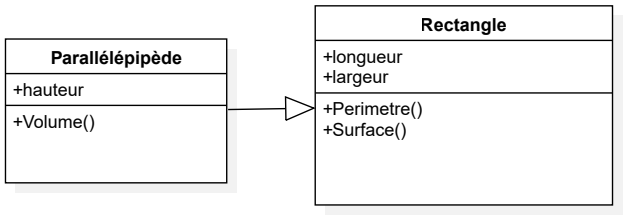
Programmation Orientée Objet

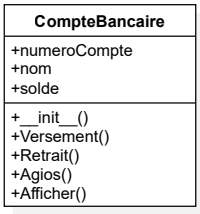
Exercices POO

Exercice 1 :



1. Écrire une classe Rectangle en langage Python, permettant de construire un rectangle dotée d'attributs longueur et largeur.
2. Créer une méthode Périmètre() permettant de calculer le périmètre du rectangle et une méthode Surface() permettant de calculer la surface du rectangle
3. Créer les getters et setters.
4. Créer une classe fille Parallélépipède héritant de la classe Rectangle et dotée en plus d'un attribut hauteur et d'une autre méthode Volume() permettant de calculer le volume du Parallélépipède.

Exercice 2 :



1. Créer une classe Python nommée CompteBancaire qui représente un compte bancaire, ayant pour attributs : numeroCompte (type numérique) , nom (nom du propriétaire du compte du type chaine), solde.
2. Créer un constructeur ayant comme paramètres : numeroCompte, nom, solde.
3. Créer une méthode Versement() qui gère les versements.
4. Créer une méthode Retrait() qui gère les retraits.
5. Créer une méthode Agios() permettant d'appliquer les agios à un pourcentage de 5
6. Créer une méthode afficher() permettant d'afficher les détails sur le compte
7. Donner le code complet de la classe CompteBancaire.

Exercice 3 :

1. Définir une classe Cercle permettant de créer un cercle C(O,r) de centre O(a,b) et de rayon r.
2. Définir une méthode Surface() de la classe qui permet de calculer la surface du cercle
3. Définir une méthode Perimetre() de la classe qui permet de calculer le périmètre du cercle
4. Définir une méthode testAppartenance() de la classe qui permet de tester si un point A(x,y) appartient ou non au cercle C(O,r)

Exercice 4 :

1. Créer une classe Calcul ayant un constructeur par défaut (sans paramètres) permettant d'effectuer différents calculs sur les nombres entiers.
2. Créer au sein de la classe Calcul une méthode nommée Factorielle() qui permet de calculer le factorielle d'un entier. Tester la méthode en faisant une instanciation sur la classe.
3. Créer au sein de la classe Calcul une méthode nommée Somme() permettant de calculer la somme des n premiers entiers 1+2+3+..+n. Tester la méthode.
4. Créer au sein de la classe Calcul une méthode nommée testPrim() permettant de tester la primalité d'un entier donné. Tester la méthode.
5. Créer au sein de la classe Calcul une méthode nommée testPrims() permettant de tester si deux nombres sont premier entre eux.
6. Créer une méthode tableMult() qui crée et affiche la table de multiplication d'un entier donné. Créer ensuite une méthode allTablesMult() permettant d'afficher toutes les tables de multiplications des entiers 1, 2, 3, ..., 9.
7. Créer une méthode statique listDiv() qui récupère tous les diviseurs d'un entier donné sur une liste Ldiv. Créer une autre méthode listDivPrim() qui récupère tous les diviseurs premiers d'un entier donné.

Exercice 5 :

Soit la classe Stagiaire qui modélise un stagiaire, comportera les attributs suivants : Matricule, Nom, Filière, Notes[]

1. Écrire la classe Stagiaire en interdissent l'accès aux attributs.
2. Ajouter les accesseurs et les modificateurs de chaque attribut.
3. Ajouter un compteur qui permet de compter le nombre des objets créés de la classe stagiaire
4. Ajouter un constructeur avec 3 arguments qui initialise le nom, le prénom, et la filière.
5. à chaque instanciation d'un objet de stagiaire, le compteur s'incrémente et le matricule du stagiaire recevra la valeur du compteur.
6. Ajouter une méthode RAZ qui initialise le compteur à 0
7. Ajouter la méthode qui teste l'égalité de 2 objets Stagiaire (2 objets stagiaires sont égaux s'ils ont le même matricule)
8. Écrire une méthode qui permet d'ajouter une note à la liste Notes
9. Écrire une méthode qui permet de faire l'itération des notes du stagiaire
10. Ajouter une méthode CALCUL qui permet de calculer la moyenne générale de chaque stagiaire

Exercice 6 :

Considérons une classe appelée Point ayant deux attributs privés __abs (abscisse du point) et __ord (ordonnée du point) et un attribut statique nb qui représente le nombre de point crées

1. Définissez la classe Point et un constructeur à deux paramètres.
2. Définissez les getters et setters pour les deux attributs en utilisant le décorateur @property
3. Définissez la méthode__ str ()__ qui retourne la représentation mathématique d'un point : (abs,ord)
4. Proposer une surcharge de l'opérateur == permettant de vérifiant si deux point p1(x1,y1) et p2(x2,Y2) sont égaux ou non(x1=x2 et y1=y2)
5. Écrivez la méthode calculerdistance(self) qui permet de calculer la distance entre le point de l'objet courant (self) et l'objet p passé en paramètre. Nous rappelons que la distance entre deux points A(x1,y1) et B(x2,y2), en mathématiques, est égale à:
$$D = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2}$$
6. Ecrivez la fonction calculermilieu(self) qui permet de calculer et de retourner un objet correspondant au milieu du segment défini par le point de l'objet courant (this) et l'objet Point p passé en paramètre. Nous rappelons que les coordonnées d'un point M(xM,yM) milieu de A(x1,y1) et B(x2,y2), sont :
$$x_M = (x1 + x2)/2, y_M = (y1 + y2)/2$$

Considérons maintenant une deuxième classe appelée TroisPoints ayant les attributs __point1, __ point2 et __ point3 qui sont de type Point

7. Définissez les getters/setters (avec le décorateur @property) et un constructeur acceptant trois paramètres.
8. Écrivez une méthode sontalignes(self) qui retourne True si les trois points point1 , point2 et point3 sont alignés, False sinon. Nous rappelons que trois points A, B et C sont alignés si AB = AC + BC, AC = AB + BC ou BC = AC + AB (AB désignant la distance séparant le point A du point B, pareillement pour AC et BC).
9. Écrivez une méthode estisocèle(self) qui retourne True si les trois points point1, point2 et point3 forment un triangle isocèle, False sinon. Nous rappelons qu'un triangle ABC est isocèle si AB = AC ou AB = BC ou BC = AC.
10. Implémentez une version statique (méthode décorée par @staticmethod) des deux méthodes calculant la distance et le milieu.
11. Dans un fichier main.py, testez toutes les classes et méthodes que vous avez implémentées

Exercice 7 :

On souhaite traiter des comptes bancaires. Un compte bancaire possède à tout moment une donnée entière, son solde. Il est caractérisé par ailleurs par un code donné sous forme de chaîne de caractères, et un titulaire qui est une personne physique. Ce solde peut être positif (compte créditeur) ou négatif (compte débiteur). Au départ, un compte bancaire a un solde nul, un code nul par défaut, et un titulaire inconnu. Il est aussi possible de créer un compte en précisant son solde initial, son titulaire et son code initial.

Utiliser son compte consiste à pouvoir y faire des dépôts et des retraits. Pour ces deux opérations, il faut connaître le montant de l'opération et vérifier que le code proposé est bien le code du compte. L'utilisateur peut aussi consulter le solde de son compte

1. Écrivez une classe PersonnePhysique, elle est décrite par un nom et prénom, et comprendra les méthodes suivantes :
   - constructeur
   - une méthode de consultation du nom

- une méthode de consultation du prénom

2. Écrivez une classe CompteBancaire prenant en compte la description précédente. Cette classe comprendra les méthodes suivantes :
   - constructeurs
   - une méthode de modification du code
   - une méthode de consultation du solde
   - une méthode de dépôt
   - une méthode de retrait
3. La banque souhaite autoriser pour certains clients un découvert. Par défaut, ce découvert autorisé est nul Une méthode decouvertAutorise permettra d'en spécifier la valeur. Un retrait est alors possible tant que le client ne dépasse pas ce découvert autorisé. Le cas échéant, le retrait est refusé. Dans ce dernier cas, la méthode de retrait lancera une exception.
   - précisez le(s) attribut(s) à ajouter à la classe CompteBancaire et modifiez le(s) constructeur(s).
   - écrivez la méthode decouvertAutorise.
   - définissez une classe RetraitInterditException qui sera lancée par la méthode de retrait dans le cas d'une tentative de retrait impossible.
   - modifiez la méthode de retrait.
4. La banque souhaite conserver le nombre de comptes bancaires existants, le nombre de comptes débiteurs. Proposez une solution pour cela
5. La banque souhaite maintenant gérer des comptes "rémunérés". Il s'agit de comptes bancaires avec un attribut supplémentaire, un "taux d'intérêt". Les titulaires de tels comptes voudront savoir :
   - quel sera leur solde dans x années, en supposant qu'aucune opération ne sera effectuée sur leur compte ;
   - combien d'années sont nécessaires pour doubler leur solde, en supposant qu'aucune opération ne sera effectuée sur leur compte.

Exercice 8 :

Un Individu est décrit par ses attributs privées :

- un numéro de CIN
- Un nom
- Un prénom
- Une adresse

1. Définir une classe Individu avec ses données membres et une méthode Affichage () qui affiche les informations de chaque objet créer a partir de cette classe.
2. Créer une classe de test pour dérouler un scénario activant des objets particuliers de cette classe : créer deux individus et afficher leurs informations

On dérive la classe Individu déjà vue par une classe Formateur. En plus des données membres Individu, la classe Formateur a comme données membres privées :

- Masse horaire effective
- Heure supplémentaire
- Taux horaire effectif

- Taux horaire supplémentaire

La méthode CalculSalaire () intègre un mode de calcul comme suit : CalculSalaire = (Masse horaire effectif* taux horaire effectif) + (Heure supplémentaire* taux horaire supplémentaire)

3. Construire la classe Formateur qui hérite de la classe Individu
4. Au niveau de la Classe de Test :
   - Prévoir deux Formateurs et les instancier avec les constructeurs adéquats.
   - Les afficher notamment CalculSalaire ( ).

On dérive la classe Individu déjà vue par une classe Stagiaire. En plus des données membres Individu, la classe Stagiaire a comme données membres privées:

- Filière
- Moyenne générale

5. Construire la classe Stagiaire qui hérite de la classe Individu
6. Au niveau de des Test : Prévoir deux Stagiaires et les instancier avec les constructeurs adéquats
7. On souhaite Enregistrer des objets Stagiaires dans un fichier
8. Écrire des procédures de mise a jour de ce fichier : Ajout, Modification, Suppression
9. Prévoir une procédure qui permet de recopier les enregistrements du fichier vers un Vecteur.
10. afficher à partir du vecteur les Stagiaires ayant une moyenne >= 10

Exercice 9 :

1. Créer la classe Article caractérisée par les attributs : Référence, Désignation, PrixHT, TauxTVA.
   Ces attributs doivent seulement être accessibles par le biais des accesseurs (get / set) en lecture/écriture mis en œuvre par les propriétés.
2. Ajouter les constructeurs suivants :
   - Un constructeur par défaut
   - Un constructeur initialisant tous les attributs.
   - Un Constructeur qui permet de renseigner la référence et la désignation lors de l'instanciation
3. Implémentez la méthode CalculerPrixTTC() ; Cette méthode doit retourne le prix TTC d'un article $PrixHT * (1 + TauxTVA)$
4. Ajouter la méthode AfficherArticle() qui affiche les informations de l'article.
5. Le taux de TVA est en fait commun à tous les articles. Pour éviter toute redondance de cet attribut, vous devriez donc la déclarer comme partagée au niveau de la classe Article et non comme un attribut spécifique des objets instanciés à partir de la classe. Proposer une solution et tester de nouveau.
6. Créer un programme de test où il faut créer des objets (en utilisant les différents constructeurs) et leur calculer le prix TTC.

Exercice 10 :

1. Définir une classe Client avec les attributs suivants : CIN, Nom, Prénom, Tél.
2. Définir à l'aide des propriétés les méthodes d'accès aux différents attributs de la classe.
3. Définir un constructeur permettant d'initialiser tous les attributs.
4. Définir un constructeur permettant d'initialiser le CIN, le nom et le prénom.
5. Définir la méthode Afficher ( ) permettant d'afficher les informations du Client en cours.
6. Créer Une classe Compte caractérisée par son solde et un code qui est incrémenté lors de sa création ainsi que son propriétaire qui représente un client.
7. Définir à l'aide des propriétés les méthodes d'accès aux différents attributs de la classe (le numéro de compte et le solde sont en lecture seule)
8. Définir un constructeur permettant de créer un compte en indiquant son propriétaire.
9. Ajouter à la classe Compte les méthodes suivantes :
   - Une méthode permettant de Crediter() le compte, prenant une somme en paramètre.
   - Une méthode permettant de Crediter() le compte, prenant une somme et un compte en paramètres, créditant le compte et débitant le compte passé en paramètres.
   - Une méthode permettant de Debiter() le compte, prenant une somme en paramètre
   - Une méthode permettant de Débiter() le compte, prenant une somme et un compte bancaire en paramètres, débitant le compte et créditant le compte passé en paramètres
   - Une méthode qui permet d'afficher le résumé d'un compte.
   - Une méthode qui permet d'afficher le nombre des comptes crées.
10. Créer un programme de test pour la classe Compte

Exercice 11 :

1. Définir une classe Employé caractérisée par les attributs : Matricule, Nom, Prénom, DateNaissance, DateEmbauche, Salaire.
2. Définir à l'aide des propriétés les méthodes d'accès aux différents attributs de la classe.
3. Définir un constructeur permettant d'initialiser les attributs de la méthode par des valeurs saisies par l'utilisateur.
4. Ajouter à la classe la méthode Age( ) qui retourne l'âge de l'employé.
5. Ajouter à la classe la méthode Anciennete( ) qui retourne le nombre d'années d'ancienneté de l'employé.
6. Ajouter à la classe la méthode AugmentationDuSalaire( ) qui augmente le salaire de l'employé en prenant en considération l'ancienneté.
   Si Ancienneté < 5 ans, alors on ajoute 2%. - Si Ancienneté < 10 ans, alors on ajoute 5%. - Sinon, on ajoute 10%.
7. Ajouter la méthode AfficherEmployé() qui affiche les informations de l'employé comme suit :

```
 - Matricule : [...]
 - Nom complet : [NOM Prénom]
 - Age : [...]
- Ancienneté : [...]
 - Salaire : [...]
```

   Le nom doit être affiché en majuscule. Pour le prénom, la première lettre doit être en majuscule, les autres en minuscule.

Exercice 12 :

1. Définir une classe Rectangle ayant les attributs suivants : Longueur et Largeur.
2. Définir à l'aide des propriétés les méthodes d'accès aux attributs de la classe.
3. Ajouter un constructeur d'initialisation.
4. Ajouter les méthodes suivantes :
   - Périmètre ( ) : retourne le périmètre du rectangle.
   - Aire( ) : retourne l'aire du rectangle.
   - EstCarre( ) : vérifie si le rectangle est un carré.
   - AfficherRectangle( ) : expose les caractéristiques d'un rectangle comme suit :
     Longueur : [. . .] - Largeur : [. . .] - Périmètre : [. . .] - Aire : [. . .] - Il s'agit d'un carré / Il ne s'agit pas d'un carré