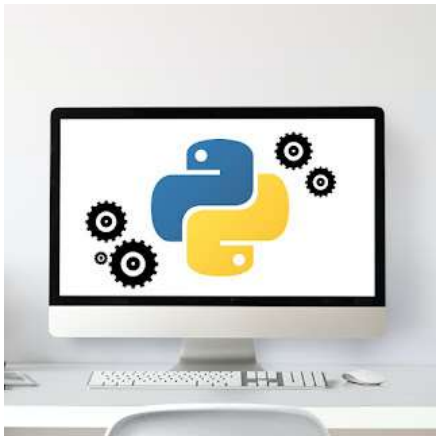
📚 60 Exercices corrigés en Python & POO👍👍



Exercice 1

Écrire un programme Python qui permet d'afficher le message Bonjour.

(Solution)

Exercice 2

Écrire un programme Python permettant de saisir deux nombres et d'afficher leur produit.

(Solution)

Exercice 3

Écrire un programme Python qui permet d'échanger le contenu de deux entiers A et B saisis par l'utilisateur. et afficher ces entiers après l'échange.

(Solution)

Exercice 4

Écrire un un programme Python qui permet d'afficher si un nombre entier saisi au clavier est pair ou impair.

(Solution)

Exercice 5

Écrire un programme Python qui permet d'afficher le plus grand de trois entiers saisis au clavier.

(Solution)

Exercice 6

Écrire un programme Python qui permet d'évaluer une note saisi au clavier (si la note est supérieur à 10 alors il affiche validé sinon non validé (NB : la note comprise entre 0 et 20).

(Solution)

Exercice 7

Écrire un programme Python qui demande deux nombres m et n à l'utilisateur et l'informe ensuite si le produit de ces deux nombres est positif ou négatif. On inclut dans le programme le cas où le produit peut être nul.

(Solution)

Exercice 8

Écrire un programme Python qui permet de calculer la valeur absolue d'un entier saisi par l'utilisateur.

(Solution)

Exercice 9

Écrire un programme Python qui permet de calculer la moyenne de trois entiers saisis par l'utilisateur.

(Solution)

Exercice 10

Une boutique propose à ces clients, une réduction de 15% pour les montants d'achat supérieurs à 200 dh. Écrire un programme Python permettant de saisir le prix total HT et de calculer le montant TTC en prenant en compte la réduction et la TVA=20%.

(Solution)

Exercice 11

Le centre de photocopie facture 0,25 DH pour les 10 premières photocopies, 0,20 DH les vingt suivantes et 0,10 DH pour plus de vingt. Ecrire un programme Python qui demande à l'utilisateur de saisir le nombre de photocopies effectuées et qui affiche la facture correspondante.

(Solution)

Exercice 12

Écrire un programme Python qui demande l'âge d'un enfant et permet d'informer de sa catégorie sachant que les catégories sont les suivantes:
"poussin de 6 a 7 ans"
"pupille de 8 a 9 ans "
"minime de 10 a 11 ans "
" cadet après 12 ans ".

(Solution)

Exercice 13

Écrire un programme Python permettant d'afficher le mois en lettre selon le numéro saisi au clavier. (Si l'utilisateur tape 1 le programme affiche janvier, si 2 affiche février , si 3 affiche mars...).

(Solution)

Exercice 14

Écrire un programme Python qui permet d'afficher le message "Bonsoir" 10 fois. Utilisant la boucle while.

(Solution)

Exercice 15

Écrire un programme Python permettant de calculer la somme S= 1+2+3+...+ 10. Utilisant la boucle while.

(Solution)

Exercice 16

Écrire un programme Python permettant de calculer la somme S=1+2+3+...+ N, où N saisi par l'utilisateur. Utilisant la boucle while.

(Solution)

Exercice 17

Écrire un programme Python qui permet d'afficher le message "bonjour" 10 fois . Utilisant la boucle for.

(Solution)

Exercice 18

Écrire un programme Python qui permet de calculer la somme S=1+2+3+...+ 10. Utilisant la boucle for.

(Solution)

Exercice 19

Écrire un programme Python qui permet de calculer la somme S=1+2+3+4+....+ N. où N saisi au clavier par l'utilisateur.Utilisant la boucle for.

(Solution)

Exercice 20

Écrire un programme Python qui permet d'afficher la table de multiplication de 5. Utilisant la boucle For.

(Solution)

Exercice 21

Écrire un programme Python qui permet d'afficher la table de multiplication d'un entier saisie par l'utilisateur, Utilisant la boucle for.

(Solution)

Exercice 22

Écrivez un programme Python, entrez deux nombres de l'utilisateur et trouvez le plus grand diviseur commun en utilisant la boucle for.

(Solution)

Exercice 23

Écrivez un programme Python pour entrer un nombre et vérifiez si le nombre est parfait ou non.
Un nombre parfait est un entier positif qui est égal à la somme de ses diviseurs positifs appropriés.
Par exemple: 6 est le premier nombre parfait
Les diviseurs appropriés de 6 sont 1, 2, 3.
Somme de ses diviseurs stricts = 1 + 2 + 3 = 6.
Par conséquent, 6 est un nombre parfait.

(Solution)

Exercice 24

Écrivez un programme Python pour saisir un nombre et calculer sa factorielle à l'aide de la boucle for.

La factorielle d'un nombre "n" est le produit de tous les entiers positifs inférieurs ou égaux à n. Il est noté n!.

Par exemple, factorielle de 5!= 1*2*3*4*5= 120.

(Solution)

Exercice 25

Écrivez un programme Python pour afficher tous les nombres impairs de 1 à n en utilisant la boucle for et while.

(Solution)

Exercice 26

Écrivez un programme Python pour entrer un nombre de l'utilisateur et comptez le nombre de chiffres dans l'entier donné en utilisant une boucle.

(Solution)

Exercice 27

Écrivez un programme Python pour saisir un nombre de l'utilisateur et recherchez le premier et le dernier chiffre d'un nombre en utilisant une boucle.

(Solution)

Exercice 28

Écrire un programme Python qui permet d'inverser les chiffres d'un entier N saisi par l'utilisateur. par exemple N=35672 le résultat affiché doit être 27653.

(Solution)

Exercice 29

Écrivez un programme Python pour saisir un nombre et calculer la somme de ses chiffres en utilisant la boucle for.

(Solution)

Exercice 30

Écrivez un programme Python pour saisir le numéro de l'utilisateur et vérifiez que le numéro est palindrome ou non, en utilisant une boucle.

(Solution)

Exercice 31

Écrivez un programme Python pour déclarer et initialiser un tableau, puis saisissez ses éléments à partir de l'utilisateur et affichez le tableau.

(Solution)

Exercice 32

Écrivez un programme Python pour déclarer un tableau, puis saisissez ses éléments par l'utilisateur et affichez tous les éléments négatifs.

(Solution)

Exercice 33

Écrire un programme Python pour déclarer un tableau, puis saisir ses éléments à partir de l'utilisateur et trouver la somme des éléments du tableau.

(Solution)

Exercice 34

Écrire un programme Python pour déclarer un tableau, puis saisir ses éléments à partir de l'utilisateur et rechercher les éléments maximum et minimum dans le tableau.

(Solution)

Exercice 35

Écrivez un programme Python pour déclarer un tableau, puis saisissez ses éléments à partir de l'utilisateur et recherchez l'élément le plus grand et le deuxième dans ce tableau.

(Solution)

Exercice 36

Écrivez un programme Python pour déclarer un tableau, puis saisissez ses éléments à partir de l'utilisateur et comptez le nombre d'éléments pairs et impairs dans ce tableau.

(Solution)

Exercice 37

Écrivez un programme Python pour déclarer deux tableaux, puis entrez les éléments du premier tableau de l'utilisateur et copiez tous ses éléments dans le deuxième tableau.

(Solution)

Exercice 38

Créez un programme Python qui crée et initialise un tableau, puis insère un élément à la position spécifiée dans ce tableau (de 0 à N-1).
Pour insérer un nouvel élément dans le tableau, déplacez les éléments de la position d'insertion donnée vers une position vers la droite.

(Solution)

Exercice 39

Créez un programme Python qui crée et initialise un tableau, puis supprimez un élément de ce tableau à la position spécifiée (de 0 à N-1).
Pour supprimer un élément du tableau, déplacez les éléments juste après la position donnée vers une position à gauche et réduisez la taille du tableau.

(Solution)

Exercice 40

Créez un programme Python qui crée et initialise un tableau, puis trouve la fréquence de chaque élément de ce tableau.

(Solution)

Exercice 41

Créez un programme Python qui crée et initialise un tableau, puis affichez tous les éléments uniques de ce tableau
Idée: utiliser un tableau de fréquences.

(Solution)

Exercice 42

Créez un programme Python qui crée et initialise un tableau, puis comptez les éléments en double dans ce tableau.

(Solution)

Exercice 43

Créez un programme Python qui crée et initialise un tableau, puis supprimez les éléments en double dans ce tableau.

(Solution)

Exercice 44

Créez un programme Python qui crée et initialise un tableau, puis inversez ce tableau sans utiliser un tableau supplémentaire.

(Solution)

Exercice 45

Ecrire un programme Python qui permet de calculer la somme :

$S = 1^1 + 2^2 + 3^3 + ... + n^n$

***** Solution ******

```
S = 0

n=int(input("Donner un entier :"))

for i in range (1,n+1):

  S=S+i**i

print("la somme est :",S)
```

POO en Python

Exercice 46 : classe Rectangle

Écrire en Python une classe «Rectangle» ayant deux variables « a » et « b » et une fonction membre « surface() » qui retournera la surface du rectangle.

Corrigé

```
class Rectangle:
  def __init__ (self,a=0,b=0):
     self.a=a
     self.b=b
  def surface(self):
    return self.a*self.b

r1 = Rectangle ();
r2 = Rectangle (5,4);
print("la surface est:",r1.surface());
print("la surface est:",r2.surface());
```

Exercice 47 : classe Somme

Écrire en Python une classe « Somme » ayant deux variables « n1 » et « n2 » et une fonction membre « som() » qui calcule la somme. Dans la méthode principale main demandez à l'utilisateur d'entrez deux entiers et passez-les au constructeur par défaut de la classe « Somme » et afficher le résultat de l'addition des deux nombres.

Corrigé

```
class Somme:
   def __init__(self,nbr1=0,nbr2=0):
      self.n1=nbr1
      self.n2=nbr2

   def som(self):
     return n1 + n2

n1 = int(input("Entrer N1:"))
n2 = int(input("Entrer N1:"))
obj = Somme(n1,n2);
print("Le resultat de l'addition est :",obj.som())
```

Exercice 48 : classe Etudiant

Écrire classe Python appelée « Etudiant » avec les membres suivant :
nom : (de type char),
note1, note2 : (de type float)
calc_moy() : calcule la note moyenne.
afficher () : affiche le nom et la note moyenne.
Le programme principal (main) demande à l'utilisateur d'entrer le nom et les notes d'un étudiant. et affiche leur nom et la note moyenne.

Corrigé

```
class Etudiant:
  def __init__(self,nom,note1,note2):
    self.nom = nom
    self.note1 = note1
    self.note2 = note2

  def calc_moy(self):
   return (self.note1 + self.note2)/2

  def afficher (self):
   print("Etudiant: ",self.nom, " moyenne: ",self.calc_moy())

nom = input("Entrer le nom: ")
note1= int(input("Entrer la note 1: "))
note2= int(input("Entrer la note 2: "))
E = Etudiant (nom, note1, note2)
E.afficher()
```

Exercice 49 : classe point

Réaliser en Python une classe point permettant de manipuler un point d'un plan.on prévoira :

1) un point est définit par ses coordonnées x et y (des membres privés)

2) les constructeurs

3) une fonction membre déplace effectuant une translation définie par ses deux arguments dx et dy (double)

4)une fonction membre affiche se contentant d'afficher les coordonnées cartésiennes du point.

5)une fonction membre saisir se contentant de saisir les coordonnées cartésiennes du point.

6)une fonction membre distance effectuant calculant la distance entre deux point.

7)une fonction membre milieu donnant le milieu d'un segment.

8)un petit programme d'essai (main) gérant la classe point.

Corrigé

```
from math import *

class Point:
  def __init__(self,a=0,b=0):
    self.x=a
    self.y=b
  def get_x(self):
    return self.x
  def get_y(self):
    return self.y

  def set_x(self,a):
    self.x=a
```

```
  def set_y(self, b):
    self.y=b

  def deplace(self,dx,dy):
    self.set_x(self.get_x()+dx)
    self.set_y(self.get_y()+dy)


  def affiche(self):
    print("x=",self.get_x())
    print("y=",self.get_y())

  def saisir(self):
    print("donner les coordonnées")
    self.x = int(input ("x = "))
    self.y = int(input ("y = "))


  def distance (self,p):
    x1=(self.get_x()-p.get_x())*(self.get_x()-p.get_x());
    x2=(self.get_y()-p.get_y())*(self.get_y()-p.get_y());
    d=sqrt(x1+x2)
    return d

  def milieu(self, p):
    p1 = Point();
    p1.x=(self.get_x()+p.get_x())/2
    p1.y=(self.get_y()+p.get_y())/2
    return p1

p = Point(1,1)
x = Point(5,5)
c = Point()
p.affiche()
p.deplace(5,5)
p.affiche();
print("la distance px est: ",p.distance(x));
c=p.milieu(x)
print("le milieu de [px] est: (",c.get_x(),",",c.get_y(),")")
```

Exercice 50 : classe Compte

Écrire un programme en Python qui simule la gestion d'un simple compte bancaire. Le compte est créé avec un solde initial. Il est possible de déposer et de retirer des fonds, d'ajouter des intérêts et de connaître le solde actuel. Cela devrait être implémenté dans une classe nommée Account qui comprend:

1) Un constructeur par défaut qui met le solde initial à zéro.
2) Un constructeur qui accepte une balance initial comme paramètre.
3) Une fonction getBalance qui renvoie le solde actuel.
4) Une méthode deposer pour déposer un montant spécifié.
5) Une méthode retirer pour retirer un montant spécifié.
6) Une méthode ajouter_Interet pour ajouter de l'intérêt au compte.
La méthode ajouter_Interet prend le taux d'intérêt comme paramètre et modifie le solde du compte en solde * (1 + taux d'intérêt).

Corrigé

```
class Compte:
  def __init__(self,balance=0):
    self.balance = balance

  def getBalance(self):
    return self.balance

  def deposer(self,amount):
    self.balance += amount
```

```
  def retirer (self,amount):
    self.balance -= amount

  def ajouter_Interet (self,rate):
    self.balance = self.balance*(1 + rate)

compte1 = Compte()
compte2 = Compte(3000)
compte1.deposer(100)
compte2.retirer(1000)
compte1.ajouter_Interet (0.3)
print(compte1.getBalance())
print(compte2.getBalance())
```

Exercice 51 : Classe temps

Créer en Python une classe appelée Temps, qui a des membres de type int tels que heures, minutes et secondes.(rendez-les private)

1) Un constructeur doit initialiser ces données à 0

2) Un autre constructeur devrait l'initialiser à des valeurs fixes.

3) Une fonction membre devrait l'afficher, au format 17h 59min 59s.

4) Une autre fonction pour renvoyer les données de chaque membre nommez-les getheurs, getMin et getSec

5) Une fonction membre doit ajouter deux objets de type Temps passé en arguments.

Corrigé

```
class Temps:
  def setTemps(self,h,m,s):
    self.heures = h
    self.minutes = m
    self.seconds = s

  def __init__ (self,h=0,m=0,s=0):
    self.setTemps(h,m,s)

  def getHours(self):
    return self.heures
  def getMin(self) :
    return self.minutes
  def getSec(self) :
    return self.seconds

  def getTemps(self):
    print(self.heures,"h ",self.minutes,"min ",self.seconds,"s")


  def ajouterTemps(self,t1,t2):
    self.seconds= t1.seconds + t2.seconds
    self.minutes = t1.minutes + t2.minutes + (int(self.seconds/60))
    self.heures = t1.heures + t2.heures + (int(self.minutes/60))
    self.minutes %= 60
    self.seconds %= 60

t1 = Temps(4,43,59)
t2 = Temps(1,20,32)
t3 = Temps()
t1.getTemps()
t2.getTemps()
t3.ajouterTemps(t1, t2)
t3.getTemps()
```

Exercice 52 : Classe rectangle

Écrire en Python un programme utilisant une classe rectangle dont le constructeur prend deux paramètres, largeur et hauteur et qui offre les fonctions suivantes :

1) calcul du périmètre

2) calcul de la surface

3) affichage

ainsi que les accesseurs et mutateurs triviaux (lecture et modification de la largeur et de la hauteur).

Corrigé

```
class Rectangle:
  def __init__(self,L,h):
    self.largeur = L
    self.hauteur = h
  def getLargeur(self):
    return largeur
  def getHauteur(self):
    return hauteur
  def perimetre(self) :
    return 2*(self.largeur + self.hauteur)
  def surface(self):
    return self.largeur * self.hauteur
  def setLargeur(self,newLargeur):
    self.largeur = newLargeur
  def setHauteur(self,newHauteur):
    self.hauteur = newHauteur

  def afficher(self):
    print("la longueur :",self.hauteur)
    print("la largeur :",self.largeur)

R= Rectangle (5,4)
R.afficher()
print("la surface est ",R.surface());
print("le perimetre est ",R.perimetre())
```

Exercice 53 : L'héritage

Écrivez un programme en Python qui définit une classe appelée Forme avec un constructeur qui donne de la valeur à la largeur(x) et à la hauteur(y). Définir la méthode aire() dans les deux sous-classes Triangle et Rectangle, qui calculent l'aire. Dans la méthode principale main, définissez deux variables, un triangle et un rectangle, puis appelez la fonction aire() dans ces deux variables.

Notez que:

l'aire du triangle est = largeur * hauteur / 2

l'aire du rectangle est = largeur * hauteur.

Corrigé

```
class Forme:
  def __init__(self,x=0,y=0):
    self.x =  x
    self.y =  y

class Rectangle (Forme):
  def __init__(self,x=0,y=0):
    Forme.__init__(self,x,y)

  def aire(self):
    return (self.x * self.y)
```

```
class Triangle (Forme):
  def __init__(self,x=0,y=0):
     Forme.__init__(self,x,y)
  def aire(self):
    return (self.x * self.y / 2)


R = Rectangle (2,3)
T = Triangle (2,3)
R1 = Rectangle ()
print(R.aire())
print(T.aire())
print(R1.aire())
```

Exercice 54: Classe Rectangle

1) Ecrire une classe Rectangle en langage Python, permettant de construire un rectangle dotée d'attributs longueur et largeur.
2) Créer une méthode Perimetre() 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 Parallelepipede 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.

Réponse

```
#coding: utf-8
class Rectangle:
  def __init__(self,longueur,largeur):
    self.longueur = longueur
    self.largeur = largeur

  # Méthode qui calcul le périmètre
  def Perimetre(self):
    return 2*(self.longueur + self.largeur)

  # Méthode qui calcul la surface
  def Surface(self):
    return self.longueur*self.largeur

class Parallelepipede(Rectangle):
  def __init__(self,longueur,largeur, hauteur):
    Rectangle.__init__(self,longueur,largeur)
    self.hauteur = hauteur

  # méthode qui calcul le volume
  def Volume(self):
    return self.longueur*self.largeur*self.hauteur

monRectangle = Rectangle(7, 5)
monParallelepipede = Parallelepipede(7,5,2)
print("Le périmètre de mon rectangle est : ",monRectangle.Perimetre())
print("La surface de mon rectangle est : ", monRectangle.Surface())
print("Le volume de mon parallelepipede est : ", monParallelepipede.Volume())
```

Exercice 55: Compte bancaire

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 % du solde

6) Créer une méthode afficher() permettant d'afficher les détails sur le compte
7) Donner le code complet de la classe CompteBancaire.

Réponse

```
#coding: utf-8
class CompteBancaire:
  def __init__(self, idNumber, nomPrenom, solde):
    self.idNumber = idNumber
    self.nomPrenom = nomPrenom
    self.solde = solde

  def versement(self, argent):
    self.solde = self.solde + argent

  def retrait(self, argent):
    if(self.solde < argent):
      print(" Impossible d'effectuer l'opération. Solde insuffisant !")
    else:
      self.solde = self.solde - argent

  def agios(self):
    self.solde =self.solde*95/100

  def afficher(self):
    print("Compte numéro : " , self.idNumber)
    print("Nom & Prénom : ", self.nomPrenom)
    print(" Solde  : ", self.solde , " DH ")


monCompte = CompteBancaire(16168891, " Mohamed Rachid", 22300)
monCompte.versement(1500)
monCompte.retrait(24000)
#monCompte.agios()
monCompte.afficher()
```

Exercice 56: Classe Cercle

1) Définir une classe Cercle permettant de créer un cercle C(O,r) de centre O(a,b) et de rayon r à l'aide du constructeur :
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).

Réponse

```
#coding: utf-8
from math import *
class Cercle:
  def __init__(self , a , b , r):
    self.a = a
    self.b = b
    self.r = r

  def perimetre(self):
    return 2*pi*self.r


  def surface(self):
    return pi*self.r**2

  def formEquation(self,x,y):
```

```
    return (x-self.a)**2 + (y-self.b)**2 -self.r**2
  def test_appartenance(self,x,y):
    if(self.formEquation(x,y)==0):
      print("le point : (",x,y,") appartient au cercle C")
    else:
      print("le point : (",x,y,") n'appartient pas au cercle C")


# Instanciation
C = Cercle(1,2,1)

print("le périmètre du cercle C est: ", C.perimetre())
print("le surface du cercle C est: ", C.surface())
C.test_appartenance(1,1)
```

Exercice 57: Calcul arithmétique

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 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.

Réponse

```
#coding: utf-8
class Calcul:
  def __init__(self):
    pass
#---Factorielle ------------
  def factorielle(self, n):
    j=1
    for i in range(1,n+1):
      j = j*i
    return j
#---Somme des n premiers nombres----
  def somme(self, n):
    j=1
    for i in range(1,n+1):
      j = j+i
    return j
#---Test primalité d'un nombre------------
  def testPrim(self, n):
    j=0
    for i in range(1,n+1):
      if(n%i==0):
        j = j + 1
    if(j == 2):
      return True
    else:
      return False

# ---Test primalité de deux nombres entiers------------
  def testprims(self , n , m):
```

```
    divCommun = 0
    for i in range(1 , n+1):
      if (n%i == 0 and m%i == 0):
        divCommun = divCommun + 1
    if divCommun == 1:
      print("Les nombres " , n , " et ", m , " sont premiers entre eux")
    else:
      print("Les nombres " , n , " et ", m , " ne sont pas premiers entre eux")

#---Table de multiplication----------
  def tableMult(self,k):
    for i in range(1,10):
      print(i," x ",k," = ",i*k)

#---Toutes les tables de multiplication des nombres 1, 2, .., 9
  def toutesLesTables(self):
    for k in range(1,10):
      print("\nla table de multiplication de : ",k, " est : ")
      for i in range(1,10):
        print(i," x ",k," = ",i*k)

#----- liste des diviseurs d'un entier
  def listDiv(self , n):
    # initialisation de la liste des diviseurs
    lDiv = []
    for i in range(1 , n+1):
      if ( n%i == 0):
        lDiv.append(i)
    return lDiv

# ------liste des diviseurs premiers d'un entier----------------
  def listDivPrim(self , n):
    # initialisation de la liste des diviseurs
    lDiv = []
    for i in range(1 , n+1):
      if ( n%i == 0 and self.testPrim(i)):
        lDiv.append(i)
    return lDiv

# Exemple Instanciation
Cal = Calcul()
Cal.testprims(13 , 7)
print("Liste des diviseurs de 18 : ", Cal.listDiv(18))
print("Liste des diviseurs premiers de 18 : ", Cal.listDivPrim(18))
Cal.toutesLesTables()
```

Exercice 58

Coder une classe myString permettant de doter les chaines de caractères des méthodes append() et pop() faisant les mêmes opérations que celles des listes. Exemple si on crée des chaines via l'instanciation s1 = myString("Hello") et s2 = "bonjour", et on lui applique les méthodes :
print(s1.append(" world !")) # affiche 'Hello world !'
print(s2.pop(2)) # affiche 'bojour'.

Réponse

```
class myString:
  def __init__(self,s):
    self.s = s
  def append(self,x):
    self.s = self.s + x
    return self.s

  def pop(self,i):
    s1 = self.s[0:i]
    s2 = self.s[i+1:len(self.s)]
```

```
        return s1+s2
    def modifier(self,i):
        pass

# Tester la classe
S = myString("hello")
print(S.pop(1)) # affiche 'hllo'
print(S.append(" world !")) # affiche 'hello world !'
```

Exercice 59

1. Définir une classe Book avec les attributs suivants : Titre, Auteur (Nom complet), Prix.
2. Définir un constructeur ayant comme attributs: Titre, Auteur, Prix.
3. Définir la méthode View() pour afficher les informations d'une instance object Book.
4. Ecrire un programme pour tester la classe Book.

Réponse

```
#coding: utf-8
# Question 1
class Book:
    # Question 2
    def __init__(self , Title , Author , Price):
        self.Title   = Title
        self.Author  = Author
        self.Price   = Price

    # Question 3
    def view(self ):
        return ("Book Title: " , self.Title ,  "Book Author: " , self.Author, "Book Price: "
, self.Price)

# Question 4
MyBook = Book("Python" , "Mohamed" , "23 Dh")
print( MyBook.view())
```

Exercise 60: Classe Geometry

Ecrire une classe Python nommée Geometry avec un constructeur par défaut sans paramètres.
1) Ajouter une méthode nommée distance() à la classe geometry qui permet de calculer la distance entre deux points
A = (a1, a2), B = (b1, b2) (avec la convention: un point est est identifié à ses coordonnées M = (x_M , y_M))
2) Ajouter une méthode nommée middle() à la classe geometry qui permet de déterminer le milieu d'un bipoint (A , B).
3) Ajouter une méthode nommée trianglePerimeter() à la classe geometry qui permet de calculer le périmètre d'un triangle ABC.
4) Ajouter une méthode nommée triangleIsoscel() qui renvoie True si le triangle est isoscel et False sinon.

ALGORITHME

📚 Algorithme, C, C#, C++, JAVA, Python, SQL,UML👍

March 14, 2023

ALGORITHME

📚 50 Exercices corrigés et bien détaillés en Java et POO👍

February 15, 2023

ALGORITHME

📚 60 Exercices corrigés en Python & POO👍👍

December 29, 2022