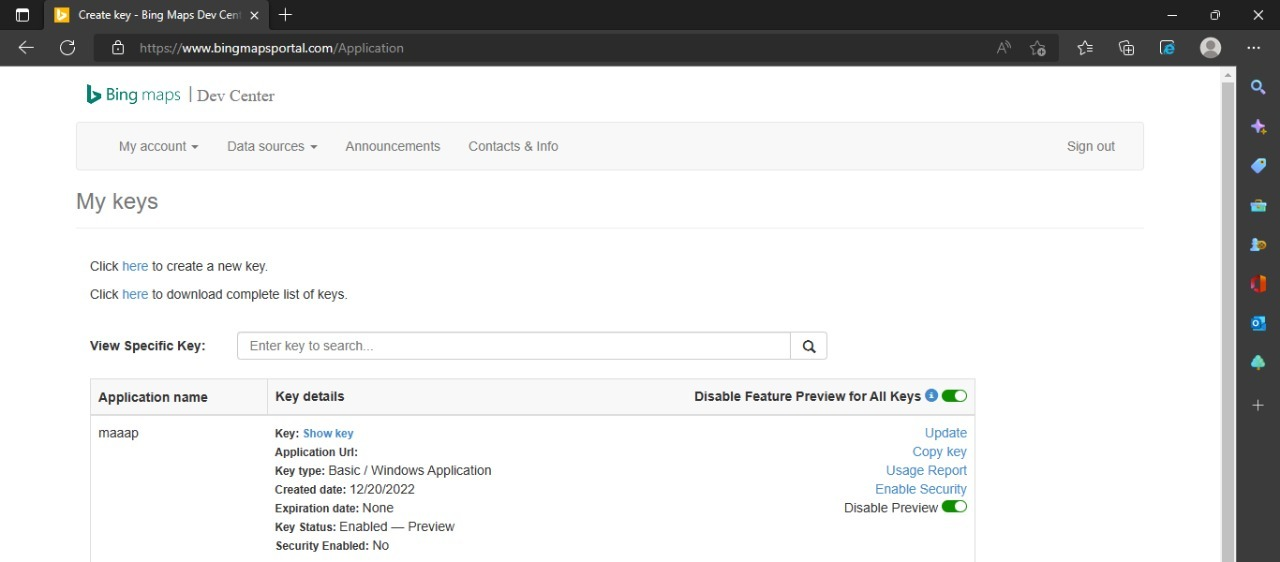
Puisque l’API est un outil payant ,on a trouvé après plusieurs recherches qu’on peut utiliser un moyen plus adaptable et gratuit ,c’est le **Bing Maps**



Bing Maps est un service de cartographie en ligne développé par Microsoft. Il permet aux utilisateurs de visualiser, d'explorer et de planifier des itinéraires sur une carte en ligne, ainsi que de rechercher des points d'intérêt et des adresses. Bing Maps propose également des



images aériennes et des vues de rue à 360 degrés, ainsi que des données sur la circulation et les trajets en transport en commun.

Bing Maps est disponible gratuitement et peut être utilisé sur différents appareils, tels que les ordinateurs, les téléphones mobiles et les tablettes. Il est également possible de l'intégrer à des applications et à des sites Web en utilisant l'API (Application Programming Interface) de Bing Maps. L'API de Bing Maps permet aux développeurs de créer des applications qui utilisent les données et les fonctionnalités de Bing Maps.En général, c’est un service de cartographie en ligne complet et facile à utiliser qui peut être utile pour visualiser, explorer et planifier des itinéraires, ainsi que pour rechercher des points d'intérêt et des adresses. Il peut être particulièrement utile pour les entreprises et les organisations qui souhaitent intégrer les données de cartographie à leurs applications et sites Web.

Ce service Bing maps permet de générer un Bing key (Bing API) ,ce qui va nous aider à extraire longitude et la latitude de chaque station **shell**

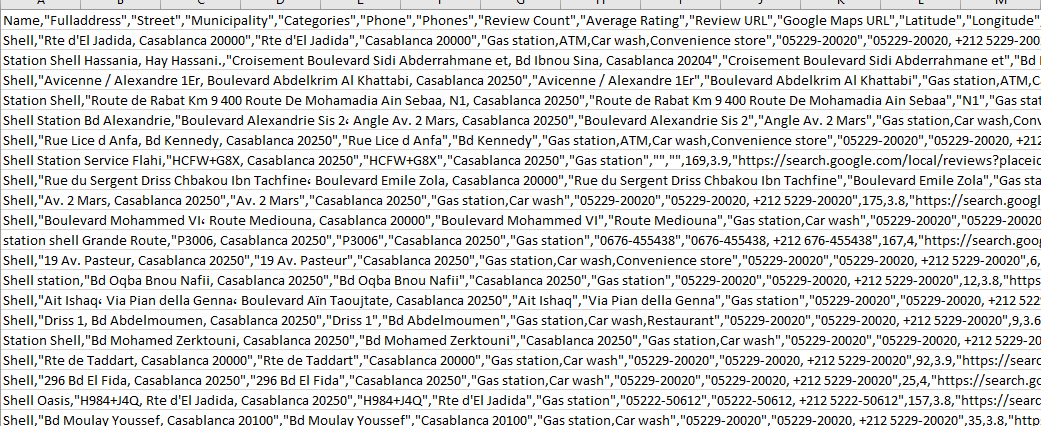
* **Génération d’un fichier csv :**

Après l’étape précedente,c-à-d après la génération d’un Bing API .on utilise le web scraping ,ce qui consiste à récupérer automatiquement des données sur le web à partir de sites web.Dans notre cas, on va utiliser cet outil informatique pour produire un fichier csv contenant tous les stations désirées avec la latitude ainse que la longitude de chaque point (chaque station shell) , on a utilisé un site nommé Selenium qui a déjà une bibliothèque convenable dans python

On convertit fichier csv généré en format Excel puisque l’API peut manipuler et exploiter juste les fichiers Excel



Et dans ce fichier,on trouve toutes les informations concernant chaque station du MAP (nom de station,adresse,catégorie,longitude,latitude…)



* **Génération des matrices :**

Après avoir le fichier csv, on va exploiter ce dernier dans l’API en utilisant une bibliothèque de lecture de CSV, comme csv pour Python pour générer deux matrices :une matrice qui définit le temps,et l’autre définit la distance entre tous les stations shell .Les deux matrices sont en format Excel comme le montre les deux figures suivantes



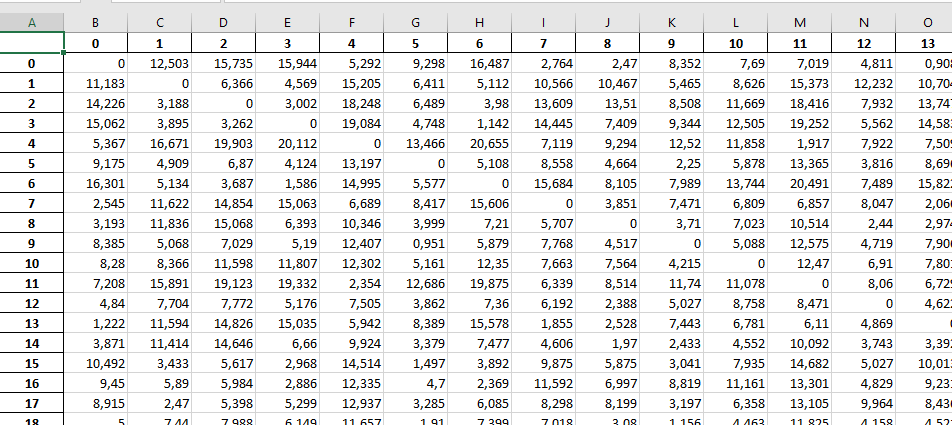
***Matrice de temps*** :



Cette matrice identifie la durée entre chaque station shell de Map avec tous les autres stations restantes

***Matrice de distance :***





Cette matrice identifie la distance entre chaque station de map avec tous les autres stations shell restantes .

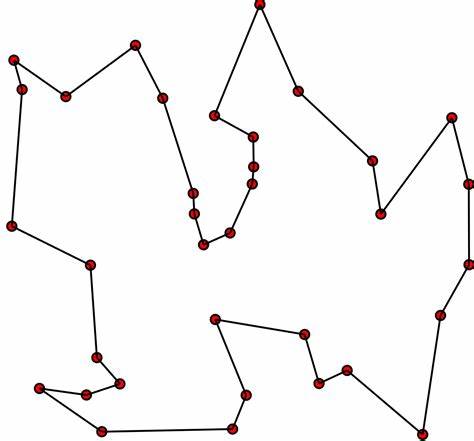
* **Trouver le chemin optimal :**

Pour trouver le plus court chemin qui va nous permettre de passer par tous les stations shell de Casablanca ,plusieurs algorithmes ont été proposé .En revanche ,on a remarqué que notre problème correspond au TSP (Traveling Salesman Problem) c’est un problème d'optimisation classique en informatique qui consiste à trouver l'itinéraire le plus court possible qui visite un ensemble donné d'emplacements, à revenir au point de départ et à visiter chaque emplacement exactement une fois.

Pour la résolution de ce problème ,on peut utiliser la biliothèque tsp existante dans python ou bien pytsp…,où on peut utiliser comme entrées les matrices générées précédement afin de trouver le plus court chemin en visitant tous les stations

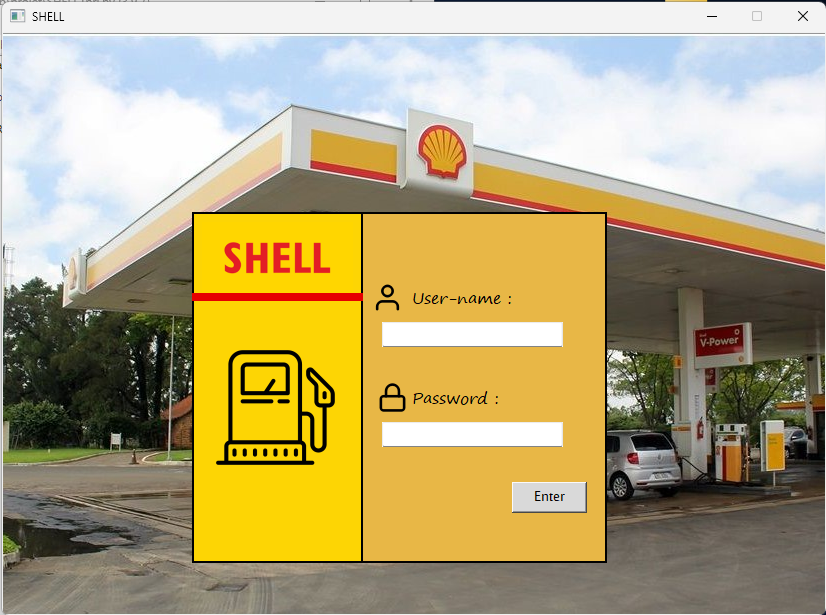
Voici un exemple de fonctionement :





**Explication de GUI :**

**1ère fenêtre :**





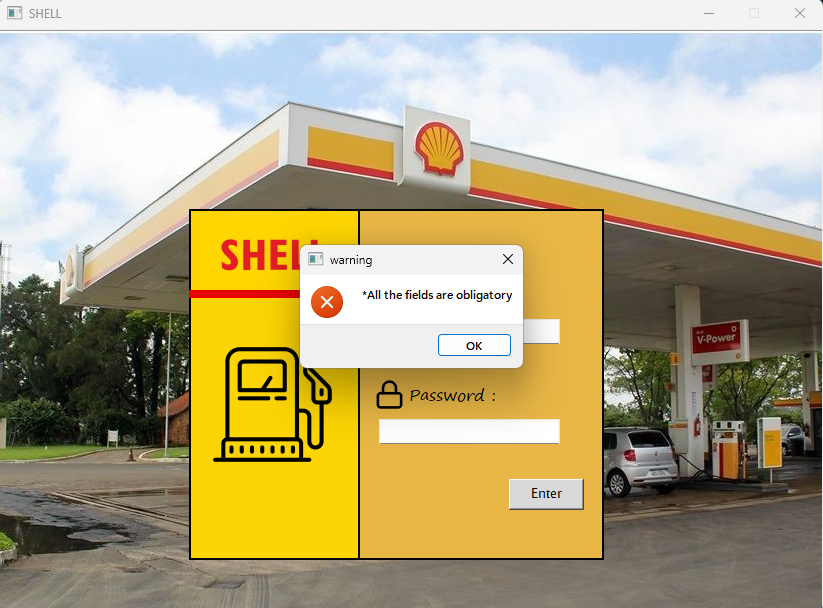
Cette fenêtre permet à l’utilisateur (Chauffeur du camion) de se connecter à l’interface en entrant le login et le mot de passe

Cette fenêtre possède un boutton « Enter » pour permettre à l’utilisateur d’entrer et poursuivre les prochaines étapes

**Affichage d’un message d’erreur :**



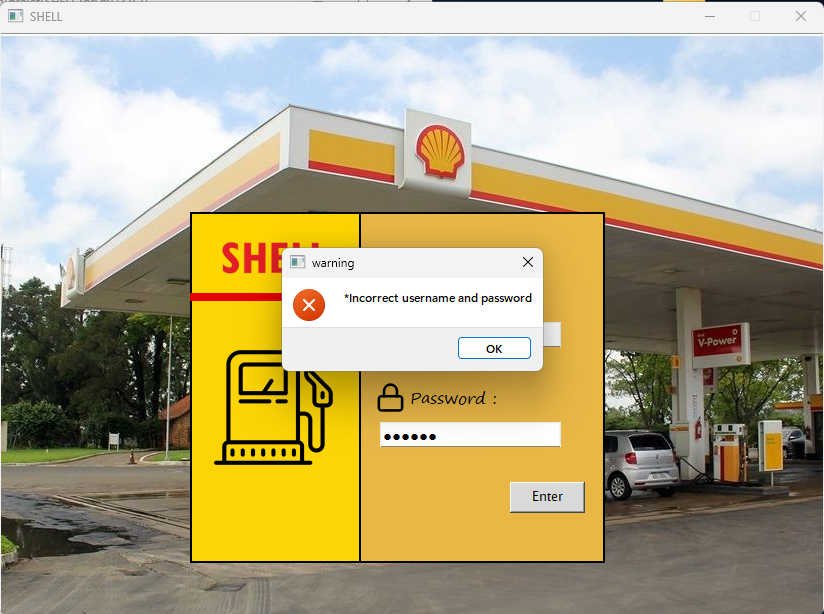
**1er cas :**



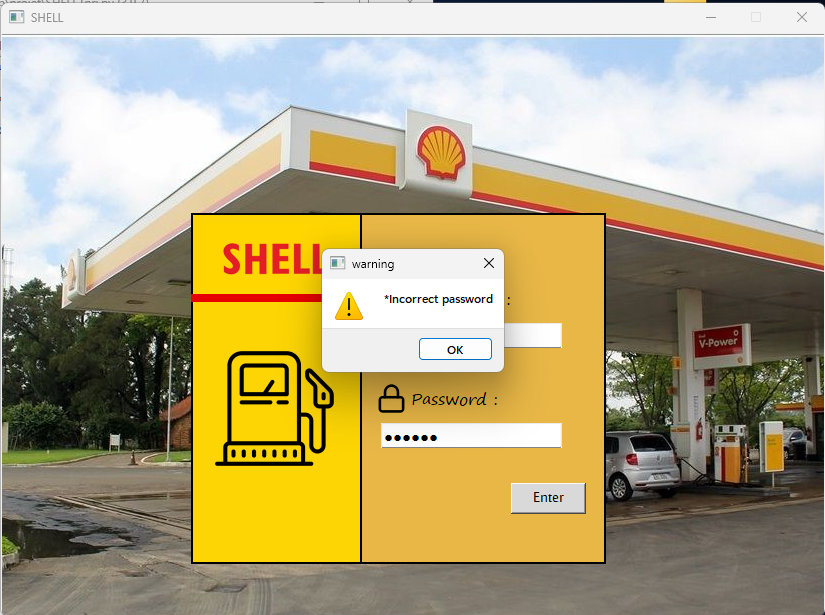
Si on remplit pas les valeurs demandés ,un message d’erreur s’affichera comme quoi on est obligé de remplir tous les cases.

**2ème cas :**

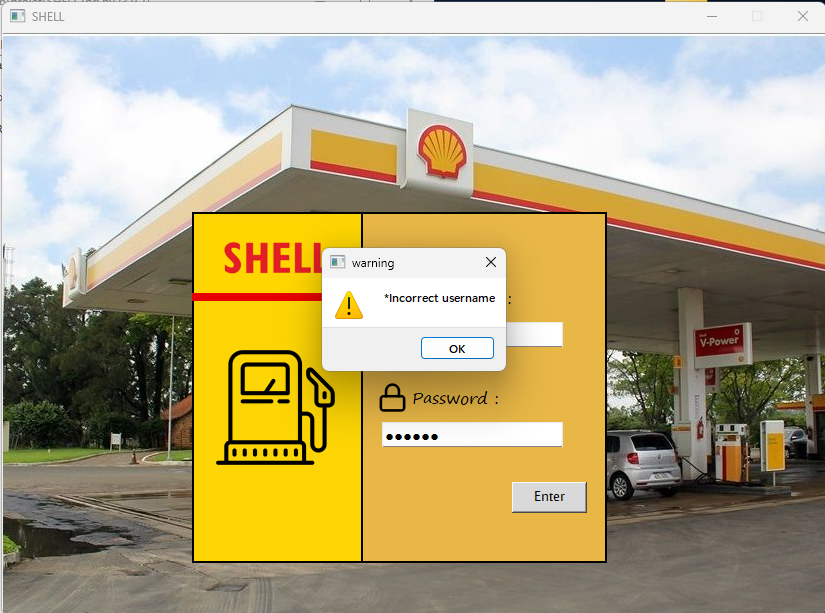




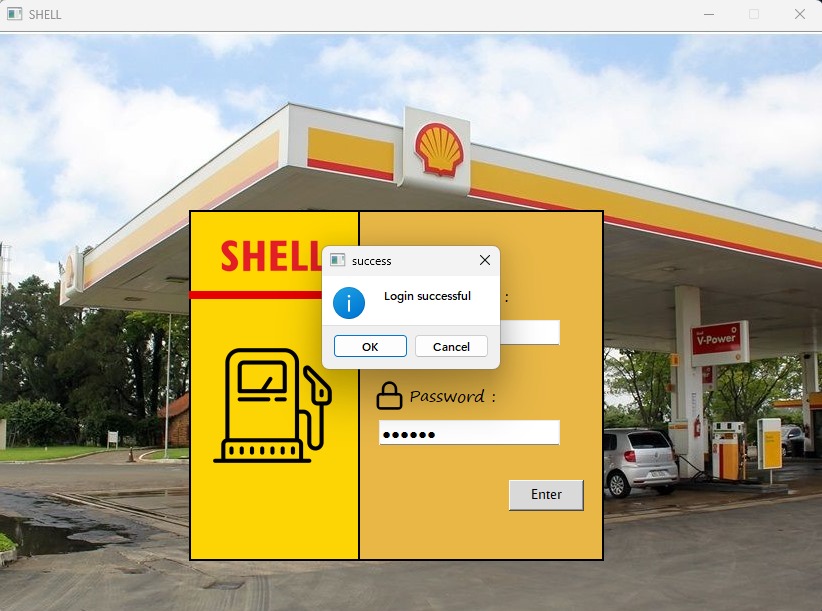
Si on entre pas les valeurs correctes ,un message d’’erreur s’affichera ,par exemple dans le cas où on entre un login et un mot de passe invalide ou bien l’un des deux est incorrecte comme le montre les captures ci-dessous.

****



****

**3ème cas :**

****



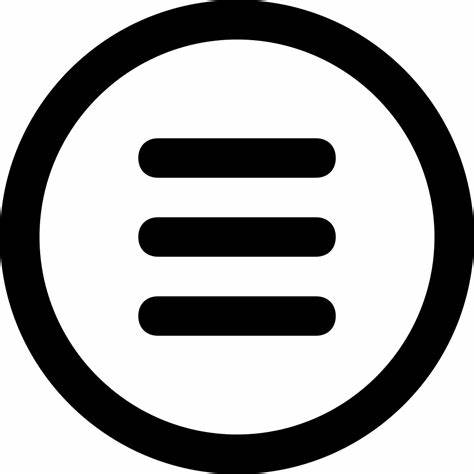
Ce dernier cas est le cas de ‘Tout va bien’,c-à-d :Lorsqu’on entre un nom d’utilisateur et un mot de passe correcte.Alors ,là le message « Login Successful » s’affichera ,donc l’utilisateur peut passer à la deuxième fenetre .

**2ème fenêtre :**



Lorsqu’on clique sur les symboles indiquées sur la figure ,une explication des symboles s’affichera sur l’interface

Par exemple si on clique sur le symbole « Menu » :



Ça donne :





C-àd,l’utilisateur peut effectuer que les options suivantes :

Consulter les stations shell de casablanca

Cliquer sur « Paths » pour voir le plus court chemin liant tous les stations à partir d’un point de départ bien précis

Si on clique sur le boutton « Map », On remarque alors l’affichage d’une zone géographique ‘MAP’ qui contient tous les stations shell qui se trouvent dans Casablanca ,comme le montre la figure ci-dessous.



On remarque que l’utilisateur peut controler l’image comme il veut c-à-d ,il peut effectuer des zooms ……



Maintenant,si on veut voir le plus court chemin permettant de visiter tous les stations **shell** en minimisant la distance ,on clique sur le boutton « Paths »





On remarque ainsi qu’on peut se déconnecter en tappant sur le boutton « Log-Out »

**Codes sous python de GUI :**

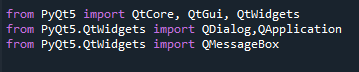
Pour réaliser cette interface ,on a pensé a utiliser Qt-designer



**Pour la 1ère interface :**

Parmi les principaux bibliothèques utilisées,on trouve :





QtCore, QtGui et QtWidgets sont des modules (c'est-à-dire des bibliothèques) du framework Qt qui fournissent respectivement des fonctionnalités de base non graphiques et graphiques.

**QtCore** fournit un ensemble de classes non GUI, y compris les types de données et la gestion des fichiers.

**QtGui** fournit des classes pour l'intégration du système de fenêtrage, les graphiques 2D et l'imagerie.

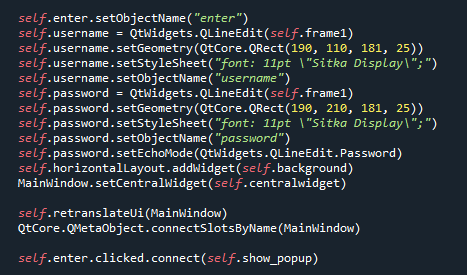
**QtWidgets** est un module qui fournit un ensemble de widgets GUI (c'est-à-dire des contrôles), tels que des boutons, des cases à cocher et des curseurs.

**Des parties du code**

****



* Ce code définit une classe appelée Ui\_MainWindow, qui est utilisée pour configurer l'interface utilisateur d'une fenêtre principale dans une application Qt. La setupUiméthode est utilisée pour configurer la fenêtre principale et ses widgets
* La fenêtre principale est créée avec une taille par défaut de 824x583 pixels et reçoit le nom d'objet "MainWindow". Un widget central est créé et ajouté à la fenêtre principale. Ce widget contiendra les autres widgets qui composent l'interface utilisateur.
* Un cadre d'arrière-plan est créé et ajouté à la mise en page. Ce cadre reçoit une feuille de style qui définit son image d'arrière-plan sur un fichier spécifique. Le cadre est également doté d'une forme de cadre de panneau stylé et d'une ombre en relief.

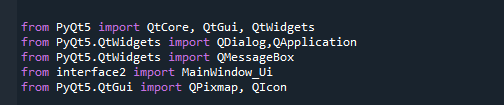


* Le code crée un bouton "entrer" et deux widgets d'édition de ligne, un pour le nom d'utilisateur et un pour le mot de passe. Les widgets d'édition de ligne reçoivent une taille et une position fixes dans le widget "frame1" et un style de police via une feuille de style.
* L'édition de la ligne de mot de passe a son mode d'écho défini sur "Mot de passe", ce qui signifie que les caractères saisis par l'utilisateur seront remplacés par des astérisques pour masquer le mot de passe.
* Enfin, une connexion est établie entre le signal cliqué du bouton "entrer" et une méthode appelée "show\_popup", de sorte que la méthode "show\_popup" sera appelée chaque fois que le bouton est cliqué

**Pour la deuxième interface :**

**Les principaux bibliothèques utilisées :**



****

* QPixmap est une classe utilisée pour afficher des images dans des widgets. Il est optimisé pour le dessin et peut être utilisé pour afficher des images à l'écran.
* QIcon est une classe utilisée pour représenter une icône dans une interface graphique. Il peut être utilisé pour créer une petite représentation graphique d'un objet ou d'un concept, tel qu'un fichier, un dossier ou une application.

**Des parties du code :**

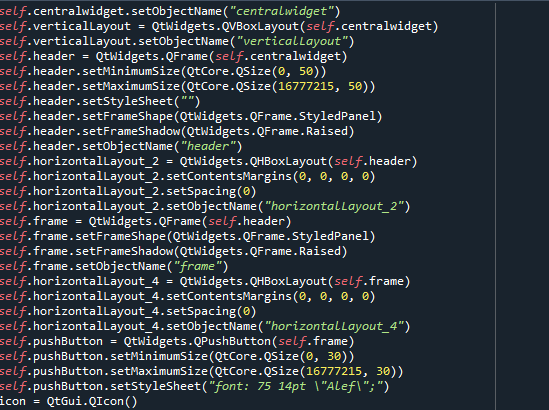




* Le premier cadre, "frame2", reçoit une forme de cadre "WinPanel" et une ombre unie. On lui donne également une largeur de ligne et une largeur de ligne médiane de 10 pixels.



* Cette partie consiste à définir le texte affiché sur chaque bouton. Le texte de chaque bouton est spécifié en tant qu'argument de chaîne à la méthode setText et est passé par la QCoreApplication.translateméthode avant d'être défini.



**Explication des codes sous-python :**





**Partie de l’extraction des données avec l’API  et la production d’une matrice de cout:**

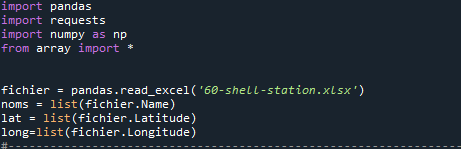
**Les bibliothèques utilisées :**

**Requests** : est une bibliothèque pour faire des requêtes HTTP en Python. Il vous permet d'envoyer des requêtes HTTP à un serveur Web et de recevoir la réponse du serveur. Ceci est utile pour interagir avec les API Web, télécharger des données depuis Internet ou tester les performances d'un serveur Web.

**Panda** : est une bibliothèque d'analyse et de manipulation de données pour Python. Elle fournit des structures de données rapides, flexibles et expressives conçues pour rendre le travail avec des données « relationnelles » ou « étiquetées » à la fois facile et intuitif. C'est un excellent outil pour gérer et analyser de grands ensembles de données.

**Le code sous python :**

**Première partie :**

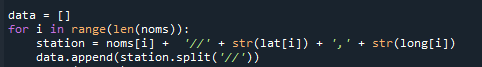
****

La fonction **pandas.read\_excel** lit le fichier Excel '60-shell-station.xlsx'et extrait les valeurs des colonnes (Name, Latitude et Longitude). Ces valeurs sont stockées dans les variables noms, lat et long, respectivement. Donc,après cette partie on va avoir 3 listes :Une contient



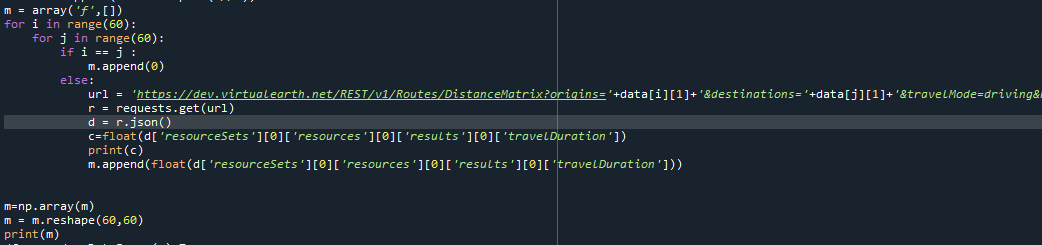
les noms des différents stations,l’autre possède la latitude de chaque station,et la dernière contient la longitude de chaque station **shell**

**Deuxième partie :**

****

Cette partie permet de créer une liste vide appelée ‘data’.Ainsi ,de itérer sur la liste des noms et combiner les valeurs correspondantes dans lat et long dans une chaîne au format 'name//latitude,longitude'. Chaque chaîne est ensuite divisée selon le délimiteur '//'

**Troisième partie :**

****

Cette partie du code consiste à :

* Créer un tableau m initialement vide
* Répeter la boucle ‘For’ 60 fois pour les deux indices i et j
* Si i est égal à j, on ajoute la valeur 0 au tableau ‘m’. Cela représente la distance entre un emplacement et lui-même, qui est toujours 0.
* Si i n'est pas égal à j, le code envoie une requête à «  **l'API Bing Maps Distance Matrix** » à l'aide de la bibliothèque requests. La requête API utilise les i et j éléments de la liste data générée de la partie précédente comme origine et destination, respectivement. La réponse de l'API est stockée dans une variable d.
* Le code extrait la valeur travelDuration de la réponse de l'API et la convertit en flottant. Cette valeur est ensuite ajoutée au m tableau.
* Le tableau m est converti en un tableau NumPy et remodelé en une matrice 60x60.



**Quatrième partie :**

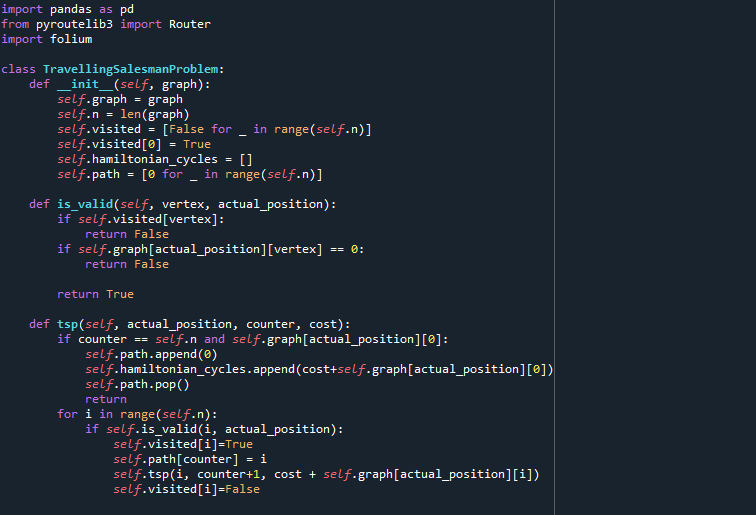
****

* Le DataFrame aura 60 lignes et 60 colonnes, chaque élément du DataFrame représentant la durée du trajet entre deux stations. Les lignes et les colonnes seront étiquetées avec les indices de la liste data, qui contiennent les noms et les coordonnées des stations shell.
* To\_excel() une fonction Pandas qui vous permet d'écrire un DataFrame dans un fichier Excel. Le excel\_writer paramètre spécifie le nom du fichier Excel dans lequel le DataFrame sera écrit. Si le fichier n'existe pas, il sera créé. Si le fichier existe déjà, le DataFrame sera ajouté au fichier existant.
* Le DataFrame transposé est écrit dans un fichier Excel appelé 'matrice\_duration.xlsx'.
* D’où la production d’une matrice de temps

**Partie de plus court chemin :**







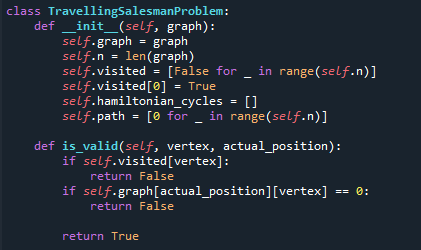
**Les principaux bibliothèques utilisées :**

* **Panda** ,c’est une bibliothèque Python consiste à travailler avec les données. Elle fournit des outils pour lire et écrire des données de différents formats, ainsi que des outils pour manipuler et analyser les données.Comme dans notre cas , on va utiliser panda pour lire les données qui existent dans la matrice de distance qui est en format **Excel**

Les parties de code :

**1ère partie :**

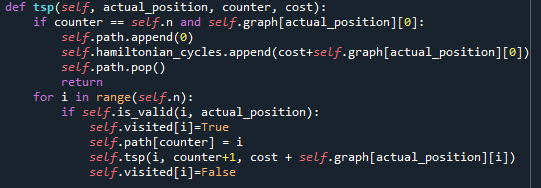




Cette partie du code permet de créer une classe nommée ‘TravellingSalesmanProblem’ .dans laquelle on définit :

* self.n qui représente est le nombre de noeuds dans le graphe, qui est déterminé en prenant la longueur du graphe.
* Self.visited est une liste de valeurs booléennes, avec un élément pour chaque nœud du graphe. La liste est initialisée à toutes les valeurs ‘**False**’ , à l'exception du premier élément, qui est défini comme ‘ **True’**.
* self.hamiltonian\_cycles :est une liste vide qui sera utilisée pour stocker les cycles hamiltoniens trouvés dans le graphe ,càd les cycles qui passent par tous les points une et une seule fois.
* self.path est une liste d'entiers, avec un élément pour chaque nœud du graphe. La liste est initialisée à 0 toutes les valeurs.

**2ème partie :**





Cette partie permet de résoudre le problème de ‘ Travelling Salesman Problem ‘ en utilisant la fonction tsp qui prend comme arguments :

* actual\_position: un entier représentant la position actuelle du chauffeur dans le graphe.
* counter: un entier représentant le nombre de stations visitées jusqu'à présent .
* cost: un entier représentant le coût total du chemin parcouru par le chauffeur jusqu'à présent.
* self: l'objet sur lequel la fonction est appelée

La fonction vérifie d'abord si le chauffeur a visité toutes les stations ( counter == self.n) et s'il est possible de revenir à la station de départ ( self.graph[actual\_position][0]). Si ces deux conditions sont vraies, cela signifie que le chauffeur a terminé un cycle hamiltonien (un cycle qui visite chaque station exactement une fois) et la fonction ajoute la station de départ (0) à la liste des stations visitées ( self.path), met à jour la liste des Cycles hamiltoniens avec le coût total du cycle ( self.hamiltonian\_cycles.append(cost+self.graph[actual\_position][0])), et supprime la station de départ de la liste des stations visitées ( self.path.pop()).

Si les conditions de réalisation d'un cycle hamiltonien ne sont pas remplies, la fonction itère sur toutes les noeuds ( for i in range(self.n)) et vérifie s'il est possible pour le chaufeur de visiter la station ( self.is\_valid(i, actual\_position)). Si la station est valide, la fonction la marque comme visitée ( self.visited[i]=True), l'ajoute à la liste des points visités ( self.path[counter] = i) et s'appelle de manière récursive avec les arguments mis à jour : i comme la nouvelle position réelle, counter+1 comme le compteur mis à jour et cost + self.graph[actual\_position][i]comme le Coût. Enfin, la fonction marque la ville comme non visitée ( self.visited[i]=False) avant de passer à l'itération suivante ou de revenir, selon les conditions

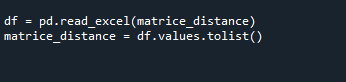


**Remarque :**

**Cette fonction est une implémentation récursive du problème TSP, ce qui signifie qu'elle résout le problème en le décomposant en sous-problèmes plus petits et en les résolvant de manière récursive. La fonction garde une trace des points visitées jusqu'à présent et du coût total du chemin emprunté, et utilise ces informations pour déterminer si un cycle hamiltonien a été complété. Il utilise une approche de retour en arrière pour explorer tous les chemins possibles et trouver la solution optimale, qui est le cycle hamiltonien avec le coût minimum.**

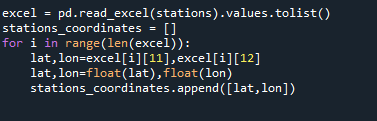
**3ème partie :**





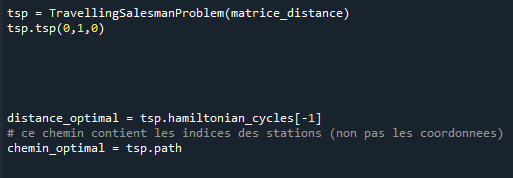
Cette partie permet de convertir un fichier excel en une matrice en utilisant la bibliothèque panda

**4ème partie :**

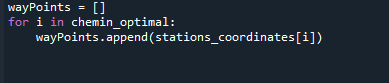
****

Cette partie permet d’importer tous les coordonnées des stations en utilisant le fichier excel

**5ème partie :**

****

Cettepartie consiste à lancer le tsp et afficher le résultat de l’algorithme

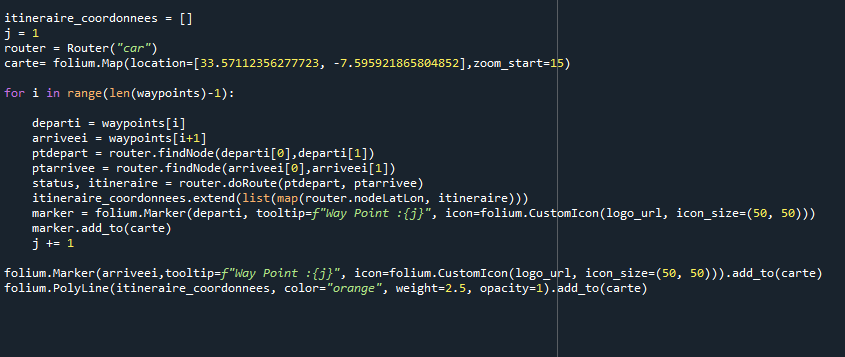


Et celle-ci permet de remplacer les indices par les coordonnées

**Partie de création du MAP** :







Parmi les bibliothèques utilisées dans cette partie ,on a :

* Folium : est une bibliothèque Python qui permet de créer des cartes interactives à l'aide de la bibliothèque Leaflet.js. Avec Folium, vous pouvez créer des cartes de différentes régions du monde et ajouter des marqueurs, des polygones et d'autres données géospatiales à la carte.
* Router : est une bibliothèque qui permet de créer des routes dans une application web en utilisant Python. Elle fonctionne en associant une URL à une fonction de traitement de la requête, ce qui permet de gérer les différentes actions que l'on souhaite effectuer en fonction de l'URL demandée.

**Explication du code :**

La première ligne déclare une liste vide qui sera utilisée pour stocker les coordonnées de l'itinéraire trouvé par la bibliothèque Router.

La variable "j" est initialisée à 1 et sera utilisée pour ajouter un marqueur sur la carte pour chaque point de passage.

La variable "router" est initialisée avec une chaîne de caractères "car", ce qui indique que l'itinéraire sera calculé en utilisant le mode de transport "voiture".

La variable "carte" est initialisée avec un objet de type Map de la bibliothèque Folium, centré sur les coordonnées [33.57112356277723, -7.595921865804852] et affichant un niveau de zoom de 15



Le code suivant utilise une boucle "for" pour parcourir tous les points de passage de la liste "waypoints" (sauf le dernier). Pour chaque point de passage, le code effectue les étapes suivantes :

Il récupère les coordonnées du point de départ et du point d'arrivée dans les variables "departi" et "arriveei" respectivement.

Il utilise la méthode "findNode" de la bibliothèque Router pour trouver les points correspondants aux coordonnées du point de départ et du point d'arrivée. Ces points sont stockés dans les variables "ptdepart" et "ptarrivee" respectivement.

Il utilise la méthode "doRoute" de la bibliothèque Router pour calculer l'itinéraire entre les points "ptdepart" et "ptarrivee". Le résultat est stocké dans les variables "status" et "itineraire".

Il utilise la méthode "nodeLatLon" de la bibliothèque Router pour convertir chaque point de l'itinéraire en coordonnées latitude et longitude. Les coordonnées sont ajoutées à la liste "itineraire\_coordonnees".

Il crée un marqueur avec la bibliothèque Folium à l'emplacement du point de départ et l'ajoute à la carte. La variable "j" est incrémentée de 1 pour passer au point de passage suivant.

Enfin, le code ajoute un marqueur à la carte pour le dernier point de passage de la liste "waypoints" et utilise la bibliothèque Folium pour tracer une ligne sur la carte entre chaque point de l'itinéraire grâce à la liste "itineraire\_coordonnees".

**Conclusion :**

**Dans ce projet ,on a amélioré nos compétences dans la recherche,et dans l’extraction de toutes les données qu’on veut de differents sites web possibles(API) et les exploiter par la suite . Ainsi,on a découvert des nouveaux outils et moyens dans Qtdesigner ,et tout cela c’est grace a python qui a rendu plusieurs projets possibles .En outre,** **sans Python, ce projet n'aurait pas été possible .**

**Nous sommes vraiment reconnaissants envers Python pour sa simplicité et sa facilité d'utilisation. Cela nous a permis de nous concentrer sur la résolution du problème au lieu de nous perdre dans les détails techniques du langage de programmation.**