Python pas à pas

Table des matières

I. Cours no 1 : « Premiers pas en Python » 2
II. Cours no 2 : «Contrôle du flux d'instructions » 2
III. Cours no 3 : « Les fonctions » 3
IV. Cours no 4 : « Structures de données Python » 5
V. Cours no 5 : Interlude : nombres parfaits et nombres chanceux 7
VI. Cours no 6 : «Modules et fichiers » 8
VII. Cours no 7 : « Programmation Orientée Objet » 8
VIII. Cours no 8 : «Notions de COO et d'encapsulation » 9

I. Cours n° 1 : « Premiers pas en Python »

1. Affectez les variables temps et distance par les valeurs 6.892 et 19.7.
   - Calculez et affichez la valeur de la vitesse.
   - Améliorez l'affichage en imposant un chiffre après le point décimal.
2. Saisir un nom et un âge en utilisant l'instruction input(). Les afficher.
   - Enfin, utilisez la « bonne pratique » : recommencez l'exercice en *transtypant* les saisies effectuées avec l'instruction input()

II. Cours n° 2 : «Contrôle du flux d'instructions »

1. Saisissez un flottant. S'il est positif ou nul, affichez sa racine, sinon affichez un message d'erreur.
2. L'ordre *lexicographique* est celui du dictionnaire.
   - Saisir deux mots, comparez-les pour trouver le « plus petit » et affichez le résultat.
   - Refaire l'exercice en utilisant l'instruction ternaire :
   - <res> = <a> if <condition> else <b>
3. On désire sécuriser une enceinte pressurisée.
   - On se fixe une pression seuil et un volume seuil : pSeuil = 2.3, vSeuil = 7.41.
   - On demande de saisir la pression et le volume courant de l'enceinte et d'écrire un script qui simule le comportement suivant :
     - Si le volume *et* la pression sont supérieurs aux seuils : arrêt immédiat ;
     - Si seule la pression est supérieure à la pression seuil : demander d'augmenter le volume de l'enceinte ;
     - Si seul le volume est supérieur au volume seuil : demander de diminuer le volume de l'enceinte ;
     - Sinon déclarer que « tout va bien ».
4. Ce comportement sera implémenté par une alternative multiple.
   - Initialisez deux entiers : a = 0 et b = 10.
   - Écrire une boucle affichante et incrémentant la valeur de a tant qu'elle reste inférieure à celle de b.
   - Écrire une autre boucle décrémentant la valeur de b et affichant sa valeur si elle est impaire. Boucler tant que b n'est pas nul.

5. Écrire une *saisie filtrée* d'un entier dans l'intervalle 1 à 10, bornes comprises. Affichez la saisie.
6. Affichez chaque caractère d'une chaîne en utilisant une boucle for.
   Affichez chaque élément d'une liste en utilisant une boucle for.
7. Affichez les entiers de 0 à 15 non compris, de trois en trois, en utilisant une boucle for et l'instruction range().
8. Utilisez l'instruction break pour interrompre une boucle for d'affichage des entiers de 1 à 10 compris, lorsque la variable de boucle vaut 5.
9. Utilisez l'instruction continue pour modifier une boucle for d'affichage de tous en-tiers de 1 à 10 compris, sauf lorsque la variable de boucle vaut 5.
10. Utilisez une *exception* pour calculer, dans une boucle évoluant de -3 à 3 compris, la valeur de sin(*x*)/*x*.
11. La clause else des boucles.Dans cet exercice, effectuez les saisies avec des integerbox et les affichages avec des msgbox, tous deux appartenant au module easygui.
    1. Initialisez une liste avec 5 entiers de votre choix puis saisissez un entier.
    2. Dans une boucle for, parcourez la liste. Si l'entier saisie appartient à la liste, sauvez-le et interrompez la boucle (puisque vous l'avez trouvé). Si la boucle s'est bien terminée,
    3. Utilisez une clause else pour afficher un message l'annonçant.
    4. Entrez maintenant un autre entier, cette fois-ci positif.
    5. Écrivez une boucle while pour déterminer si cet entier est premier. S'il ne l'est pas,la boucle devra afficher le premier diviseur trouvé et s'interrompre. S'il est premier,l'afficher dans une clause else.

III. Cours n° 3 : « Les fonctions »

1. Écrire une procédure table avec quatre paramètres : base, debut, fin et inc.
    1. Cette procédure doit afficher la table des bases, de debut à fin, de inc en inc.
    2. Tester la procédure par un appel dans le programme principal.
2. Écrire une fonction cube qui retourne le cube de son argument.
    - Écrire une fonction volumeSphere qui calcule le volume d'une sphère de rayon r fourni en argument et qui utilise la fonction cube.
    - Tester la fonction volumeSphere par un appel dans le programme principal.

3. Écrire une fonction maFonction qui retourne *f* (*x*) *Æ* 2*x*3*Åx ¡*5.
   - Écrire une procédure tabuler avec quatre paramètres : fonction, borneInf, borneSup et nbPas. Cette procédure affiche les valeurs de fonction, de borneInf à borneSup, tous les nbPas. Elle doit respecter *borneIn f Ç borneSup*.
   - Tester cette procédure par un appel dans le programme principal après avoir saisi les deux bornes dans une floatbox et le nombre de pas dans une integerbox (utilisez le module easyguiB).
4. Écrire une fonction volMasseEllipsoide qui retourne le volume et la masse d'un ellipsoïde grâce à un tuple. Les paramètres sont les trois demi-axes et la masse volumique. On donnera à ces quatre paramètres des valeurs par défaut.
   - On donne : $v = \frac{4}{3}\pi abc$
   - Tester cette fonction par des appels avec différents nombres d'arguments.
5. Écrire une fonction somme avec un argument « tuple de longueur variable » qui calcule la somme des nombres contenus dans le tuple.
   - Tester cette fonction par des appels avec différents tuples d'entiers ou de flottants.
6. Écrire une autre fonction somme avec trois arguments, et qui renvoie leur somme.
   - Dans le programme principal, définir un tuple de trois nombres, puis utilisez la syntaxe d'appel à la fonction qui *décompresse* le tuple. Affichez le résultat.
7. Écrire une fonction unDictionnaire avec un argument « dictionnaire de longueur variable », et qui affiche son argument.
   - Dans le programme principal, définir un dictionnaire, puis utilisez la syntaxe d'appel à la fonction qui *décompresse* le dictionnaire. Affichez le résultat.

IV. Cours n°4 : « Structures de données Python »

1. Définir la liste : liste =[17, 38, 10, 25, 72], puis effectuez les actions suivantes :
   - Triez et affichez la liste ;
   - Ajoutez l'élément 12 à la liste et affichez la liste ;
   - Renversez et affichez la liste ;
   - Affichez l'indice de l'élément 17 ;
   - Enlevez l'élément 38 et affichez la liste ;
   - Affichez la sous-liste du 2eau 3eélément ;
   - Affichez la sous-liste du début au 2eélément ;
   - Affichez le dernier élément en utilisant un indiçage négatif.
   - Affichez la sous-liste du 3eélément à la fin de la liste ;
   - Affichez la sous-liste complète de la liste ;
   - Affichez le dernier élément en utilisant un indiçage négatif.

Bien remarquer que certaines méthodes de liste ne retournent rien.

---

2. Initialisez truc comme une liste vide, et machin comme une liste de cinq flottants nuls.
   - Affichez ces listes.
   - Utilisez la fonction range() pour afficher :
     - Les entiers de 0 à 3 ;
     - Les entiers de 4 à 7 ;
     - Les entiers de 2 à 8 par pas de 2.

   Définir chose comme une liste des entiers de 0 à 5 et testez l'appartenance des éléments 3 et 6 à chose.
3. Utilisez une liste en compréhension pour ajouter 3 à chaque élément d'une liste d'entiers de 0 à 5.
4. Utilisez une liste en compréhension pour ajouter 3 à chaque élément d'une liste d'entiers de 0 à 5, mais seulement si l'élément est supérieur ou égal à 2.
5. Utilisez une liste en compréhension pour obtenir la liste ['ad', 'ae', 'bd', 'be','cd', 'ce'] à partir des chaînes "abc" et "de".
   3. *Indication* : utilisez deux boucles for imbriquées.
6. Utilisez une liste en compréhension pour calculer la somme d'une liste d'entiers de 0 à 9.

7. Définir deux ensembles (*sets*) : *X Æ* {*a*,*b*,*c*,*d*} et *Y Æ* {*s*,*b*,*d*}, puis affichez les résultats suivants:
   - Les ensembles initiaux ;
   - Le test d'appartenance de l'élément 'c' à *X* ;
   - Le test d'appartenance de l'élément 'a' à *Y* ;
   - Les ensembles *X ¡Y* et *Y ¡X* ;
   - L'ensemble *X [Y* (union) ;
   - L'ensemble *X \Y* (intersection).
8. Écrire une fonction compterMots ayant un argument (une chaîne de caractères) er qui renvoie un dictionnaire qui contient la fréquence de tous les mots de la chaîne entrée.
9. Le type dictionnaire (ou tableau associatif) permet de représenter des tableaux structurés. En effet, à chaque clé un dictionnaire associe une valeur, et cette valeur peut elle-même être une structure de donnée (liste, tuple ou un dictionnaire. . .).
   - Soit le tableau suivant représentant des informations physico-chimiques sur des éléments simples (température d'ébullition (T_e) et de fusion (T_f), numéro (Z) et masse (M) atomique :

Au	T_e / T_f	2970	1063
	Z / A	79	196.967
Ga	T_e / T_f	2237	29.8
	Z / A	31	69.72

   - Affectez les données de ce tableau à un dictionnaire dico python de façon à pouvoir écrire par exemple :

```
>>> print dico["Au"]["Z/A"][0] # affiche : 79
```

10. Implémentez une pile LIFO avec une liste.
    - Pour cela, définir trois fonctions :
    - Pile : qui retourne une pile à partir d'une liste variable d'éléments passés en paramètre ;
    - Empile : empile un élément en « haut » de la pile ;
    - Depile : dépile un élément du « haut » de la pile.
11. De la même manière, implémentez une queue FIFO avec une liste. Essayez d'ajouter un menu de manipulation de la queue.

Conseil : *N'utilisez que des procédures sans argument et une liste en variable globale.*

V. Cours n° 5 : Interlude : nombres parfaits et nombres chanceux

Définitions :

- On appelle nombre premier tout entier naturel supérieur à 1 qui possède exactement deux diviseurs, lui-même et l'unité ;
- On appelle diviseur propre de n, un diviseur quelconque de n, n exclu ;
- Un entier naturel est dit parfait s'il est égal à la somme de tous ses diviseurs propres ;
- Les nombres a tels que : (a Ån Ån2) est premier pour tout n tel que 0 6 n Ç (a ¡1), sont appelés nombres chanceux.

1. Écrire un module (parfait_chanceux_m.py) définissant quatre fonctions : somDiv, estParfait,estPremier, estChanceux et un auto-test :
    - La fonction somDiv retourne la somme des diviseurs propres de son argument ;
    - Les trois autres fonctions vérifient la proprriété donnée par leur définition et retourne un booléen. Plus précisément, si par exemple la fonction estPremier vérifie que son argument est premier, elle retourne True, sinon elle retourne False.
2. La partie de test doit comporter quatre appels à la fonction verif permettant de tester somDiv(12), estParfait(6), estPremier(31) et estChanceux(11).
3. Puis écrire le programme principal (parfait_chanceux.py) qui comporte :
    - L'initialisation de deux listes : parfaits et chanceux ;
    - Une boucle de parcours de l'intervalle [2,1000] incluant les tests nécessaires pour remplir ces listes ;
    - Enfin l'affichage de ces listes dans des boîtes de message du module easygui.

VI. Cours n°6 : «Modules et fichiers »

1. Écrire un module de calcul des racines du trinôme réel : ax2Åbx Åc.
   - Le module définit une fonction trinôme avec les trois paramètres du trinôme, a, b et c. La fonction doit retourner un tuple dont le premier élément est le nombre de racines du trinôme (0, 1 ou 2), et les autres éléments sont les racines éventuelles.
   - Testez votre fonction avec les trois jeux de valeurs suivantes : 1,¡3,2, 1,¡2,1 et 1,1,1.
2. Écrire un programme principal utilisant le module précédent.
   - Les trois paramètres seront saisis dans une flotbox du module easyguiB et les résultats seront affichés dans une msgbox.

VII. Cours n°7 : « Programmation Orientée Objet »

1. Définir une classe MaClasse possédant les attributs suivants :
   - Données : deux attributs de classes : x = 23 et y = x + 5.
   - Méthode : une méthode affiche contenant un attribut d'instance z = 42 et les affichages de y et de z.
   - Dans le programme principal, instanciez un objet de la classe MaClasse et invoquez la méthode affiche.
2. Définir une classe Vecteur2D avec un constructeur fournissant les coordonnées par défaut d'un vecteur du plan (par exemple : x = 0 et y = 0).
   - Dans le programme principal, instanciez un Vecteur2D sans paramètre, un Vecteur2D avec ses deux paramètres, et affichez-les.
3. Enrichissez la classe Vecteur2D précédente en lui ajoutant une méthode d'affichage et une méthode de surcharge d'addition de deux vecteurs du plan.
   - Dans le programme principal, instanciez deux Vecteur2D, affichez-les et affichez leur somme.

VIII. Cours n°8 : «Notions de COO et d'encapsulation »

1. Définir une classe Rectangle avec un constructeur donnant des valeurs (longueur et largeur) par défaut et un attribut nom = "rectangle", une méthode d'affichage et une méthode surface renvoyant la surface d'une instance.
   - Définir une classe Carre héritant de Rectangle et qui surcharge l'attribut d'instance : nom = "carré".
   - Dans le programme principal, instanciez un Rectangle et un Carre et affichez-les.
2. Définir une classe Point avec un constructeur fournissant les coordonnées par défaut d'un point du plan (par exemple : x = 0.0 et y = 0.0).
   - Définir une classe Segment dont le constructeur possède quatre paramètres : deux pour l'origine et deux pour l'extrémité. Ce constructeur définit deux attributs : orig et extrem, instances de la classe Point. De cette manière, vous concevez une classe composite : La classe Segment est composée de deux instances de la classe Point.
   - Ajouter une méthode d'affichage.
   - Enfin écrire un auto-test qui affiche une instance de Segment initialisée par les valeurs 1, 2, 3 et 4.
3. Définir une fonction fabrique creer_plus renvoyant une fonction fermeture plus. cree_plus a un argument ajout. Son code ne renferme que la fonction plus qui, elle aussi, possède un argument increment et dont le code se contente de renvoyer la somme : (ajout + increment).
   - Dans le programme principal, créez deux fonctions, par exemple p = creer_plus(23) et p = creer_plus(42), puis affichez les valeurs données par p(100) et q(100).
4. Écriture d'une fonction fabrique renvoyant une instance de classe.
   - Définir une classe CasNormal contenant une méthode uneMethode qui affiche "normal".
   - Définir une classe CasSpecial contenant une méthode uneMethode qui affiche "spécial".
   - Enfin définir la fonction fabrique casQuiConvient avec un paramètre estNormal initialisé par défaut à True. Si le paramètre est vérifié, le corps de la fonction renvoie une instance de la classe CasNormal, sinon il renvoie une instance de la classe CasSpecial.
   - Dans le programme principal, créez l'instance que vous désirez grâce à la fabrique, puis vérifiez son type en appelant dessus la méthode uneMethode.