jeu de devinette

Rappel

Projet 1:

Un jeu de devinette consiste à générer un nombre x(Aléatoire entre 1 et 100) puis essayer de le deviner .
(Ajouter perdu, si le nombre n'est pas trouvé dans 10 essais)

Objectif 1 : *Générer un entier x au hasard entre 1 et 100.*

Point d'information

Aléa Permet de générer un entier au hasard.

Algorithme	Python
x←Aléa(0,10)	`from random import *` `x=randint(0,10)`

Remarque : Pour utiliser **randint** en Python, il faut commencer par importer la bibliothèque **random**

Point d'information

Pour affecter une valeur à une variable, on utilise le symbole d'affectation.

Algorithme	Python
x←10	`x=10`

Attention : ne pas confondre avec le test d'égalité

Algorithme	Python
Si x=10 alors	`if x==10 :`

On va avoir la solution suivante :

Algorithme	python
Algorithme devinette Début x←Aléa(1,100)	`from random import *` `#ou from random import randint` `x=randint(1,100)`

Objectif 2 : Afficher un message contenant la question « Donner un entier entre 1 et 100 = ? » et demander de l'utilisateur une réponse.

Point d'information

Pour afficher une variable ou un message :

Algorithme	Python
Ecrire("Message",x)	`print('Message',x)`

- En Python on peut délimiter un texte avec ' ' ou " "

Exemple : `print("C'est un texte")`

- Pour rester sur la même ligne et empêcher un retour à la ligne suivante avec print, on pourra utiliser l'instruction suivante : print('___',**end=' '**)

Point d'information

Pour saisir une variable :

Algorithme	Python
lire(n)	`n=input()`

- En Python : saisir une chaîne (str) : n=input()
  saisir un entier (int): n=**int**(input())
  saisir un réel (float) : n=**float**(input())

Pour lire une variable en affichant un message:
n=input('Message=')

On va avoir la solution suivante :

Algorithme	python

Algorithme devinette
Début
x←Aléa(1,100)
Ecrire("Donner un entier entre 1 et 100:"),Lire(n)
	`from random import randint`
	`x=randint(1,100)`
	`n=int(input('Donner un entier (entre 1-100) = '))`

Objectif 3 : On veut afficher un message suivant la valeur entrée : « Bravo » si la réponse est juste et « plus grand » ou « plus petit » sinon .

Algorithme	python

Algorithme multiplication
Début
x←Aléa(1,100)
Ecrire("Donner un entier entre 1 et 100:")
Lire(y)
Si x=y alors
Ecrire("Bravo")
Sinon si y>x alors
Ecrire("Plus petit")
Sinon
Ecrire("Plus grand")
Fin
	from random import *
	x=randint(1,100)
	y=int(input('Donner un entier (entre 1-100) = '))
	if y==x :
	print('Bravo')
	elif y>x:
	print('plus petit !')
	else:
	print('Plus grand :')

Point d'information

Structure conditionnelle si

Algorithme	Python
Si condition **alors** Traitement1 **Sinon** Traitement 2 **FinSi**	**if** condition : ↔ traitement1 **else :** ↔ traitement2

- En Python on doit utiliser la tabulation ↔ pour exprimer un bloc d'instructions
- Pour rester sur la même ligne et empêcher un retour à la ligne suivante avec print, on pourra utiliser l'instruction suivante : print('___',**end=' '**)

Objectif 4 : On veut **répéter** la saisie d'un nombre et l'affichage du message correspondant **jusqu'à** ce que l'utilisateur trouve la solution **(x=n)** .

Il y a un traitement qui se répète, quelle est la boucle à choisir ?

Point d'information

La boucle **Pour**

Algorithme	Python
Pour i de 0 à n-1 faire Traitement FinPour	for i in range(n) : ↔ Traitement

- En Python on doit utiliser la tabulation ↔ pour exprimer un bloc d'instructions
- range(4) donne 0 1 2 3 range(1,4) donne 1 2 3

Point d'information

La boucle **Répéter jusqu'à**

Algorithme	Python
Répéter Traitement **Jusqu'à** condition(s)	valide=False while valide==False: Traitement valide=(cond(s))

La boucle **Tant que faire**

Algorithme	Python
Tant que Cond(s) **Faire** Traitement **Fin Tantque**	while condition : Traitement

On va avoir la solution suivante :

Algorithme	python

Algorithme nombre
Début
X←Aléa(1,100)
Répéter
Ecrire("Donner un entire entre 1 et 100:"),Lire(y)
Si x=y alors Ecrire("Bravo")
Sinon si y>x alors Ecrire("Plus petit")
Sinon Ecrire("Plus grand")
Finsi
Jusqu'à (x=y)
Fin
	from random import *
	x=**randint**(1,100)
	valide=False
	while valide==False:
	y=int(input('Donner un entier (entre 1-100) = '))
	if y==x :
	print('Bravo')
	elif y>x:
	print('plus petit !')
	else:
	print('Plus grand :')
	valide= (x==y)

On pourra aussi implémenter l'algorithme en utilisant While :

```
from random import *
x=randint(1,100)
y=-1
while x !=y:
   y=int(input('Donner un entier (entre 1-100) = '))
   if y==x :
      print('Bravo')
   elif y>x:
      print('plus petit !')
   else:
      print('Plus grand :')
```

Objectif 5 : Ajouter le nombre d'essai nb, et le jeu se termine si le joueur trouve le nombre caché et un message « Bravo » est affiché ou si le joueur fait 10 essais sans trouver le nombre et un message « Perdu ! » et affiché, ainsi que le nombre à trouver.

On va avoir la solution suivante :

Algorithme

```
Algorithme nombre
    Début
    X←Aléa(1,100)
    Nb←0
    Répéter
        Nb←nb+1
        Ecrire("Donner un entire entre 1 et 100:"),Lire(y)
        Si x=y alors Ecrire("Bravo")
        Sinon si y>x alors Ecrire("Plus petit")
                    Sinon Ecrire("Plus grand")
        Finsi
    Jusqu'à (x=y) ou (nb=10)
    Si x≠y alors Ecrire("Perdu !, le nombre à trouver=", x)
    Finsi
    Fin     Déclaration des objets
```

Objets	Types/Nature
x ,y, nb	entier

python

```
from random import *
x=randint(1,100)
nb=0
valide=False
while valide==False:
  nb=nb+1
  y=int(input('Donner un entier (entre 1-100) = '))
  if y==x :
    print('Bravo')
  elif y>x:
    print('plus petit !')
  else:
    print('Plus grand :')
  valide= (x==y)or (nb==10)

if x!=y:
        print('perdu !, le nombre à trouver=',x)
```

Exercice 2:

Soit l'algorithme permettant de compter le nombre des voyelles dans une chaine CH.


```
Algorithme Lettres
Début
    Ecrire("Donner ch="),lire(ch)
    Nb←0
    Pour i de 0 à long(ch)-1 faire
    Si Majus(ch[i]) ∈ {"A","E","I","O","U","Y"} alors
    nb←nb+1
    Finsi
    FinPour
    Ecrire('Le nombre de voyelles=',nb)
Fin
```

Déclaration des objets

Objets	Types/Nature
Ch	chaîne
i, nb	entier

Donner une implimentation en Python:

```
ch=input('Donner ch=')
nb=0
for i in range(len(ch)):
  if ch[i].upper() in {'A','E','I','O','U','Y'}:
    nb=nb+1

print('Le nombre de voyelles=',nb)
```

Ajouter le cas de consonnes :

```
Algorithme Lettres2
Début
    Ecrire("Donner ch="),lire(ch)
    Nb←0
    nbc←0
    Pour i de 0 à long(ch)-1 faire
    Si Majus(ch[i]) ∈ {"A","E","I","O","U","Y"} alors nb←nb+1
      Sinon si Majus(ch[i])∈["A".."Z"] alors nbc←nbc+1
    Finsi
    FinPour
    Ecrire('Le nombre de voyelles=',nb)
    Ecrire('Le nombre de consonnes=',nbc)
Fin
```

Déclaration des objets

Objets	Types/Nature
Ch	chaîne
i, nb,nbc	entier

en Python:

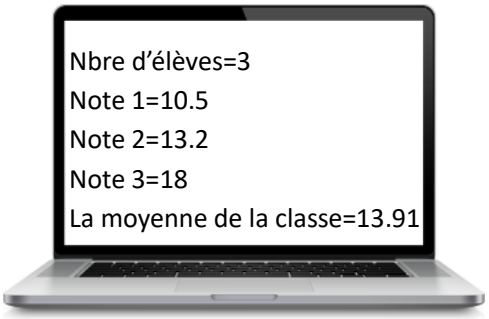
```
ch=input('Donner ch=')
nb=0
nbc=0
for i in range(len(ch)):
  if ch[i].upper() in {'A','E','I','O','U','Y'}:
    nb=nb+1
  elif  "A"<=ch[i].upper()<="Z":
    nbc=nbc+1
print('Le nombre de voyelles=',nb)
print('Le nombre de consonnes=',nbc)
```

Moyennes

Projet 3:

On veut faire la moyenne des notes d'une classe.

Exemple d'exécution :



Objectif 1 : *Saisir le nombre d'élèves, puis la note de chaque élève, enfin afficher la moyenne de la classe.*

1ère solution :

Algorithme
Algorithme Moyenne **Début** Ecrire("Nbre d'élèves="),Lire(n) S←0 Pour i de 0 à n-1 faire Ecrire("Note ",i+1,"="), Lire(x) S←S+x Finpour Moy←S/n Ecrire("moy classe=",Moy) **Fin**

Déclaration des objets

Objets	Types/Nature
i,n	Entier
x,S, moy	réel

2ère solution : (en utilisant les tableaux)

On va avoir la solution suivante :

Algorithme	python
Algorithme classe **Début** Ecrire("Nbre d'élèves=") Lire(n) Pour i de 0 à n-1 faire Ecrire("Note ",i+1,"=") Lire(T[i]) Finpour S←0 Pour i de 0 à n-1 faire S←S+T[i] FinPour Moy←S/n Ecrire("moy classe=",Moy) **Fin** Déclaration des objets	from **numpy** import * n=**int**(input('Nombres élèves=')) T=array([float()]*n) for i in range(n): print('Note',i+1,'=') T[i]=**float(**input()**)** s=0 for i in range(n): s=s+T[i] moy=s/n print(' moy de la classe=',moy)

Objets	Types/Nature
i,n	Entier
T	Tableau de n réels
S, moy	réel

Point d'information

On utilisera la bibliothèque numpy pour implémenter les tableaux

Un tableau numpy est :

- Homogène : constitué d'éléments de même type,
- Statique : sa taille est fixée lors de la création

Importation
from numpy import *
ou
from numpy import array
ou
import numpy as alias

Déclaration
T = array ([Type_élément] * N)
ou bien
T = array ([valeur_initiale] * N)

Exemple :

```
from numpy import array
T=array([float()]*10)
```

Objectif 2 : Ajouter le calcul *puis l'affichage de la meilleure note.*

On va avoir la solution suivante :

SOLUTION

Algorithme

```
Algorithme Classe
    Début
    Ecrire("Nbre d'élèves="), Lire(n)
    Pour i de 0 à n-1 faire
        Ecrire("Note ",i+1,"="),   Lire(T[i])
    Finpour
      S←0
    Pour i de 0 à n-1 faire
        S←S+T[i]
    FinPour
    Moy←S/n
    M←T[0]
    Pour i de 1 à n-1 faire
        Si T[i]>M alors M←T[i]   Finsi
    FinPour
    Ecrire("moy classe=",Moy, "Meilleure note=",M)
    Fin   Déclaration des objets
```

Objets	Types/Nature
i,n	Entier
T	Tableau de n réels
S, moy, M	réel

python

```
from numpy import *
n=int(input('Nombres élèves='))

T=array([float()]*n)
for i in range(n):
    print("Nbre d'élèves=")
    T[i]=float(input())
s=0
for i in range(n):
    s=s+T[i]
moy=s/n

M=T[0]
for i in range(1,n) :
    if T[i]>M :
        M=T[i]
print(' moy de la classe=',moy,'\nMeilleure note=',M)
```

Point d'information

Dans un print "\n" permet d'ajouter un retour à la ligne.

Objectif 3 : Ajouter un contrôle de saisie sur les notes pour être entre 0 et 20 et sur le nombre des élèves pour être entre 2 et 25.

On va avoir la solution suivante :

SOLUTION

Algorithme

```
Algorithme classe
    Début
    Répéter
        Ecrire("Nbre d'élèves="), Lire(n)
    Jusqu'à 2<=n<=25
    Pour i de 0 à n-1 faire
        Répéter
            Ecrire("Note ",i+1,"="),   Lire(T[i])
        Jusqu'à 0<=T[i]<=20
    Finpour
    S←0
    Pour i de 0 à n-1 faire
        S←S+T[i]
    FinPour
    Moy←S/n
    M←T[0]
    Pour i de 1 à n-1 faire
        Si T[i]>M alors M←T[i]   Finsi
    FinPour
    Ecrire("moy classe=",Moy, "Meilleure note=",M)
    Fin   Déclaration des objets
```

Objets	Types/Nature
i,n	Entier
T	Tableau de n réels
S, moy, M	réel

Point d'information

La boucle **Répéter jusqu'à**

Algorithme
Répéter
Traitement
Jusqu'à condition(s)

python

```
from numpy import *
T=array([float()]*25)
valide=False
while valide==False :
    n=int(input('Nombres élèves='))
    valide=(2<=n<=25)
for i in range(n):
    valide=False
    while valide==False :
        print("Note", i+1, '=',end=' ')
        T[i]=float(input())
        valide=(0<=T[i]<=20)

s=0
for i in range(n):
    s=s+T[i]
moy=s/n

M=T[0]
for i in range(1,n) :
    if T[i]>M :
        M=T[i]

print(' moy de la classe=',moy,'\nMeilleure note=',M)
```

Point d'information

La boucle **Répéter jusqu'à**

```
valide=False
while valide==False:
    Traitement
    valide=(cond(s))
```

Objectif 4 : Décomposer le problème en modules.

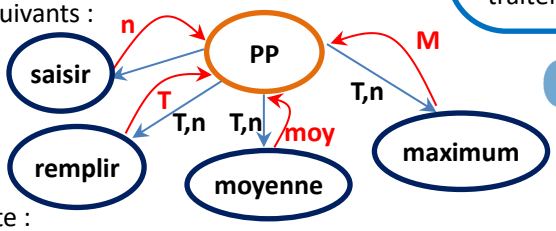
Point d'information

La **décomposition modulaire** consiste à diviser un problème en sous problème de difficultés moindres. Ces derniers sont aussi soumis à cette division jusqu'à ce qu'on arrive à un niveau abordable de difficulté.

Point d'information

Intérêts de la décomposition modulaire :
- Plus d'organisation en séparant les difficultés et les taches.
- S'occuper d'un seul problème à la fois.
- En cas d'erreur la division en module permet de savoir quel module à corriger
- Plus facile à faire évoluer.
- Permet d'éviter la répétition d'un même traitement dans un programme.

On peut utiliser les modules suivants :



Point d'information

La décomposition modulaire permet de diviser un programme en plusieurs sous-programmes. (procédures ou une fonctions).

On va avoir la solution suivante :

Algorithmes

Algorithme du programme principal :

```
Algorithme Classe:
    DEBUT
        Saisir(n)
        Remplir(T1, n)
        Moy←moyenne(T1,n)
        M←maximum(T1,n)
        Ecrire("moy classe=",Moy, "Meilleure note=",M)
    FIN
```

Appel d'une procédure — **Appel d'une fonction** — **Paramètres effectifs**

Nouveaux Types
TAB= tableau de 25 réels

T.D.O.globaux

Objet	Type/Nature
n	Entier
moy, M	réel
T1	TAB
Saisir, remplir	Procédures
Moyenne,maximum	fonctions

Algorithme de la procédure saisir :

```
Procédure saisir(@n :entier)
    DEBUT
        Répéter
            Ecrire("Nbre d'élèves="), lire(n)
        Jusqu'à 2<=n<=25
    FIN
```

Algorithme de la procédure remplir :

```
Procédure Remplir(@T:tab,n :entier)
    Début
        Pour i de 0 à n-1 faire
            Répéter
                Ecrire("Note ",i+1,"="),   Lire(T[i])
            Jusqu'à 0<=T[i]<=20
        Finpour
    Fin
```

T.D.O.Locaux

Objets	Types/Nature
i	Entier

Algorithme de la fonction moyenne :

```
fonction moyenne(T:tab,n :entier) :réel
    Début
        S←0
        Pour i de 0 à n-1 faire
            S←S+T[i]
        FinPour
        Moy←S/n
        Retourner Moy
    Fin
```

Type de retour de la fonction

T.D.O.Locaux

Objets	Types/Nature
i	Entier
s,moy	réel

Algorithme de la fonction maximum :

```
fonction maximum(T:tab,n :entier) :réel
    Début
        X←T[0]
        Pour i de 1 à n-1 faire
            Si T[i]>M alors X←T[i]    Finsi
        FinPour
        Retourner X
    Fin
```

Paramètres formels

T.D.O.Locaux

Objets	Types/Nature
i	Entier
X	réel

Point d'information

On distingue deux types de paramètres :
1-**Les paramètres formels** : qui figurent dans la définition de la procédure ou la fonction.
2-**Les paramètres effectifs** : qui figures dans l'appel de la procédure ou la fonction et qui sont manipulés par celle-ci.

Point d'information

Il y a deux modes de passage par (**valeur/adresse**). Si le mode de passage est par référence (par adresse), on ajoutera le symbole **@** avant le nom du paramètre.

Point d'information

Remarque : Les paramètres **formels** et les paramètres **effectifs** doivent s'accorder de point de vue nombre et ordre et leurs types doivent être identiques ou compatibles.

Point d'information

En Algorithmique :
Les **procédures** sont des sous-programmes qui peuvent avoir plusieurs résultats.
Une **fonction** est un sous-programme qui renvoie une valeur de type simple, ce type sera le type de la fonction..

Implémentations en python

python Méthode 1

```
from numpy import *
T1=array([float()]*25)

def saisir():
    valide=False
    while valide==False :
        n=int(input('Nombres élèves='))
        valide=(2<=n<=25)
    return n

def remplir(T,n):
    for i in range(n):
        valide=False
        while valide==False :
            print("Note", i+1, '=', end=' ')
            T[i]=float(input())
            valide=(0<=T[i]<=20)

def moyenne(T,n):
    s=0
    for i in range(n):
        s=s+T[i]
    moy=s/n
    return moy

def maximum(T,n):
    X=T[0]
    for i in range(1,n) :
        if T[i]>X :
            X=T[i]
    return X

#programme principal
n=saisir()
remplir(T1,n)
moy=moyenne(T1,n)
M=maximum(T1,n)
print('moy de la classe=',moy,'\nMeilleure note=',M)
```

python Méthode 2

```
from numpy import *
T1=array([float()]*25)

def saisir():
    global n
    valide=False
    while valide==False :
        n=int(input('Nombres élèves='))
        valide=(2<=n<=25)

def remplir(T,n):
    for i in range(n):
        valide=False
        while valide==False :
            print("Note", i+1, '=', end=' ')
            T[i]=float(input())
            valide=(0<=T[i]<=20)

def moyenne(T,n):
    s=0
    for i in range(n):
        s=s+T[i]
    moy=s/n
    return moy

def maximum(T,n):
    X=T[0]
    for i in range(1,n) :
        if T[i]>X :
            X=T[i]
    return X

#programme principal
saisir()
remplir(T1,n)
moy=moyenne(T1,n)
M=maximum(T1,n)
print('moy de la classe=',moy,'\nMeilleure note=',M)
```

Point d'information

En Python : Un module (fonction ou procédure) se définit en utilisant le mot clé **def**

Point d'information

La portée des variables en python :

• Toute variable déclarée au sein d'un module a une portée locale.

• Toute variable déclarée au sein d'un module précédée par le mot clé **global** a une portée globale. Par conséquent, elle ne devra pas figurer parmi les paramètres de ce module.

- Il est possible de définir un sous-programme sans paramètres. La communication avec l'appelant se produit grâce aux ressources (objets) communes partagées entre l'appelé et l'appelant.

Objectif 5 : Modifier le programme principal et le module **Remplir** pour saisir le nom et chaque élève et sa note.

Point d'information

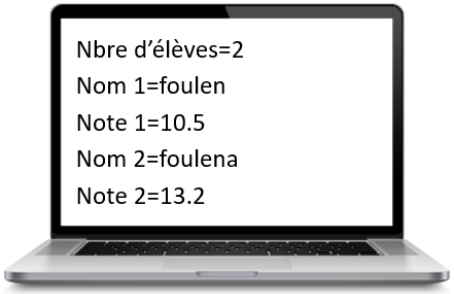
Déclarer un tableau de caractère :

Déclaration
T1 = array ([**str( )**] * N)

Déclarer un tableau de chaînes de caractère :

Déclaration
T2 = array ([**str**] * N)

Exemple d'exécution



Algorithme

Algorithme du programme principal :

```
Algorithme Classe:
    DEBUT
        Saisir(n)
        Remplir(T1,T2 n)
        Moy←moyenne(T1,n)
        M←maximum(T1,n)
        Ecrire("moy classe=",Moy, "Meilleure note=",M)
    FIN
```

Nouveaux Types
TAB= tableau de 25 réels
TAB2=Tableau de 25 chaînes

T.D.O.globaux

Objet	Type/Nature
n	Entier
moy, M	réel
T1	TAB
T2	**TAB2**
Saisir, remplir	Procédures
Moyenne,maximum	fonctions

Algorithme de la procédure remplir :

```
Procédure Remplir(@T:tab,@T2:tab2,n :entier)
    Début
        Pour i de 0 à n-1 faire
            Ecrire("Nom  ",i+1,"="), Lire(T2[i])
            Répéter
                Ecrire("Note ",i+1,"="),   Lire(T[i])
            Jusqu'à 0<=T[i]<=20
        Finpour
    Fin
```

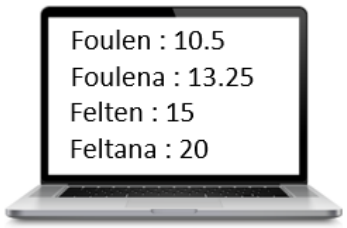
T.D.O.Locaux

Objets	Types/Nature
i	Entier

python

```
from numpy import *
T1=array([float()]*25)
T2=array([str]*25)
def saisir():
    valide=False
    while valide==False :
        n=int(input('Nombres élèves='))
        valide=(2<=n<=25)
    return n

def remplir(T,T2,n):
    for i in range(n):
        print("Nom:", i+1, '=', end=' ')
        T2[i]=input()
        valide=False
        while valide==False :
            print("Note", i+1, '=', end=' ')
            T[i]=float(input())
            valide=(0<=T[i]<=20)

def moyenne(T,n):
    s=0
    for i in range(n):
        s=s+T[i]
    moy=s/n
    return moy

def maximum(T,n):
    X=T[0]
    for i in range(1,n) :
        if T[i]>X :
            X=T[i]
    return X

#programme principal
n=saisir()
remplir(T1,T2,n)
moy=moyenne(T1,n)
M=maximum(T1,n)
print('moy de la classe=',moy,'\nMeilleure
note=',M)
```

Objectif 6 : Modifier le programme principal et ajouter un module **Afficher pour afficher les élèves et leurs notes**



Algorithme

Algorithme du programme principal :

```
Algorithme Classe:
    DEBUT
        Saisir(n)
        Remplir(T1,T2, n)
        Moy←moyenne(T1,n)
        M←maximum(T1,n)
        Afficher(T1,T2,n)
        Ecrire("moy classe=",Moy, "Meilleure note=",M)
    FIN
```

Nouveaux Types
TAB= tableau de 25 réels
TAB2=Tableau de 25 chaînes

T.D.O.globaux

Objet	Type/Nature
n	Entier
moy, M	réel
T1	TAB
T2	**TAB2**
Saisir, remplir, **afficher**	Procédures
Moyenne,maximum	fonctions

Algorithme de la procédure Afficher :

```
Procédure Afficher(T1:tab,T2:tab2,n :entier)
    Début
        Pour i de 0 à n-1 faire
            Ecrire(T2[i] , " : ", T1[i] )
        Finpour
    Fin
```

T.D.O.Locaux

Objets	Types/Nature
i	Entier

python

```
from numpy import *
T1=array([float()]*25)
T2=array([str]*25)
def saisir():
    valide=False
    while valide==False :
        n=int(input('Nombres élèves='))
        valide=(2<=n<=25)
    return n

def remplir(T,T2,n):
    for i in range(n):
        print("Nom:", i+1, '=', end=' ')
        T2[i]=input()
        valide=False
        while valide==False :

            print("Note", i+1, '=', end=' ')
            T[i]=float(input())
            valide=(0<=T[i]<=20)

def moyenne(T,n):
    s=0
    for i in range(n):
        s=s+T[i]
    moy=s/n
    return moy

def maximum(T,n):
    X=T[0]
    for i in range(1,n) :
        if T[i]>X :
            X=T[i]
    return X

def afficher(T1,T2,n):
    for i in range(n):
        print(T2[i], ' : ', T1[i])

#programme principal
n=saisir()
remplir(T1,T2,n)
moy=moyenne(T1,n)
M=maximum(T1,n)
afficher(T1,T2,n)
print('moy de la classe=',moy,'\nMeilleure
note=',M)
```

Objectif 7 : Ajouter un contrôle de saisie sur les noms des élèves pour être composés par des lettres et des espaces seulement.

Algorithme

Algorithme du programme principal :

```
Algorithme Classe:
    DEBUT
        Saisir(n)
        Remplir(T1,T2, n)
        Moy←moyenne(T1,n)
        M←maximum(T1,n)
        Afficher(T1,T2,n)
        Ecrire("moy classe=",Moy, "Meilleure note=",M)
    FIN
```

Nouveaux Types
TAB= tableau de 25 réels
TAB2=Tableau de 25 chaînes

T.D.O.globaux

Objet	Type/Nature
n	Entier
moy, M	réel
T1	TAB
T2	**TAB2**
Saisir, remplir	Procédures
Moyenne,maximum	fonctions

Algorithme de la fonction verif:

```
Fonction  verif(ch:chaine):booléen
    Début
     i← -1, ok←vrai
     Répéter
          i←i+1
          si NON(Majus(ch[i]) ∈  ["A".."Z"] ou ch[i]=" ")
           alors ok ←Faux
          Finsi
     Jusqu'a (i=long(ch)-1) ou (ok=faux)
    Retourner ok
    Fin
```

T.D.O.Locaux

Objet	T/N
I	Entier
ok	booléen

Algorithme de la procédure remplir :

```
Procédure Remplir(@T:tab,@T2:tab2,n :entier)
    Début
        Pour i de 0 à n-1 faire
            Répéter
             Ecrire("Nom  ",i+1,"="), Lire(T2[i])
            Jusqu'à vérif(T2[i])=vrai
            Répéter
                 Ecrire("Note ",i+1,"="),   Lire(T[i])
            Jusqu'à 0<=T[i]<=20
        Finpour
    Fin
```

T.D.O.Locaux

Objets	Types/Nature
i	Entier
verif	fonction

python

```
from numpy import *
T1=array([float()]*25)
T2=array([str]*25)
def saisir():
  valide=False
  while valide==False :
    n=int(input('Nombres élèves='))
    valide=(2<=n<=25)
  return n
def verif(ch):
  ok=True
  i=-1
  while (ok==True) and (i<len(ch)-1):
    i=i+1
    if not(('A'<=ch[i].upper()<='Z') or (ch[i]==' ')):
      ok=False
  return ok
def remplir(T,T2,n):
  for i in range(n):
    valide=False
    while valide==False:
      print("Nom:", i+1, '=', end=' ')
      T2[i]=input()
      valide=verif(T2[i])

    valide=False
    while valide==False :
      print("Note", i+1, '=', end=' ')
      T[i]=float(input())
      valide=(0<=T[i]<=20)
def moyenne(T,n):
  s=0
  for i in range(n):
    s=s+T[i]
  moy=s/n
  return moy
def maximum(T,n):
  X=T[0]
  for i in range(1,n) :
    if T[i]>X :
      X=T[i]
  return X
def afficher(T1,T2,n):
  for i in range(n):
    print(T2[i], ' : ', T1[i])
#programme principal
n=saisir()
remplir(T1,T2,n)
moy=moyenne(T1,n)
M=maximum(T1,n)
afficher(T1,T2,n)
print('moy de la classe=',moy,'\nMeilleure note=',M)
```

Correction de devoir de contrôle n°1

Ajouter l'affichage de la liste des candidats, en ordre décroissant, suivant le score.

Objectif 8 : Ajouter un module Résultat pour afficher le rang , le nom et la moyenne de chaque élève en ordre décroissant suivant la moyenne.

Exemple d'affichage :

Le tri d'un tableau: Le tri d'un tableau consiste à le mettre en ordre (croissant ou décroissant). Pour un tableau d'entiers de taille n:

	0	1	2	3
T	15	20	3	7

Tri par sélection

principe: (pour un tableau de taille n)
-Rechercher le plus petit élément du tableau et le placer à la 1ère case (T[0])
-Ensuite rechercher le second plus petit élément du tableau et le placer à la 2ème case (T[1])
-ainsi de suite jusqu'a trier tout le tableau.
Déroulement de l'algorithme:
-Commencer par i=0 et on cherche la position de l'élément le plus petit du tableau (pos_min).
-Une fois cet emplacement trouvé, on compare son contenu avec T[0] et s'ils sont différents(T[0]≠T[pos_min]), on permute l'élément de l'emplacement trouvé par l'élément de la première position T[0] sinon T[0] reste à sa place
→ Après ce parcours le premier élément est bien placé.
-On recommence le même procédé pour le reste du tableau (T[1],..,T[n-1]), ainsi on recherche le plus petit élément de cette nouvelle partie du tableau et on l'échange éventuellement avec T[1].
-Ainsi de suite jusqu'à la dernière partie du tableau formée par les deux derniers éléments(T[n-2],..T[n-1]).

pour un tableau d'entiers de taille n:

```
Procédure tri_select(@T:TAB,n:Entier)
  Début
    pour i de 0 à n-2 faire
      pos_min←i
      pour j de i+1 à n-1 faire
        si T[j]<T[pos_min] alors pos_min←j Finsi
      finPour
      si i ≠ pos_min alors
                          aux←T[i]
                          T[i]←T[pos_min]
                          T[pos_min]←aux
      finsi
    finpour
  Fin
```

T.D.O.Locaux

Objet	Type/Nature
i,j,pos_min,aux	Entier

Tri à bulles

Principe :
Faire remonter le plus grand élément du tableau en comparant les éléments successifs.
Déroulement de l'algorithme :
-On commence par i=0, on compare le 1er(T[0]) et le 2éme élément(T[1]) du tableau, s'il ne sont pas dans le bon ordre, on les permute, on passe ensuite au 2ème (T[1])et 3ème (T[2]), puis 3ème et 4ème et ainsi de suite jusqu'au (T[n-2])et (T[n-1]).
→À la fin du premier parcours, on aura poussé le plus grand élément du tableau vers sa place finale qui est le nième élément du tableau.
-On recommence cette opération en parcourant de 0 à n-2 puis de 0 à n-3 et ainsi de suite.
→On arrête quand la partie à trier est réduite à un seul élément ou que le tableau est devenu trié (c.à.d aucune permutation n'a été faite lors du dernier parcours à vérifier par un indicateur)

pour un tableau d'entiers de taille n:

```
Procédure tri_bulle(@T:TAB,n:Entier)
  Début
    Répéter
        Echange← faux
        Pour i de 0 à n-2 faire
            Si ( T[i] > T[i+1])Alors aux←T[i]
                                     T[i]←T[i+1]
                                     T[i+1]←aux
                                     Echange← vrai
            FinSi
        FinPour
        n← n-1
    Jusqu'à (Echange = Faux) ou (n=1)
  Fin
```

T.D.O.Locaux

Objet	Type/Nature
i,aux	Entier
echange	booléen

Le tri par insertion

Principe : Insérer le i-ème élément à sa place parmi ceux qui précèdent
Déroulement de l'algorithme :
Pour T[i] (par exemple i=4, on suppose que les i premiers élément sont triés

1. Commencer par sauvegarder T[i] dans une autre variable TMP
2. Trouver où l'élément doit être inséré dans la partie du tableau commençant de 0 à i-1
3. Décaler les éléments afin de pouvoir effectuer l'insertion
4. Insérer l'élément à partir de TMP dans la position d'insertion.

Remarque : En pratique l'étape 2 et 3 se font au même temps, on décale les éléments en cherchant la position d'insertion (on décale les éléments jusqu'à rencontrer un élément plus petit)
On commence ce procédé à partir du 2ème éléments i=1 jusqu'à atteindre la fin du tableau.

```
Procédure Tri_Insertion (@ T :TAB ; n :entier)
  Début
    Pour i de 1 à n-1 faire
      Tmp← T[i]
      j← i
      Tant que ( j >0) et (T[j-1] > Tmp ) faire
        T[j]←T[j -1]
        j← j – 1
      FinTantQue
      T[j ]←Tmp
    FinPour
  FIN
```

T.D.O.Locaux

Objet	Type/Nature
i,Tmp,j	Entier

La recherche séquentielle:

La méthode de recherche séquentielle d'un élément x dans un tableau T consiste à parcourir le tableau élément par élément progressivement de début vers la fin en les comparant avec l'élément à chercher jusqu'à trouver ce dernier ou achever le tableau. (Voir la fonction Chercher Exercice 9: vaccination)

Exercice 13: Ecrire un algorithme puis un script python permettant de saisir un entier n entre 1 et 15 puis remplir un tableau T par des entiers au hasard (entre 10 et 50), ensuite afficher T, puis trier T en ordre croissant enfin afficher le résultat (le contenu de T).

```
Algorithme tri:
    DEBUT
      Saisir(n)
      Remplir(T,  n)
       Afficher(T, n)
       tri_select(T,n)
       Afficher(T,n)
    FIN
```

Nouveaux Types
TAB= tableau de 15 entiers

T.D.O.Globaux

Objet	Type/nature
n	Entier
T	Tab
Saisir,remplir	procédure
Afficher, Tri_select	procédure

Algorithme de la procédure saisir :

```
Procédure saisir(@n :entier)
    Début
        Répéter
    Ecrire("Donner le nombre de personnes"), Lire(n)
        Jusqu'a n ∈[1..15]
    Fin
```

Algorithme de la procédure remplir :

```
Procédure remplir(@T :TAB, n :entier)
    Début
      Pour i de 0 à n-1 faire
       T[i]=Aléa(10,50)
      Finpour
Fin
```

T.D.O.Locaux

Objet	T/N
i	Entier

Algorithme de la procédure Afficher:

```
Procédure remplir(@T :TAB, n :entier)
    Début
      Pour i de 0 à n-1 faire
       Ecrire(T[i])
      Finpour
Fin
```

T.D.O.Locaux

Objet	T/N
i	Entier

pour un tableau d'entiers de taille n:

```
Procédure tri_select(@T:TAB,n:Entier)
  Début
    pour i de 0 à n-2 faire
      pos_min←i
      pour j de i+1 à n-1 faire
         si T[j]<T[pos_min] alors pos_min←j Finsi
      finPour
      si i ≠ pos_min alors
                          aux←T[i]
                          T[i]←T[pos_min]
                          T[pos_min]←aux
      finsi
    finpour
  Fin
```

T.D.O.Locaux

Objet	Type/Nature
i,j,pos_min,aux	Entier

Remarque: On peut aussi utiliser la procédure permute:

```
procédure permute(@x:entier,@ y:entier)
   Début
     aux←x
     x←y
```

T.D.O.Locaux

Objet	T/N

```
from numpy import *
from random import *
T=array([int()]*15)

def saisir():
   valide=False
   while valide==False:
      n=int(input("n="))
      valide=1<=n<=15
    return n

def remplir (T,n):
   for i in range(n):
      T[i]=randint(10,50)

def afficher(T,n):
   for i in range(n):
      print(T[i], end=" ")
    print()

def permute(x,y):
   aux=x
   x=y
   y=aux
   return x,y

def tri_select(T,n):
   for i in range(n-1):
      pos_min=i
      for j in range(i+1,n):
         if T[j]<T[pos_min]:
            pos_min=j
      if i!=pos_min:
         T[i],T[pos_min]=permute(T[i],T[pos_min])
   return T

#programme principal
n=saisir()
T=remplir(T,n)
afficher(T,n)
T=tri_select(T,n)
afficher(T,n)
```

implémentation du tri à bulle en Python:

```
def tri_bulle(T,n):
   echange=True
   while (echange==True) and (n!=1):
      echange=False
      for i in range(n-1):
         if ( T[i] > T[i+1]):
            echange=True
            aux=T[i]
            T[i]=T[i+1]
            T[i+1]=aux
      n=n-1
```

y←aux Fin	i	Entier
return T		

Remarque: En Python return T :est facultatif puisque le module tri_select modifie directement le tableau T(T par défaut passé par référence).

Exercice 17:

Écrire un programme qui permet de saisir une chaîne Ch composée uniquement de chiffres et de lettres puis extraire les lettres dans une chaîne CH1 et les chiffres dans une chaîne CH2.

Algorithme du programme principal :

```
Algorithme extraction
    Début
        saisir(ch)
        extraire(ch,ch1,ch2)
        Ecrire ("Lettres=",ch1,"Chiffres =",ch2)
    Fin
```

T.D.O.Globaux

Objet	Type/nature
Ch,ch1,ch2 Saisir , extraire	chaîne procédure

Algorithme de la procedure saisir:

```
Procédure saisir( @ ch:chaine)
    Début
        Répéter
            Ecrire("ch="),lire(ch)
        Jusqu'a verif(ch)=vrai
    Fin
```

T.D.O.Locaux

Objet	T/N
verif	fonction

Algorithme de la fonction verif:

```
Fonction  verif(ch:chaine):booléen
    Début
     i← -1, ok←vrai
     Répéter
          i←i+1
          si NON(ch[i] dans ["0.."9", "a".."z","A".."Z"])
           alors ok ←Faux
          Finsi
     Jusqu'a (i=long(ch)-1) ou (ok=faux)
    Retourner ok
    Fin
```

T.D.O.Locaux

Objet	T/N
I	Entier
ok	booléen

Algorithme de la procédure extraire :

```
procédure extraire(ch :chaine ,@ ch1 : chaine ,@  ch2 :chaine)
    Début
        CH1←"", CH2← ""
        Pour i de 0 à long(ch)-1 faire
          Si ch[i] dans ["0".."9"] alors  CH2 ←CH2+ch[i]
          Sinon CH1←CH1+ch[i]    FinSi
```

Estnum(ch[i])

Une implémentation en python:

```
def verif(ch):
    i=-1
    ok=True
    while (ok==True) and (i!=len(ch)-1):
        i=i+1
        if not(('0'<=ch[i]<='9') or ('A'<=ch[i].upper()<='Z')):
            ok=False
    return ok
def saisir():
    ok=False
    while ok==False:
        ch=input('ch=')
        ok=verif(ch)
    return ch
def extraire(ch):
    ch1=''
    ch2=''
    for i in range(len(ch)):
        if '0'<=ch[i]<='9':              # if ch[i].isdecimal() :
            ch2=ch2+ch[i]
        else:
            ch1=ch1+ch[i]
    return ch1,ch2
#programme principal
ch=saisir()
ch1,ch2=extraire(ch)
print('Lettres=',ch1,'\nChiffres =',ch2)
```

Activité : Trier les lettres d'une chaine en ordre alphabétique :

Exemple : ch= "sciences" -> "cceeinss"

```
Fonction trier (ch) :chaine
    Début
          n=long(ch)
        Répéter
          Echange← faux
          Pour i de 0 à n-1 faire
            Si ( ch[i] > ch[i+1])Alors aux←ch[i]
                                       ch[i]←ch[i+1]
                                       ch[i+1]←aux
                                       Echange← vrai
              FinSi
          FinPour
          n← n-1
        Jusqu'à (Echange = Faux) ou (n=1)
    Retourner ch
    Fin
```

Remarques: - "1"<"9", "A"<"B" puisque ord("A")<ord("B")

- ch est immutable par conséquent la permutation de ch[i] et de ch[i+1] en Python est: **ch=ch[:i]+ch[i+1]+ch[i]+ch[i+2:]**

```
def tri(ch):
    n=len(ch)
    echange=True
    while (echange==True) and (n!=1):
```

```
    FinPour
Fin
```

T.D.O.Locaux

Objet	T/N
I	Entier

```
    echange=False
    for i in range(n-1):
        if ( ch[i] > ch[i+1]):
            echange=True
            ch=ch[:i]+ch[i+1]+ch[i]+ch[i+2:]
    n=n-1
return ch
```

Remarque : Pour convertir la chaine résultat ch2 en un entier, on peut utiliser la fonction **Valeur(**ch2**)** en Python **int(**ch2**)**

Exercice 9: (*Vaccination):* On se propose de réaliser un programme, permettant de s'inscrire dans la campagne de vaccination contre le covid-19. Le script permet de saisir un numéro d'identification, puis le stocker dans un tableau ID.

- Si le numéro n'existe pas, il sera ajouté au tableau,
- Si le numéro existe déjà, le numéro ne sera pas ajouté et on demande un autre numéro.

	0	1	2	3
ID	09990999	08888888	07777777	09898888

Exemple d'exécution:

```
Donner le nombre de personnes = 4
Donner ID n°0: 09990999
Donner ID n°1: 08888888
Donner ID n°2: 08888888
Donner ID n°2: 07777777
Donner ID n°3: 09898888
```

Travail à faire: Ecrire un algorithme puis un script Python permettant de saisir le nombre de personne n puis leurs n° ID (sous forme de chaîne de caractères) à inscrire dans un tableau ID.

Algorithme Vaccination:

```
DEBUT
    Saisir(n)
    Remplir(ID,  n)
FIN
```

Nouveaux Types
TAB= tableau de n chaînes

T.D.O.Globaux

Objet	Type/nature
n	Entier
ID	Tab
Saisir	procédure
remplir	procédure

Algorithme de la procédure saisir :

```
Procédure saisir(@n :entier)
    Début
    Ecrire("Donner le nombre de personnes")
    Lire(n)
    Fin
```

Algorithme de la procédure remplir :

```
Procédure remplir(@T :TAB, n :entier)
    Début
    Ecrire("Donner ID n° 0")
    Lire(T[0])
        Pour i de 1 à n-1 faire
            Répéter
            Ecrire("Donner ID n°",i), Lire(T[i])
           Jusqu'à chercher(T[i], T, i-1)=faux
        Finpour
    Fin
```

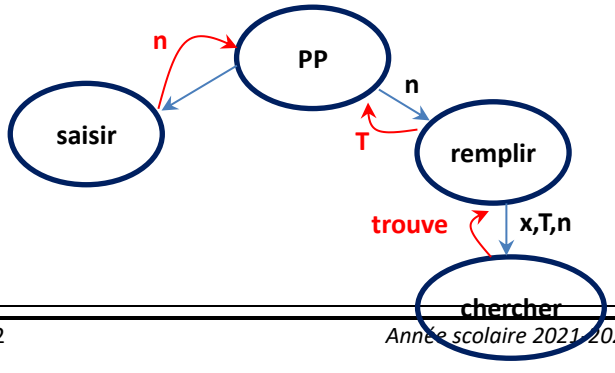
T.D.O.Locaux

Objet	Type/Nature
i	Entier
chercher	fonction

Algorithme de la fonction Chercher :

```
Fonction chercher(x : chaîne , T :Tab, n :entier) :booléen
    Début
    Trouve←Faux
    i←-1
    Répéter
    i←i+1
    Si T[i]=x alors trouve←vrai FinSi
```

```
from numpy import *

def saisir():
    global n
    n=int(input('Donner le nbre de personnes='))

def chercher(x,T,n):
    trouve=False
    i=-1
    while (trouve==False) and (i!=n):
        i=i+1
        if T[i]==x:
            trouve=True
    return trouve

def remplir(T,n):
    print('Donner ID n°0 :')
    T[0]=input()
    for i in range(1,n):
        valide=False
        while valide==False:
            print('Donner ID n°',i,':')
            T[i]=input()
            valide=chercher(T[i],T,i-1)==False

#Programme principal
saisir()
T=array(n*[str])
remplir(T,n)
```



Jusqu'à (trouve=vrai) ou (i=n)
Retourner trouve
Fin

T.D.O.Locaux

Objet	Type/Nature
i	Entier
trouve	booléen

N.B. : En python, les paramètres de type tableau sont, par défaut passés par référence
• Toute variable déclarée au sein d'un module a une portée locale.
• Toute variable déclarée au sein d'un module précédée par le mot clé global a une portée globale. Par conséquent, elle ne devra pas figurer parmi les paramètres de ce module.

►Ajouter un contrôle de saisie sur les identificateurs pour être composés de 8 chiffres exactement:

```
Procédure remplir(@T :TAB, n :entier)
    Début
Répéter
  Ecrire("Donner ID n°0:"), lire(T[0])
Jusqu'a long(T[0])=8 et EstNum(T[0])

Pour i de 1 à n-1 faire
 Répéter
  Ecrire("Donner ID n°",i,":"), lire(T[i])
Jusqu'à  (long(T[i])=8) et EstNum(T[i])
              et chercher(T[i],T,i-1)=faux
FinPour
    Fin
```

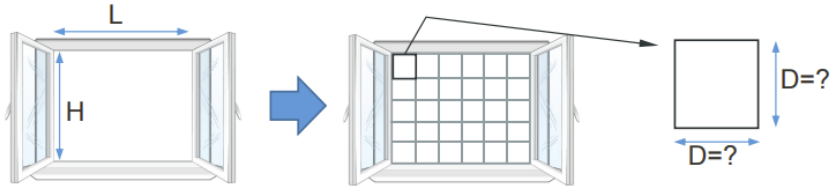
```
def remplir(T,n):
  ok=False
  while ok==False:
    print('Donner ID n° 0:')
    T[0]=input()
    ok=len(T[0])==8 and T[0].isdecimal()
 for i in range(1,n):
    ok=False
    while ok==False:
      print('DonnerID n°',i,':')
      T[i]=input()
      ok= len(T[i])==8 and T[i].isdecimal() and chercher(T[i],T,i-1)==False
```

Pour un tableau T de taille n. par exemple n=4 x=3

La recherche séquentielle:

La méthode de recherche séquentielle d'un élément x dans un tableau T consiste à parcourir le tableau élément par élément progressivement de début vers la fin en les comparant avec l'élément à chercher jusqu'à trouver ce dernier ou achever le tableau. (Voir la fonction Chercher Exercice 9: vaccination)

Exercice 4 : *PGCD(Plus grand commun diviseur)*



On veut poser un barreaudage pour une fenêtre de longueur L et de hauteur H, de manière que la distance D maximale entre les barreaux soit identique sur toute la longueur et la hauteur.

Solution : Calculer le PGCD(*Plus grand commun diviseur*) de L et H.

Calcul de **PGCD** de 2 entiers a et b par la méthode de différence :
Exp : a=15 b =27
pgcd(15,27) =pgcd(15 , 27-15)
=pgcd(15 , 12) =pgcd(15-12 ,12)
=pgcd(3 , 12)=pgcd(3 , 12-3)
=pgcd(3 , 9)=pgcd(3 , 9-3)=pgcd(3,6)
=pgcd(3 , 6-3)=pgcd(3 , 3) → a=b
donc pgcd(a,b)=a=b=3

Travail à faire: Ecrire un algorithme puis un script Python permettant de saisir 2 entiers (la longueur L(en cm), puis la hauteur H (en cm), ensuite calculer le PGCD de L et H (définir un module pour réaliser le calcul) enfin afficher la distance maximale D entre les barreaux équidistants.

Algorithme barreaudage:
```
DEBUT
   Ecrire("L=") , Lire(L)
   Ecrire("H=") , Lire(H)
   Ecrire("Distance D=",pgcd(L,H))
FIN
```
T.D.O

Objet	*Type/Nature*
L , H	Entier
pgcd	fonction

Fonction pgcd(a, b : entier) : entier
```
DEBUT
   Tant que a≠b faire
      Si a>b  alors a←a-b Finsi
      Si b>a  alors b←b-a Finsi
   FinTantQue

   Retourner a
FIN
```

Une implémentation en python:
```
def pgcd(a,b):
   while a!=b:
      if a>b:
         a=a-b
      if b>a:
         b=b-a
   return a
L=int(input('L='))
H=int(input('H='))
print('Disatance  maximale D=',pgcd(L,H))
```

Exercice 5 : *Nombres premiers*

1-Ecrire un module en python permettant de vérifier si un nombre n est premier.premier(n)(retourne True ou False)

2-Afficher à la suite tous les entiers premiers entre 1 et 99 :

Rq :Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs (1 et lui-même).

Algorithme nbrpremiers:
```
DEBUT
   Pour i de 1 à 99 faire
     si premier(i)=vrai  alors
       Ecrire(i)
     Finsi
   FinPour
FIN
```
T.D.O

Objet	*Type/Nature*
i	Entier
premier	fonction

Fonction premier(n : entier) : booléen
```
DEBUT
  nb←0
  Pour i de 1 à n faire
    Si n mod i =0 Alors nb←nb+1 Finsi
  FinPour
  Retourner nb=2
FIN
```
T.D.O.Locaux

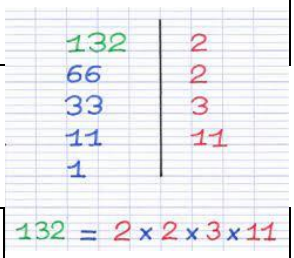
Objet	*Type/Nature*
i,nb	Entier

Une implémentation en python:
```
def premier(n):
   nb=0
   for i in range(1,n+1):
      if n % i==0:
         nb=nb+1
   return nb==2
for i in range(1,100):
   if premier(i):    # if premier(i)==True:
      print(i)
```

```
if nb==2:
   return True
else:
   return False
```

Exercice 6: *(Décomposition en produit de facteurs premiers)*

Ecrire l'algorithme d'un programme qui permet de chercher puis d'afficher la décomposition en produit de facteurs premiers d'un entier n donné. La décomposition d'un entier en produit de facteurs premiers consiste à écrire cet entier sous la forme d'un produit de ces diviseurs premiers.



Algorithme facteurs:
```
DEBUT
   Ecrire("n à décomp=")
   Lire(n)
   afficher_fact(n)
FIN
```
T.D.O

Objet	*T/N*
n	entier
afficher_fact	Procédure

Procédure afficher_fact(n : entier)
```
DEBUT
  i=2
  Répéter
     Tant que n mod i =0 faire
        Ecrire ( i )
        n←n div i
     FinTantque
     i←i+1
  Jusqu'a n=1
FIN
```
T.D.O.Locaux

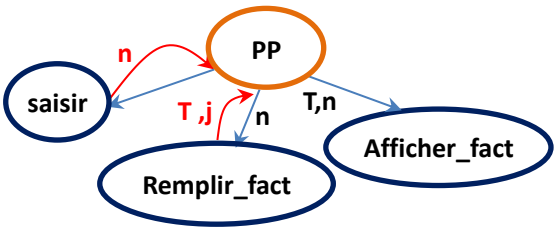
Objet	*Type/Nature*
i	Entier

Une implémentation en python:
```
def afficher_fact(x):
   i=2
   valide=False
   while valide==False:
      while x % i==0:
         print(i, end=" ")
         x=x//i
      i=i+1
      valide=( x==1)

n=int(input('n='))
afficher_fact(n)
```

autre méthode :
```
def afficher_fact(x):
   i=2
   while x!=1:
      while x % i==0:
         print(i, end=" ")
         x=x//i
      i=i+1
n=int(input('n='))
afficher_fact(n)
```

Exercice 7:

Modifier le programme de l'exercice 6, pour saisir un entier n entre (10 et 1000), remplir un tableau T par la décomposition en facteurs premiers de n enfin afficher la décomposition à partir de T. Pour résoudre ce problème, on va le décomposer en modules :



Algorithme

Algorithme du programme principal :

```
Algorithme Décomposition:
DEBUT
   Saisir(n)
   Remplir_fact(T, i , n)
   Afficher_fact(T,i)
FIN
```

Paramètre effectif

Appel d'une procédure

Nouveaux Types
TAB= tableau de 10 entiers

T.D.O.globaux

Objet	Type/Nature
i,n	Entier
T	TAB
Saisir, remplir_fact,afficher_fact	Procédures

Algorithme de la procédure saisir :

```
Procédure saisir(@n :entier)
    DEBUT
        Répéter
         Ecrire("Donner un entier (entre 10 et 1000)=")
         lire(n)
        Jusqu'à n>=10 et n<=1000
    FIN
```

Algorithme de la procédure rempli_fact :

```
Procédure Remplir_fact(@T:tab, @ j:entier , x:entier)
    DEBUT
        i←2    j←-1
        Tantque  x≠1 faire
            si x mod i = 0 alors
                j←j+1
                T[j]←i
                x←x div i
            sinon   i← i+1
            Finsi
        FinTantque
    Fin
```

Passage par variable

Paramètre formel

T.D.O.Locaux

Objets	Types/Nature
i	Entier

Algorithme de la procédure Afficher_fact :

```
Procédure afficher_fact(T :tab,j :entier)
    DEBUT
        Pour i de 0 à j-1 faire
            Ecrire(T[i], "x")
        Finpour
        Ecrire(T[j])
    FIN
```

Passage par valeur

T.D.O.Locaux

Objets	Types/Nature
i	Entier

python

```
from numpy import *
T=array([int()]*10)
def afficher_fact(T,j):
    for i in range(j):
        print(T[i],'x',end=' ')
    print(T[j])
def remplir_fact(T,x):
    global j
    i=2
    j=-1
    while x!=1:
        if x % i==0:
            j=j+1
            T[j]=i
            x=x//i
        else:
            i=i+1
def saisir():
    valide=False
    while valide==False:
       n=int(input('n à décomposer ?='))
       valide= 10<=n<=1000
    return n
#programme principal
n=saisir()
remplir_fact(T,n)
afficher_fact(T,j)
```

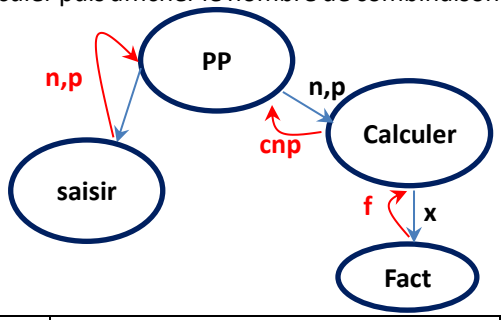
Autre méthode :

```
from numpy import *
T=array([int()]*10)
def afficher_fact(T,j):
    for i in range(j):
        print(T[i],'x',end=' ')
    print(T[j])
def remplir_fact(T,x):
    i=2
    j=-1
    while x!=1:
        if x % i==0:
            j=j+1
            T[j]=i
            x=x//i
        else:
            i=i+1
   return j
def saisir():
    valide=False
    while valide==False:
       n=int(input('n à décomposer ?='))
       valide= 10<=n<=1000
    return n

#programme principal
n=saisir()
j=remplir_fact(T,n)
afficher_fact(T,j)
```

Exercice14:

Écrire un programme qui permet de saisir deux entiers n et p (tq 1≤p≤n) et de calculer puis afficher le nombre de combinaison de p éléments parmi n : CNP, sachant que:

$$C_n^p = \frac{n!}{p!*(n-p)!}$$



Algorithme combinaison:

```
DEBUT
   Saisir(n,p)
   cnp←calculer(n,p)
    Ecrire("La CNP=",cnp)
FIN
```

T.D.O.Globaux

Objet	Type/nature
n,p,cnp	Entier
Saisir	procédure
calculer	fonction

Algorithme de la procédure saisir :

```
Procédure saisir(@n ,@p :entier)
    Début
       Ecrire("n="), Lire(n)
        Répéter
    Ecrire("p="), Lire(p)
       Jusqu'a p ∈[1..n]
    Fin
```

Algorithme de la fonction calculer :

```
Fonction calculer(n,p :entier) :entier
    Début
    c←fact(n) div (fact(p)*fact(n-p))
    Retourner  c
    Fin
```

T.D.O.Locaux

Objet	Type/Nature
c	Entier
fact	fonction

Algorithme de la fonction fact:

```
Fonction fact(x :entier) :entier
    Début
      f←1
      pour i de 2 à x faire
       f←f*i
      finpour
      retourner  f
    Fin
```

T.D.O.Locaux

Objet	Type/Nature
i,f	Entier

Une implémentation en python:

```
def saisir():
   n=int(input('n='))
   ok=False
   while ok==False:
      p=int(input('p='))
      ok=1<=p<=n
   return n,p
def fact(x):
   f=1
   for i in range(2,x+1):
      f=f*i
   return f
def calculer(n,p):
   c=fact(n)//(fact(p)*fact(n-p))
   return c
#programme principal
n,p=saisir()
cnp=calculer(n,p)
print('cnp=',cnp)
```

Exercice 15: La propagation de l'épidémie Covid-19 suit une croissance exponentielle. Pour déterminer et afficher le nombre total de personnes contaminées pendant un nombre de jours donné (N) et pour x personnes initialement contaminées on utilise la formule suivante :

$$e^x = \sum_{i=0}^{N} \frac{(x)^i}{i!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots \ldots \ldots \ldots + \frac{(x)^N}{N!}$$

Donner un algorithme solution à ce problème.

Algorithme exponentielle:

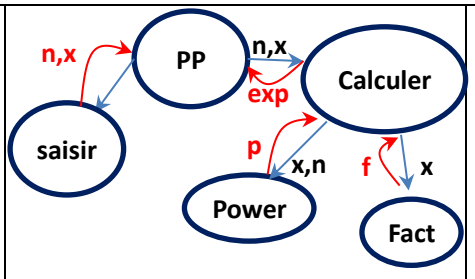
```
DEBUT
   Saisir(n,x)
   exp←calculer(n,x)
    Ecrire("Nbr contaminés=",exp)
FIN
```

T.D.O.Globaux

Objet	Type/nature
n,x,exp	Entier
Saisir	procédure
calculer	fonction

Algorithme de la procédure saisir :

```
Procédure saisir(@n ,@x :entier)
    Début
       Ecrire("nbr jours=")
       Lire(n)
       Ecrire("nbr personnes initial=")
       Lire(x)
    Fin
```



Algorithme de la fonction calculer:

```
Fonction calculer(n,x :entier) :réel
    Début
      e←1
      pour i de 1 à n faire
       e←e+power(x,i)/fact(i)
      finpour
      retourner  e
    Fin
```

T.D.O.Locaux

Objet	Type/Nature
e	Entier
fact, power	fonction

Algorithme de la fonction fact:

```
Fonction fact(x :entier) :entier
    Début
      f←1
      pour i de 2 à x faire
       f←f*i
      finpour
      retourner  f
    Fin
```

T.D.O.Locaux

Objet	Type/Nature
i,f	Entier

Algorithme de la fonction power:

```
Fonction power(x ,n:entier) :entier
    Début
      p←1
      pour i de 1 à n faire
       p←p*x
      finpour
      retourner  p
    Fin
```

T.D.O.Locaux

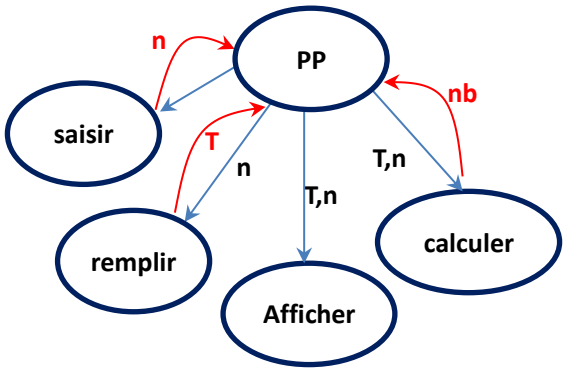
Objet	Type/Nature

			i,p	Entier

Exercice 10 :

Ecrire un script python permettant de saisir un entier **n** entre (5 et 15) puis remplir un tableau **T** par des entiers aléatoires (entre 10 et 50), ensuite afficher le tableau T, enfin calculer et afficher **nb** (le nombre des entiers pairs).

```
from numpy import *
from random import *
def saisir():
    global n
    while True:
        n=int(input('Donner n(entre 5 et 15)='))
        if 5<=n<=15:break
def remplir(n):
    global T
    for i in range(n):
        T[i]=randint(10,50)
def afficher(T,n):
    for i in range(n):
        print(T[i])
def calculer(T,n):
    nb=0
    for i in range(n) :
        if T[i] % 2==0 :
            nb=nb+1
    return nb
#Programme principal
saisir()
T=array([int()]*n)
remplir(n)
afficher(T,n)
print('le nombre des pairs=',calculer(T,n))
```



Rq :Afficher le nom de l'élève ayant la meilleure note.

Objectif 8 : Ajouter un contrôle de saisie sur les noms des élèves pour être distincts .

Objectif 9 : Ajouter un module Résultat pour afficher le rang , le nom et la moyenne de chaque élève en ordre décroissant suivant la moyenne.

Exemple d'affichage :