

projet station essence

1. les types de données

1.1 la gestion des type de clients

- le type de *clients* sont dans un fichier csv qui contient les informations sur les types de clients leurs **temps** de **remplissage** et leur **particularité**

	A	B	C
1	clients	temps	special
2	lamba	150	1
3	lent	300	1
4	stupide	0	2
5	malin	200	3
6	pas de regle	400	4
7	file mechant	150	5
8	angry	175	6
9	cops	150	7

1.1.1 retour sur le csv clients

- nous avons preferer utiliser les "case" pour une meilleur lisibilité du code et pour simplifier le fichier csv donc la derniere colonne est inutile avec les "case"

1.2 la gestion des voitures

- les *voiture* sont dans un fichier csv qui contient les **informations** sur les voitures leurs **carburant** et leur **capacité** de remplissage

	A	B	C
1	marque	type	reservoir
2	fiat pundo	diesel	45
3	renault clio	sans_plomb95	55
4	cybertruck	sans_plomb95	100
5	SUV	sans_plomb95	80
6	camion	gasoil	200
7	moto	sans_plomb95	10
8	scooter custom	gasoil	7
9	bus a essence	sans_plomb98	300
10	Tracteur	gasoil	150
11	lambo	sans_plomb98	75

1.3 dans python

- nous convertissons les fichiers csv en liste avec la methode **Reader** de la librairie **CSV**

2. structure du code python

2.1 la class pompe yanis

- la pompe est une **class** qui utilisera la **class File** (vue predecament en cours) pour combiner les informations des fichiers csv et efilier les informmations dans la File d'attente de la station essence

2.1.1 l'attribut random_pompe

- cette attribut permet de crée une combinaison aléatoire d'un type de **client** et d'une **voiture**

2.1.2 l'attribut remplir_pompe_debut

- cette attribut permet de remplir chaque pompe avec un **client** et une **voiture** de plus cette attribut est utiliser dans le init de la class pompe

2.1.3 l'attribut remplir_pompe

- cette attribut utilise l'attribut random_pompe pour remplir une pompe aleatoire avec un **client** et une **voiture** de plus cette attribut est utiliser dans le init de la class pompe

2.1.4 l'attribut vide_pompe

- cette attribut permet de vider la pompe indique en parametre d'un cran et verifie si la pompe est vide ou non avant de la vider pour eviter les erreurs

2.1.5 l'attribut pompe_vide

- cette attribut permet de verifier si les pompe sont vide ou non

2.2 la class essence dylan

- la class essence est une **class** qui permet de gerer les **prix** des **carburants** et elle initialiser les **type de carburent** et les **prix** des **carburants**

2.2.1 l'attribut augmenter_prix

- cette attribut permet d'augmenter le prix du carburant de maniere aleatoire

2.2.2 l'attribut diminuer_prix

- cette attribut permet de diminuer le prix du carburant de maniere aleatoire

2.2.3 l'attribut round

- cette attribut permet d'arrondir le prix du carburant a 2 chiffres apres la virgule

2.3 la class client dylan

- la class Client est une **class** qui permet de gerer les **clients** et leurs **particularités** et leurs **temps de remplissage**

2.3.1 l'attribut special

- cette attribut va renvoyé un entier qui correspond a la vitesse et la particularité du client et la presence ou non d'un vigile

2.3.2 l'attribut `affichage_client`

- cette attribut va renvoyé un string qui correspond au type de client et a la vitesse du client

2.4 la class voiture yanis

- WIP(Work In Progress), la class voiture sera une **class** qui permettra de gerer les **voitures** , leurs **carburant** et leurs **capacité** de remplissage

2.5 la class station dylan


- la class Station est une class qui regroupe toutes les autres class et permet de gerer la station essence et ses **Clients** , **Voitures** , **Pompe** , **essence** , elle gere aussi les evenements de la station essence

3 Structure du code

3.1 le fichier class.py

- il contient les class

3.2 le fichier main.py

- il contient la boucle principale du code  wakatime 13 hrs 28 mins