PYTHON

DEVOIR MAISON

Notes et Commentaires

Notes

Nom	Note
Alexandre	15.00
Aréna	19.50
Lilou	20.00
Lucas	19.75
Lucie	20.00
Lyvahne	14.50
Marie	20.00
Mirana	20.00
Raphaël	18.25
Simon	18.75
Yann	19.75

Alexandre

15/20

Exercice	Note	Commentaire
`debug.py`	0.5 / 1	(0.375 arrondi à 0.5) Je n'ai pas trouvé le reste de l'exercice
`list_easy.py`	2.25 / 2	
`list_medium.py`	3 / 4	`3 derniers jours de la semaine` : `days[3:6]` affiche `["thursday", "friday", "saturday"]`, les 3 derniers jours sont: `["friday", "saturday", "sunday"]`. Privilégie la méthode `days[-3:]` qui récupère tous les éléments à partir du 3ème en partant de la fin jusqu'au dernier
`dict_easy.py`	3 / 3	
`dict_medium.py`	4.25 / 4	
`dict_hard.py`	2 / 6	`Ajout de données` : tu ajoutes une `list` et non un `dict`. La fin de l'exercice était réalisable même en ne réussissant pas cette partie

Aréna

19.5/20

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	1.75	`Ajoutez éléments de la liste add_to_lst à lst_easy` :

Exercice	Note	Commentaire
	/ 2	il fallait utiliser la méthode `.extend()` pour ajouter tous les éléments d'une liste dans une autre liste
`list_medium.py`	4 / 4	
`dict_easy.py`	3 / 3	
`dict_medium.py`	4.25 / 4	
`dict_hard.py`	5.5 / 6	`"in_stock" de "laptop" à False` : False est un `boolean` et non une `string` => `False` au lieu de `"False"`

Lilou

20/20

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	2.25 / 2	
`list_medium.py`	3.5 / 4	`Afficher "thursday"` : le concept est compris mais tu affiches `"tuesday"` à la place de `"thursday"`
		`3 derniers jours de la semaine` : c'est correct, mais privilégie l'utilisation de `days[-3:]` qui récupère tous les éléments à partir du 3ème en partant de la fin jusqu'au dernier
`dict_easy.py`	3 / 3	Tu utilises parfois `.update()` et parfois `dct[key]` pour mettre à jour la valeur d'une clef. Même si les deux méthodes font la même chose, essaye de ne t'en tenir qu'à

Exercice	Note	Commentaire
		une seule.
`dict_medium.py`	4.25 / 4	
`dict_hard.py`	6.25 / 6	

Lucas

19.75/20

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	2.25 / 2	
`list_medium.py`	3 / 4	`3 derniers jours de la semaine` : `days[3:6]` affiche `["thursday", "friday", "saturday"]`, les 3 derniers jours sont: `["friday", "saturday", "sunday"]`. Privilégie la méthode `days[-3:]` qui récupère tous les éléments à partir du 3ème en partant de la fin jusqu'au dernier
`dict_easy.py`	3 / 3	
`dict_medium.py`	4.25 / 4	
`dict_hard.py`	6.25 / 6	

Lucie

20/20

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	2.25 / 2	`lst_easy = lst_easy + add_to_lst` peut s'écrire `lst_easy += add_to_lst` mais ta façon est juste
`list_medium.py`	3 / 4	`3 derniers jours de la semaine` : c'est correct, mais privilégie l'utilisation de `days[-3:]` qui récupère tous les éléments à partir du 3ème en partant de la fin jusqu'au dernier
`dict_easy.py`	3 / 3	
`dict_medium.py`	4.25 / 4	
`dict_hard.py`	5.75 / 6	`"in_stock" de "laptop" à False` : False est un `boolean` et non une `string` => `False` au lieu de `"False"`

Lyvahne

14.50/20

Consigne non-respectée: `Creéz un fichier .py par exercice, ...`

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	1.5 / 2	`lst_easy = [1]` : tu crées une liste avec un élément, cela est différent d puis d'ajouter un élément à cette liste. La solution était `lst_easy.appen`

Exercice	Note	Commentaire
		`Bonus` : La taille est de 4 car la liste `[4, 5]` compte pour un seul éléme `lst_easy` .
`list_medium.py`	3 / 4	`3 derniers jours de la semaine` : `days[3:6]` affiche `["thursday", "friday", "saturday"]` , les 3 derniers jours sont: `["friday", "saturday", "sunday"]` . Privilégie la méthode `days[-3:]` les éléments à partir du 3ème en partant de la fin jusqu'au dernier
`dict_easy.py`	3 / 3	
`dict_medium.py`	4 / 4	
`dict_hard.py`	2 / 6	Il manque une paranthèse sur ton `print()` ligne 113
		`Ajoutez la clef "keyboard"` : tu ajoutes une `list` et non un `dict` => pour les éléments suivants
		`Dans "watch" , creéz la clef "tags"` : Comme tu peux le voir avec t 105, tu écrases tous les éléments de `"watch"` par une liste contenant `["tags", ["tech", "smartwatch", "iot"]]` . La solution était `products_hard["watch"].update({"tags": ["tech", "smartwatch",` `products_hard["watch"]["tags"] = ["tech", "smartwatch", "iot"]`
		`Ajouter le tag "iot" au "tags" de "smartphone"` : Même comment tu écrases au lieu d'ajouter un élément => `products_hard["laptop"]["tags"].append("iot")`
		`"screen" pour Little Screen Corp` : `products_hard["watch"]` est u n'ont pas de méthodes `.replace()` , donc le terminal t'indique une erre `products_hard["watch"]["components"].update({"screen": "Little`
		`"in_stock" de "laptop" a`False` : `"in_stock"` est une clef de `"lapt` même une clef de `products_hard` => `products_hard["laptop"].update({"in_stock": False})` ou `products_hard["laptop"]["in_stock"] = False`

Marie

20/20

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	2.25 / 2	
`list_medium.py`	4 / 4	
`dict_easy.py`	3 / 3	
`dict_medium.py`	4.25 / 4	Pas besoin de boucle `for` pour le bonus
`dict_hard.py`	6.25 / 6	Tu utilises parfois `.update()` et parfois `dct[key]` pour mettre à jour la valeur d'une clef. Même si les deux méthodes font la même chose, essaye de ne t'en tenir qu'à une seule.

Mirana

20/20

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	2 / 2	`Ajoutez les eleḿents de la liste add_to_lst` : Il fallait utiliser la méthode `.extend()` qui ajoute tous les éléments d'une liste dans une autre liste
`list_medium.py`	4 / 4	

Exercice	Note	Commentaire
`dict_easy.py`	3 / 3	
`dict_medium.py`	4.25 / 4	
`dict_hard.py`	6.25 / 6	Bonus: tu utilises `,` et `+` pour afficher les string. Même si les deux opérateurs font la même chose, essaye de ne t'en tenir qu'à un seul.

Note: il n'est pas nécessaire `print()` à chaque fin de question

Raphaël

18.25

Consigne non-respectée: `Creéz un fichier .py par exercice, ...`

Exercice	Note	Commentaire
`debug.py`	1 / 1	`var_dict_b` : il manque les `:` entre une paire clef/valeur
`list_easy.py`	2.25 / 2	
`list_medium.py`	4 / 4	`3 derniers jours de la semaine` : c'est correct, mais privilégie l'utilisa `days[-3:]` qui récupère tous les éléments à partir du 3ème en partant jusqu'au dernier
`dict_easy.py`	3 / 3	Tu utilises parfois `.update()` et parfois `dct[key]` pour mettre à jour la d'une clef. Même si les deux méthodes font la même chose, essaye de n tenir qu'à une seule.
`dict_medium.py`	3 / 4	`Ici 0 représente le premier caractère de la string` => Ici 0 repre premier caractère de la **list**
		`prix de "smartphone" pour 1500` :

Exercice	Note	Commentaire
		`products_medium["smartphone"] = [1500, 16]` change l'intégralité de valeur, afin de ne changer que le prix => `products_medium["smartphone"][0] = 1500`
		`Bonus` : `if products_medium["smartphone"] == [1500, 0]` vérifie si `products_medium["smartphone"]` est une liste de 2 éléments avec `150` l'index 0 et `0` à l'index 1. Ce qui est différent de vérifier si le produit est => `if products_medium["laptop"][1] == 0`
`dict_hard.py`	5 / 6	`Dans "watch" , creéz la clef "tags"` : `products_hard["watch"]["tags"] = ("tech", "smartwatch", "iot")` un `tuple` (structure de données similaire au `list`), il fallait insérer une avec `["tech", "smartwatch", "iot"]`
		`"screen" pour Little Screen Corp.` : `products_hard["watch"]["components"].clear()` supprime tous les é du dict, or il ne fallait changer que `"screen"`, les autres éléments ne de pas être modifiés => `products_hard["watch"].update({"tags": ["tech", "smartwatch",`

Simon

18.75

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	2.25 / 2	`lst_easy = lst_easy + add_to_lst` peut s'écrire `lst_easy += add_to_lst` mais ta façon est juste
`list_medium.py`	4 / 4	
`dict_easy.py`	3 / 3	

Exercice	Note	Commentaire
`dict_medium.py`	3.25 / 4	`prix de "smartphone" pour 1500` : `products_medium["smartphone"] = [1500, 16]` change l'intégralité de la valeur, afin de ne changer que le prix => `products_medium["smartphone"][0] = 1500`
		`Bonus` : tu n'as fait que la moitié du bonus, il faut afficher la quantité si celle-ci n'est pas égale à 0 => `else: print(products_medium["laptop"][1])`
`dict_hard.py`	4.75 / 6	`Dans "keyboard" , ajoutez la clef "price"` : tu as oublié d'ajouter `"price"`
		`Dans "keyboard" , ajoutez la clef "quantity"` : `products_hard["keyboard"]={"quantity":72}` écrases la valeur pour en créer une nouvelle, c'est pour ça que tu es obligé de réécrire `"quantity":72` quand tu ajoutes `"components"`
		`Bonus` : pas besoin boucle `for` imbriquées, voir correction

Yann

19.75

Exercice	Note	Commentaire
`debug.py`	1 / 1	
`list_easy.py`	2.25 / 2	`lst_easy = lst_easy + add_to_lst` peut s'écrire `lst_easy += add_to_lst` mais ta façon est juste
`list_medium.py`	4 / 4	
`dict_easy.py`	3 / 3	

Exercice	Note	Commentaire
`dict_medium.py`	4.25 / 4	
`dict_hard.py`	5.25 / 6	`Affichez la quantité de "laptop" en stock` : tu affiches la quantité de smartphone en stock: `products_hard["smartphone"]["quantity"]` . Le raisonnement est correct donc pas de points en moins, mais ça ne respecte pas la consigne
		`"components" de "watch"` : Il fallait afficher les clefs/ valeurs sans boucle `for` (nous ne les avions pas encore vu). Lucie a d'ailleurs posé la question sur Slack sur ce qu'il fallait afficher => `print(products_hard["watch"]["components"])`
		`Dans "keyboard" , ajoutez la clef "price"` : `products_hard["keyboard"] = {"price": 500}` écrase le `dict` par `{"price": 500}` , tu n'ajoutes donc pas de clef mais crée un nouveau `dict` => `products_hard["keyboard"].update({"price": 500})`