Chapitre 1: Structures de données avec Python

Programme de la séance 1:

- Introduction à l'EDI Jupyter-lab
- Python un langage à typage dynamique
- Les variables
- Les listes
- Les dictionnaires
- Les structures conditionnelles
- Les boucles for
- Les boucles while

Projet

Une entreprise de transport routier dispose d’une flotte de n camions assurant le transport des marchandises entre plusieurs villes.

Calculer avec Python

- Python permet une utilisation en mode interactif.
- On peut dialoguer avec lui directement depuis le clavier.

Code

```
[1]: # Une addition
     30+20
```

```
[1]: 50
```

```
[2]: # Une soustraction
     13 - 14 # Les espaces sont optionnels
```

```
[2]: -1
```

```
[3]: # Une multiplication
     (7+2)*3 #Les parenthèses sont fonctionnelles
```

```
[3]: 27
```

```
[4]: # Une division
     15/2
```

```
[4]: 7.5
```

Code

```
[5]: # Le quotient de la division euclidienne
     14.0 // 3.0
```

```
[5]: 4.0
```

```
[6]: # Le reste de la division euclidienne
     14.0 % 3.0
```

```
[6]: 2.0
```

```
[7]: # L'exponentiation
     3 ** 2
```

```
[7]: 9
```

```
[8]: # Autrement
     pow(3,2)
```

```
[8]: 9
```

Les variables

- Un nom de variable doit commencer par une lettre et peut comporter une séquence de lettres (a→z, A→Z) et de chiffres (0→9).

- Les caractères majuscules et minuscules sont distingués (a et A sont différents): **On dit que Python est sensible à la casse.**

- Attention: les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que $, #, @,… sont interdits.

- Les mots réservés en Python sont aussi interdits:
And, del, for, is, raise, assert, elif, from, lambda, return, break, else, global, not, try, class, except, if, or, while, continue, exec, import, pass, yield, def, finally, in, print

```
[ ]: print
     if
     elif
```

- Ces mots clés se colorient en une couleur différente de celle des variables.

- On dit que Python est **un langage à typage dynamique** puisque on ne spécifie pas le type de la variable pendant la déclaration:

Par exemple en C, C++ et Java, on utilise cette ligne pour déclarer et initialiser la variable n

```
int n = 3;
```

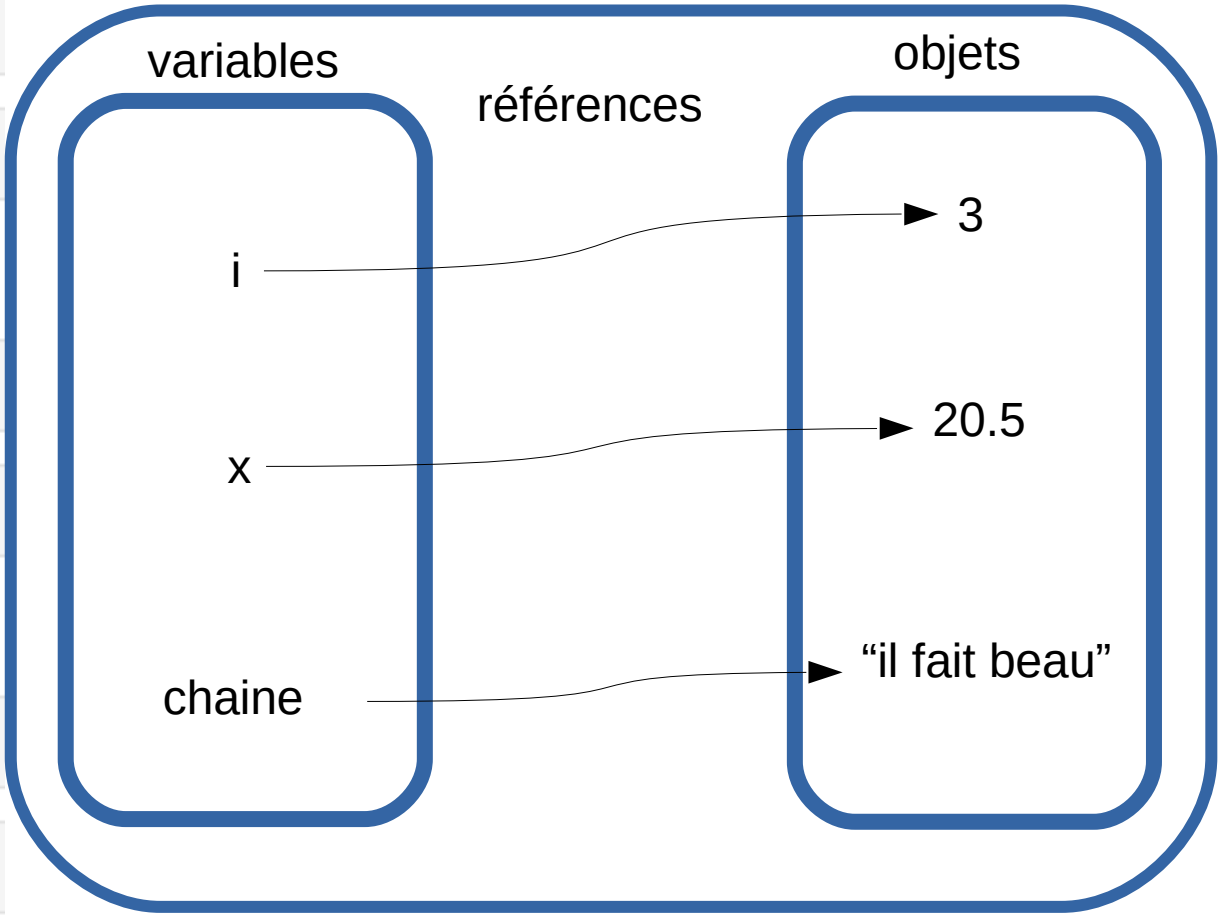
Question : Comment déclarer une variable en Python?

Les variables : définition et affectation

- L'affectation permet d'établir un lien entre le nom de la variable et une valeur donnée (son contenu).
- Autrement, on associe une référence entre une variable et un objet.

```
[1]: i = 1

[2]: type(i)

[2]: int

[3]: x = 20.5

[4]: type(x)

[4]: float

[5]: chaine = "il fait beau"

[6]: type(chaine)

[6]: str
```

Situation dans la mémoire de l'ordinateur



Les variables : définition et affectation

```
# Affectation du résultat d'une opération mathématique
somme = 23 + 34
```

```
# Affectation utilisant la variable elle-même
x = x*x
```

```
# Affectations avancées
x += 3 # remplace x = x + 3
x -= 11 # remplace x = x - 11
x *= 10 # remplace x = x * 10
x **= 5 # remplace x = x ** 5 (puissance)
x /= 2 # remplace x = x / 2 (division réelle)
x //= 2 # remplace x = x // 2 (division entière)
x %= 4 # remplace x = x % 4 (reste modulo 4)
```

Les variables : affectations multiples

- On peut affecter une valeur à plusieurs variables simultanément.
- On peut aussi réaliser des affectations parallèles en une seule ligne.

```
[1]: a = b = 13
```

```
[2]: a
```

```
[2]: 13
```

```
[3]: b
```

```
[3]: 13
```

```
[4]: a = b = c = 13
```

```
[5]: c
```

```
[5]: 13
```

```
var1,var2 = 4,100
```

```
var1
```

```
4
```

```
var2
```

```
100
```

```
x , y , z = 1.00, 2., 3.1
```

```
x
```

```
1.0
```

Comment afficher la valeur d'une variable?

- On écrit le nom de la variable et on appuie sur shift + entrée sur jupyter-notebook.
- Dans un autre environnement, on écrit le nom de la variable et on tape sur entrée.
- On peut aussi utiliser la fonction print()

Sans print

```
i
```

```
1
```

```
x
```

```
20.5
```

```
chaine
```

```
'il fait beau'
```

Avec print

```
print(i)
```

```
1
```

```
print(x)
```

```
20.5
```

```
print(chaine)
```

```
il fait beau
```

Obtenir de l'aide pour en ligne la fonction print

```
print? # help(print) fonctionne aussi
```

```
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

Comment saisir une valeur à partir du clavier?

- La saisie d'une information venue de l'utilisateur, s'effectue avec la fonction input()

```
prenom_chauffeur = input()
```

```
prenom_chauffeur
```

```
'karim'
```

```
print(prenom_chauffeur)
```

```
karim
```

```
help(input)
```

```
Help on method raw_input in module ipykernel.kernelbase:

raw_input(prompt='') method of ipykernel.ipkernel.IPythonKernel instance
    Forward raw_input to frontends

    Raises
    ------
    StdinNotImplentedError if active frontend doesn't support stdin.
```

Pour cela:

1 – On écrit le nom de la variable.

2 – On effectue une affectation de la fonction input() puis on tape sur shift + entrée. Une fenêtre s'ouvre.

4 – A partir du clavier, on saisit une chaîne de caractères.

5 – On affiche le résultat: directement ou à à l'aide de la fonction print().

Comment saisir une valeur à partir du clavier?

- La fonction input() retourne toujours une chaîne de caractères alors comment saisir des valeurs numériques?
Solution: Tout d'abord, on affecte la sortie de la fonction input() à une variable
- Ensuite, on change le type de cette variable ==> c'est un "type casting" ou forçage de type (une technique bien connue en programmation informatique).

```
age = input("Quel est l'age du chauffeur?")
```

```
Quel est l'age du chauffeur? 40
```

```
age
```

```
'40'
```

```
type(age) # La variable age est une chaine de caractères!
```

```
str
```

```
age = int(age) # On force le type de la variable age
```

```
# Maintenant, age est une variable de type int
# (entier ou integer)
type(age)
```

```
int
```

Comment saisir une valeur à partir du clavier?

On peut forcer le type directement sur une seule ligne en utilisant int(input()) ou float(input())

```
distance = float(input("Donner la distance entre les camions en km"))
```

```
Donner la distance entre les camions en km 14.1289
```

```
distance
```

```
14.1289
```

```
type(distance)
```

```
float
```

```
numero = int(input("Quel est le numéro du camion?"))
```

```
Quel est le numéro du camion? 111
```

```
numero
```

```
111
```

```
type(numero)
```

```
int
```

Type et Identifiant

- Chaque variable possède un type et un identifiant.
- Le type d'une variable: la nature de l'objet stocké dans la variable
- L'identifiant d'une variable: l'adresse mémoire associée à cette variable

```
type(x) # affiche le type de la variable
```

```
float
```

```
print(type(x)) # affiche la classe de l'objet
```

```
<class 'float'>
```

```
id(x) # affiche l'identifiant (l'adresse mémoire)
```

```
140347929306528
```

```
int # Entiers, de longueur non bornée
float # Flottants (réels, en norme IEEE 754 sur 64 bits)
str # Chaînes de caractères (string)
list # Listes
dict # Dictionnaires
complex # Nombres complexes
bool # Booléens (True / False)
set # Ensembles
tuple # n-uplets
```

Saisie et affichage de plusieurs valeurs? Exercice

Exercice: 1 - Exécuter les commandes suivantes
2 - Donner à chaque fois le type de la variable saisie
3 - Afficher ces variables
4 – Exécuter les fonctions who et whos. Que remarquez-vous?

```
prenom = str(input(" Quel est le prenom du chauffeur ? "))
```

```
age = int(input(" Quel est l'age du chauffeur ? "))
```

```
numero = float(input("Quel est le numéro du camion conduit ? "))
```

```
print("Bonjour",prenom,"!")
```

```
print("Vous avez ",age," ans et vous conduisez le camion numéro ",numero,". Bon travail!")
```

Les chaînes de caractères

- Une chaîne de caractères est une structure de données immuable (non mutable).
- Il s'agit d'un assemblage non interprété de caractères.
- Il existe plusieurs manières de définir une chaîne de caractères:

```
chaine1 = 'Tunisie' # avec simple apostrophe
```

```
print(chaine1) # affichage du contenu de chaine1
```

```
Tunisie
```

```
chaine2 = "Tunisie" # avec double guillemets
chaine2 # affichage du contenu de chaine1
```

```
'Tunisie'
```

```
# Importance de double guillemets
chaine3 = 'l'apostrophe'
```

```
  File "<ipython-input-60-0a8972788a20>", line 2
    chaine3 = 'l'apostrophe'
                          ^
SyntaxError: invalid syntax
```

Erreur

```
# Pour contourner cette erreur
# On utilise les doubles guillemets
chaine3 = "l'apostrophe"
```

```
chaine3
```

```
"l'apostrophe"
```

```
chaine4 = """
Ce commentaire est
intelligent,
puisqu'il est utilisé
pour documenter les
fonctions
"""
```

```
print(chaine4)
```

```
Ce commentaire est
intelligent,
puisqu'il est utilisé
pour documenter les
fonctions
```

Les chaînes de caractères: quelques méthodes

- Quelques méthodes sur les chaînes de caractères:

```
chaine = "il fait beau en Tunisie"
```

```
chaine
```

```
'il fait beau en Tunisie'
```

```
chaine.upper() #
```

```
'IL FAIT BEAU EN TUNISIE'
```

```
chaine.replace("en Tunisie","à Siliana")
```

```
'il fait beau à Siliana'
```

```
chaine
```

```
'il fait beau en Tunisie'
```

```
L = chaine.split()
L
```

```
['il', 'fait', 'beau', 'en', 'Tunisie']
```

```
L[2] = "mauvais"
```

```
" ".join(L)
```

```
'il fait mauvais en Tunisie'
```

Les chaînes de caractères: accès, indexation et slicing

- Comment accéder à un élément dans une chaîne de caractères?

```
chaine1 = "monde" # affectation d'une chaîne
```

```
# Accès aux éléments de chaine1 par indexation
chaine1[0] # élément numéro 1
```

```
'm'
```

```
chaine1[1] # accès à l'élément numéro 2
```

```
'o'
```

```
chaine1[2] # accès à l'élément numéro 3
```

```
'n'
```

```
# Détermination de la longueur de chaine1
len(chaine1)
```

```
5
```

```
# Accès au dernier élément
chaine1[4]
```

```
'e'
```

```
# Attention à cette faute
chaine1[5]
```

```
---------------------------------------------------------
IndexError                                Traceb
<ipython-input-9-9825746489a1> in <module>()
      1 # Attention à cette faute
----> 2 chaine1[5]

IndexError: string index out of range
```

- Il est vrai que la longueur de cette chaîne
- est de 5 mais le dernier élément est indexé
- par 4 et non pas par 5 car en Python on
- commence à compter à partir de 0!

Les chaînes de caractères: accès, indexation et slicing

- Indexation négative

```
"""
Une deuxième méthode
pour accéder au dernier
élément d'une chaîne
de caractères avec
une indexation négative
"""
chaine1[-1]
```

```
'e'
```

```
# Accès à l'avant dernier
# élément
chaine1[-2]
```

```
'd'
```

- Slicing d'une chaîne de caractères

```
chaine1[0:5:1] #accés à toute la chaîne
```

```
'monde'
```

```
chaine1[:5:1] #Autrement
```

```
'monde'
```

```
chaine1[0:5:2] #avec un pas de 2
```

```
'mne'
```

```
# Si on ne spécifie pas
# le pas alors il est 1
# par défaut
chaine1[1:3]
```

```
'on'
```

Les chaînes de caractères: formatage

Question: Comment formater une chaîne de caractère?

Méthode 1:
On utilise la méthode format() qui peut de remplacer son contenu entre deux accolades présentes déjà dans la chaîne.

```
prenom_client = "karim"
```

```
age = 25
```

```
"{} a {} ans".format(prenom_client,age)
```

```
'karim a 25 ans'
```

Les chaînes de caractères: formatage

Question: Comment formater une chaîne de caractère?

- Il existe une deuxième méthode plus expressive

```
f"{prenom_client} a {age} ans"
```

```
'karim a 25 ans'
```

- Une solution plus avancée

```
age = 30
prenom_client = "karim"

print("%s" %prenom_client, "a" ,"%d" %age)
```

```
karim a 30
```

Quelques méthodes opérant sur les chaînes de caractères

```
len(chaine1) # longueur (nombre de caractères)
print(chaine1) # affichage à l'écran de chaine1
"je" + "suis" # concaténation des chaînes
2 * "je" # concaténation répétée 2 fois
n * "a" # concaténation répétée n fois
"a" in "karim" # teste si a est sous-chaîne de karim
"b" not in "karim" # teste si b n'est pas une sous-chaîne de karim
```

Pour plus d'information sur les chaînes de caractères

https://docs.python.org/3.3/library/string.html

Les listes

- La liste linéaire est la forme la plus simple d'organisation de données que l'on puisse rencontrer. Celles-ci sont stockées les unes à la suite des autres dans des places et permettent divers traitements séquentiels. L'ordre des éléments dans une liste ne dépend pas des éléments eux-mêmes, mais de la place de ceux-ci dans la liste.

- Une liste est une suite ordonnée d'objets (int, str, bool, float, ou même des listes), pouvant être de types différents.

- Une liste est une structure de données dynamique et mutable.

- Il s'agit de la structure de données la plus maniable en Python.

```
semaine = ['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi']
```

```
print(semaine)
```

```
['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi']
```

```
len(semaine)
```

```
5
```

Les listes: déclaration et notations

```
L = [] # L est une liste vide
```

```
L # Affichage de L
```

```
[]
```

```
len(L) # Aucun élément
```

```
0
```

```
len(L) # Aucun élément
```

```
0
```

```
L = [1,2,3,4] # L est remplie
```

```
print(L) # affichage de L
```

```
[1, 2, 3, 4]
```

```
len(L)
```

```
4
```

Les listes: déclaration et notations

```
L1 = [1,True,12.3,"esprit",'A',[1,2,3,4]]
```

```
print(L1) # L est hétérogène
```

```
[1, True, 12.3, 'esprit', 'A', [1, 2, 3, 4]]
```

```
L2 = list("esprit") # En utilisant list()
```

```
L2 # contient toutes les lettres
```

```
['e', 's', 'p', 'r', 'i', 't']
```

```
len(L2)
```

```
6
```

Les listes: accès, indexation et slicing

Question: Comment on peut accéder à un élément d'une liste?

Méthode 1: On peut utiliser un indice positif qui commence à partir de 0.

```
id_camion = ["12","45","14","3"] # Création d'une liste
```

```
id_camion[0] # accès au premier élément
```

```
'12'
```

```
id_camion[1] # accès au 2ème élément
```

```
'45'
```

```
len(id_camion) # longueur de cette liste
```

```
4
```

```
id_camion[4] # Attention à cette erreur!
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-6-338517b9cfa2> in <module>()
----> 1 id_camion[4]

IndexError: list index out of range
```

Les listes: accès, indexation et slicing

Question: Comment on peut accéder à un élément d'une liste?

Méthode 2: On peut utiliser un indice négatif qui commence à partir de -1.

```
id_camion[-1] # accès au dernier élément
```

```
'3'
```

```
# accès à l'avant dernier élément
id_camion[-2]
```

```
'14'
```

```
id_camion[-5] # Attention erreur
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-13-564a7af9d71a> in <module>()
----> 1 id_camion[-5] # Attention erreur

IndexError: list index out of range
```

Les listes: accès, indexation et slicing

Question: Comment on peut accéder à un ensemble d'éléments d'une liste?

Méthode: On utilise une technique dite de slicing permettant d'éviter les boucles pour!

```
# Une autre méthode pour accéder au dernier élément
id_camion[len(id_camion)-1]
```

```
'3'
```

```
prenoms = ["Karim","Selim","David","Paul","Pierre"]
```

```
prenoms[0:5:1] # afficher toute la liste
```

```
['Karim', 'Selim', 'David', 'Paul', 'Pierre']
```

```
- Essayer aussi: prenoms[0:5], prenoms[:5], prenoms[:]
- prenoms[:5:1] et prenoms[::1]
```

```
 prenoms[0:5:2] # on change de pas et on choisit 2
```

```
['Karim', 'David', 'Pierre']
```

```
prenoms[1:3] # on change l'intervalle
```

```
['Selim', 'David']
```

Les listes: opérations et méthodes

```
num_camions_jour1 = [23,114,12,13,1]
num_camions_jour2 = [10,3]
```

```
#concaténation des listes
num_camions_jour1 + numero_camions_jour2
```

```
[23, 114, 12, 13, 1, 10, 3]
```

```
L = [None]
L
```

```
[None]
```

```
# Pour n entier: concaténation
# répétée de L avec elle-même
n=6
L * n
```

```
[None, None, None, None, None, None]
```

```
# ajout de 6 à la fin de la liste
num_camions_jour1.append(6)
num_camions_jour1
```

```
[23, 114, 12, 13, 1, 6]
```

```
# Suppression du 1er élément
del num_camions_jour1[0]
```

```
num_camions_jour1
```

```
[114, 12, 13, 1, 6]
```

```
# Insertion de 1000 en position 3
num_camions_jour1.insert(3,1000)
num_camions_jour1
```

```
[12, 13, 1, 1000, 6]
```

```
# Supprimer le dernier élément
num_camions_jour1.pop()
# ou
# Suppression d'un élément par indice
numero_camions_jour1.pop(i)
```

```
# Suppression d'un élément par valeur
num_camions_jour2.remove(10)
num_camions_jour2
```

```
[3]
```

Les dictionnaires : quelques définitions

Une première définition facile: Un dictionnaire permet de définir une association entre un ensemble de clés et des valeurs

La vraie définition d'un dictionnaire: Un dictionnaire est une table de hash alors on dispose donc d'un temps d'accès, d'insertion, d'effacement et d'un test d'appartenance qui est indépendant du nombre d'éléments.

Un dictionnaire est objet mutable, on peut donc le modifier en place avec une excellente efficacité mémoire.

Dans un dictionnaire on peut avoir comme clé n'importe quel objet hashable (un objet où on peut calculer une fonction de hash).

En python, tous les objets immuables sont hashables et tous les objets mutables ne sont pas hashables.

Création et remplissage d'un dictionnaire

Attention l'accès s'effectue par clé et non pas par indice comme dans le cas d'une liste.

```
camion = {} # création d'un dictionnaire vide
```

```
type(camion)
```

```
dict
```

```
print(type(camion))
```

```
<class 'dict'>
```

```
# Remplissage du dictionnaire
# avec des clés
camion["chauffeur"] = "karim"
camion["age"] = 40
camion["identifiant"] = 124
```

```
camion # affichage du contenu
```

```
{'age': 40, 'chaufeur': 'karim', 'chauffeur': 'karim', 'identifiant': 124}
```

Création d'un dictionnaire à partir d'une liste des listes

```
# création d'une liste des listes
chauffeurs = [["Paul",30], ["Pierre",45], ["François",25]]
```

```
# création d'un dictionnaire à
# partir de la liste chauffeurs
dictionnaire = dict(chauffeurs)
```

```
print(dictionnaire)
```

```
{'Paul': 30, 'Pierre': 45, 'François': 25}
```

```
"karim" in dictionnaire # test d'appartenance
```

```
False
```

```
"Paul" in dictionnaire # test d'appartenance
```

```
True
```

Quelques manipulations sur un dictionnaire

```
camion['age'] # accès à la valeur ayant age comme clé
```

```
40
```

```
camion["chauffeur"]
```

```
'karim'
```

```
len(camion) # nombre d'éléments dans un dictionnaire
```

```
3
```

```
"""
Attention: un dictionnaire
ne supporte pas un accès par indice.
"""
camion[0] # On obtient un message d'erreur
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-80-83fce78f9b0e> in <module>()
      3 ne supporte pas un accès par indice.
      4 """
----> 5 camion[0] # On obtient un message d'erreur

KeyError: 0
```

Comment supprimer une valeur dans un dictionnaire?

```
# création et remplissage d'un dictionnaire
clients = {"c1": "Pierre", "c2": "David", "c3": "Mohamed"}
```

```
print(clients) # affichage du dictionnaire
```

```
{'c1': 'Pierre', 'c2': 'David', 'c3': 'Mohamed'}
```

```
# suppression par clé avec del
del clients["c2"]
```

```
clients
```

```
{'c1': 'Pierre', 'c3': 'Mohamed'}
```

```
clients.clear() # supprimer tout le contenu
clients # vérification par affichage
```

```
{}
```

Comparaison entre deux structures de données: les listes et les dictionnaires

Prenons l'exemple suivant afin de comparer entre les deux structures de données:

```
flotte_liste = [["Tunis",3,[["Karim", "camion1"],
                            ["Mohamed","camion2"],
                            ["Faouzi","camion3"]]],
                ["Kasserine",2,[["Naoufel", "camion1"],
                                ["Aziz","camion2"]]],
                ["Siliana",1,[["Mouguith","camion1"]]]
               ]
```

```
flotte_dict = {"Tunis": [3,[["Karim", "camion1"],
                            ["Mohamed","camion2"],
                            ["Faouzi","camion3"]]], "Kasserine": [2,[["Naoufel", "camion1"],
                                ["Aziz","camion2"]]], "Siliana": [1,[["Mouguith","camion1"]]]}
```

```
# On veut accéder aux données de Tunis
flotte_liste[0] # peu pratique puisqu'il faut connaître l'indice à l'avance
```

```
['Tunis',
 3,
 [['Karim', 'camion1'], ['Mohamed', 'camion2'], ['Faouzi', 'camion3']]]
```

```
# On veut accéder aux données de Tunis
flotte_dict["Tunis"] # Il suffit d'utiliser la clé correcte "Tunis"
```

```
[3, [['Karim', 'camion1'], ['Mohamed', 'camion2'], ['Faouzi', 'camion3']]]
```