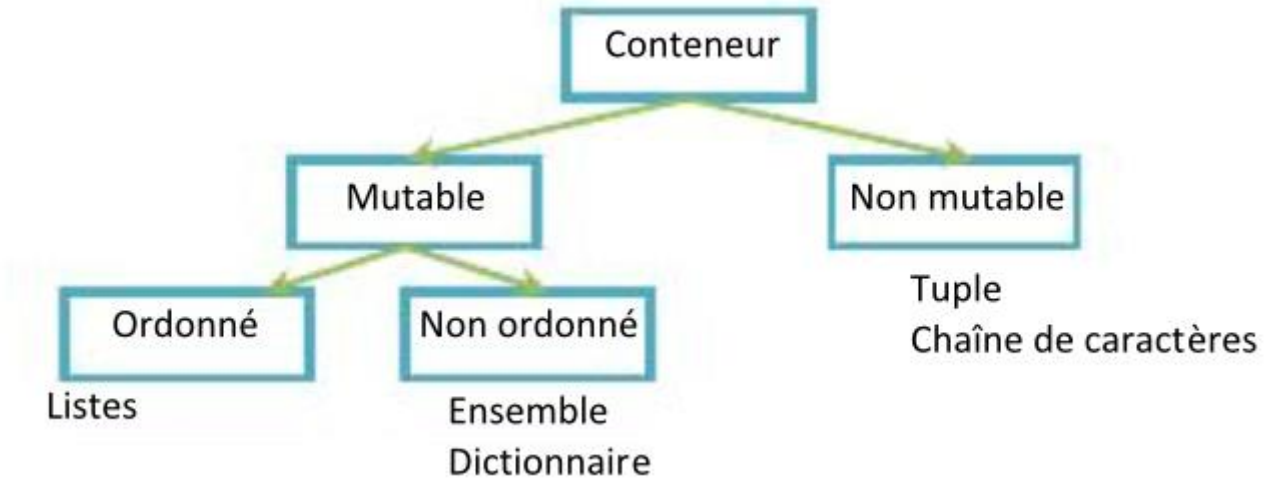
STRUCTURES DE DONNEES A 1 DIMENSION (3) : LES DICTIONNAIRES



Me prévenir de toute erreur éventuelle.

- *Sites internet et logiciels* : pythontutor.com, éditeur et console Python (Thonny, VS Code etc.), franceioi.org.
- *Pré-requis pour prendre un bon départ :*

	☹	😐	☺	☺☺
Types de base.				
Listes.				

Lorsque le logo Python apparait, cela signifie que l'activité doit être aussi faite sur ordinateur.

VERIFIEZ TOUTES VOS REPONSES SUR ORDINATEUR !

NOM et prénom : …… *Première spécialité NSI*

I. PROBLEMATIQUE : CORRESPONDANCES DE DONNEES.

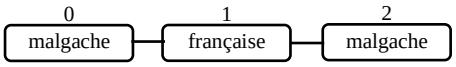
Dans la vraie vie, beaucoup d'informations se présentent sous la forme d'associations de données.

Ex : On s'intéresse à la nationalité d'élèves : Mahe est malgache, Célia est française, Moeve est malgache.

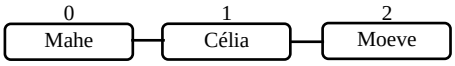
Comment représenter simplement ces correspondances de données ?

- 1ère idée : Sous forme de liste de listes : [[Mahe , malgache] , [Célia , française] , [Moeve , malgache]].
  Inconvénient ? ……..
- 2ème idée : Sous forme d'une liste simple : [malgache , française , malgache].
  Inconvénients ? ……

Cette liste simple peut être vue comme un train de wagons numérotés :



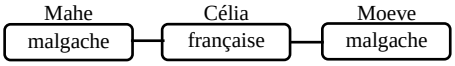
Si la représentation est plus simple que pour la liste de listes, on a une perte d'informations : les nationalités ne sont plus directement associées aux prénoms, mais à des numéros ! Il nous faudrait donc sauvegarder dans une 2ème liste les correspondances Numéros - Prénoms !



Soient au total 2 listes pour une simple association Prénoms – Nationalités ? Impensable !

- 3ème idée : Une sorte de « liste » qui serait simple et sans perte d'informations.

Débarrassons-nous donc des numéros intermédiaires sans intérêt et faisons jouer directement aux prénoms le rôle des indices. On obtient alors un train de wagons nommés :



Cela a l'air de bien répondre à notre problématique : représenter simplement une association de données.

On dirait une liste sans en être tout à fait une : **ici, les index ne sont pas des numéros mais les prénoms !**

Cette nouvelle structure de données a un nom : les ……..

II. DEBUTER AVEC LES DICTIONNAIRES.

Définition	• Un dictionnaire est un **ensemble modifiable fini non numéroté de couples d'objets liés modifiables** *(Train de wagons nommés. Train modifiable, contenu des wagons aussi).* • **Les dictionnaires font donc partie des objets muables.**
Vocabulaire	• Les couples d'objets liés contenus dans un dictionnaire *(les wagons)* s'appellent les **items**. Chaque item est donc une association **Clé : Valeur** (en anglais **Key : Value**). • **Les clés *(les noms des wagons)* jouent le même rôle que les index numérotés (indices) pour les listes. Ces clés permettent donc le repérage et l'accès aux items du dictionnaire.** Ces **clés de repérage et d'accès** sont aussi appelées **index nommés, étiquettes, labels, etc.** ⚠ Afin que les clés repèrent correctement les items, 2 conditions nécessaires : · **clés uniques** *(impossible d'avoir 2 wagons avec le même nom !).* · **clé objet immuable !** Donc pas de liste ou de dictionnaire comme clé ! • Valeur *(contenu du wagon)* peut être n'importe quel objet y compris un autre dictionnaire ! • On parle de dictionnaire car un vrai dictionnaire peut être vu comme un ensemble de couples Mot : Définition. Grosse différence : un vrai dictionnaire est trié, en informatique non.

Syntaxe d'un dictionnaire 3 règles.	1. Chaque item décrit une unique correspondance : **clé unique : valeur** ⚠ « : » et non « = » ! ⚠ Pas de parenthèses autour de chaque couple clé : valeur. Ce ne sont pas des tuples ! 2. Items **séparés par des virgules**. 3. Ensemble des items enfermé **entre 2 accolades { }**. **{ clé unique : valeur (item unique) , clé unique : valeur (autre item unique) , clé unique : valeur (autre item unique) , etc. }**
Définir un dictionnaire.	• Définir un dictionnaire, c'est écrire tous ses items entre accolades. • Exemples : ① { } : dictionnaire ……. **Utile pour initialiser un dictionnaire à vide.** ② { (3 , 4) : 'rouge' } : dictionnaire contenant ….. seul item. 1 clé (3 , 4) de type …….. associée à 1 valeur 'rouge' de type …….. ③ { 'Anakin' : 27 , 'Maul' : 31 , 'Obi' : 40 } : dictionnaire contenant …… items. 3 …….. (les prénoms) de type str associées à 3 …… (les âges) de type int. ④ { True : { 'a' : 5} , ('3', 5) : [1 , 4] } : dictionnaire farfelu contenant …… items. 1 clé de type ……..…….. associée à une valeur de type …….……. 1 clé de type …..…..... associée à une valeur de type ….……. ⑤ { 0 : valeur , 1 : autre valeur , 2 : autre valeur }. Clés ? ……. • Remarques sur les exemples : ④ Ce dictionnaire complètement artificiel montre qu'en théorie les clés ne sont pas forcément toutes du même type, de même pour les valeurs. En pratique, les clés représentent souvent soit des critères soit des entités donc les clés sont souvent de type str ou tuple. Les valeurs quant à elles peuvent être de n'importe quel type ! ⑤ Ce dictionnaire peut être vu comme une liste, les clés entières jouant le rôle des ……. de la liste. Aucun intérêt, autant utiliser une vraie liste dans ce cas ! Inversement, une liste peut être vue comme un dictionnaire ordonné à clés ……
Nommer un dictionnaire.	• C'est affecter le dictionnaire à un nom de variable. Ex : personnels = { (Alain , Provist) : dirlo , (Beth , Emechan) : prof }
Type d'un dictionnaire.	• Un dictionnaire est un objet de type **dict**. Ex : type({1 : 2}) renverra < class '…..…….' > « dict » est donc un mot réservé de Python. Evitez de l'utiliser comme nom de qq chose !
Ordre d'un dictionnaire. Python ≥ 3.6	• Depuis cette version 3.6, les dictionnaires ont un ordre naturel : **l'ordre d'ajout des items.** Ex : print({1 : 'z' , 'a' : 5}) affiche bien {1 : 'z' , 'a' : 5}, ce qui n'était pas le cas avant 3.6. Comme on le voit, cet ordre temporel ne signifie pas que le dictionnaire est trié ! • Cet ordre comptera dans le parcourt des dictionnaires à l'aide d'une boucle For.

III. QUE REPRESENTER PAR UN DICTIONNAIRE ?

En général, un dictionnaire implémente (modélise informatiquement) l'une des 2 situations suivantes :

Fiche d'informations : 1 seule entité ↔ plusieurs critères	***Situation de comparaison :*** plusieurs entités ↔ 1 critère commun
On a affaire à 1 seule et même entité et l'on souhaite regrouper des informations la concernant : Ex : {'Pays' : 'USA' , 'numéro_président' : 44 } De qui s'agit-il ? …… Remarque : Dans ce type de dictionnaire : • les critères sont les clés et explicitement nommés. • l'entité se devine implicitement par le dictionnaire. D'où l'importance de bien nommer un dictionnaire faisant office de fiche d'informations : Obama = {'Pays' : 'USA','numéro_président' : 44}	On a affaire à plusieurs entités et l'on souhaite regrouper leur comparaison selon le même critère : Ex : {'Eli' : 'NSI' , 'Ella' : 'SVT} Critère de comparaison ? …….. Remarque : Dans ce type de dictionnaire : • les entités sont les clés et explicitement nommées. • le critère se devine implicitement par les valeurs. D'où l'importance de bien nommer un dictionnaire faisant office de comparaison : Spécialités_élèves = {'Eli' : 'NSI' , 'Ella' : 'SVT}
• Créer une fiche d'informations sous forme d'un dico de 3 items au choix. La nommer judicieusement. • Ecrire un dictionnaire de 3 items au choix rendant compte d'une comparaison. Le nommer judicieusement.	

IV. TRAVAILLER AVEC LES DICOS ; COMPARAISON AUX LISTES.

Création	
Création simple et initialisation d'un dictionnaire : **nom_dictionnaire = { clé : valeur , clé : valeur , etc. }** Ex : • Initialiser un dico Moi avec vos 'nom' et 'prénom' : ……... • Initialiser un dico Chats à vide : ……	*Création simple et initialisation d'une liste :* **nom_liste = [élément , élément , etc.]**
Sans objet : un dictionnaire ne peut pas avoir des items identiques car les clés sont toutes ……..…… !	*Liste b de n éléments tous identiques :* **b = [élément] * n ou n * [élément]**
Compréhension de dictionnaire : (avec f et g 2 fonctions) **b ={ f(x) : g(y) for (x , y) in liste couples if condition }**	*Compréhension de liste ou liste image :* **b =[fonction(x) for x in séquence if condition]**
Créer un dictionnaire en convertissant une liste de couples : **nom_dico = dict (liste couples)** Ex : dict ([('Nom', 'Vador')]) → { 'Nom' : 'Vador' } dict([['a', 1] , ['b' , 2]]) → …….	*Créer une liste par conversion d'un itérable (chaîne str, tuple, dico, range etc.) :* **nom_liste = list (objet itérable)**
Créer un dictionnaire par accolement de 2 dicos (vers. 3.9) : **dico3 = dico1 | dico2**	*Créer une liste par accolement de 2 listes :* **liste3 = liste1 + liste2**

Gestion Maintenance	
Modifier la valeur d'un item de dico connaissant sa clé : **dico[clé] = nouvelle_valeur** Ex : Soit Darth = {'Nom' : 'Vadir' , 'âge' : 45}. Ecrire l'instruction qui corrigera le nom : ……	*Modifier la valeur de l'élément d'indice k de liste :* **liste[k] = nouvelle_valeur**
Supprimer un item de dico connaissant sa clé : **……..** • Supprime du présent dictionnaire l'item d'index clé. **Pas d'affectation !** • Si clé non dans le dictionnaire → KeyError. • Ex : Soit Dic = {3 : 4 , 4 : 3 }. Ecrire l'instruction qui supprime l'item de valeur 4 : ……	*Supprimer un élément d'indice k de liste :* **del liste[indice k]**
Vider un dictionnaire : **……** Efface tous les items du dictionnaire lui-même qui devient alors vide ({ }). **Pas d'affectation !**	*Vider une liste :* **liste.clear( )**
Ajouter un seul item en toute fin de dictionnaire : **dico[nouvelle clé] = valeur** • Pas de méthode .append( ) pour les dictionnaires ! • ⚠ si clé est déjà dans le dictionnaire, l'ancienne valeur associée à cette clé sera écrasée par cette nouvelle valeur !	*Ajouter un seul élément en toute fin de liste :* **liste.append(élément)**
Sans objet : les clés d'un dictionnaire sont des index nommés et non des index numérotés !	*Insérer un objet à l'indice k de liste :* **liste.insert(k , objet)**
Etendre un dico1 avec les items d'un dico2. **dico1.update(dico2)** • Etend dico1 en rajoutant à la fin de dico1 lui-même tous les items de dico2. Si 2 clés sont égales, la valeur associée à cette clé dans dico1 est remplacée par la valeur associée à cette clé dans dico2. • Ex : Soient a = {3 : 4 , 7 : 3 } et b = {5 : 3 , 3 : 8 } a.update(b) → …… b.update(a) → …… • Remarque : En fait, n'importe quel itérable de couples (liste de couples, etc.) peut être utilisé à la place de dico2.	*Etendre liste1 avec les éléments de liste2.* **liste1.extend(liste2 ou itérable)**

<table>
<tr><th colspan="2">Tirer de l'information</th></tr>
<tr><td>Nombres d'items d'un dictionnaire :
……</td><td>Nombres d'éléments d'une liste :
len(liste)</td></tr>
<tr><td>Récupérer dans une variable la valeur associée à une clé d'un dictionnaire : 2 façons.
① a = dico[clé] (crochets et non ……..… !)
Si clé hors dico → KeyError
Ex : Soit D = { 7 : 4 , 'a' : 7 , '7' : 5 , 5 : 'a' }, compléter :
D['7'] → …… ……..…. → 7 ……..… → 'a'
② a = dico.get(clé , trucbidule)
Si clé dans dico, renvoie alors la valeur associée à clé.
Si clé hors dico, renvoie trucbidule (au lieu de KeyError !).
Ex : a = D.get('7', 'clé absente') → a vaut 5
Mais a = D.get('5', 'clé absente') → …….</td><td>Récupérer dans une variable l'élément d'indice k d'une liste :
a = liste[indice k]
Si k hors liste → IndexError</td></tr>
<tr><td>Sans objet : tous les items sont différents dans un dictionnaire car les clés sont ……. !</td><td>Nombre de fois où apparait objet dans liste :
liste.count(objet)</td></tr>
<tr><td>Oui ou non clé est-elle présente dans dictionnaire ?
……
• ⚠ recherche parmi les clés du dictionnaire et non parmi les valeurs !
• Ex : 'A' in { 'a' : 1 , 1 : 'A' } → False.
2 in { 'a' : 2 , '2' : 'A' } → ……</td><td>Oui ou non objet est-il présent dans liste ?
objet in liste</td></tr>
<tr><td>Sans objet : les clés d'un dictionnaire sont des index nommés et non des index ……. !</td><td>Connaître le 1er indice d'objet dans liste :
liste.index(objet)</td></tr>
<tr><th colspan="2">Copie</th></tr>
<tr><td>• Copies liées de dictionnaires :
……..
• Copies non liées de dictionnaires :
……..</td><td>• Copies liées de listes :
liste2 = liste1
• Copies non liées de listes :
liste2 = liste1.copy( )</td></tr>
<tr><th colspan="2">Tranches (slices)</th></tr>
<tr><td>Sans objet : les clés d'un dictionnaire sont des index …….. et non des index …… !</td><td>Créer une tranche a (slice) à partir de liste :
a = liste[d : f]</td></tr>
</table>

Il existe encore quelques autres méthodes pour les dictionnaires :

- dico.pop(clé) pour supprimer un item connaissant sa clé et renvoyer sa valeur.
- dico.setdefault(clé , valeur) pour ajouter un item clé : valeur si clé n'est pas dans dico.

Etc. Je vous renvoie vers la documentation officielle Python.

V. EXPLOITATION DES DONNEES DANS UN DICTIONNAIRE.

Les méthodes et fonctions vues auparavant assurent les commodités de base : création, maintenance et récupération d'informations.

Qu'en est-il maintenant de la pleine exploitation des données contenue dans un dictionnaire ?

A. Liste d'items, liste de clés et liste de valeurs liées à un dictionnaire :

« Liste » des items : méthode .items( )	
a = list(dico.items( )) • Récupère dans la variable a tous les items de dico sous forme d'une **liste de tuples**. • Ex : Soit dico = { 'a' : 1 , 1 : 'A' } b = list(dico.items( )) → b vaut [('a' , 1) , (1 , 'A')].	Personnages = { 'Marge' : 'Simpson' , 'Clown' : 'Krusty' } Ecrire l'instruction qui affecte à b la liste des items de Personnages : …… b[1] → …….. b[1][0] → ……
« Liste » des clés : méthode .keys( )	
a = list(dico.keys( )) • Récupère dans la variable a toutes les clés des items de dico sous forme d'une **liste simple**. • Ex : Soit dico = { 'a' : 1 , 1 : 'A' } c = list(dico.keys( )) → c vaut ['a' , 1].	Personnages = { 'Homer' : 'Simpson' , 'Ned' : 'Flanders' } Ecrire l'instruction qui affecte à c la liste des clés de Personnages : …… c[1] → …….
« Liste » des valeurs : méthode .values( )	
……. • Récupère dans la variable a toutes les valeurs des items de dico sous forme d'une **liste simple**. • Ex : Soit dico = {'a' : 1 , 1 : 'A'} f = list(dico.values( )) → f vaut …….	Personnages = {'Lisa' : 'Simpson' , 'Waylon' : 'Smithers'} Ecrire l'instruction qui affecte à f la liste des valeurs de Personnages : …… f[0] → ……..

B. Parcours des listes liées au dictionnaire :

Exploiter les données d'un dictionnaire nécessite évidemment de le parcourir à l'aide d'une boucle …… !

Seulement voilà, **on ne parcourt pas directement un dictionnaire** comme on parcourt une liste.

On parcourt en fait soit **dico.items( ), soit dico.keys( ), soit dico.values( )**. Ces 3 itérables (sans la fonction list( ) devant) sont liés dynamiquement au dictionnaire : on les appelle des **vues de dictionnaires**.

Parcours direct sur la « liste » des tuples-items.	***Parcours direct sur la « liste » des clés.***	***Parcours direct sur la « liste » des valeurs.***
① **for k in dico.items( ) :** k vaudra successivement chaque tuple de la « liste » des tuples. ② **for j , k in dico.items( ) :** j ↔ 1ère valeur de chaque tuple. k ↔ 2ème valeur de chaque tuple.	**for k in dico.keys( ) :** k vaudra successivement chaque clé de la « liste » des clés issues du dictionnaire.	**for k in dico.values( ) :** k vaudra successivement chaque valeur de la « liste » des valeurs issues du dictionnaire.

➢ Exemple : Soit le dictionnaire Noms = { 'Cersei' : 'Lannister' , 'Theon' : 'Greyjoy' , 'Khal' : 'Drogo' }.

for k in Noms.items( ) : print (k) affichera : (Cersei , Lannister) (Theon , Greyjoy) (Khal , Drogo)	for j,k in Noms.items( ) : print (f"{ j } : { k }") affichera : Cersei : Lannister Theon : Greyjoy Khal : Drogo	for k in Noms.keys( ) : print (k) affichera : Cersei Theon Khal	for k in Noms.values( ) : print (k) affichera : Lannister Greyjoy Drogo

➢ Application : Ecrire les boucles qui afficheront les colonnes suivantes :

affichera : Lannister : Cersei Greyjoy : Theon Drogo : Khal	affichera : (Lannister , Cersei) (Greyjoy , Theon) (Drogo , Khal)	affichera : CERSEI THEON KHAL

C. Remarques sur les listes liées :

❶ On peut appliquer aux « listes » liées dico.items( ) dico.keys( ) dico.values( ) (qui sont des itérables en fait) toutes les fonctions et méthodes utilisables sur les vraies listes.

En particulier, on ne trie pas directement un dictionnaire mais on trie la « liste » liée dynamiquement dico.items( ) avec la fonction sorted( ).

❷ « for k in dico » revient au même que « for k in dico.keys( ) » !

Parcourir directement un dictionnaire revient en fait à parcourir la « liste » liée de ses clés.

VI. EXERCICES CLASSIQUES SUR LES DICTIONNAIRES.

❶	***Fabrication de dictionnaires : techniques de base. (difficulté = *)***

Soient les listes : Prénoms = ['Riri' , 'Fifi' , 'Loulou'] Ages = [5 , 5 , 5]

1. Ecrire à la main le dico NeveuxDonald avec pour clés les prénoms et pour valeurs les âges :

……

2. Fabriquer ce dictionnaire des 2 façons suivantes :

Avec une boucle For :	Avec la fonction zip(liste1, liste2) :
NeveuxDonald = { } for	*La fonction zip( ) fabrique à partir de 2 listes une seule liste de couples. Ex : zip(['a' , 'b'] , [1 , 2]) renvoie [('a', 1) , ('b', 2)].* *Et convertir ce zippage en dictionnaire avec la fonction dict( ) !* NeveuxDonald = ……

❷ *Fabrication à la volée d'un dictionnaire par l'utilisateur (*)*

1. Ecrire une fonction Construct_dico qui :

- En entrée reçoit le nombre voulu d'items n.
- A chaque étape de la construction :
  demande à l'utilisateur 'Entrez la clé : '.
  demande à l'utilisateur 'Entrez la valeur : '.
  affiche le dictionnaire intermédiaire.
- En sortie renvoie le dictionnaire final.

2. Les nombres entrés sont considérés comme du texte ! Améliorer le programme en faisant en sorte que les nombres entrés soient considérés comme de vrais nombres (entiers) !
Utiliser pour cela la méthode .isnumeric( ).

```
def Construct_dico(n) :
```

❸ *Nombres d'occurrences de chaque caractère d'une chaîne de caractères (**)*

Un exercice bien plus facile à réaliser avec un dictionnaire qu'avec une liste ! (France IOI Niv2 chap3 B2)

Soit une chaîne de caractères entrée par l'utilisateur.
Déterminer le nombre d'occurrences de chaque caractère sous forme d'un dictionnaire.

❹ *Recherche par valeur dans un dictionnaire (*)*

La recherche par clé est très simple dans un dictionnaire : il y a pour cela l'instruction toute faite : clé …… dico
Hélas, pas d'instruction …….. !
Ecrire une fonction Recherche_valeur avec :

- en entrée : le dico et la valeur cherchée.
- en sortie : la clé associée si la valeur est dans le dico, sinon la phrase « la valeur n'est pas dans le dico ! ».

Plus facile par dico.items( ) que par dico.values( ) !

❺ *« Trier un dictionnaire » (*)*

En fait, comme déjà dit p.8 en remarque sur les listes liées, **ce n'est pas le dictionnaire qu'on trie (il n'existe pas de fonction ou de méthode de tri qui agissent sur les objets dict) mais la « liste » de tuples dico.items( )** tirée du dictionnaire. On peut alors trier selon 2 critères :

Tri selon les clés

b = sorted(dico.items( ))

• Convertit le dico en liste de tuples (dico.items( )), puis trie cette liste selon la 1ère coordonnée (donc ce qui correspondait aux clés dans dico), puis affecte à la variable b la liste ainsi triée.

• Exemple : Soit le dictionnaire a = { 'b' : 2 , 'a' : 5}.

a.items( ) → équivalent de [('b' , 2) , ('a' , 5)].

sorted(a.items( )) → [('a' , 5) , ('b' , 2)].

Rien n'empêche après de convertir ce truc en dictionnaire !

a = dict(sorted(a.items( ))) → a vaut maintenant { 'a' : 5 , 'b' : 2 }.

• Application : Soit un dictionnaire Noms.

Ecrire l'instruction qui renverra le dictionnaire Noms trié selon ses clés. Puis tester sur Pythontutor !

…….

Tri selon les valeurs

• Par défaut, la fonction sorted( ) renvoie une liste de tuples triée selon la 1ère coordonnée.

• Si on veut trier selon la 2ème coordonnée (la valeur), il faut l'indiquer en précisant le paramètre key :

b = sorted(dico.items( ) , key = lambda élément : élément[1])

• Explications :

Qui trie-t-on ? dico.items( ) qui renvoie la « liste » des tuples issue du dictionnaire.

Selon quel critère ? key = lambda élément : élément[1].

Cette instruction fait agir la fonction anonyme lambda qui prend chaque élément de la « liste » de tuples dico.items( ) (donc un tuple) et va y sélectionner élément[1] (qui est bien la 2ème coordonnée de l'élément c-à-d ce qui correspondait à la valeur dans dico).

Cette action de sélection est souvent appelée clé de sélection (ou clé de tri) d'où le mot key. Dans les langages de bases de données, on utilise d'ailleurs plutôt le mot Select que Key.

• Exemple : Soit liste_tuples une liste de tuples ayant chacun 5 coordonnées.

sorted(liste_tuples, key = lambda tuple : tuple[3]) trie liste_tuples selon quelle coordonnée ? La ……..

Ecrire l'instruction qui permet de trier liste_tuples selon la 1ère coordonnée :

• Application : Entrer un dictionnaire Ages_personnes avec pour clés les noms et pour valeurs les âges.

Puis écrire l'instruction qui, à partir de Ages_personnes, affecte à b la liste triée selon les âges. Afficher b.

VII. MINI-BASE DE DONNEES. (**) (***)

On va construire une mini-base de données constituée des fiches d'informations de personnes (mais cela aurait pu être n'importe quoi d'autre : des écoles, des requins, des sentiments etc.)

Chaque fiche de personne contiendra le Nom, le Prénom, l'Age et la Profession.

1. Sous quelle forme doit se présenter une fiche d'informations : …….

Ecrire une fonction Construct_fiche qui permettra d'entrer les informations demandées et renverra le dictionnaire correspondant à cette fiche.

2. Toutes les fiches doivent être réunies dans une **liste Team** qui sera donc notre mini-base de données.

Demander d'abord combien de personnes doit contenir cette Team.

Compléter le programme précédant pour construire petit à petit Team par ajout successif des fiches-dicos.

3. Trier Team par exemple par rapport à l'âge ou une autre clé.

Pour trier une liste de dictionnaires Liste_dico selon l'un des critères communs des dictionnaires, on peut utiliser l'instruction suivante : **Liste_dicos = sorted(Liste_dicos, key = lambda dico : dico[critère])**

Exemple : Soit Liste_chats une liste de dictionnaires de chats.

Alors Liste_chats = sorted(Liste_chats, key=lambda chat : chat['taille']) renverra la liste des fiches (liste des dictionnaires) de chats triée selon la clé 'taille'.

4. Ecrire une fonction qui recherche si une valeur est dans Team et renvoie à quelle personne et à quel critère cette valeur correspond. Sinon affiche 'La valeur n'est pas dans la base de données.'

On pourra tester le programme sur les fiches d'identité des personnes suivantes :

Organa , Leia , 27 , Princesse Vador , Dark , 46 , Sith Ren , Kylo , 20 , Charlo

VIII. POUR FINIR.

- Dans d'autres langages, les dictionnaires s'appellent Tableaux associatifs, p-upplets nommés.
- Les dictionnaires sont un conteneur d'informations tout comme les listes et les tuples.

Placer Listes, Tuples et Dictionnaires dans les cases appropriées :

	Modifiable par les traitements	*Non modifiable par les traitements*
Index nommés		
Index numérotés		

Données plutôt hétérogènes	*Données plutôt homogènes*	*Associations de données*

Exemple : Pour les données suivantes, quelle structure semble la plus adaptée pour l'implémenter ?

- Une carte d'identité : ……
- Des notes à un contrôle : ……..
- Login / Mot de passe : ……..

- **La maîtrise des dictionnaires est incontournable en Algorithmique-Programmation : les dictionnaires constituent avec les listes l'un des fondements des Bases de Données.**

Ai-je tout compris ? Dictionnaires.	☹	😐	☺	☺☺
Problématique des dictionnaires.				
Définition et vocabulaire des dictionnaires.				
Que représenter par un dictionnaire.				
Création d'un dictionnaire.				
Maintenance d'un dictionnaire.				
Tirer de l'information d'un dictionnaire.				
Itérables liées à un dictionnaire : dico.items( ), dico.keys( ), dico.values( ).				
Recherche par valeur dans un dictionnaire.				
Trier une liste tirée d'un dictionnaire : fonction sorted(liste, key = lambda ….).				
Différencier Listes, Tuples et Dictionnaires.				

Sur la page de garde, il y a une erreur dans l'illustration ! Laquelle ?