Programmer avec Python
Exercices

Séance n° 4	**La programmation orientée objet**

Objectif : Créer des programmes utilisant plusieurs classes

1 Classe date personnalisée

1.1 Écriture de la classe

On se propose d'écrire une classe Date pour représenter des dates (on verra plus tard que Python nous fournit également des possibilités de gérer les dates). Pour ceci, on veut retenir le jour, le mois et l'année de la date en question, sous forme d'entiers.

1. Écrivez cette classe. Les 3 champs sont obligatoires
2. Ajoutez une méthode **afficher**. Cette méthode doit prendre en paramètre une langue. Si elle vaut "US", on l'affichera sous la forme MM-JJ-AAAA. Si elle vaut une quelconque autre valeur, on affichera la date sous la forme JJ/MM/AAAA. Attention, il faut bien que les jour et mois s'affichent sur deux caractères (= on met un zéro devant s'il est inférieur à 10)
3. Testez votre classe dans un programme principal.

1.2 Utilisation basique

Maintenant que l'on dispose de cette classe, on voudrait l'utiliser dans d'autres classes.

1. Écrivez une classe Personne, qui retient le nom, le prénom, la nationalité et la date de naissance d'une personne. Ces 4 éléments doivent être fournis lorsque l'on crée une personne.
2. Écrivez une méthode **afficher** dans la classe Personne, qui affiche toutes ses caractéristiques. Si la personne est américaine, on doit afficher sa date de naissance au format américain. Dans tous les autres cas, on l'affichera au format français.
3. Testez

1.3 Comparaisons

1. Ajouter une méthode **est_plus_grande** dans la classe date. Cette méthode prendra en paramètre une autre date, et renverra vrai si la date courante est plus grande que celle passée en paramètre, faux sinon
2. Ajouter ensuite une méthode **est_plus_jeune** dans la classe Personne. Cette méthode prend en paramètre une autre personne, et renvoie vrai si la personne courante est plus jeune que celle passée en paramètre, faux sinon.
3. Testez ces méthodes

2 Halloween

Pour Halloween, on veut superviser la distribution des friandises aux enfants

1. Écrivez une classe **Friandise**. Une friandise est représentée par son nom, son poids en grammes, et sa note de popularité auprès des enfants (sur 10).

2. Écrivez une classe **Bol**. Un bol possède une contenance maximum en grammes ainsi que la liste de friandises qu'il contient.
3. Ajoutez une méthode **prendre_meilleure_friandise** à la classe Bol, qui retire du bol la friandise la mieux notée (s'il y en a plusieurs, on n'en retire qu'une).
4. Ajoutez une méthode **ajouter_friandise** à la classe Bol. S'il n'y a plus assez de place dans le bol, on n'ajoute pas la friandise et la méthode renvoie False. Sinon, elle renvoie True.
5. Ajoutez une méthode **calculer_moyenne**, qui prend en paramètre un booléen nommé *ponderee*. Si le boolean vaut False, on calcule la moyenne des notes des friandises du bol. Sinon, on calcule la moyenne des friandises pondérée par leur poids.
   Ex : J'ai 2 friandises, une de poids 100 et de note 2 et une de poids 300 et de note 8. La moyenne non pondérée vaudra $\frac{2+8}{2} = 5$ et la moyenne pondérée vaudra $\frac{2\times100+8\times300}{100+300} = 6.5$
6. Testez vos classes et méthodes dans un programme principal.

3 Contraventions

On se propose d'écrire un programme permettant de gérer les infractions de certain usagers de la route. Pour ceci, on va représenter les infractions, les procès-verbaux (contenant une ou plusieurs infractions), et les usagers ayant commis des infractions.

1. Écrivez la classe **Infraction**. Une infraction est représentée par son libellé et son tarif. Ajoutez une méthode **afficher** à cette classe.
2. Écrivez la classe **Conducteur**, contenant le nom, prenom et numéro de permis d'une personne. Ajoutez une méthode **afficher** à cette classe.
3. Écrivez la classe **ProcesVerbal**, contenant un numéro de PV, le conducteur à qui il est rattaché, ainsi que la liste des infractions ayant été commises. Ajoutez-y les méthodes suivantes :
   - **afficher**, qui affiche les caractéristiques du PV, y compris celle du conducteur associé ainsi que celles de toutes les infractions associées à ce PV.
   - **montantTotal**, qui renvoie le montant total à payer pour le PV courant.
   - **infractionMax**, qui renvoie l'infraction du PV qui coûte le plus cher (on veut bien l'objet infraction en entier, pas juste le montant le plus cher).
4. Écrivez une fonction **sommeParConducteur** qui prend en paramètre une liste de PV et retourne un dictionnaire : pour chaque conducteur, on veut savoir le montant total de toutes les infractions de tous ses PV. La clé du dictionnaire sera donc le numéro de permis de conduire, et la valeur la somme totale de toutes les amendes à payer.
5. Faites un programme principal afin de tester toutes les méthodes et fonctions de votre code. Afin de simplifier vos tests, il est fortement conseillé de créer des infractions/PV/Conducteurs directement dans votre programme sans faire de saisie utilisateur.

4 L'Armada de Rouen

On se propose d'écrire un programme afin de gérer l'Armada, événement qui a lieu tous les 4-5 ans à Rouen et qui rassemble les plus grands voiliers du monde sur les quais de la ville.

1. Écrire une classe **Voilier**. Un voilier est représenté par son nom, sa longueur et sa voilure. Lui ajouter une méthode afficher
2. Écrire une méthode **comparer** qui compare le voilier courant à un autre passé en paramètre. La méthode renvoie le voilier dont le rapport voilure/taille est le plus grand
3. Écrire une classe **Visiteur**. Un visiteur est représenté par son nom et le planning de ses visites. Pour ceci, on utilisera un dictionnaire, dont les clés seront les numéros des jours de l'événement, et les valeurs seront une liste des bateaux à visiter ce jour-là.

4. Ajouter une méthode **ajouter_bateau** qui prend en paramètres un jour et un bateau et ajoute le bateau dans le planning à ce jour-ci. Attention, un jour donné ne peut contenir que 3 bateaux au maximum. Si le jour est déjà plein, l'ajout ne se fait pas et la méthode renvoie False. Sinon, elle renvoie True.
5. Ajouter une méthode **meilleur_voilier_par_jour** qui renvoie un dictionnaire dont les clés sont les numéros des jours, et les valeurs sont, pour chaque jour, le voilier qui a le meilleur rapport taille/voilure

5 Do it Py way *

On se propose maintenant d'utiliser la surcharge des opérateurs afin de rendre certains codes plus simples et plus lisibles.

5.1 Classe date personnalisée

1. Remplacez les méthodes afficher par des surcharges de l'opérateur `__str__`. Adaptez ensuite votre code là où c'est nécessaire pour supprimer les appels à "afficher"
2. Faites en sorte que l'on puisse comparer les dates et les personnes avec les symboles "<", ">" ou "==". Adaptez votre code de comparaison de dates et de personnes

5.2 Halloween

1. Faites en sorte que l'on puisse comparer 2 bols avec les symboles "<", ">" ou "==". Cette comparaison fera appel à la méthode de la moyenne pondérée.
2. Faites en sorte que l'on puisse fusionner 2 bols avec le symbole "+" : on obtient un nouveau Bol, sa contenance max est la somme des deux contenances max et son contenu est la concaténation des deux listes de friandises
3. Testez !