Application 1:

Ecrire un programme qui permet de remplir une matrice **M** carré par **N*N** entiers positifs, (avec 3<=**N**<=7), puis de calculer et d'afficher la somme des éléments de la matrice.

```
Algorithme PP
Debut
    Saisir(N)
    Remplir(M,N)
    S← somme(M,N)
    Ecrire(''la somme :'', S)
Fin
```

TDO

Objet	Type
N, S	Entier
M	Mat
Saisir, Remplir	Procédure
somme	Fonction

TDNT

Type
Mat=**Tableau** de 7*7 entier

```
Procedure saisir(@N : entier)
Debut
    Repeter
        Ecrire(''N='')
        Lire(N)
    Jusqu'à N ∈ [3..7]
Fin
```

```
Procedure remplir(@M : Mat, N : entier)
Debut
    Pour i de 0 à N-1 faire
       Pour j de 0 à N-1 faire
          Repeter
               Ecrire(''M= '')
               Lire(M[i,j])
          Jusqu'à M[i,j]>=0
        Fin pour
     Fin pour
Fin
```

TDO

Objet	Type
i,j	Entier

Fonction somme (M :Mat, N :entier) :entier
Debut
S←0
Pour i de 0 à N-1 faire
Pour j de 0 à N-1 faire
S← S+M[i,j]
Fin pour
Fin pour
Retourner S
Fin

TDO

Objet	Type
i,j, s	Entier

Implémentation en python

```
from numpy import *
def saisir():
    N=int(input("N="))
    while not(3<=N<=7):
        N=int(input("N="))
    return N
def remplir(M,N):
    for i in range(N):
        for j in range(N):
            M[i,j]=int(input("M["+str(i)+","+str(j)+"]="))
            while not (M[i,j]>=0):
                M[i,j]=int(input("M[{i},{j}]="))
def somme(M,N):
    S=0
    for i in range(N):
        for j in range(N):
            S=S+M[i,j]
    return S
#Programme Principale
N=saisir()
M=array([[int()]*N]*N)
remplir(M,N)
S=somme(M,N)
print("la somme=", S)
```

Application 2:

Un **employé** est définit par :

- **Matricule** (une chaine numérique)
- **Salaire** (un réel)
- **Etat_Civil** (M : Marié ou C : Célibataire ou D : Divorcé)

Ecrire un algorithme puis l'implémentation en python d'un programme permettant de :

- Saisir un entier **N**<=10
- Remplir un tableau **T** par **N** employés
- Calculer le nombre d'employés célibataires
- Afficher le nombre d'employés célibataires

<table>
<tr><th>Solution en algorithme</th><th>Implémentation en Python</th></tr>
<tr><td></td><td>

```
from numpy import*
Emp=dict(
   mat=str(),
   sal=float(),
   ec=str()
)
```

</td></tr>
<tr><td>

```
Procédure saisir(@n : entier)
Début
      Répéter
           Ecrire (''Donner N :'')
           Lire(n)
      Jusqu'à n<=10
Fin
```

</td><td>

```
def saisir():
   n=int(input("n="))
   while not (n<=10):
      n=int(input("n="))
   return n
```

</td></tr>
<tr><td>

```
Procédure remp(@t : tab, n : entier)
Début
   Pour i de 0 à n-1 faire
      Répéter
           Ecrire (''Maticule='')
           Lire(t[i].mat)
      Jusqu'à t[i].mat ∈ [" "M","C","D"]
Fin
```

TDO

Objet	Type/Nature
i	entier

</td><td>

```
def remp(t,n):
   for i in range (n):
      t[i]=dict()
      t[i]["mat"]=input("Matricule=")
      while not(t[i]["mat"].isnumeric()):
         t[i]["mat"]=input("Matricule=")
      t[i]["sal"]=float(input("Salaire="))
      t[i]["ec"]=input("Etat Civil=")
      while not(t[i]["ec"] in["M","C","D"] :
         t[i]["ec"]=input("Etat Civil=")
```

</td></tr>
</table>

Fonction nombre(t :tab, n :entier) :entier
Début
Nb← 0
Pour i **de** 0 **à** N-1 **faire**
Si t[i].ec=''M'' **alors**
Nb← Nb+1
Fin si
Fin pour
Retourner Nb
fin

TDO	
Objet	**Type/Nature**
i,Nb	entier

```
def nombre(t,n):
    nb=0
    for i in range(n):
        if(t[i]["ec"]=="C"):
            nb=nb+1
    return nb
```

Algorithme PP
Debut
saisir(n)
remp(t,n)
Nb← nombre(t,n)
ecrire(Nb)
Fin

TDNT
Type
Emp=**enregistrement** Mat : chaine Sal : réel Ec : chaine **Fin** Tab= **tableau de** 10 Emp

TDO	
Objet	**Type/Nature**
Nb, n	Entier
T	Tab
Saisir, remp	Procédure
nombre	fonction

```
n=saisir()
t=array([Emp]*n)
remp(t,n)
nb=nombre(t,n)
print(nb)
```

Application n°3:

Ecrire un algorithme puis l'implémentation en python d'un programme permettant de remplir un fichier ''**personne.dat**'' par **N** (N<=50) personne définit par :

- **Nom** : chaine alphabétique
- **Age** : entier >=20
- **Salaire** : réel >0

De supprimer à partir de fichier les personnes dont leurs âges supérieur ou égale à 60 ans, enfin afficher le nouveau contenu de ''**personne.dat**''.

Exemple :

Pour N=5, soit le contenu suivant du fichier ''**personne.dat**'' :

Touati	25	19000.000
Barhoumi	32	2000.000
Ben Yahiya	61	2500.000
Andolsi	64	3200.000
Ben Achour	55	2700.000

Après suppression, le nouveau contenu sera :

Touati	25	19000.000
Barhoumi	32	2000.000
Ben Achour	55	2700.000

<table>
<tr><th>Solution algorithmique</th><th>implémentation en Python</th></tr>
<tr><td></td><td>from pickle import*
fom numpy import*
Personne=dict(
Nom = str,
Age=int(),
Salaire=float()
)</td></tr>
<tr><td colspan="2">Procédure saisir permet de saisir un entier <=50</td></tr>
<tr><td>Procedure saisir(@N :entier)
Debut
Repeter
Ecrire(''donner N :'')
Lire(N)
Jusqu'à N<=50
Fin</td><td>def saisir() :
N=int(input(''donner N :''))
while not (N<=50) :
N=int(input(''donner N :''))
return N</td></tr>
</table>

Procédure permettant de remplir un tableau T par N personne

```
Procedure Remplir(@f : fichier personne, N : entier)
Début
    ouvrir(''personne.dat'',f,''wb'')
    Pour i de 1 à N faire
       Repeter
          Ecrire(''Nom='')
          Lire(E.nom)
       Jusqu'à Alpha(E.nom)
       Repeter
           Ecrire(''Age='')
           Lire(E.age)
       Jusqu'à E.age>=20
       Repeter
            Ecrire(''Salaire='')
            Lire(E.salaire)
       Jusqu'à E.salaire>0

       Ecrire(F, E)
    Fin pour
    Fermer(F)
Fin
```

TDO	
Objet	**Type/Nature**
i	Entier
E	Personne

```
def Remplir(N):

    f=open("personne.dat","wb")
    for i in range(N):
        E=dict(Personne)
        E["Nom"]=input("Nom=")
        while not Alpha(E["Nom"]):
            E["Nom"]=input("Nom=")

        E["Age"]=int(input("Age="))
        while not (E["Age"]>=20):
            E["Age"]=int(input("Age="))

        E["Salaire"]=float(input("Salaire:"))
        while not (E["Salaire"]>0):
            E["Salaire"]=float(input("Salaire:"))


        dump(E,f)

        f.close()
```

Fonction permettant de vérifier si une chaine est alphabétique

```
Fonction Alpha (ch : chaine) : booléen
Début
    ch← majus(ch)
    i←0
    tantque (i<long(ch) et (ch[i])∈[''A''..''Z'']) faire
          i←i+1
    fin tantque
    retourner i=long(ch)
fin
```

TDO	
Objet	**Type/Nature**
i	Entier

```
def Alpha (ch):
    ch=ch.upper()
    i=0
    while (i<len(ch)) and ("A"<=ch[i]<="Z"):
        i=i+1
    return i==len(ch)
```

Procédure permettant de supprimer à partir d'un fichier les personnes dont leurs âges>=60

```
Procedure supprimer (@f :fichier personne,
n :entier)
Début
        ouvrir(''personne.dat'',f,''rb'')
        pour i de 0 à n faire
                lire(f,T[i])
        fin pour
        fermer(f)
        ouvrir(''personne.dat'',f,''wb'')
        pour i de 0 à n faire
                si T[i].age<60 alors
                        ecrire(f,T[i])
                fin si
        fin pour
        fermer(f)
fin
```

TDO	
Objet	**Type/Nature**
i	Entier
T	Tab

TDNT
Type
Tab= **tableau de** 50 personne

```
def Supprimer(n):
        T=array([Personne]*n)
        f=open("personne.dat","rb")
        for i in range(n):
           T[i]=load(f)
        f.close()

        f=open("personne.dat","wb")
        for i in range(n):
           if T[i]["Age"]<60:
              dump(T[i],f)
        f.close()
```

Procédure permettant d'afficher le contenu d'un fichier binaire

```
Procedure Afficher (@f :fichier personne )
Début
        ouvrir(''personne.dat'',f,''rb'')
        Tantque (Non finfichier(f)) faire
                lire(f,E])
                ecrire(E.nom, E.age, E.salaire)
        fin pour
        fermer(f)
fin
```

TDO	
Objet	**Type/Nature**
i	Entier
E	personne

```
def Afficher(n):
   f=open("personne.dat","rb")
   finfichier=False
   while not finfichier:
      try:
         E=load(f)

print(E["Nom"],["Age"],E["Salaire"])
      except:
         finfichier=True
   f.close()
```