2.5 Correction des exercices

Exercice 1 :

Reprendre l'algorithme `MaxDeDeuxEntiers` *en affichant les résultats dans l'ordre décroissant*

```
#-*-coding:Latin-1 -*
#algo MaxDeDeuxEntiers_1

n1=float(input("Donner le premier entier "))
n2=float(input("Donner le second entier "))
if n1>n2:
  max=n1
  min=n2
else:
  max=n2
  min=n1
print("Le plus grand des deux est ",max)
print("Le plus petit des deux est ",min)
```

Il y a une façon plus « Pythonnique » d'écrire ce programme :

```
#-*-coding:Latin-1 -*
#algo MaxDeDeuxEntiers_1

n1=float(input("Donner le premier entier "))
n2=float(input("Donner le second entier "))
if n1>n2:
  max, min=n1, n2
else:
  max , min =n2,n1

print("Le plus grand des deux est ",max)
print("Le plus petit des deux est ",min)
```

Exercice 2 :

1. *Ecrire un algorithme qui trie trois nombres réels n_1, n_2, n_3 et qui donne ce tri par ordre décroissant.*

```
#-*-coding:Latin-1 -*
#algo TriDe3Nombres
  #Cet algorithme trie 3 nombres réels et les présente sous forme décroissante


x1= float(input("Donner le premier réel: "))
x2= float(input("Donner le second réel: "))
x3= float(input("Donner le troisième réel: "))
      # On commence le traitement des données
if x1>=x2:
    if x1>=x3:
        print("Le plus grand des 3 nombres est x1 ",x1)
        if x2>=x3:
            print("Le nombre moyen est x2 ",x2)
            print("Le petit nombre est x3 ",x3)
        else:
            print("Le nombre moyen est x3 ",x3)
            print("Le petit nombre est x2 ",x2)
    else:
        print("Le plus grand des 3 nombres est x3 ",x3)
        print("Le nombre moyen est x1 ",x1)
        print("Le plus petit des 3 nombres est x2 ",x2)
```

```
elif x2>=x3:
    print("Le plus grand des 3 nombres est x2 ",x2)
    if x1>=x3:
        print("Le nombre moyen est x1 ",x1)
    else:
        print("Le plus petit nombre est x3 ",x3)
else:
    print("Le plus grand nombre est x3 " , x3)
    print("Le nombre moyen est x2 ",x2)
    print("Le plus petit nombre est x1 ",x1)
```

2. *Ecrire un algorithme qui résoud une équation du second degré à coefficients dans* $\mathbb{R}$

   Nous allons, dans un premier temps, analyser le problème en réalisant un « arbre de décision » :

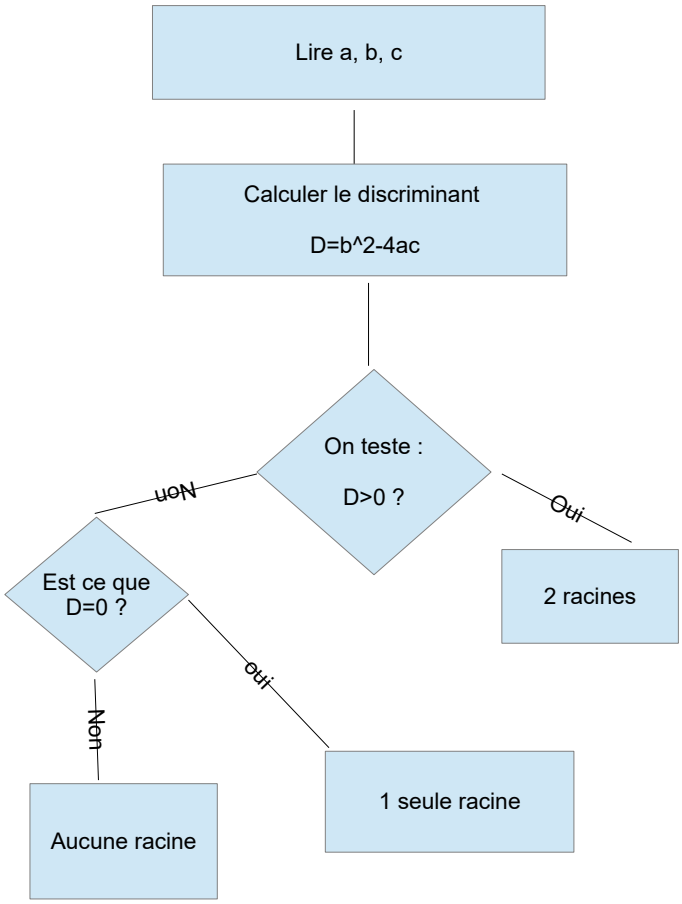

FIGURE 2.1 – L'organigramme de résolution d'une équation du second degré

```
#-*-coding:Latin-1 -*
# Algorithme de résolution d'une équation du second degré


import math #nécessité d'importer le module math:
            #nous en aurons besoin pour calculer la racine carrée
```

```
a=float(input ("Saisissez la valeur de \'a\'  "))
# l'utilisateur entre la valeur de a, le coefficient du second degré
b=float(input("saisissez la valeur de \'b \' "))
# l'utilisateur entre la valeur de b, le coefficient du terme du premier degré
c=float(input("saisissez la valeur de \'c \' "))
# l'utilisateur entre la valeur de c, le terme constant
if a==0 and b==0 and c==0:          #dans le cas où a=b=c=0
  print ("a, b, c sont égaux à 0;polynôme nul, tous les réels sont racines de l'équation
elif a==0 and b==0 and not(c==0):
  print("ce n'est pas un polynôme du second degré")
elif a==0 and not(b==0):
  S=-(c/b)
  print("Nous sommes devant un polynôme du premier degré la solution est S=",S)
elif not(a==0) and b==0 and c==0:
  print("L'équation est du type ax^2 =0; la seule solution est S=0")
elif not(a==0) and not(b==0):
  D=(b**2)-4*(a*c)
  if D<0:
    print("Le discriminant est négatif: pas de racine")
  elif D==0:
    S=-(b/(2*a))
    print("Le discriminant est nul; Il y a donc une seule solution S=",S)
  else:
    S1=(-b+math.sqrt(D))/(2*a)
    S2=(-b-math.sqrt(D))/(2*a)
    print("Le discrimant est positif; il y a 2 solutions \n")
    print("La première solution est S1= ",S1)
    print("La seconde solution est S2= ",S2)
```

Nous avons importé, ici, le module "math". Nous reviendrons, un peu plus loin sur les imports de modules

Exercice 3 :

Ecrire un algorithme qui affiche les n premiers entiers, n étant choisi par l'opérateur.

Exercice pas très difficile ; juste une adaptation de celui vu dans le cours!!!!

```
#-*-coding:Latin-1 -*

#algo affichage_des_N_premiers_entiers

n=int(input("Quel est le plus grand entier à afficher "))
compteur=1 #initialisation du compteur
while compteur<=n :
  print(compteur)
  compteur=1+compteur
```

Exercice 6 :

Ecrire un programme qui affiche tous les nombres pairs entre 0 et n, dans l'ordre croissant, le nombre n étant donné par l'opérateur.

```
#-*-coding:Latin-1 -*



#algo nombres pairs

n = int(input("Donner  le nombre jusqu'où vous souhaitez aller "))
```

```
for i in range(0,n+1,2):
    print(i)
```

Exercice 7 :

Ecrire un script qui calcule $p!$ pour un $p \in \mathbb{N}$ donné par l'utilisateur

```
#-*-coding:Latin-1 -*


#algo Factorielle

n = int(input("Donner l'entier dont on veut connaître la factorielle "))
p=1
for i in range(1,n+1):
  p=p*i

print(p)
```

Ce petit script se rapproche d'une notion de récursivité que nous verrons plus loin

Exercice 8 :

Ecrire un script qui calcule les n premiers termes de la suite :

$$\begin{cases} u_0 \in \mathbb{R}^* \\ u_{n+1} = \dfrac{1}{2}\left(u_n + \dfrac{a}{u_n}\right) \text{ avec } a \in \mathbb{R}^* \end{cases}$$

C'est le calcul de la racine carrée par la méthode des babyloniens. On remarque que si $u_0 < 0$, la suite converge vers $-\sqrt{a}$

```
#-*-coding:Latin-1 -*


#algo Racine_Carree

n = int(input("Donner le nombre de termes de la suite que vous souhaitez imprimer "))
a = float(input("Donner le réel strictement positif dont vous souhaitez calculer la racine ")
u0 = float(input("Donner le premier terme de la suite "))
while a<0:
  a = float(input("On a dit un réel strictement positif!!. "))
for i in range(1,n+1):
  u0 = (u0 + (a/u0))/2
  print("Pour n= ",i,"u= ",u0)
```

Exercice 8 :

Une assurance propose 3 tarifs selon l'âge et le nombre d'accidents des automobilistes :

1. *Le tarif Vert*
2. *Le tarif Orange*
3. *Le tarif Rouge*

	Moins de 25 ans	25 ans et plus
0 accident	Orange	vert
1 ou 2 accidents	Rouge	Orange
3 à 6 accidents	Pas assuré	Rouge
7 accidents ou plus	Pas assuré	Pas assuré

Ecrire un programme qui affiche le tarif après avoir saisi l'âge et le nombre d'accidents de l'automobiliste

Si je vous disais que ce n'est pas très difficile....Me croiriez vous ?

```
#-*-coding:Latin-1 -*


#algo Assurance_Accident


age= int(input("Quel est votre âge?  "))
accident= int(input("Combien avez vous eu d'accidents?  "))

if age <= 25:
    if accident == 0:
        print("Vous aurez un tarif Orange ")
    elif (accident == 1) or (accident == 2):
        print("Vous aurez un tarif rouge")
    else:
        print ("On n'assure pas")
else:
    if accident == 0:
        print("Vous aurez un tarif vert ")
    elif (accident == 1) or (accident == 2):
        print("Vous aurez un tarif orange")
    elif (accident >=3) and (accident <=6):
        print("Vous aurez un tarif rouge")
    else:
        print ("On n'assure pas")
```

Exercice 9 :

Vous désirez comparer 2 offres d'abonnement téléphonique. La facture est déterminée avec une somme fixe à payer obligatoirement tous plus mois, plus une partie proportionnelle au temps passé à téléphoner (indiqué en minutes)

Offre	Fixe	Prix à la minute
Opérateur N° 1	*10,00€*	*0,50 €*
Opérateur N° 2	*15,00 €*	*0,42 €*

Ecrire l'algorithme qui indique l'opérateur le plus intéressant après avoir saisi la consommation individuelle mensuelle en minutes

Cet exercice, bien que d'énoncé qui peut paraître long est de résolution très facile

```
#-*-coding:Latin-1 -*


#algo Operateur


minute= float(input("Combien de minutes?  "))
operateur1=10+0.5*minute
operateur2=15+0.42*minute
if operateur1 < operateur2 :
   print("C'est l'opérateur 1 qui est le plus intéressant")
elif operateur2 < operateur1 :
   print("C'est l'opérateur 2 qui est le plus intéressant")
else:
   print ("L'un ou l'autre, à votre choix")
```

Question : quand donc pouvons nous prendre l'un ou l'autre des opérateurs ?

Exercice 15 :

Une file d'entiers est toujours terminée par -1. Elaborer l'algorithme qui compte le nombre d'occurences du premier entier de la file.

```
#-*-coding:Latin-1 -*


#algo Occurences
#Cet algo compte le nombre d'occurence du premier entier dans une file


integ = int(input("Donner le premier entier "))

while integ == -1:
     print("Vous avez fait une erreur")
     integ = int(input("Donner le premier entier "))

premier = integ
compteur = 1
while integ != -1:
     integ = int(input("Donner l'entier suivant "))
     if integ == premier:
          compteur +=1


print("Le nombre d'occurences du premier entier ", premier," est ", compteur)
```

Exercice 16 :

Calculer le minimum et le maximum d'une suite de nombres terminée par -1

Cet algorithme est un raffinement du précédent

```
#-*-coding:Latin-1 -*



#algo OccurencesMaxMin
#Cet algo donne le maximum et le minimum d'une suite de nombres rÃ©els terminÃ©e par -1


integ = int(input("Donner le premier entier "))

while integ == -1:
     print("Vous avez fait une erreur")
     integ = int(input("Donner le premier entier "))
minimum = integ
maximum = integ
while integ !=-1:
    integ = int(input("Donner l'entier suivant "))
    if (integ >= maximum) and (integ !=-1):
        maximum = integ
    if (integ <= minimum) and (integ !=-1):
        minimum = integ



print("Le plus petit nombre de la file est:  " , minimum)
```

```
print("Le plus grand nombre de la file est:  " ,maximum)
```

Exercice 17 :

Calculez la somme et la moyenne d'une suite de notes terminées par -1 ; ces notes devant être comprises entre 0 et 20.

Pas exactement difficile!!.....Quoique!!

```
#-*-coding:Latin-1 -*

#algo SommeMoyenne
#Cet algo donne le maximum et le minimum d'une suite de nombres réels
#compris entre 0 et 20 terminée par -1


nombre= int(input("Donner le premier nombre "))#C'est l'initialisation
while nombre==-1 or nombre >20 or nombre<0:
    print("Vous avez fait une erreur, recommencez")
    nombre= int(input("Donner le premier nombre "))
somme = nombre
compteur = 1
#On continue la liste
while nombre != -1:
    nombre= int(input("Donner le nombre suivant "))
    while (nombre >20 or nombre<0)and nombre !=-1:
        nombre= int(input("Erreur: donner nombre entre 0 et 20 "))
    somme += nombre
    compteur+= 1


print("La somme est ",somme +1)
print("le compteur est ", compteur-1)
print("La moyenne est ", (somme+1)/(compteur-1))
```

Exercice 18 :

Ecrire le programme complet d'une petite application qui affiche à l'écran un triangle rectangle isocèle rempli d'étoiles et dont les côtés de ce triangle est de longueur N étoiles (N étant saisi par l'utilisateur).

```
#-*-coding:Latin-1 -*



#algo Triangle
n = int(input("Quel est la longueur du côté? "))
z = input("De quel caractère souhaitez vous remplir le triangle? ")
#rappel: la fonction input appelle un caractère de type string
for i in range(n+1):
    for j in range(i+1):
        print(z , end=""),
    print("\n")
```

Dans ce listing, on remarquera la syntaxe de retour à la ligne

Exercice 19 :

Ecrire le programme qui calcule les valeurs successives de cette suite

1. *Prendre comme valeur initiale un naturel A.*
2. *Si $A = 1$ alors* **STOP**

3. *Si A est pair, remplacer A par $\frac{A}{2}$ et aller en 2).*
4. *Si Aest impair, remplacer A par $3A+1$ et aller en 2)*

Le programme de calcul des termes de cette suite

```
#-*-coding:Latin-1 -*
#algo Distance Euclidienne

A=int(input("Donner le premier entier "))
print (A ,"\n")
i=1
while A !=1:
    if A%2==0:
        i+=1
        A=(A/2)
        print( A,"\n" )
    else:
        A=3*A+1
        i+=1
        print (A,"\n")
print("Le nombre de termes est ",i)
```

On remarquera que la suite s'arrête toujours par une suuite de termes appelée « cycle trivial » : 5, 16, 8, 4, 2 1.
Si le terme initial est 27, il faut 117 itérations pour terminer le programme et si le premier terme est 31 466 383, il en faut 706, la suite se terminant toujours par le cycle trivial.

La conjecture de Syracuse

La conjecture de Syracuse est une conjecture mathématique qui reste improuvée à ce jour. Jusqu'à présent, la conjecture de Syracuse, selon laquelle depuis n'importe quel entier positif la suite de Syracuse atteint 1, n'a pas été mise en défaut.
Par exemple, les premiers éléments de la suite de Syracuse pour un entier de départ de 10 sont : 10, 5, 16, 8, 4, 2, 1 . . .

Pour en savoir plus sur la suite de Syracuse, consultez la page Wikipedia de la suite de Syracuse

Le programme ci-après donne seulement le nombre d'itérations nécessaires pour que la suite arrive à 1

```
#-*-coding:Latin-1 -*
#algo Distance Euclidienne

A=int(input("Donner le premier entier "))
print (A)
i=1
while A !=1:
    if A%2==0:
        i+=1
        A=(A/2)
    else:
        A=3*A+1
        i+=1

print("Le nombre d'itérations est ",i)
```

Exercice 20 :

Voici un algorithme définissant une suite :

1. *Choisir un entier naturel A quelconque comme valeur initiale*
2. *Si $A = 4$, alors* **STOP**

3. *Si A se termine par le chiffre 4, barrer ce chiffre et aller en 2*
4. *Si A se termine par le chiffre 0, barrer ce chiffre et aller en 2*
5. *Mutiplier A par 2 et aller en 2*

Ecrire le programme de calcul des termes de la suite. Essayer différentes valeurs de A ; la valeur $A = 1249$ est particulièrement intéressante.

Cet exercice ne pose pas de difficultés

```
#-*-coding:Latin-1 -*
#algo Suite Amelioré

A=int(input("Donner le premier entier A "))
print (A )
i=1
while A !=4:
    if A%4==0 or A%10==0:
        i+=1
        A=(A-(A%10))/10
        print( A)
    else:
        A=2*A
        i+=1
        print (A)
print("Le nombre de termes est ",i)
```