Université Hassan 1er
École Nationale des Sciences Appliquées – Berrechid

MANUEL DES EXERCICES CORRIGÉS ET ANNALS DES EXAMENS

PROGRAMMATION PYTHON

Professeur : MOUMOUN Lahcen

Table des matières

Introduction

Python est un langage de programmation de haut niveau, interprété et polyvalent. Il est largement utilisé en raison de sa syntaxe simple, claire et proche du langage humain, ce qui le rend accessible aussi bien pour les débutants que pour les développeurs expérimentés. Grâce à son vaste écosystème de bibliothèques et de frameworks, Python permet de créer des solutions rapides et efficaces dans divers secteurs industriels. Aujourd'hui, ce langage est au cœur des avancées technologiques modernes, y compris dans des domaines comme l'Intelligence Artificielle, l'analyse des données et l'informatique en nuage.

Ce manuel d'exercices corrigés et d'annales d'examens de la programmation Python a été conçu pour accompagner les étudiants dans leur préparation académique. Il leur offre une sélection de sujets variés, inspirés des examens passés, ainsi que des éléments de corrections pour les guider dans leur apprentissage et leur permettre de s'entraîner efficacement, en leur fournissant non seulement des exercices pratiques, mais aussi des corrigés complets. Chaque exercice a été soigneusement sélectionné pour couvrir les points essentiels du programme et assurer une progression adaptée de renforcement des compétences.

Les annales d'examens permettent de se familiariser avec le format des épreuves et de tester des connaissances dans des conditions réelles. En s'exerçant sur ces sujets, les étudiants pourront non seulement mieux comprendre les attentes des examens, mais aussi développer des stratégies pour gérer leur temps et améliorer leur performance.

Chapitre 1 : Eléments de base du langage Python

Exercice 1.1

Énoncé

Une liste l = [x1, x2, x3, x4, . . . , xn] est dite alternée si :

(1) chaque élément est du signe contraire de celui qui le précédé,

(2) les éléments de rang (d'indice) pair sont égaux à eux-mêmes

(3) les éléments de rang (d'indice) impair sont égaux à leur opposé.

Par exemple, la liste l = [0, −1, 2, −3, 4, −5] est une liste alternee.

Ecrire la fonction isAlternate(l) qui retourne True si la liste l passée en argument est alternee, et False sinon.

Élément de correction

```
def isAlternate(l):
      for i in range(len(l)):
          if i%2==0 and l[i]!=i:
              return False
          if i%2!=0 and l[i]!=-i:
              return False
      return True

  l= [0, -1, 2, -3, -4, -5]
  print( isAlternate(l))
```

Exercice 1.2

Énoncé

Soit a un entier strictement supérieur à 1. La suite $(U_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} U_0 = a \\ U_{n+1} = \frac{1}{2}\left(U_n + \frac{a}{U_n}\right) \end{cases}$$

S'appelle la suite de Héron (d'Alexandrie). Cette suite converge très rapidement vers $\sqrt{a}$.

Ecrire une fonction alternative puis récursive qui retourne la racine carrée d'un réel (reçu en paramètre) en se basant sur la suite de Héron.

Élément de correction

```
from math import sqrt as racine
   def heron(a):
        u=a
        u2= 1 / 2*(u + a / u)
        while ( abs(u2-u) > 0.01 ):
            u=u2
            u2=1 / 2*(u + a / u)
        return u2
   a= float(input ("Donner un réel :"))
   print ("Racine reel de %.2f est %f" %(a, racine(a)))
   print("Racine Heron de %.2f est %f" %(a, heron(a)))
```

Exercice 1.3

Énoncé

Dans cet exercice nous rappelons les notions suivantes :

- 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 n tels que : (n + a + a2) est premier pour tout a tel que $0 \leq a < (n - 1)$, sont appelés nombres chanceux.

1. Écrire un module définissant quatre fonctions : somDiv, estParfait, estPremier et estChanceux:

- 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. Ecrire un programme principal 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

Élément de correction

```
def somDiv(n):
        som=1
        for i in range(2, n//2+1):
            if n%i == 0: som+=i
        return som
    def estParfait(n): return n==somDiv(n)
    def estPremier(n): return somDiv(n)==1
    def estChanceux(n,a):
        if n >= a-1:
            return False
        if estPremier(a + n + n*n ):
            return True
        return False
    def estChanceux(n):
        for a in range(0, n-1):
            if not estPremier(n + a + a**2):
                return False
        return True
     parfaits, chanceux = [], list()
    intervalle = range(2, 1001)
    for i in intervalle:
        if estParfait(i):
            parfaits.append(i)
        if estChanceux(i):
            chanceux.append(i)
    msg=" Nombres remarquables dans[2..1000]".center(70,'-')
    msg+="\n\nParfaits:\t{}\n\nChanceux:\t{}".format(parfaits,
                                chanceux)
    print(msg)
```

Exercice 1.4

Énoncé

On appelle couple de nombres premiers jumeaux toute liste [p,q] telle que p,q sont des nombres premiers vérifiant p < q et q = p + 2. Par exemple, [3,5] ou [11,13] sont des couples de nombres premiers jumeaux alors que [2,3] ne l'est pas.

1. Ecrire une fonction python nommée jumeau, prenant comme argument un entier N et renvoyant le couple [p,q] de nombres premiers jumeaux tels que p strictement supérieur à N et le plus petit possible. Par exemple jumeau(5) renvoie comme valeur [11,13].
2. Ecrire avec les mêmes consignes une fonction lesJumeaux prenant en argument un entier N et renvoyant la liste de tous les couples de nombres premiers jumeaux [p,q] tels que q ≤ N. Par exemple, lesJumeaux(18) retourne [[3,5],[5,7],[11,13]] (le couple [17,19] n'en fait pas partie).

Élément de correction

```
def isPremier(n):
        for i in range(2, n // 2 + 1):
            if n % i== 0 : return False
        return True
    def jumeau(n):
        nb=n
        while True :
            nb=nb+1
            if isPremier(nb) and isPremier(nb+2):return [nb,
    nb+2]
    def lesJumeaux(n):
        lst = list()
        for nb in range(3, n-1):
            if isPremier(nb) and isPremier(nb + 2):
                l = list()
                l.append(nb)
                l.append(nb + 2)
                lst.append(l) #lst.append([nb,nb+2])
        return lst
    print(lesJumeaux(4))
```

Exercice 1.5

Énoncé

On souhaite créer un jeu de dés sur ordinateur. Le jeu se joue à deux joueurs, le gagnant étant celui qui le premier atteint 50 points. Un tour se déroule de la façon suivante:

- Un joueur lance le dé et rejoue tant qu'il n'obtient pas de '1'. A chaque lancé:
  - S'il obtient un chiffre pair (2, 4, 6), il augmente ses points du chiffre obtenu
  - S'il obtient un '3', ses points sont multipliés par 2
  - S'il obtient un '5', il perd deux points.
- Il perd la main quand il obtient un '1'.

Ecrire le programme complet simulant ce jeu.

Élément de correction

```
from random import randint as rd
    def lanceDe(nbPt):
        chif=rd(1,6)
        print("    DE : ", chif)
        if chif % 2==0 : return nbPt+chif
        if chif==3: return nbPt * 2
        if chif == 5 :
            if nbPt > 2 : return nbPt-2
```

```
            else: return 0
        return -1
    def jouer(nbptJoueur):
        arret=False
        nbessai=0
        while not arret and nbessai<3 and nbptJoueur < 50 :
            nbessai=nbessai+1
            input("  Appuyer sur entrée pour lancer le dé" )
            nbPt=lanceDe(nbptJoueur)
            if nbPt==-1: arret=True
            else: nbptJoueur=nbPt
        return nbptJoueur
    joueurs= dict()
    for i in range (1,3):
        joueurs[input("Donner le nom du joueur n° %d :"%i)]=0
    arret=False
    while not arret:
        for joueur in joueurs.keys():
            print ("Tour du  %s" %joueur)
            print ("======================================")
            joueurs[joueur]=jouer(joueurs[joueur])
            print(" nb points de %s est %d"%(joueur,joueurs[joueur]))
            if joueurs[joueur] >= 50:
                arret=True ;  break
    print("\nRésultat\n======================================")
    for joueur in joueurs.keys():
        if joueurs[joueur] >= 50 :
            print ("le joueur gagnant est %s" % joueur)
            break
```

Exercice 1.6

La formule de Zeller permet de déterminer le jour de la semaine correspondant à une date donnée. On l'obtient grâce à l'expression suivante, où la notation $\lfloor x \rfloor$ désigne le plancher de x (ou simplement partie entière de x), c'est-à-dire le plus grand entier inférieur ou égal à x (par exemple : $\lfloor 15.8 \rfloor = 15$:

$$\text{jour} = \left(j + \left\lfloor \frac{26(m+1)}{10} \right\rfloor + a + \left\lfloor \frac{a}{4} \right\rfloor + \left\lfloor \frac{s}{4} \right\rfloor + 5s \right) \%7$$

Où :

- **jour** est le numéro du jour (0 = samedi, 1 = dimanche, ..., 6 = vendredi);
- **j** est le jour du mois (entre 1 et 31) et **m** est le numéro du mois (3 = mars, 4 = avril, ..., 12 = décembre, 13 = janvier, 14 =février) ;
- **s** est le siècle (par exemple 19 si l'année est 1987) et **a** est l'année dans le siècle (par exemple 87 si l'année est 1987).

1. Ecrivez un fonction python qui reçoit une date et qui retourne, en utilisant la formule de Zeller, le jour de la semaine correspondant à cette date ;
2. En utilisant la fonction précédente, écrire un programme python qui réalise les tâches suivantes
   - Saisir une date « **strDate** » sous forme d'une chaine de caractère 'dd-mm-yyyy'
   - Tester si le jour et le mois de la date « strDate » sont valides (utiliser une fonction « nbJour(mois, annee) » qui retourne le nombre de jours correspondant au mois et l'année reçu en paramètre).
   - Affichage du jour de la semaine (lundi, mardi, … samedi ou dimanche) correspondant à la date « strDate ».

Exercice 1.7

On considère dans cet exercice le **jeu de Marienbad** – appelé également « jeu des allumettes » – qui nécessite deux joueurs et 21 allumettes. Les 21 allumettes sont réparties en 6 tas, avec i allumettes dans le $i^{ème}$ tas : une allumette dans le premier tas, deux dans le deuxième, etc.

Chacun à son tour, les joueurs piochent dans un seul tas (on se limite au tas non vide) un nombre d'allumette souhaité (le choix du tas et celui du nombre d'allumette sont fait par le joueur encours). Le joueur qui prend la dernière allumette perd la partie.

Le programme à réaliser devra faire jouer deux joueurs. Au cours du jeu, l'état du jeu est affiché simplement sous la forme suivante (si ENSAB est l'un des deux joueurs)

```
Tas        : (1, 2, 3, 4, 5, 6)
Allumettes : [0, 2, 3, 2, 4, 4]
-------- Prochain joueur : ENSAB --------
```

Ecrire un programme python (avec utilisation de fonctions) permettant de réaliser les tâches suivantes :

- Saisir du nom des deux joueurs ;
- Lancer le jeu (c'est une boucle tant que le jeu n'est pas terminé) ;
- À chaque tour vous devez :
  - Afficher l'état du jeu,
  - Demander au joueur en cours le numéro du tas t et le nombre d'allumettes n qu'il désire ôter,

- Vérifier si son choix est valide, sinon expliquer l'erreur et refaire le choix,
- Supprimer n allumettes du tas t,
- Vérifier si le jeu est terminé,

▪ Annoncer le gagnant et le perdant.

Exercice 1.8

Introsort est un algorithme hybride de tri qui combine les avantages du tri rapide (quicksort) et du tri par fusion (merge sort) et du tri par insertion. Il a été conçu pour fournir un tri efficace avec une bonne performance en moyenne tout en évitant les scénarios de complexité temporelle de pire cas auxquels le tri rapide peut être confronté. Introsort réalise cela en basculant entre différents algorithmes de tri en fonction de certaines conditions, en particulier de la profondeur de récursion. Voici un bref aperçu du fonctionnement de l'Introsort:

1. Étape de tri rapide : Initialement, l'Introsort commence par le tri rapide parce que le tri rapide a une excellente performance en moyenne. Il choisit un élément pivot et partitionne le tableau en de plus petits sous-tableaux.
2. Vérification de la profondeur de récursion : Pendant l'étape de tri rapide, l'Introsort suit la profondeur de récursion. Si la profondeur de récursion devient trop grande, il bascule vers un autre algorithme de tri pour éviter les scénarios de pire cas.
3. Etape de tri fusion ou de tri par insertion : Lorsque la profondeur de récursion dépasse un certain seuil, généralement basé sur le logarithme de la taille du tableau, l'Introsort passe soit au tri fusion (merge sort) soit au tri par insertion. Le tri fusion a une complexité temporelle de pire cas garantie de log n), tandis que le tri par insertion est utilisé pour les petits sous-tableaux.
4. Étape finale : Après le tri effectué avec l'algorithme de tri choisi (quicksort, tri fusion ou tri par insertion), l'Introsort renvoie le tableau trié.

En combinant le tri rapide pour sa performance en moyenne avec le tri fusion et le tri par insertion en tant que mécanismes de secours, l'Introsort offre un tri efficace tout en veillant à ce que la complexité temporelle de pire cas ne devienne pas un problème.

L'estimation de la taille des petits sous-tableaux (ou la profondeur de récursion à laquelle Introsort bascule vers le tri par insertion) est un paramètre important de l'algorithme. Voici in exemple à suivre :

a. Seuil pour basculer du tri rapide au tri fusion (Mergesort) : Vous pouvez utiliser une valeur de profondeur de récursion, généralement basée sur le logarithme en base 2 de la taille du tableau (log2(n)), comme seuil pour basculer du tri rapide au tri fusion. Par exemple, si vous avez un tableau de 1 000 éléments, le seuil serait log2(1000) z 10. Donc, lorsque la profondeur de la récursion atteint 10, vous basculeriez vers le tri fusion.

b. Seuil pour basculer du tri fusion au tri par insertion : Le seuil pour basculer du tri fusion au tri par insertion dépend de la taille des sous-tableaux de tri fusion. Par exemple, si la taille des soustableaux de tri fusion est inférieure à 16, vous pourriez basculer vers le tri par insertion.

1. **Programmer une fonction** qui effectue le tri Introsort d'une liste d'entiers reçue en paramètre.
2. **Ecrire une fonction récursive** qui reçoit un entier x et une liste d'entiers I puis effectue une recherche séquentielle de x dans L, la fonction retourne le plus grand indice i tel que L[i]= x.
3. **Ecrire une fonction itérative** qui reçoit un entier x et une liste d'entiers L puis retourne, en effectuant une recherche dichotomique, le plus grand indice i tel que L[i]=x.
4. En aimerait savoir qu'il est le nième plus petit élément dans une liste d'entiers. Une première solution sera de trier la liste puis de retourner l'élément désiré. Une deuxième démarche retourne le nième élément sont effectuer le tri, elle est appelée « Quick Select ». On considère que le nième élément est celui d'indice n dans la liste triée. **Ecrire une fonction récursive** qui reçoit une liste d'entiers L et un entier n, puis retourne le nième plus petit entier de L sans effectuer le tri de L. Une première piste à suivre sera de choisir le premier élément de la liste L comme un pivot, puis partitionner L en deux groupes : la sous liste inf qui contient les éléments inférieurs au pivot et la sous liste sup qui contient les éléments supérieurs ou égaux au pivot.

Chapitre 2 : La programmation objet en Python

Exercice 2.1

Énoncé

Une pile est une structure de données qui permet de stocker des objets et de les retirer en mode dit LIFO : Last In First Out (resp. FIFO First in First Out). On s'intéresse ici à la programmation de pile à capacité bornée. Une pile offre les méthodes suivantes :

- push(obj) permet d'empiler l'élément obj si la pile n'est pas pleine.
- pop() permet de retourner l'élément en tête de la pile. Cette méthode lève une exception si la pile est vide.

1. Proposer une classe Pile qui modélise une pile bornée.
2. Ecrire un programme de la classe Pile.

Élément de correction

```
Class Pile(object):
   def __init__(self,size):
          assert isinstance(size,int) and size>0
          self.size=size
          self.pointer=size
          self.tab=[0]_size
   def empty(self):
          return(self.pointer==self.size)
   def full(self):
          return(self.pointer==0)
   def push(self,obj):
          if not self.full():
                self.pointer=self.pointer - 1
                self.tab[self.pointer]=obj
          else:
                raise Exception
   def pop(self):
          if not self.empty():
                res=self.tab[self.pointer]
                self.pointer=self.pointer+1
                return res
          else:
                raise Exception
```

Exercice 2.2

Énoncé

On souhaite développer un programme de gestion de salaires des employées d'une entreprise. Un employé est caractérisé par son prénom, nom et âge. Nous distinguons entre deux catégories d'employés : les commerciaux et les techniciens. Parmi les commerciaux on distingue entre les vendeurs et les représentants. Les techniciens peuvent être des techniciens de production ou de manutention. On considère que, le salaire minimum mensuel (SMIC) est de 1480,27 Dh et que le SMIC journalier est fixé à 65 Dh Les règles de calcul des salaires dépendent de la catégorie des employés. Les règles suivantes sont appliquées :

- Le salaire d'un vendeur est égal au SMIC mensuel auquel on ajoute 10% du chiffre d'affaires réalisé par le vendeur
- Le salaire d'un représentant est égal au SMIC mensuel auquel on ajoute 15% du chiffre d'affaires réalisé par le représentant.
- Le salaire d'un technicien de production est à égale au SMIC journalier multiplié par le nombre de jours travaillés. On y ajoute 5 Dh par pièce produit.
- Le salaire d'un technicien de manutention est égal au SMIC journalier multiplié par le nombre de jours travaillés.

1. Donner les classes nécessaires pour la gestion des salaires des employées de cette entreprise.
2. Proposer un programme de test des fonctionnalités de ces classes.

Élément de correction

```
class Employe (object):
       smicMensuel= 1480.27
       smicJournalier= 65
       def __init__( self , prenom , nom, age ) :
           assert  isinstance( prenom , str ) and len ( prenom)>0
           assert  isinstance( prenom , str ) and len (nom)>0
           assert  isinstance ( age , int ) and age >=16 and age <=65
           self.prenom=prenom
           self.nom=nom
           self.age=age
           raise NotImplementedError #classe abstraite
       def salaire (self):
           raise NotImplementedError #methode abstraite
       def __str__(self):
        return "Nom: "+self.nom+" Prenom: "+self.prenom+" Age:
   "+str(self.age)
```

```
class Commerecial(Employe):
        bonus=0
        def __init__( self , prenom , nom, age , ca ) :
            assert ( isinstance (ca,float) and ca>=0)
            self.ca=ca
            super(Commerecial,self).__init__(prenom,nom,age)
            raise NotImplementedError
        def salaire (self):
            return(Employe.smicMensuel+self.bonus*self.ca)
        def __str__(self):
            return super(Commerecial,self).__str__()+" CA : "+str(self.ca)\
                   + " Bonnus : " + str(self.bonus)
    class Vendeur (Commerecial):
        bonus=0.1
        def __init__(self, prenom, nom, age, ca):
           try:
              super(Vendeur,self).__init__(prenom,nom,age,ca)
           except NotImplementedError:
              pass
    class Representant(Commerecial):
       bonus=0.15
       def __init__(self,prenom,nom,age,ca):
           try:
              super(Representant,self).__init__(prenom,nom,age,ca)
           except NotImplementedError:
              pass
    class Technicien(Employe):
        def __init__(self,prenom,nom,age,nbJour):
            assert (isinstance (nbJour, int) and (0 <= nbJour <= 31) )
            self.nbJour=nbJour
            super(Technicien, self).__init__(prenom, nom, age)
        def __str__(self):
            return super(Technicien,self).__str__()+" NBJours:
    "+str(self.nbJour)
    class TechProd(Technicien):
        prixPiece=5
        def __init__ (self , prenom , nom , age , nbJour , nbPiece):
            assert (isinstance(nbPiece, int)and nbPiece >= 0)
            self.nbPiece = nbPiece
            try:
                super(TechProd, self).__init__(prenom, nom, age, nbJour)
            except NotImplementedError:
                pass
        def salaire(self):
            return(Employe.smicJournalier*self.nbJour+
                       self.nbPiece*TechProd.prixPiece )
        def __str__(self):
          return super(TechProd,self).__str__()+" NBPiece: "+str(self.nbPiece)
    class TechManutentation (Technicien):
        def __init__(self,prenom,nom,age,nbJour):
            try:
                super(TechManutentation,self).__init__(prenom,nom, age,nbJour)
            except NotImplementedError:
                pass
        def salaire(self):
            return(self.nbJour * Employe.smicJournalier)
    v1= Vendeur("FI SIBD", "ENSAB", 30, 2000.0)
    r1= Representant("FI IA", "ENSAB", 30, 1000.0)
```

```
tp1= TechProd("DUT GI", "ENSAB", 30, 30, 100)
tm1= TechManutentation("DUT GE", "ENSAB", 30, 30)
print (v1, "Salaire : ", v1.salaire())
print (r1 , "Salaire : ",r1.salaire())
print (tp1, "Salaire : ",tp1.salaire() )
print (tm1 , "Salaire : ", tm1.salaire())
```

Exercice 2.3

Énoncé

On considère l'ensemble de nombres rationnels Q défini comme suit :

$$\mathbb{Q} = \left\{\frac{n}{d} : \ (n, d) \in \mathbb{Z} \times (\mathbb{Z}\{0\})\right\}$$

où $\mathbb{Z}$ est l'ensemble d'entiers relatifs. On souhaite définir une classe Fraction qui modélise les éléments de $\mathbb{Q}$

1. On propose de décrire l'état d'une fraction par trois attributs : les deux valeurs absolues de n et d et le signe de la fraction. Donner la méthode constructrice de la classe Fraction. Il faut s'assurer qu'une fraction est toujours conforme à sa définition.
2. Redéfinir la méthode __str__ afin de pouvoir afficher une fraction sous la forme n:d.
3. Proposer une méthode inverse qui permet d'inverser une fraction. Par exemple l'inverse de 5 7 est 7 5
4. Proposer une méthode inverseSigne qui permet d'inverser le signe d'une fraction.
5. Proposer une méthode qui permet d'additionner deux fractions.
6. Proposer une méthode qui permet de faire la soustraction de deux fractions.
7. Proposer une méthode qui permet de faire la multiplication de deux fractions.
8. Proposer une méthode qui permet de faire la division de deux fractions.
9. Proposer une méthode simplifier qui permet de simplifier une fraction (penser à calculer le PGCD de n et d.).

   Exemple : 9 27 peut être simplifié à 1 3 , 16 20 peut être simplifié à 4 5 .
10. Proposer une méthode irréductible qui renvoie : True si la fraction ne peut pas être simplifiée, False sinon.

Élément de correction

```
class Fraction:
        def __init__(self,n,d):
            assert isinstance(n,int);assert(isinstance(d,int)and d!=0)
            self.n,self.d=abs(n),abs(d)  #n:numerateur,d:denominateur
            if(n*d)>=0:self.s=1 #lesigne
            else: self.s=-1
        def __str__(self):
            res = ""
            if self.s ==-1: res = "-"
            res = res + str(self.n) + ":" + str(self.d)
            return res
        def __setattr__(self,nom,value):
            if nom not in["n","d","s"]: raise ValueError
            if nom=="n":
                assert isinstance(value,int)
                self.__dict__[nom]=abs(value)
            if nom=="d":
                assert(isinstance(value,int)and value!=0)
                self.__dict__[nom]=abs(value)
            if nom=="s":
                assert value in(-1,1)
                self.__dict__[nom]=value
        def inverseSigne (self):
            self.s*=-1
        def inverse (self):
            if (self.d !=0): self.n,self.d=self.d,self.n
            else: raise ValueError
        def add(self, other):
            assert isinstance(other, Fraction)
            num = (self.s_self.n_other.d) + (other.s_other.n_self.d)
            den = self.d*other.d;  return Fraction(num, den)
        def sub(self, other):
            assert isinstance(other, Fraction)
            aux = Fraction(other.n, other.d)
            aux.s = other.s ; aux.inverseSigne(); return self.add(aux)
        def mul(self,other):
            assert isinstance(other,Fraction)
            res=Fraction(self.n_other.n,self.d_other.d)
            res.s=self.s_other.s; return res
        def div(self,other):
            assert isinstance(other,Fraction)
            aux=Fraction(other.n,other.d); aux.s=other.s;aux.inverse()
            return(self.mul(aux))
        def pgcd(n,m):
```

```
        assert isinstance(n,int)and isinstance(m,int)and n>0 and m>0
        if n<m: n,m=m,n #Euclide algorithm
        while m>0: n,m= m, n%m
        return n
    pgcd=staticmethod(pgcd)
    def irréductible (self):
        return (Fraction.pgcd(self.n, self.d)==1)
    def simplify(self):
        diviseur=Fraction.pgcd(self.n,self.d)
        if diviseur!=1:
            self.n=self.n/diviseur
            self.d=self.d/diviseur
```

Exercice 2.4

Énoncé

1. Définissez une classe JeuDeCartes() permettant d'instancier des objets dont le comportement soit similaire à celui d'un vrai jeu de cartes. La classe devra comporter au moins les trois méthodes suivantes:

   - Méthode **constructeur** : création et remplissage d'une liste de 52 éléments, qui sont eux-mêmes des tuples de 2 entiers. Cette liste de tuples contiendra les caractéristiques de chacune des 52 cartes. Pour chacune d'elles, il faut en effet mémoriser séparément un entier indiquant la valeur (2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, les 4 dernières valeurs étant celles des valet, dame, roi et as), et un autre entier indiquant la couleur de la carte (c'est-à-dire 3,2,1,0 pour Cœur, Carreau, Trèfle et Pique). Dans une telle liste, l'élément (11,2) désigne donc le valet de Trèfle, et la liste terminée doit être du type: [(2, 0), (3,0), (3,0), (4,0), ... (12,3), (13,3), (14,3)]
   - Méthode **nom_carte()** : cette méthode doit renvoyer, sous la forme d'une chaîne, l'identité d'une carte quelconque dont on lui a fourni le tuple descripteur en argument. Par exemple, l'instruction : print(jeu.nom_carte((14, 3))) doit provoquer l'affichage de : As de pique
   - Méthode **battre()** : comme chacun sait, battre les cartes consiste à les mélanger. Cette méthode sert donc à mélanger les éléments de la liste contenant les cartes, quel qu'en soit le nombre.
   - Méthode **tirer()** : lorsque cette méthode est invoquée, une carte est retirée du jeu. Le tuple contenant sa valeur et sa couleur est renvoyé au programme appelant. On retire toujours la première carte de la liste. Si cette méthode est invoquée alors qu'il ne reste plus aucune carte dans la liste, il faut alors renvoyer l'objet spécial None au programme appelant.

   Exemple d'utilisation de la classe JeuDeCartes() :

```
jeu = JeuDeCartes() # instanciation d'un objet
jeu.battre() # mélange des cartes
for n in range(53): # tirage des 52 cartes :
  c = jeu.tirer()
  if c == None: # il ne reste plus aucune carte dans la liste
          print('Termine !')
  else: print(jeu.nom_carte(c)) #valeur et couleur de la carte
```

2. Instancier deux jeux de cartes A et B (un pour chaque joueur) et les mélanger. Ensuite, à l'aide d'une boucle, tirer 52 fois une carte de chacun des deux jeux et comparer leurs valeurs. Si c'est la première des deux qui a la valeur la plus élevée, on ajoute un point au joueur A. Si la situation contraire se présente, on ajoute un point au joueur B. Si les deux valeurs sont égales, on passe au tirage suivant. Au terme de la boucle, comparer les comptes de A et B pour déterminer le gagnant.
3. Modifier la classe **JeuDeCartes** pour obtenir un même jeu de cartes de l'ensemble des joueurs Reprendre le traitement précédant pour 3 joueurs (tirer successivement une carte pour chaque joueur) afin de déterminer le gagnant de ces 3 joueurs

Élément de correction

```
from random import randrange
class JeuDeCartes(object):
    # attributs de classe (communs a toutes les instances) :
    couleur = ('Pique', 'Trefle', 'Carreau', 'Coeur')
    valeur=(0,0,2,3,4,5,6,7,8,9,10,'valet','dame','roi','as')
    def __init__(self):"Construction de la liste des 52 cartes"
        self.carte =[]
        for coul in range(4):
            for val in range(13):
                self.carte.append((val +2, coul)) # Commence a 2
    def nom_carte(self, c):
        "Renvoi du nom de la carte c, en clair"
        return "%s de %s" %(self.valeur[c[0]],self.couleur[c[1]])
    def battre(self):
        "Melange des cartes"
        t = len(self.carte) # nombre de cartes restantes
        for i in range(t):
          # tirage au hasard de 2 emplacements dans la liste :
          h1, h2 = randrange(t), randrange(t)
          # echange des cartes situees a ces emplacements :
          self.carte[h1],self.carte[h2]=self.carte[h2],self.carte[h1]
    def tirer(self):
        "Tirage de la premiere carte de la pile"
```

```
        t = len(self.carte) # verifier qu'il reste des cartes
        if t >0:
            carte = self.carte[0] # choisir la premiere carte du jeu
            del(self.carte[0]) # la retirer du jeu
            return carte # en renvoyer copie au prog. appelant
        else: return None # facultatif

jeuA = JeuDeCartes() # instanciation du premier jeu
jeuB = JeuDeCartes() # instanciation du second jeu
jeuA.battre() # melange de chacun
jeuB.battre()
pA, pB = 0, 0 # compteurs de points des joueurs A et B


# tirer 52 fois une carte de chaque jeu :
for n in range(52):
    cA, cB = jeuA.tirer(), jeuB.tirer()
    vA, vB = cA[0], cB[0] # valeurs de ces cartes
    if vA > vB: pA += 1
    elif vB > vA: pB += 1 # (rien ne se passe si vA = vB)
    # affichage des points successifs et des cartes tirees :
    print ("%s * %s ==> %s * %s" % (jeuA.nom_carte(cA),
            jeuB.nom_carte(cB), pA, pB))
    print ("le joueur A obtient %s points, le joueur B en obtient
%s."
            % (pA, pB))
```

Chapitre 3 : Calcul scientifique en Python

Exercice 3.1

Énoncé

On considère la matrice A définie par :

$$A = \begin{pmatrix} 4 & 5 & 6 & -1 \\ 5 & 10 & 15 & 2 \\ 6 & 15 & 1 & 4 \\ -1 & 2 & 4 & -2 \end{pmatrix}$$

1. Définir la matrice A comme un np.array()
2. Determiner le rang de la matrice A (utiliser « np.linalg.matrix_rank »). Puis à l'aide de « np.linalg.eig » , calculer le vecteur propre V et les valeurs propres W de A et donner une base de ces valeurs propres.
3. Calculer l'inverse de A a partir de la question précedente (et en utilisant « np.linalg.inv ». Comparer le resultat obtenu avec celui qui peut être obtenu par dot(V * 1 / W, V.transpose())
4. A l'aide de « np.linalg.solve » Resoudre le système AX = B suivant :

$$A = \begin{pmatrix} 4 & 5 & 6 & -1 \\ 5 & 10 & 15 & 2 \\ 6 & 15 & 1 & 4 \\ -1 & 2 & 4 & -2 \end{pmatrix} \; et \; B = \begin{pmatrix} 3 \\ -2 \\ 4 \\ 1 \end{pmatrix}$$

5. La factorisation LU décompose une matrice comme le produit de deux matrices triangulaire supérieure (U = upper) et triangulaire inférieure (L = lower). La décomposition LU est effectuée dans SciPy (« scipy.linalg ») par la fonction lue_factor, qui retourne une matrice contenant la matrice L et la matrice U. Si la diagonale de la matrice L est composée de 1, la factorisation LU est unique. Une fois la matrice mise sous forme LU, la fonction lue_solve permet de résoudre le système d'équations AX=B.
6. Reprendre la résolution du système précédent en utilisant la factorisation LU. Faire une comparaison entre les 2 résultats obtenus.

Élément de correction

```
import numpy as np
import scipy.linalg as sclin
import scipy.sparse as sparse
```

```
def showMatrice(nom, mat):
    dimMat=mat.shape
    print ("Matrice %s :" %nom)
    for i in range(dimMat[0]):
        strMat=""
        for j in range(dimMat[1]):
            strMat+= (str("%.2f" % mat[i,j]) + "\t\t")
        print(strMat)

A = np.array([[4,5,6,-1],[5,10,15,2],[6,15,1,4],[-1,2,4,-2]])
showMatrice("Matrice A",A)

rank = np.linalg.matrix_rank(A)
print("Rang de la Matrice A : ", rank)

W, V = np.linalg.eig(A)
print("Valeurs propores de la Matrice A")
print (W)
print("Vecteurs propores de la Matrice A")
print(V)

W = np.sort(W)
print("Classement valeur propores de la Matrice A")
print(W)

A_inv_1 = np.dot(V * 1 / W, V.transpose())
print("Inverse 1 de la Matrice A")
print (A_inv_1)
A_inv_2 = np.linalg.inv(A)
print("Inverse2 de la Matrice A")
print (A_inv_1)
print("dif inv de la Matrice A")
print (A_inv_2 - A_inv_1)

b = np.array([3,-2,4,1])
x = np.linalg.solve(A,b)

print("Matrice A")
print (A)
print("Vecteur B")
print (b)
print("Solution Ax=b")
print (x)

lu,piv=sclin.lu_factor(A)
x = sclin.lu_solve((lu, piv), b)
print("Solution Ax=b")
print (x)
```

Exercice 3.2

Énoncé

La fonction $f(x,t) = e^{-(x-3t)^2} \sin\big(3\pi(x-t)\big)$ décrit pour une valeur fixe de t une onde localisée en espace. Faire un programme qui visualise cette fonction comme une fonction de x dans l'intervallex ∈ [−4, 4] pour t = 0.

Élément de correction

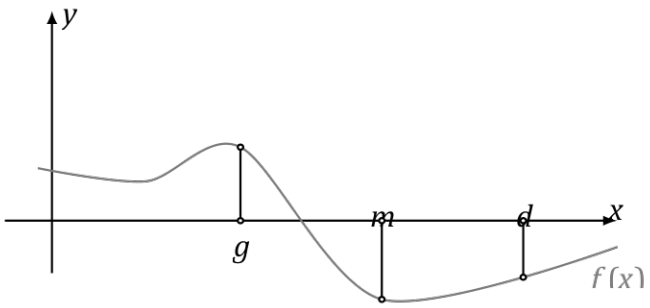
```
from math import exp, pi,sin
   import numpy as np
   import matplotlib.pyplot as plt
   def f(x,t):
       return exp(-(x-3*t)**2) * sin(3*pi*(x-t))
   f = np.vectorize(f)
   x = np.linspace(-4,4)
   print("x=",x)
   y=f(x,0)
   print("y =",y)
   plt.plot(x,y)
   plt.xlabel("x")
   plt.ylabel("f(x,0)")
   plt.show()
```

Exercice 3.3

Énoncé

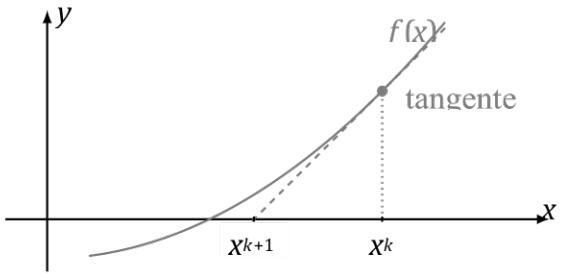
On se donne une fonction f : [a,b] → R continue, vérifiant f(a)f(b) ≤ 0. Le fait que f change de signe entre les deux extrémités du segment assure par théorème des valeurs intermédiaires que f possède au moins un zéro sur l'intervalle [a,b]. Le but est de trouver une approximation d'un tel zéro à ε près.

1. La méthode de résolution dichotomique est procédée ainsi :
   - on démarre avec g = a et d = b;
   - à chaque étape, on calcule le milieu m de l'intervalle [g,d], et on fait prendre cette valeur à g ou à d, pour maintenir l'invariant de boucle f(g)f(d) ≤ 0 (qui assure que f possède un zéro dans l'intervalle [g,d]).
   - on s'arrête lorsque d − g ≤ 2ε : le milieu est alors une approximation à ε près d'un zéro de f.



Ecrire une fonction dichoM(f,a,b,eps) prenant en entrée la fonction f, les bornes a et b et le réel ε, et renvoyant un zéro à ε près de f sur [a,b]. Donner le code de test de la fonction dichoM.

2. La méthode de Newton pour le calcul d'une solution de $f(x) = 0$ nécessite que f soit dérivable. On part simplement d'une abscisse x_0, et à chaque étape x_{k+1} est l'intersection de la tangente à f au point $(x_k f(x_k))$ avec l'axe des abscisses. On espère que la suite (x_k) ainsi définie converge vers un zéro de f.



Écrire une fonction newtonM(f,g,a,e) réalisant *la résolution avec* la méthode de Newton pour la fonction f à partir du point x_0. La fonction g est la dérivée de f.

3. Le module scipy possède un sous module scipy.optimize dédié à l'optimisation. Dans ce module on trouve :
   - la fonction bisect : c'est la méthode dichotomique. Elle s'utilise sous la forme bisect(f, a, b) comme ci-dessus (on peut ajouter une précision optionnelle).
   - la fonction newton : c'est la méthode de Newton. Elle s'utilise sous la forme newton(f, x0, fprime=...), où fprime est un argument optionnel pour la dérivée.
   - la fonction fsolve : une généralisation de la méthode de Newton.

   Effectuer un test de comparaison entre ces fonctions et les fonction dicho et newton.

Élément de correction

```
import scipy.optimize as scOpt
def dichoM(f,a,b,e):
    """résout f(x)=0 sur[a,b] avec un critère d'arrêt e par
    dichotomie"""
    inf,sup = a,b
    n=0
    while (sup-inf) > e:
        m = (sup+inf)/2
        if f(inf)*f(m)<0: sup = m
        else: inf = m
        n+=1
    return m,n
```

```
def newtonM(f,fp, x0, e):
    """ résout f(x)=0 sur [a,b] avec un critère d'arrêt e
    par la méthode de newton , parton de x0"""
    xn = x0
    xnplus1= xn - f(xn)/fp(xn)
    n=1
    while abs(xnplus1 - xn) > e:
        xn, xnplus1= xnplus1, xnplus1- f(xnplus1)/fp(xnplus1)
        n+=1
    return xnplus1,n
def compare(f,fp,a, b, e):
    """ Comparaison des méthodes de résolution approchée
    de f(x)=0 """
    print( dichoM(f,a,b,e), newtonM(f,fp,a,e),scOpt.bisect(f,a,b),
         scOpt.newton(f, a), scOpt.fsolve(f,a) )
#définir la fonction et sa dérivée
fct= lambda x: (x**2-1)
dfct= lambda x: x*2
compare(fct,dfct,-1,2,0.01)
```

Exercice 3.4

Énoncé

On lâche une bille de masse *m*. Cette bille est soumise à l'action de la pesanteur et à un effort de freinage proportionnel au carré de la vitesse. L'équation de résultante dynamique projetée dans la direction $\overrightarrow{e_z}$ conduit à :

$$m\frac{dv}{dt}(t) = mg - \mu m v^2 \Leftrightarrow \frac{dv}{dt}(t) = g - \mu v^2$$

où *v* est la vitesse de la bille, initialement nulle : $v(0) = 0$ (μ est l'indice de viscosité). La vitesse $v(t)$ est alors solution du problème de Cauchy :

$$\begin{cases} \dfrac{dv}{dt} = f(t, v(t)) \\ v(0) = 0 \end{cases}$$

On considère *f* la fonction de deux variables : $f(t, \mathrm{x}) \to g - \mu v^2$ avec les constantes du problème suivantes:

- g=9.81 # accélération de la pesanteur, m/s^2
- mu=2.5e-4 # indice de viscosité, 1/m

1. Définir une fonction *f* qui prend comme arguments une valeur de *x* et un instant *t* et qui renvoie la valeur de *f* (*t*,*x*) correspondant au problème.
2. Écrire une fonction *euler*(*f* ,*x*0,*a*,*b*,*h*) qui prend comme arguments une fonction *f* , une valeur initiale *x*0, une valeur initiale de temps *a*, une valeur finale de temps *b*, un pas de temps *h* et qui renvoie deux listes $liste_t$ et $liste_x$ contenant respectivement les instants $(ti)_{i\geq 0}$ avec $t_i = a+h\times i \leq b$ et les approximations de la solution correspondantes.
3. La fonction odeint de la bibliothèque scientifique scipy.integrate de Python permet la résolution d'équation différentielle. Résoudre l'équation différentielle précédente et tracer sur un même graphe la solution obtenue par les fonctions fonction *euler et odeint (utiliser par exemple un* intervalle de temps [0s,100s] et un pas de temps $h = 10^{-3}$s)

Élément de correction

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
g = 9.81 #accelération de la pésenteur, m/s²
mu = 2.5e-4 #indice de viscosité, l/m
def F(v,t):
    return g - mu * v**2
def euler(F, v0, a, b, h):
    t, v=a, v0
    liste_t, liste_v =[t], [v]
    while t+h<b:
        v=v+h*F(v,t)
        liste_v.append(v)
        t+=h
        liste_t.append(t)
    return liste_t, liste_v
t,v = euler(F,0,0,100,10)
plt.plot(t,v, 'r', label="euler, pas =10")
t,v = euler(F,0,0,100,1e-3)
plt.plot(t,v, 'b', label="euler, pas =1e-3")
plt.legend(loc="center right")
plt.title("Variation de la vitesse d'un corps en chute libre")
plt.xlabel("temps (s)")
plt.ylabel("vitesse v(t), m/s")
#Solution calculée avec 'odeint'
lest=np.linspace(0,100)
lesv=odeint(F,0,lest)
plt.plot(lest,lesv,'g', label="Sol odeint")
plt.grid(True); plt.show()
```

Exercice 3.5

Énoncé

Considérons dans cet exercice des données de pourboire("tips") dont les informations sur chaque pourboire, reçu par un serveur travaillant dans des fêtes, été enregistrés sur une période de quelques mois. Ce dataset comporte 244 lignes et 7 variables, sa structure est la suivante :

Variable (rubrique d'information du pourboire) :

- total_bill : valeur de la facture en DH,
- tip : valeur depourboire en DH,
- sex : sexe du payeur de factures,
- smoker: presence de fumeurs dans la fête.
- day : jour de la semaine,
- time : moment de la journée,
- size : taille de la fêteof the party.

Ecrire un code python permettant de réaliser les tâches suivantes :

1. Lire et placer dans un dataFrame les données de pourboire à partir du fichier « tips.csv ».
2. Utiliser les differents attributs du dataframe pour afficher les différents informtions relatives au dataset : Taille du dataset, type des variables, étiquette des variables.
3. Donner une statistique sommaire (count, mean, std : écart-type, min, max) et celle associé aux différentes variables quantitatives du jeu de données.
4. Représenter séparément dans un graphique l'histogramme de distribution de « total_bill » et celui de « tip ». utiliser un graphique boxplot (boîte à moustaches) pour visualiser pour comparer les medianes associées aux variables « total_bill» et « tip ».
5. Utiliser la fonction crosstab(variable, "freq") pour déterminer les éffectifs associés au variable qualitative « sex » puis tracer le diagramme en barres(type « bar ») et le diagramme circulaire(type « pie ») corespondants.
6. Reprendre ces graphes pour comparer les variable « sex » et « smoker »
7. Déterminer la corrélation (utiliser la fonction « corr() ») entre les variables « total_bill» et « tip ». comparer le resultat obtenu avec celui retourner par les fonctions pearsonr et kendalltau du sous module scipy.stats.
8. Tracer un graphe de nuage de points entre les variables « total_bill» et « tip ».

Élément de correction

1

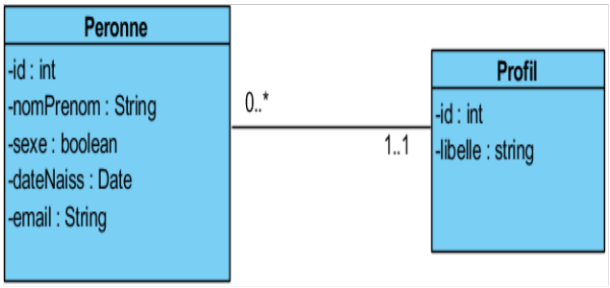
```
   import matplotlib.pyplot as plt
   import pandas as pd
   import scipy.stats as scSt
   import numpy as np
   # Lecture d'un fichier texte
   tips = pd.read_csv("tips.csv",header = 0, sep = ",")
2
   print(tips.shape)
   tips.info()
   print(tips.columns)
3
   print(tips.describe())
   print(tips.describe().round(2))
   #Il est possible de sélectionner les variables soit via
   les crochets [], soit par un point.
   tips.total_bill.describe()
   tips["total_bill"].describe()
   tips.total_bill.mean() ; tips.total_bill.std()
   tips.total_bill.var() ; tips.total_bill.median()
4
   plt.subplot(3,1,1); tips.total_bill.hist( color = "r")
   plt.subplot(3,1,2); tips["tip"].plot(kind = "hist",
   title="tip Distribution")
   plt.subplot(3,1,3); tips[["total_bill","tip"] ].boxplot()
   plt.show()
5)
   t = pd.crosstab(tips.sex, "freq" )
    #t = pd.crosstab(tips.sex, "freq", normalize=True)
   print(t)
   t.plot.bar()
   t.plot.pie(subplots=True)
   # ou
   fig, ax = plt.subplots()
   ax.pie(np.array(t).flatten(),labels=["Femmes","Hommes"])
   plt.show()
6
   t = pd.crosstab(tips.sex, tips.smoker)
   t.plot.bar()
   t = pd.crosstab(tips.sex, tips.smoker, normalize="index")
   t.plot.bar(stacked=True)
   t = pd.crosstab(tips.sex, tips.smoker)
   t.plot.pie(subplots=True, figsize = (12, 6))
7
   print(tips.corr())
   print(tips.total_bill.corr(tips.tip))
   print(scSt.stats.pearsonr(tips.total_bill, tips.tip))
   print(scSt.kendalltau(tips.total_bill, tips.tip))
8
   tips.plot.scatter("total_bill", "tip")
   plt.show()
```

Chapitre 4 : Base de données et Interfaces graphiques

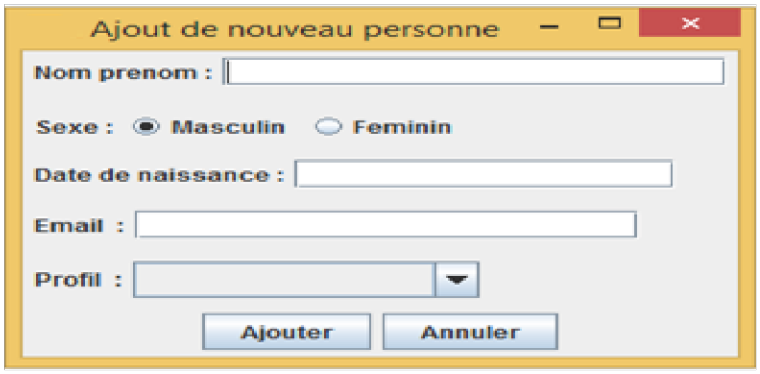
Exercice 4.1

Énoncé

Le but de cet exercice est de créer une application de gestion des profils du personnel d'une entreprise industrielle. Les informations relatives aux personnes sont stockées dans une base de données MYSQL « GestionPersonnes ». Le diagramme de classe ci-dessous représente la structure des 2 tables « personnes » et « profils » utilisées (les champs id de ces tables sont auto-increment).



1. Définir la classe « Connexion » permettant d'obtenir une connexion à la base de données « GestionPersonnes ».
2. On considère l'interface graphique suivante permettant l'ajout de l'enregistrement, regroupant les informations d'une nouvelle personne, dans la table « personnes ».

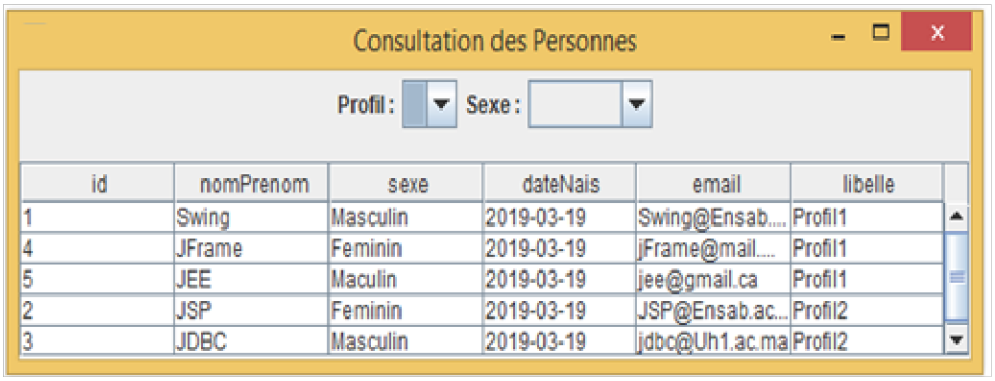


Les contrôles de l'interface sont décrits dans le tableau suivant :

Type de contrôle	Nom du contrôle
Entry	txtNomPr; txtDateNais; txtEmail
Radiobutton	rbMas; rbFem
Combobox	cbProfil
Button	btnAjouter; btnAnnuler

Définir la classe permettant d'assurer les fonctionnalités de cette interface.

3. On considère l'interface graphique suivante permettant la consultation des informations des personnes selon des critères de sexe et profil.



Donner le code python permettant d'assurer les fonctionnalités de cette interface.

Élément de correction

1

```
import pymysql as my
class Connexion  (object):
   def __init__(self):
      self.dbconection = my.connect(host='localhost',port=3306,
                  user='root', db='GestionPersonnes'   )
      self.dbcursor = self.dbconection.cursor()
   def commit_db(self):
      self.dbconection.commit()
   def close_db(self):
      self.dbcursor.close() ;    self.dbconection.close()
```

2

```
from tkinter import *
from tkinter import ttk
from connexion import *
class Interface(Frame):
    def __init__(self, fenetre, **kwargs):
        Frame.__init__(self, fenetre, width=768, height=576, ** kwargs)
        self.pack(fill=BOTH)
        lbl1 = Label(self, text="Nom Prenom :")
        lbl1.place(x=10,y=20)
        self.txtNomPrenom=Entry(master=self)
        self.txtNomPrenom.place(x=100,y=20)
        lbl2 = Label(self, text="Sexe :")
        lbl2.place(x=10, y=50)
        self.sexe=bool()
        self.rbMas = Radiobutton(master=self, text="Masculin",
                variable=self.sexe,value=True)
        self.rbMas.place(x=100, y=50)
        self.rbFem = Radiobutton(master=self, text="Femenin",
                                 variable=self.sexe, value=False)
        self.rbFem.place(x=180, y=50)
        lbl3 = Label(self, text="DateNaisance :")
        lbl3.place(x=10, y=80)
```

```
            self.txtDateNaiss = Entry(master=self)
            self.txtDateNaiss.place(x=100, y=80)
            lbl4 = Label(self, text="Email :")
            lbl4.place(x=10, y=110)
            self.txtEmail = Entry(master=self)
            self.txtEmail.place(x=100, y=110)
            lbl5 = Label(self, text="Profil :")
            lbl5.place(x=10, y=140)
            listIdProfil=list()
            self.combo = ttk.Combobox( self  , style="BW.TCombobox")
            self.combo.place(x=100, y=140)
        def ajout(self):
            try:
                req= "insert into personne (nomPrenom, idProfil ) values ('"
                req+=self.txtNomPrenom.get()
                req+= ("'," + self.combo.get()+ ")")
                self.db.dbcursor.execute(req)
                self.db.dbcursor.execute('commit')
            except Exception as ex:
                print(ex.__str__())
fenetre = Tk()
fenetre.geometry('280x220')
fenetre.title("Exemple Tk")
interface = Interface(fenetre)
interface.mainloop()
interface.destroy()

3
from connexion import Connexion
from tkinter import *
from tkinter import ttk
db = Connexion()
interf = Tk()
interf.title('Consultation du personnel')
interf.geometry("500x230")
sexe = Label(interf, text="sexe :")
sexe.place(x=5, y=30, width=50, height=30)
lstval = ["Masculin", "Féminin"]
cbsexe = ttk.Combobox(interf, values=lstval)
cbsexe.place(x=60, y=33, width=100, height=20)
profil = Label(interf, text="Profil :")
profil.plac(x=170, y=30, width=50, height=30)
db.cursor.execute('SELECT libelle FROM profil')
lstval = db.cursor.fetchall()
profil = StringVar()
cbprofil = ttk.Combobox(interf, values=lstval, textvariable=profil)
cbprofil.place(x=230, y=33, width=100, height=20)
tree = ttk.Treeview(interf, columns=(1, 2, 3, 4, 5, 6), height=5,
show='headings')
tree.place(x=0, y=70, width=500)
tree.column(1, width=40)
tree.column(2, width=100)
```

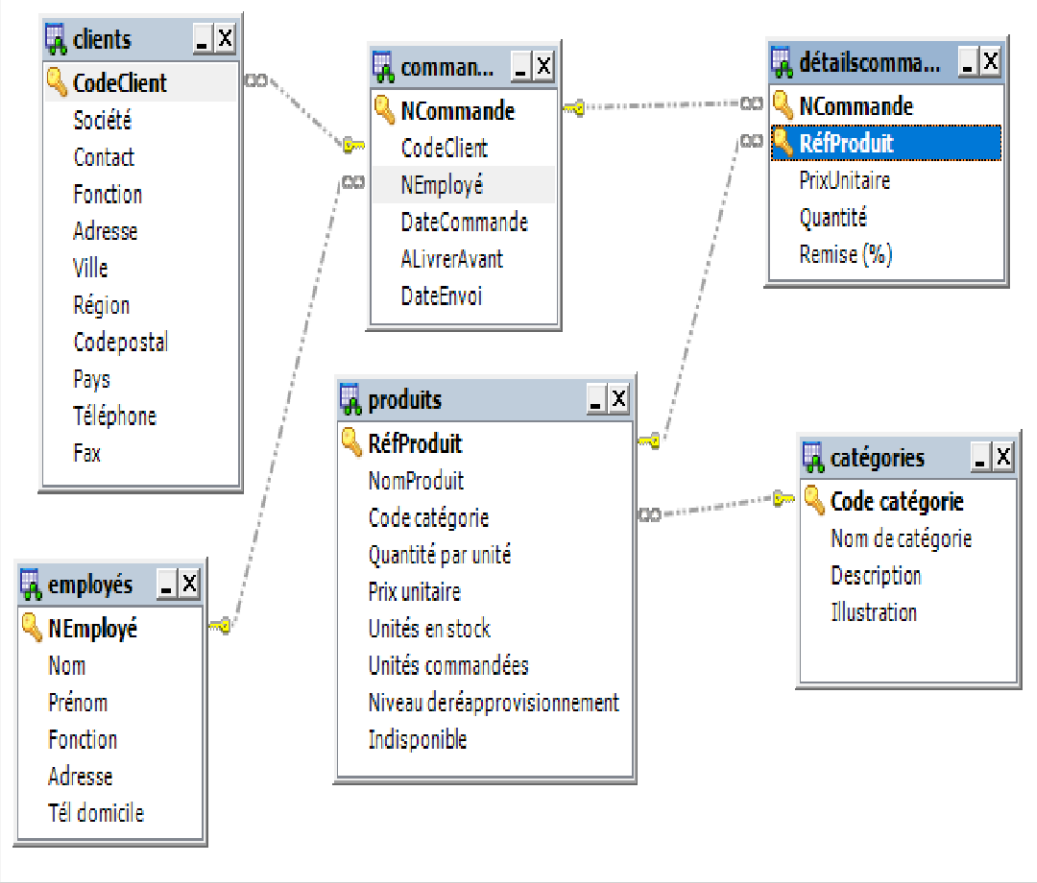
```
tree.column(3, width=60)
tree.column(4, width=80)
tree.column(5, width=140)
tree.column(6, width=80)
tree.heading(1, text='id')
tree.heading(2, text='nomprenom')
tree.heading(3, text='sexe')
tree.heading(4, text='datenaiss')
tree.heading(5, text='email')
tree.heading(6, text='libelle')
vsb = ttk.Scrollbar(interf, orient="vertical", command=tree.yview)
vsb.place(x=485, y=72, height=123)
tree.configure(yscrollcommand=vsb.set)
def listerPersonnes():
    if cbsexe.current()!=-1:
        if cbsexe.get() == 'Masculin': sx = 1
        elif cbsexe.get() == 'Féminin': sx = 0
    req='''select per.id, per.nomprenom, per.sexe, per.datenaiss,
                per.email, pro.libelle from personne per, profil pro
                 where pro.id = per.id'''
    if cbsexe.current()==-1 and cbprofil.current()==-1:
        db.cursor.execute(req)
    elif cbsexe.current()!=-1 and cbprofil.current()==-1:
        db.cursor.execute(req + ''' and per.sexe = %s''', (sx,))
    elif cbsexe.current()==-1 and cbprofil.current()!=-1:
        db.cursor.execute(req+''' and pro.libelle =
%s''',(profil,))
    else:
        db.cursor.execute(req+''' and per.sexe = %s and
pro.libelle=
                  %s''', (sx,profil))
    res = db.cursor.fetchall()
    for it in tree.get_children():
        tree.delete(it)
    for row in res:
        if row[2] == b'\x00':
            tree.insert('', 'end', values=(row[0], row[1],
'Féminin',
              row[3], row[4], row[5]))
        else:
            tree.insert('', 'end', values=(row[0], row[1],
'Masculin',
                   row[3], row[4], row[5]))
cbprofil.bind('<<ComboboxSelected>>', listerPersonnes)
cbsexe.bind('<<ComboboxSelected>>', listerPersonnes)
search = Button(interf, text='search', command=listerPersonnes)
```

```
search.place(x=130, y=200, width=80, height=20)
listerPersonnes()
interf.mainloop()
```

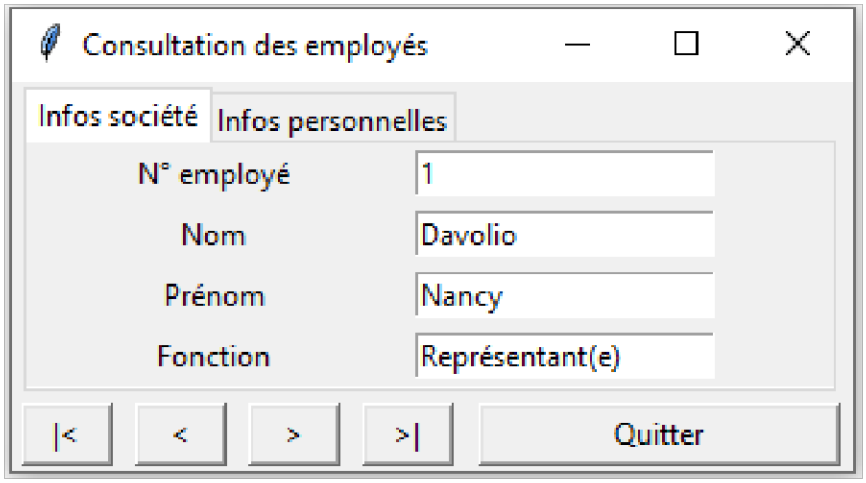
Exercice 4.2

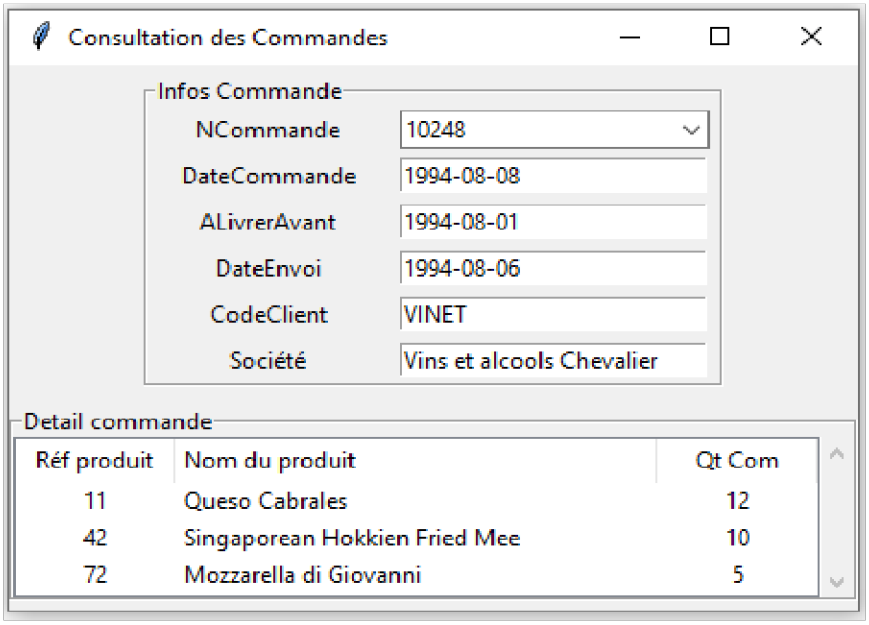
Énoncé

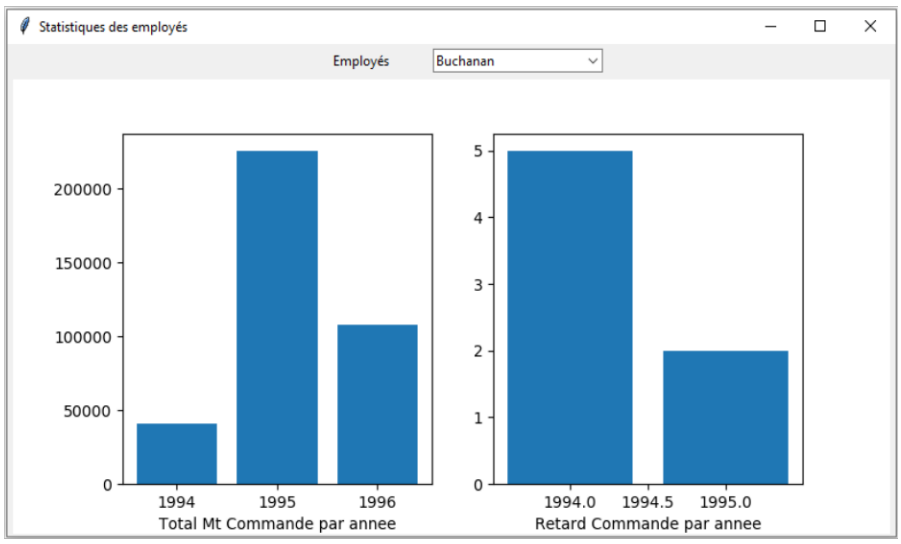
Le but de ce TP est de créer des interfaces graphiques pour une application de gestion commerciale. Les informations relatives aux ventes sont stockées dans une base de données MYSQL « dbcomptoire». Le schéma de cette base est donné par la figure ci-dessous



1. Définir la classe « Connexion » permettant d'obtenir une connexion à la base de données « GestionPersonnes ».
2. On considère les interfaces graphiques suivantes. Définir la classe permettant d'assurer les fonctionnalités de chaque interface.


Consultation des employés
Infos société
Infos personnelles
N° employé
1
Nom
Davolio
Prénom
Nancy
Fonction
Représentant(e)
|<
<
>
>|
Quitter


Consultation des Commandes
Infos Commande
NCommande
10248
DateCommande
1994-08-08
ALivrerAvant
1994-08-01
DateEnvoi
1994-08-06
CodeClient
VINET
Société
Vins et alcools Chevalier
Detail commande
Réf produit
Nom du produit
Qt Com
11
Queso Cabrales
12
42
Singaporean Hokkien Fried Mee
10
72
Mozzarella di Giovanni
5


Statistiques des employés
Employés
Buchanan
200000
150000
100000
50000
0
1994
1995
1996
Total Mt Commande par annee
5
4
3
2
1
0
1994.0
1994.5
1995.0
Retard Commande par annee

Élément de correction

```
1
from pymysql import *
class Connexion(object):
   def __init__(self):
      self.dbconection = connect(host='localhost',
                     port=3306, user='root', db='dbcomptoire'  )
      self.dbcursor = self.dbconection.cursor()
   def commit_db(self):
      self.dbconection.commit()
   def close_db(self):
      self.dbcursor.close(); self.dbconection.close()
```

```
2
Interface consultation des employés

from tkinter import Frame, Tk, ttk, DISABLED, NORMAL
from tkinter.ttk import *
from dao import Connexion
class Interface(Frame):
    def initComponents(self, window):
        Frame.__init__(self, window)
        window.geometry("385x215")
        self.tabControl = ttk.Notebook(window)
        self.tab1 = ttk.Frame(self.tabControl)
        self.tab2 = ttk.Frame(self.tabControl)
        self.tabControl.add(self.tab1, text='Infos société')
        self.tabControl.add(self.tab2, text='Info personnelles')
        self.tabControl.pack(expand=1, fill="both")
        # La premiere tab de notre interface
        Label(self.tab1, text='N° Employé :').grid(column=0,
                    row=0, padx=10, pady=10)
        self.nemp = Entry(self.tab1, width=35)
        self.nemp.grid(column=1, row=0, padx=10, pady=10)
        Label(self.tab1, text='Nom :').grid(column=0, row=1,
                  padx=10, pady=10)
        self.nom = Entry(self.tab1, width=35)
        self.nom.grid(column=1, row=1, padx=10, pady=10)
        Label(self.tab1, text='Prénom :').grid(column=0, row=2,
                 padx=10, pady=10)
        self.prnom = Entry(self.tab1, width=35)
        self.prnom.grid(column=1, row=2, padx=10, pady=10)
        Label(self.tab1, text='Fonction :').grid(column=0,
                    row=3, padx=10, pady=10)
        self.fonction = Entry(self.tab1, width=35)
        self.fonction.grid(column=1, row=3, padx=10, pady=10)
        self.firstButton = Button(window, text='|<', command='')
        self.firstButton.pack(side="left", expand=1)
```

```
        self.previousButton=Button(window, text='<', command='')
        self.previousButton.pack(side="left", expand=1)
        self.nextButton = Button(window, text='>', command='')
        self.nextButton.pack(side="left", expand=1)
        self.lastButton = Button(window, text='>|', command='')
        self.lastButton.pack(side="left", expand=1)
        style = Style()
        style.configure('W.TButton', font=('calibri', 10,
                'bold', 'underline'), foreground='red')
        self.cancelButton = Button(window, text='Quitter',
                 style='W.TButton', command=window.destroy)
        self.cancelButton.pack(side="left", expand=1)

        # La deuxieme tab de notre interface consultation
        Label(self.tab2, text='N° Employé :').grid(column=0,
               row=0, padx=10, pady=10)
        self.nemptab2 = Entry(self.tab2, width=35)
        self.nemptab2.grid(column=1, row=0, padx=10, pady=10)
        Label(self.tab2, text='Adresse Personnelle
                :').grid(column=0, row=1, padx=10, pady=10)
        self.address = Entry(self.tab2, width=35)
        self.address.grid(column=1, row=1, padx=10, pady=10)
        Label(self.tab2, text='Tel Domicile :').grid(column=0,
                   row=2, padx=10, pady=10)
        self.tel = Entry(self.tab2, width=35)
        self.tel.grid(column=1, row=2, padx=10, pady=10)

    def __init__(self, window):
        self.initComponents(window)
        self.list_emp= Connexion.Connexion().get_all_employees()
        self.pos=0;
        self.affiche()
        self.firstButton.bind("<Button>",
                              lambda x:self.move_first())
        self.lastButton.bind("<Button>",
                             lambda x:self.move_last())
        self.nextButton.bind("<Button>",
                             lambda x:   self.next())
        self.previousButton.bind("<Button>",
                                 lambda x:   self.previous())

    def affiche(self):
        self.nemp.delete(0, 'end')
        self.nemp.insert(0, self.list_emp[self.pos][0])
        self.nom.delete(0, 'end')
        self.nom.insert(0, self.list_emp[self.pos][1])
```

```
            self.prnom.delete(0, 'end')
            self.prnom.insert(0, self.list_emp[self.pos][2])
            self.fonction.delete(0, 'end')
            self.fonction.insert(0, self.list_emp[self.pos][3])
            self.nemptab2.delete(0, 'end')
            self.nemptab2.insert(0, self.list_emp[self.pos][0])
            self.address.delete(0, 'end')
            self.address.insert(0, self.list_emp[self.pos][4])
            self.tel.delete(0, 'end')
            self.tel.insert(0, self.list_emp[self.pos][5])

            if self.pos == 0:
                self.previousButton['state'] = DISABLED
            else:
                if self.pos == len(self.list_emp) - 1:
                    self.nextButton['state'] = DISABLED
                else:
                    self.previousButton['state'] = NORMAL
                    self.nextButton['state'] = NORMAL

    def move_first(self):
        self.pos=0
        self.affiche()
    def move_last(self):
        self.pos = len(self.list_emp)-1
        self.affiche()
    def next(self):
        if self.pos != len(self.list_emp)-1:
            self.pos+=1
            self.affiche()
    def previous(self):
        if self.pos != 0:
            self.pos -= 1
            self.affiche()

window = Tk()
window.title('Consultation des employés')
interface = Interface(window)
interface.mainloop()
```

Interface consultation des commandes

```
from tkinter import *
from tkinter import ttk
from Connexion import *
```

```
class Interface(Frame):
    champs=["DateCommande","ALivrerAvant","DateEnvoi","CodeClient",
            "Société"]
    def __init__(self, fenetre, **kwargs):
        self.db = Connexion()
        Frame.__init__(self, fenetre)
        self.pack(fill=BOTH)
        frm1 = Frame(self, pady=8)
        frm1.pack()
        label=Label(frm1,text="NCommande").grid(row=0,
             column=0,pady=8)
        self.db.dbcursor.execute('SELECT NCommande FROM commandes')
        results = self.db.dbcursor.fetchall()
        self.NCommande = IntVar()
        self.combo = ttk.Combobox(frm1, values=results,
             textvariable=self.NCommande )
        self.combo.grid(row=0, column=1,
                 pady=8);self.combo.current(0)
        self.widget=dict()
        for i in range(len (Interface.champs)):
            c=Interface.champs[i]
            Label(frm1, text=c, width= 20).grid(row=i+1, column=0,
                  padx=5, pady=2,sticky=NW)
            self.widget[c] = Entry(frm1)
            self.widget[c].grid(row=i+1, column=1, padx=5, pady=2)
        frm2 = Frame(self);   frm2.pack()
        self.tree = ttk.Treeview(frm2, columns=  (1, 2, 3),
                height=3, show="headings")
        self.tree.place(x=50, y=50, width=300)
        self.tree.column(1, width=50);
        self.tree.column(2, width=100)
        self.tree.column(3, width=150);
        self.tree.heading(1, text="Réf produit");
        self.tree.heading(2, text="Nom du produit")
        self.tree.heading(3, text="Qt Com");
        sbar = ttk.Scrollbar(frm2,
                orient='vertical',command=self.tree.yview)
        sbar.pack(side=RIGHT, fill=Y)
        self.tree.pack(side=LEFT, expand=YES, fill=BOTH)
        self.tree.configure(yscroll=sbar.set)
        self.combo.bind("<<ComboboxSelected>>", self.comboAction)
    def comboAction (self, event):
        req="select NCommande, DateCommande, ALivrerAvant,
              DateEnvoi, cd.CodeClient, Société from" \
            " commandes cd join clients c on c.CodeClient= " \
            " cd.CodeClient where NCommande = "+ str(self.NCommande)
```

```
            self.db.dbcursor.execute(req)
            results = self.db.dbcursor.fetchone()
            for i in range(len(Interface.champs)):
                c=Interface.champs[i]
                self.widget[c].delete(0, END)
                self.widget[c].insert(0, self.results[self.pos][i])

fenetre = Tk()
fenetre.title("Consultation des employés")
interface = Interface(fenetre)
interface.mainloop()
```

Interface des statistiques des employés

```
from tkinter import *
from tkinter.ttk import *
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from numpy import array
from dao import Connexion

class Interface(Frame):
    def __init__(self, fenetre):
        Frame.__init__(self, fenetre)
        window.geometry("485x355")
        self.pack(fill=BOTH)
        frm1 = Frame(self);  frm1.pack()
        Label(frm1, text='Employé :').grid(column=0,row=0)
        # list_values = dbconn.get_employees()
        self.nomemp = StringVar()
        self.empname=Combobox(master=frm1,textvariable=
            self.nomemp,values = list_values, state='readonly')
        self.empname.grid(column=1, row=0)
        self.empname.bind("<<ComboboxSelected>>",lambda x:
                    self.plot())
        self.fig = Figure(figsize=(8, 8), dpi=100)
        self.fig.clear()
    def plot(self):
        name=self.nomemp.get()
        res = array(dbconn.get_total_mt_cmd(name))
        annee = res.transpose()[0];  mt = res.transpose()[1]
        plot1 = self.fig.add_subplot(121)
        plot1.clear()
        plot1.bar(annee,mt)
        plot1.text(0.0,-1.5,'Total Mt Commande par année',
                horizontalalignment='center', fontsize=10)
        self.canvas = FigureCanvasTkAgg( self.fig, master=self)
```

```
            # placing the canvas on the Tkinter window
            self.canvas.get_tk_widget().pack(fill=BOTH,expand=True)
            self.canvas.draw_idle()
            res = array(dbconn.get_retard_cmnd(name))
            print(res)
            anneeRetard = res.transpose()[0]
            retard = res.transpose()[1]
            plot2 =  self.fig.add_subplot(122)
            plot2.clear()
            # plotting the graph
            plot2.bar(anneeRetard,retard)
            self.canvas = FigureCanvasTkAgg( self.fig, master=self)
            self.canvas.draw_idle()
            #placing the canvas on the Tkinter window
            self.canvas.get_tk_widget().pack(fill=BOTH, expand=True)

window = Tk()
dbconn = Connexion.Connexion()
window.title('Statistiques des employés')
interface = Interface(window)
interface.mainloop()
```

Annales des examens

Examen 1. Cycle Ingénieur SIBD S7 2019-2020

Exercice n°1 ***(5 pts)*****:**

On s'intéresse dans cet exercice aux "nombres artificiels" définis comme des séquences de chiffres d'un entier naturel, par exemple [1,0,6,3] représentant ceux du nombre décimal 1063. On considère que ces nombres artificiels sont représentés par une classe « Artificiel » dont l'attribut « listeNombre » (supposé privé) est une liste des nombres artificiels.

1. Ecrire le constructeur de la classe « Artificiel » qui accepte comme paramètre un nombre décimal arbitraire ; ***(2pts)***
2. Définir les méthodes spéciales de surcharge de l'opérateur d'addition '+'. Chacune de ces méthodes construit par instanciation la liste des nombres artificiels relatifs à un nombre décimal « other » reçu en paramètre et effectue l'opération d'addition en utilisant les chiffres de la liste construite (il faut vérifier que la somme correspondante à chaque adition ne dépasse 10 si non une propagation à gauche du retenu est nécessaire) ; ***(3pts)***
3. Donner le code d'une fonction qui vérifie qu'un nombre décimal n (reçu en argument) est narcissique. Un tel nombre est égal à la somme des puissances p-ièmes de ses chiffres, où p désigne le nombre de chiffres de n. Par exemple, 153 est narcissique car $153 = 1^3+5^3+3^3$; 548834 est également narcissique car $548834=5^6 + 4^6 + 8^6+ 8^6+ 3^6+ 4^6$. ***(2pts)***

Exercice n°2 (5 pts):

On considère dans cet exercice une grille de Sudoku de taille 9×9, découpée en 9 carrés de taille 3×3. Le but est de la remplir avec des chiffres de 1 à 9, de sorte que chaque ligne, chaque colonne et chacun des 9 carrées de taille 3×3 contienne une et une seule fois chaque entier de 1 à 9. On dira alors que la grille est complète. Alors, dans ce cas, pour chacune des lignes, chacune des colonnes et chacun des carrés de taille 3×3, la somme des chiffres est égale à 45.

On représente en Python la grille de Sudoku par un tableau array de taille 9×9 (9 lignes et 9 colonnes). On suppose que les cases non remplies sont associées au chiffre 0. Ainsi, la grille suivante est représentée par le tableau L ci-contre :

	6					2		5
4			9	2	1			
	7				8			1
					5			9
6	4						7	3
1			4					
3			7				6	
			1	4	6			2
2		6					1	

L= numpy.array([
[0,6,0,0,0,0,2,0,5], [4,0,0,9,2,1,0,0,0], [0,7,0,0,0,8,0,0,1],
[0,0,0,0,0,5,0,0,9], [6,4,0,0,0,0,0,7,3], [1,0,0,4,0,0,0,0,0],
[3,0,0,7,0,0,0,6,0], [0,0,0,1,4,6,0,0,2], [2,0,6,0,0,0,0,1,0]
])

Les 9 carrées de taille 3×3 sont numérotés du haut à gauche jusqu'en bas à droite. Ainsi, sur cette grille, le carré 0 (en haut et à gauche) contient les chiffres 6, 4 et 7 ; le carré 1(en haut et au milieu) contient les chiffres 9, 2, 1 et 8 alors que le carré 8 (en bas et à droite) contient les chiffres 6, 2 et 1.

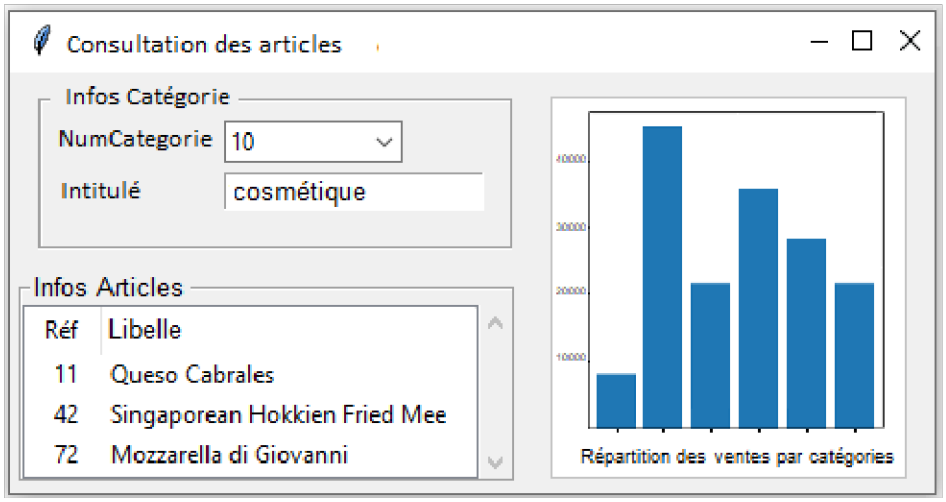
1. Ecrire une fonction carreComplet(L, i) qui prend un tableau Sudoku L et un entier i entre 0 et 8, et renvoie True si le carré i du Sudoku L vérifie les conditions de remplissage d'un Sudoku, et False sinon. ***(2pts)***
2. Ecrire une fonction complet(L) qui prend un tableau Sudoku L comme argument, et qui renvoie True si la grille est complète, False sinon. ***(3pts)***

Exercice n°3 (7 pts):

Le contexte de cet exercice est celui d'une société de vente en ligne de divers articles et l'expédition des colis correspondants. Cette société utilise une base de données MySQL «GestColis » dont le schéma simplifié est le suivant :

- Articles(ref: varchar(10), libelle: varchar(30), prixHt: double, numCategorie: smallint)
- Categorie(numCategorie: smallint, intitule: varchar(20), TauxTVA: smallint)
- Client(idClient: varchar(10), nomPrenom: varchar(30), adresse: varchar(255), email: varchar(50))
- Colis(numColis: int, dateExpedition: date, idClient: varchar(10))
- DetailColis(numColis: int, idClient: varchar(10), quantite : smallint)

1. Définir la classe « Connexion » permettant d'obtenir une connexion à la base de données « GestColis ». ***(2pts)***
2. On considère l'interface graphique suivante utilisée pour la consultation (pour chaque catégorie) des informations et la répartition des montants de vente des articles (le graphique à bar représente pour chaque catégorie le total des ventes de ses articles : $\sum prixHt * \left(1 + \frac{TauxTVA}{100}\right) * Quantite$).

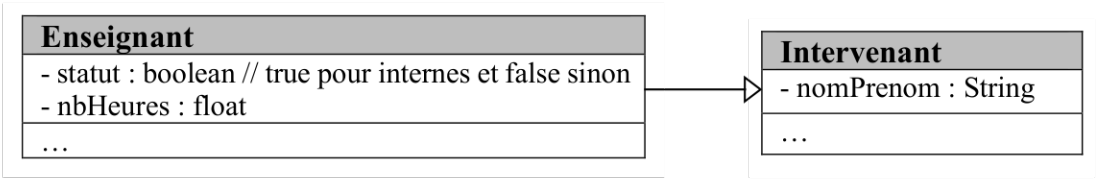


Définir la classe permettant d'assurer les fonctionnalités de l'interface. ***(5 pts)***

Examen 2. Cycle Ingénieur SIBD S7 2020-2021

Exercice n°1 (7 pts) :

Le contexte de cet exercice est celui de la gestion (en python) des heures d'enseignement complémentaires d'une université où chaque enseignant effectue un certain nombre d'heures d'enseignement dans l'année. Nous considérons qu'il y a deux types d'enseignants : les internes et les externes (exerçant dans d'autres universités) à l'université. Ainsi, un certain nombre des heures effectuées peut être considéré comme complémentaires selon le statut de l'enseignant. En effet, pour les enseignants externes toutes les heures d'enseignement sont complémentaires alors que pour les enseignants internes, seules les heures assurées au-delà d'une charge statutaire (charge horaire annuelle maximale) de 192 heures sont considérées comme complémentaires.



1. Écrire le constructeur de la classe « Enseignant ». Les valeurs du nom, du statut et du nombre d'heures total d'un enseignant sont fournies sous forme d'arguments au constructeur (seul le nom peut être librement consulté alors que les deux autres attributs sont inaccessibles) ; il faut prévoir des assertions pour le test de validation des valeurs de ces variables d'instances. (4pts)
2. Donner le code d'une méthode « getHCompl » de la classe « Enseignant » qui retourne le volume d'heures complémentaires d'un enseignant. ***(2pts)***

Exercice n°2 (6 pts) :

On considère l'approximation de la fonction $\cos(x)$ définie par la somme suivante :

$$C(x,n) = \sum_{j=0}^{n} c_j \qquad \begin{cases} c_0 = 1 \\ c_j = -c_{j-1} * \dfrac{x^2}{2j(2j-10} \end{cases}, \quad j = 1,2,\dots,n$$

1. Définir une fonction python qui reçoit un réel x et un entier n pour calculer et retourner l'approximation « C(x, n) » (indication : représenter cj par une variable,

et la mettre à jour par cj = -cj*... dans une boule for, et l'accumuler dans une autre variable pour la somme) ***(3 pts)***

2. Écrire une fonction « tableErreurs(minX, maxX, pasX) » qui un retourne un tableau (array) de numpy. Le tableau retourné représente la table des erreurs d'approximation entre C(x, n) et cos(x) pour différentes valeurs de x (variant de minX à maxX avec un pas d'incrémentation pasX) et celles de n (de valeur égale consécutivement à 5, 25, 50, 100 et 200). Ainsi, les valeurs de x parcourront la première colonne et celles de n correspondent aux autres colonnes du tableau.***(3 pts)***

   Par exemple, la table pour x = 4π, 6π, 8π, 10π (la valeur π est donnée par la constante math.pi) pourra ressembler à :

x	5	25	50	100	200
12.5664	1.61e+04	1.87e-11	1.74e-12	1.74e-12	1.74e-12
18.8496	1.22e+06	2.28e-02	7.12e-11	7.12e-11	7.12e-11
25.1327	2.41e+07	6.58e+04	-4.87e-07	-4.87e-07	-4.87e-07
31.4159	2.36e+08	6.52e+09	1.65e-04	1.65e-04	1.65e-04

Tableau array : Valeur x et (C(x, n) − cos(x))

C(25.1327 , 50) − cos(25.1327)

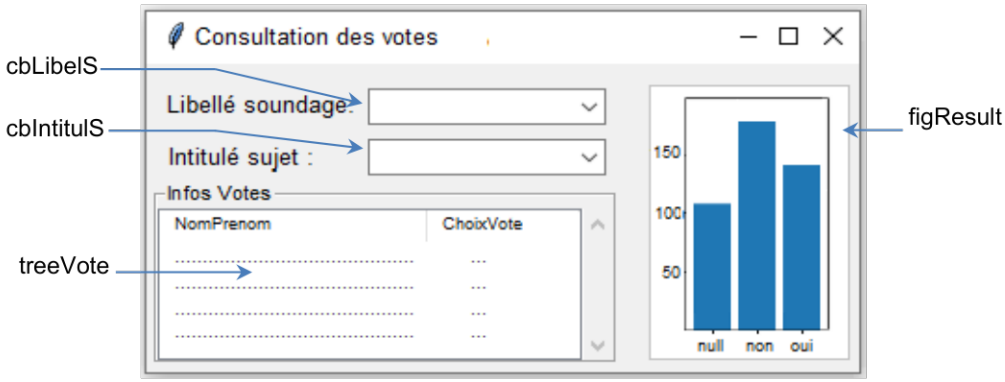
Exercice n°3 (8 pts) :

On s'intéresse dans cet exercice à la réalisation d'une application python de gestion d'un sondage. Ce dernier consiste pour des participants à choisir pour un sujet donné un vote parmi une liste de votes possibles. Pour des raisons de simplification nous nous limitons à 3 choix de votes : les votes Oui, Non et Nul. Il est possible de voter tant qu'un sondage n'est pas fermé (correspond à une date de fermeture non définie : valeur null) . Ensuite, une fois que celui-ci est fermé, il est possible de déterminer le résultat de ce sondage. Une base de données MySQL «GestSondage » a été créé pour y stocker les informations des sondages. Le schéma simplifié de cette base est le suivant :

- Soudages(numSo: smallint, libelle:varchar(20), dateOuverture:date, dateFermeture: date)
- Sujets(numSu: int, intitule: varchar(20), numSo: smallint)
- Participants(numP: int, nomPrenom: varchar(30), sexe: boolean(255))
- Votes(numV: int, dateVote: date, numP: int , numSu: int, ChoixVote: char(3))

Soit la figure ci-dessous représentant une interface graphique utilisée pour la consultation (pour chaque sondage et sujet) des informations des votes et la répartition des nombres totaux des choix de vote (pour le sondage et le sujet choisis, le graphique à bar représente le nombre total des votes de chaque choix de vote). Dans cette figure , les flèches indiquent la correspondance entre les widgets et les variables d'instance utilisées dans le constructeur de la classe de création de l'interface. Nous considérons que :

- L'accès à la base de données « GestSondage » est réaliser en utilisant une instance de la classe prédéfinie « Connexion ».
- Le remplissage du comboBox « cbLibelS » est effectué dans le constructeur de l'interface graphique alors que le remplissage du comboBox « cbIntitulS » est réalisé (à l'exécution) lors du choix d'un sondage.
- Les méthodes « selectSondage » et « selectSujet » sont appelées respectivement lors de la sélection du sondage (évènement associé à « cbLibelS ») et le choix du sujet (évènement associé à « cbIntitulS »).



1. Écrire les instructions permettant de préciser les méthodes à exécuter lorsque les évènements de sélection associés aux comboBoxs « cbLibelS » et « cbIntitulS » se sont produits. ***(2 pts)***
2. Donner le code de la méthode « selectSujet » permettant le remplissage du widget « treeVote » et l'affichage du graphique relatif au widget « figResult ». L'affichage de ce graphique est effectué uniquement pour un sondage fermé, Il faut prévoir une gestion d'exception corespondante au cas contraire ***(6 pts)***

Examen 3. Cycle Ingénieur SIBD S7 2021-2022

Exercice n°1 ***(7 pts)*****:**

On considère les données relatives à l'épidémie de l'année encours qui sont issues des différents hôpitaux du Royaume. On s'intéresse plus particulièrement dans cet exercice au temps passé dans l'hôpital. Ainsi, on connait pour chaque personne les informations:

- Sa date d'entrée (jour de l'année , exemple 67 correspond au 08/03/2022)
- Sa date de sortie (jour de l'année)
- L'issue : décès (0) ou guérison (1)

L'ensemble de ces informations sont regroupées dans un fichier « TempsHopital.csv» dont le contenu (l'entête du fichier est « entree,sortie,issue » , « 26, 31, 1 » est un exemple de ligne de ce fichier dont 26 est le jour d'entrée , 31 est le jour de sortie et 1 qui correspond à une guérison) est placé dans un DataFrame via l'instruction : df=pandas.read_csv("TempsHopital.csv",header = 0, sep = ",")

On veut calculer une fonction S(t) qui détermine la probabilité de survie des malades (une guérison) encore dans l'hopital et ceci t jours aprés leur arrivée. Cette fonction s'appelle l'estimateur de Kaplan-Meier. L'estimateur de Kaplan-Meier est dé ni par :

$$S(t) = \frac{n(t)-d(t)}{n(t)}.$$

Avec :

- n(t) est le nombre de personnes entrées dans l'hopital et qui y sont encore aprés t jours
- d(t) est le nombre de personnes qui sont décédées au jour t partir de leur entrée l'hopital (issue=0 et jourSortie – jourEnree <= t)

1. Ecrire une fonction qui retourne le résultat de l'estimateur de Kaplan-Meier pour un t donné *(3 pts)*
2. Donner le code python permettant de tracer la courbe de l'estimateur de Kaplan-Meier pour t allant de 1 à 100. *(4 pts)*

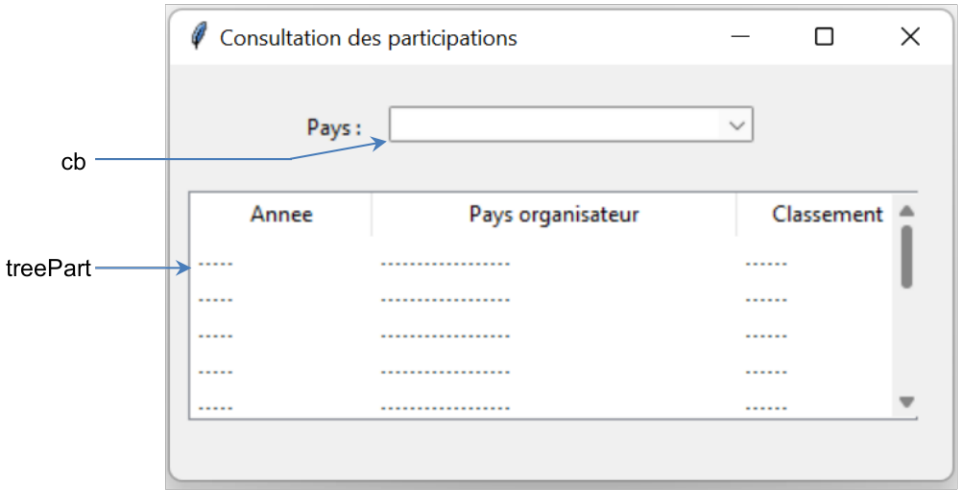
Exercice n°2 ***(13 pts)*****:**

Le contexte de cet exercice est celui de la fédération Africaine de Football (CAF) qui souhaite mettre en place une application pour gérer les informations concernant les coupes d'Afrique des nations. Une base de donnée MySQL «GestCAF» a été créé pour gérer ces informations. Le schéma relationnel simplifié de cette base est le suivant :

- Pays (numP, pays)
- Joueur (numJ, nom, prenom, **numPays**, datenaissance)
- CoupesAfrique (annee, **numPaysOrganisateur**)
- ParticipationEquipe (numPE, **annee** , **numPays**, groupe, entraineur, classement)
- ParticipationJoueur (numPJ, **numJ**, **numPE**, poste)

Les clés sont soulignées et les clés étrangères sont en gras. Les champs numP, numJ, numPE, et numPJ sont des champs dont les valeurs s'incrémentent automatiquement.

On considère l'interface graphique suivante utilisée pour la consultation (pour chaque payes) des informations des participations aux coupes d'Afrique des nations.



Nous considérons que :

- L'accès à la base de données « GestSondage » est réalisé en utilisant une instance de la classe prédéfinie « Connexion ».

- Le remplissage du comboBox « cbPays » est effectué dans le constructeur de l'interface graphique
- La méthodes « selectParticipation » est appelée lors de la sélection du pays (évènement associé à « cbPays »).

1. Écrire le code du contructeur de la classe associée à l'interface graphique *(8 pts)*
2. Donner le code de la méthode « selectParticipation » permettant le remplissage du widget « treeParticip » *(5 pts)*

Examen 4. Cycle Ingénieur SIBD et GI S7 2022-2023

Exercice n°1 (7 pts)

Dans le scrutin à un tour, un ensemble de votants expriment chacun un choix parmi une liste de candidat, et le candidat remportant le plus de suffrages remporte l'élection.

On choisit d'utiliser un dictionnaire pour compter le score de chaque alternative. Par exemple, si la liste des votes est['Alice','Bob','Alice','Dave','Alice','Charlie','Bob'], le dictionnaire résultant sera {'Alice': 3, 'Bob': 2, 'Dave': 1, 'Charlie': 1}.

1. Ecrire une fonction creer_dico_resultats prenant en argument une liste de chaines de caractères et renvoyant le dictionnaire résultant.
2. Ecrire une fonction cle_de_valeur_max prenant en argument un dictionnaire supposé non vide et renvoyant une clé associée à la valeur maximale (sur le dictionnaire de l'exemple précédent, la fonction doit donc renvoyer 'Alice'). On pourra supposer que les valeurs du dictionnaire sont des entiers positifs.
3. En déduire une fonction scrutin_un_tour prenant en argument une liste de votes et renvoyant le vainqueur du scrutin à un tour correspondant.
4. Ecrire un script python permettant de simuler un scrutin :
   - Saisir le nombre et le nom des candidats
   - Simuler l'action du vote par un choix aléatoire (utiliser la fonction random.randint et prévoir une condition d'arrêt du vote) et construire la liste des votes.
   - Afficher le vainqueur du scrutin

Exercice n°2 (6 pts)

Le but de cet exercice est de calculer la valeur approchée de sin(x) une précision donnée (16 chiffres après la virgule par exemple). On va se baser sur le développement limité de la fonction sin défini comme suit

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$$

Pour un réel x on va définir une suite U telle que pour tout entier n :

$$U_n = \sum_{k=0}^{n} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

Pour calculer la valeur approchée de sin(x) on va calculer la valeur approchée de la limite de la suite U à un certain epsilon près (pour une précision de 16 chiffres , epsilon=10^{-16}). La valeur approchée de la limite de la suite U à epsilon près est le terme de la suite U du plus petit n tel que $|U_n - U_{n-1}| <= epsilon$

1. Ecrire une fonction U(x, n) qui renvoie le terme U_n de la suite définie ci-dessus.
2. Utilisez la fonction U(x. n) pour définir la fonction sin(x).
3. Sans utilisation de la fonction U(x, n), proposer une version plus optimale de la fonction sin(x).

Exercice n°3 (7 pts)

Définir une classe **Polynome** dont les objets « seront » des polynômes de R[X] repérés par la liste de leurs coefficients où, de gauche à droite, on lit les coefficients des monômes par ordre croissant de degré.

- Par défaut il s'agira du polynôme nul.
- Un attribut .coefs sera donc défini valant cette liste.

Par exemple: P=Polynome([0,3,0,4]) définira un objet de la classe Polynôme avec P.coefs valant la liste [0,3,0,4] tout ceci représentant le polynôme $3X+4X^3$.

1. Écrire une méthode deg calculant le degré pour lequel on posera −1 comme valeur pour le polynôme nul. Indication.
2. Écrire les méthodes existantes gérant + et * : __add__ et __mul__.
3. Surcharge de la méthode __str__ pour l'affichage « standard » des polynômes (On prendra soin de ne pas afficher le premier '+' le cas échéant).
4. Surcharger la méthode existante __call__ qui permet d'évaluer un polynôme P (« P(-3) » par exemple) en utilisant la méthode de Hörner qui repose sur l'identité (avec les coefficients des monômes p0, p1, … , pn)

P(x0) = p0+ x0(p1+ x0(p2+ x0(p3+ x0(⋯+ x0(pn))))).

Examen 5. Cycle Ingénieur SIBD et GI S7 2023-2024

On considère dans cet exercice les missions d'exploration « Mars Exploration Rovers » dans lesquelles on utilise des robots pour analyser différents types de roches et de sols qui peuvent contenir des indices sur la présence d'eau. Chaque robot est équipé de plusieurs instruments d'analyse (caméra, microscope, spectromètres) et d'un bras qui permet d'amener les instruments au plus près des roches et sols dignes d'intérêt. La liste d'emplacements de la surface de la planète (**points d'intérêt ou PI**) est transmise au robot pour se rendre à chaque emplacement indiqué et y effectuer les analyses prévues. Une fois tous les points d'intérêts visités et les résultats des analyses transmis à la Terre, le robot reçoit une nouvelle liste de points d'intérêts et démarre une nouvelle exploration.

	x	y
0	345	635
1	1076	415
2	38	859
3	121	582

A. L'ensemble des points d'intérêts d'une exploration est représenté par un objet de type numpy.ndarray, à éléments entiers, à 2 colonnes et n lignes (l'élément d'indice i,0 correspondant à l'abscisse du point d'intérêt i et l'élément d'indice i,1 à son ordonnée).

1. On souhaite construire un tableau des distances entre les différents points d'intérêt d'une exploration et entre ceux-ci et la position courante du robot au moment du calcul. On dispose de la fonction d'entête « **def position_robot()** » qui renvoie un tuple donnant les coordonnées actuelles du robot.

   Écrire une fonction d'entête « **def calculer_distances(PI:np.ndarray)** » qui prend en paramètre un tableau numpy de n points d'intérêt et renvoie un tableau numpy de nombres flottants, de dimension (n+1)×(n+1), tel que l'élément d'indice i,j fournit la distance entre les points d'intérêt i et j, l'indice n désignant le point de départ du robot ($Distance_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$)
2. Écrire une fonction d'entête « **def longueur_chemin(chemin:list, d:np.ndarray)** » qui prend en paramètre un chemin à parcourir (liste des

indice des points d'intérêt » et la matrice des distances entre points d'intérêt (telle que renvoyée par la fonction calculer_distances) et renvoie la distance que doit effectuer le robot pour suivre ce chemin en partant de sa position courante (correspondant à la dernière ligne/colonne du tableau d) et en visitant tous les points d'intérêt dans l'ordre indiqué dans chemin.

3. On suppose que le nombre de points d'intérêt n'est pas trop important. Dans ce cas, nous pouvons appliqués un algorithme du plus proche voisin. Le principe que nous adoptons est simple, il consiste à construire un chemin en choisissant systématiquement le point, non encore visité, le plus proche de la position courante.

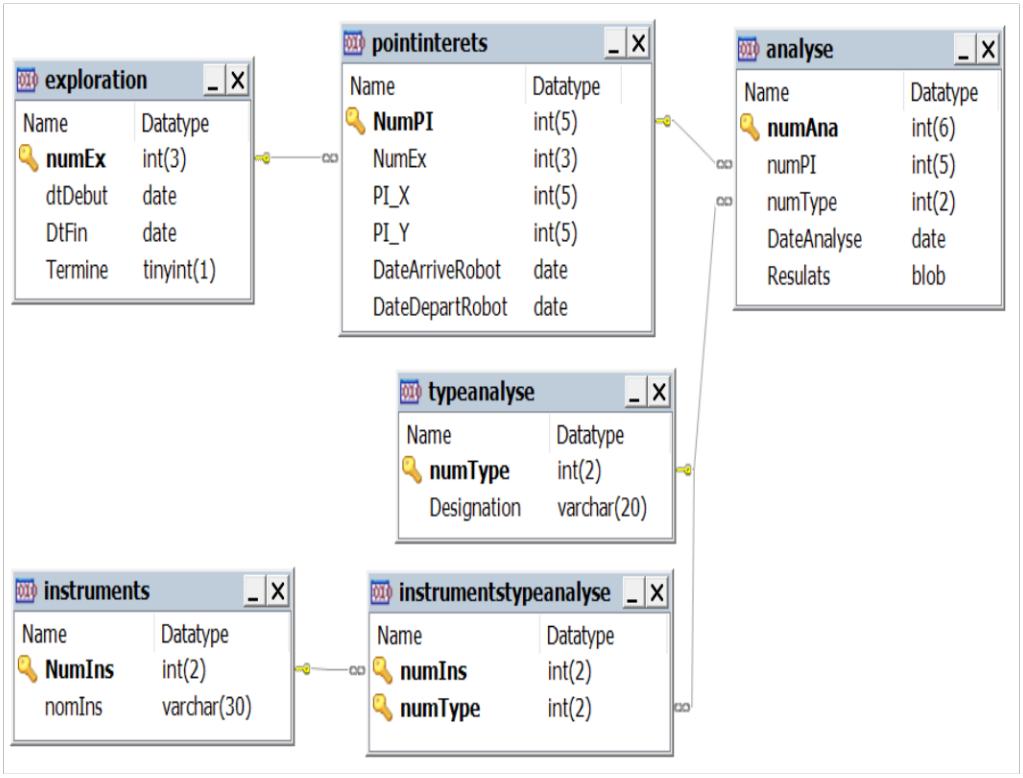
   Écrire une fonction d'entête « **def plus_proche_voisin(d:np.ndarray)** » qui prend en paramètre le tableau des distances résultat de la fonction calculer_distances et fournit (sous forme d'un tableau numpy) un chemin d'exploration en appliquant l'algorithme du plus proche voisin.

B. Une exploration est un ensemble de points d'intérêts à l'intérieur d'une zone géographique limitée, elle est codifiée et repérée par un numéro (**numEx**). Un point d'intérêt est repéré par deux entiers, positifs ou nuls, correspondant à ses coordonnées cartésiennes à l'intérieur de la zone d'exploration (**PI-X** et **PI_Y**). Le type d'analyse que le robot doit effectuer est définis à l'avance, elle est représentée par une **désignation**. Chaque robot est capable d'effectuer un certain nombre de types d'analyses géologiques correspondant aux différents instruments dont il dispose. Chaque instrument est défini par un nom (**nomIns**). On suppose dans cette partie que ces informations sont transférées au robot sur une période de quelques mois. Elles sont regroupées dans un fichier « explorations.csv » dont l'entête correspond aux variables suivantes: numEx, PI_X, PI_Y, designation et nomIns.

On considère que les données du fichier « explorations.csv » sont placées dans un dataFrame (utilisation de la méthode read_csv) nommé « **dfExploration** ».

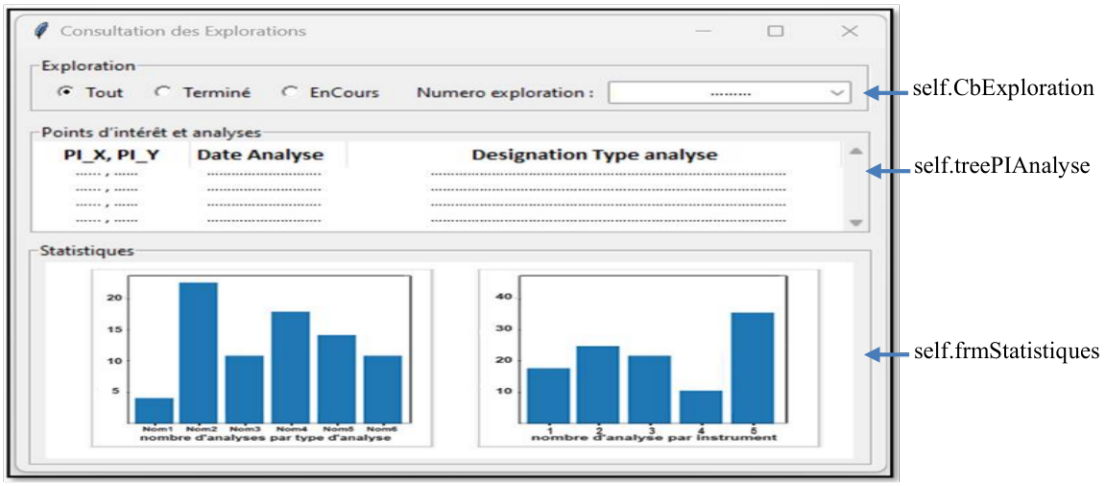
1. Donner le code permettant de tracer un graphe de nuage de points (type « scatter») entre les variables « PI_X » et « PI_Y».
2. En utiliser la fonction crosstab(variable, "freq"), écrire le code permettant de tracer 2 graphes circulaires (type « pie ») pour comparer les effectifs associés au variable « designation » et « nomIns».

C. On considère dans cette partie que les différentes informations des explorations reçues des robots sont stockées dans une base de données MySQL «DBExploration». Le modèle physique de cette dernière est schématisé par le diagramme suivant :



Une série d'analyses (effectués par un robot) étant associée à chaque point d'intérêt.

On considère l'interface graphique suivante utilisée pour la consultation (pour chaque exploration) des informations relatives aux analyses effectuées et les graphiques relatifs au nombre d'analyse par nom du type d'analyse (graphe à bar de gauche) et par numéro d'instrument (graphe à bar de droite).



Nous considérons que :

- L'accès à la base de données « DBExploration» est réalisé en utilisant une instance de la classe prédéfinie « Connexion » (vu dans le cours).
- Les widgets «Radiobutton» de l'interface sont associés à l'attribut «self.etatExploration» (valeur égale à 0 pour «Tout», à 1 pour «Terminé» et à 2 pour «EnCours»). La méthode « selectEtatExploration » est affectée à l'argument « command » de ces widgets. Cette méthode assure le remplissage du combobox « self.CbExploration » par les numéros des explorations correspondantes à l'état choisie («Tout», «Terminé» ou «EnCours»)
- La méthode « selectExploration est appelée lors de la sélection d'un numéro d'exploration (évènement associé au combobox). Cette méthode assure le remplissage du widget TreeView et le traçage des graphiques de l'interface.

1. Écrire le code de la méthode « **selectEtatExploration**»
2. Donner le code de la méthode « **selectExploration**»

Barème de notation

A.1.	A.2.	A.1.	B.1.	B.2.	C.1.	C.2.
3 pts	3 pts	3 pts	2 pts	2 pts	2 pts	5 pts