Algorithmique et Python

1 algorithme

1.1 Algorithme et Python

Définition 1.

Un algorithme est une succession d'un nombre fini d'étapes réalisées dans un ordre précis et qui a pour but de résoudre un problème.

Les instructions doivent être formulées dans un langage compréhensible par l'exécutant.

- humain ⇒ langage courant.
- ordinateur ⇒ langage de programmation (Python).
- algorithmique ⇒ pseudo-code. C'est un langage entre le langage courant et un langage de programmation, sans norme précise, qui met l'accent sur la logique de l'algorithme.

Python C'est un langage de programmation interprété. Pour écrire un programme en Python :

1. on écrit un texte contenant les instructions (avec vocabulaire et grammaire précises).
2. on demande à Python d'exécuter le programme.
3. Python lit le texte et suit les instructions.
4. Si tout va bien, il va au bout. Si il rencontre une erreur, il s'arrête et renvoi un message d'erreur.

Il faut une distribution Python et un éditeur de code pour l'utiliser. On va privilégier les tout-en-un.

- Sur ordinateur : Installer un logiciel (Spyder, Idle, Anaconda, WinPython)
- Sur internet, dans un navigateur :
  https://www.programiz.com/python-programming/online-compiler/
  https://repl.it/languages/python3
- Sur un téléphone ou une tablette : installer une application (Pythonista 3)

1.2 Ecrire un programme en Python

Dans les logiciels cités précédemment, on trouve deux fenêtres. Un Editeur (pour taper du code) et une Console (qui donne le résultat produit par Python)

1) Editeur.
le code est « texte » contenant les instructions
Run/Executer On demande à Python d'exécuter le programme. Python lit le texte et suit les instructions.

2) Console.
Les résultats ou les erreurs s'affichent.

On peut aussi taper le code dans un logiciel autre, et le faire executer ultérieurement par un compiler Python.

Quelques éléments de base en Pyhthon :

- Les commentaires (mots, phrases que python ne lit pas) commencent par le symbole #
- Les blocs d'instruction sont identifiés par l'indentation (les espace en début de ligne).
- Des mots sont réservés à Pyhton et ne peuvent pas être utilisé pour nommer autre chose
  if, then, else, while, def, int, string, bool, print, input

Exemple.

Python

```
def Calcul(n):
  if n<2 :
    return 1
  else:
    return (n*9 -2)

a= 5
b=2+a
c =Calcul(b)
print(c)
```

2 Les variables

2.1 Définition de variable

Définition 2.

Une variable est un espace de stockage dans un emplacement de la mémoire (de l'ordinateur).

Caractéristiques :
- nom : permet d'identifier la variable ;
- type : définit quelle donnée peut contenir la variable ;
- valeur : utilisée par le programme, peut éventuellement changer ;
- (adresse : l'endroit de l'ordinateur où est stockée la variable.)

Remarque : Python gère tout seul les types de variables (c'est quand même bien de savoir avec quel type on travaille).

Définition 3.

Déclarer une variable, c'est attribuer un nom et un type à une "boite mémoire".

Affecter une variable, c'est associer une valeur à la boite.

En Python, on fait les deux à la fois et c'est Python qui gère le type et la mémoire.

Exemple. A, b, prix et Total ci-dessous sont des variables. Elles ont des types différents.

Python
```
A= 50
prix = 1.52
Total = A * prix
b= "prix de l'essence"
```

Propriété 4.

Bonnes pratiques pour nommer une variable :
- Commencer par une lettre minuscule.
- choisir un nom explicite pour bien comprendre le programme
- On peut utiliser des majuscules au milieu du nom ou bien utiliser le symbole _ pour séparer des parties du nom.
- éviter les lettres accentuées mais chiffres autorisés.

Remarque : majuscules ≠ minuscules. La variable prix n'est pas la même que la variable Prix.

Fonctions utiles pour les variables
- type(.....) donne le type de la variable qu'on met dedans
- print(....) affiche le contenu de la variable.
- print(..., ...,....) imprime plusieurs éléments.
- variable = input("Entrer la valeur de la variable") permet de faire entrer au clavier une valeur et la stocke dans la variable. Attention, la variable sera de type *string*, si on veut un nombre, il faut le convertir en faisant n=int(variable) ou n=float(variable).

Exercice 1

Jean veut faire un programme inversant le contenu de deux variables. Il propose le programme suivant :

Python
```
a=7
b=2
a=b
b=a
```

1. Expliquer pourquoi le programme ne marche pas.
2. modifier le programme pour que la valeur des variables a et b soient inversées.

2.2 Types de nombres

Les nombres peuvent être de différents types :
- int entiers relatifs (entiers négatifs, entiers positifs et 0).
- float flottant (nombres décimaux). Ils s'écrivent avec un point à la place de la virgule.
- complex Nombres complexes de la forme $a + bj$. La partie imaginaire se note avec un j, a et b sont des float. Il est obligatoire de mettre un nombre devant le j, même 0 ou 1.

Exemple. -6.9 est un float. -69 est un int. 3 est un int, mais 3.0 est un float. 2+5.1j est un complex. -9.7 + 0j aussi

Conversion de type Il est possible de convertir quelque chose en un type de nombre.
- int(x) converti x en entier
- float(x) converti x en nombre flottant.

— complex(a,b) crée le nombre complexe $a + ib$

Exemples. int(3.6) $\to$ 3, float(5) $\to$ 5.0, complex(2.3)$\to$ 2.3 + 0j, complex(2.3, 6)$\to$ 2.3 + 6.0j

Remarque : Les trop petits nombres ($< 10^{-323}$) deviennent 0. Les nombres trop grands ($> 10^{308}$) renvoient une erreur.

Opérations

— + addition : $3 + 4 \to 7$
— − soustraction : $5 - 6.7 \to -1.7$
— * multiplication : $12 * (-4)$
— / division avec résultat à virgule : $81/10 \to 8.1$
— // quotient de la division euclidienne $81//10 \to 8$
— % reste de la division euclidienne $81\%10 \to 1$
— ** Puissance : $2 * *3 \to 2^3$, on peut aussi utiliser pow(2,3)

Remarque : Les variables de type entier et de type flottant sont compatibles pour les opérations arithmétiques. Mais une opération entre entiers et flottants donnera un résultat flottant.

Pour faire d'autres fonctions, il faut charger un package mathématique (library).
On le fait en début de fichier par la commande from math import *
On peut alors utiliser (entre autre) :

— la racine carrée sqrt(...)
— les nombres π pi et e e
— la fonction exponentielle exp(...)
— la fonction ln log(...)
— les fonctions trigonométriques cos(...), sin(...), tan(...)

Raccourci d'opérations On a une variable x et on change sa valeur par des opérations simples. x += a correspond à l'opération $x = x + a$, x a été remplacée par $x + a$. ça marche aussi avec $- =$, $* =$, $/ =$, $** =$, $\% =$.

imprécision avec les float

```
x=0.1
y=x+x
print("y est-il égal à
0.2 ?" , y==0.2)
z=x+x+x
print("z est-il égal à
0.3 ?" , z==0.3)
print("z=", z)
```

y est-il égal à 0.2 ? True
z est-il égal à 0.3 ? False
z= 0.30000000000000004

Les float en Python sont stockés d'une manière "non exacte" en mémoire, et Python affiche une valeur approchée.

Exemple. 0.1 est stockée en python avec une valeur de

0.1000000000000000055511151231257827021181583404541015625

il faut s'en souvenir si on veut faire des test d'égalité avec des float.
Pour éviter ça, on peut utiliser le mode fraction from fractions import Fraction

— Fraction(n,d) permet d'utiliser la fraction $\frac{n}{d}$ de manière exacte.
— On peut convertir un float f en fraction avec Fraction(f) mais ça se base sur la valeur (approchée) du float ! Donc Fraction('nombre') est préférable.

On a aussi le mode décimal, plus précis mais plus gourmand en mémoire from decimal import Decimal. Si on met Décimal(nombre à virgule), on retrouve l'imprecision, il vaut mieux utiliser Décimal('nombre à virgule').

2.3 Chaine de caractères

Définition 5.
Une chaine de caractère, type string str, s'écrit entre des "guillemets " (ou des 'guillemets').

Exemple. "Bonjour" est un string, "landfql r gfmk " aussi. "3" aussi

Opérations

— + concatène deux chaines de caractère : "chaine1"+ "chaine2" donne "chaine1chaine2"
— "une chaine de caractère"[n] donne la lettre numéro n de la chaine de caractère, en comptant à partir de 0.
— \n dans une chaine de caractère insère un retour à la ligne
— print("une chaine") affiche la chaine de caractère
— len("une chaine") donne la longueur de la chaine de caractère

— str(un objet) convertit un objet en chaine de caractère.

Exemple. "Bonjour" + "le monde" donnera "Bonjourle monde", "Bonjourle monde"[4] donne la lettre o.

Insérer des résultats dans une phrase Les f-string sont des chaines de caractère avec des places libres prévues pour des variables ou des calculs.

On met en premier la lettre *f*, puis le guillemet " puis la chaine de caractère et on finit par ". Pour insérer la valeur d'une variable, d'un calcul ou d'une fonction, on le met entre accolade dans la phrase.

f " ..... { variable } ..... { fonction(truc) } ..... "

2.4 Booléen

Un booleen bool ne peux prendre que deux valeurs : vrai True ou faux False. Par défaut, la majorité des objets Python sont considérés comme Vrai.

- Les chaines de caractères non vides sont True. Une chaine de caractère vide est False.
- Un nombre non nul est True. Le nombre 0 est False.
- Une liste non vide est True. Une liste vide est False.
- La valeur None est False

Les booléens servent à écrire des conditions et des tests.

Exercice 2

Donner le nom, la valeur et le type des variables ci-dessous

Python

```
A = 5.2
B = \Bonjour"
C =6
D = \Choisir un nombre entier"
E =False
F =A+C
G= C - 10
H =(26 + 4*2 - (12 + 6*3))/2
```

3 Fonctions

Définition 6.

Une fonction est une suite d'instructions qui définissent un sous-programme et qui renvoient un résultat pouvant être utilisé autant de fois que nécessaire dans un programme plus général. ça évite d'écrire plusieurs fois la même séquence d'instructions.

Une fonction peut avoir besoin de paramètres : des valeurs à entrer dans la fonction. Elle peut renvoyer un résultat. Mais pas obligatoirement.

Exemple. print() et sqrt() sont des fonctions.

- print() est une fonction qui nécessite un ou plusieurs arguments : print("Hello"), print(a, b, c). Elle affiche quelque chose à l'écran, mais elle ne renvoie pas de résultat.
- sqrt() est une fonction qui nécessite un argument : sqrt(9). Elle renvoie un résultat sqrt(9) → 3.

Créer une fonction

Python

```
def NomDeLaFonction ( parametre1, parametre2, ...) :
  instructions de la fonction
  avec un décalage vertical (indentation) obligatoire
  tout ce qui est décalé est dans la fonction
  return resultat
```

Remarque :

- return n'est pas obligatoire. Si il y est, la fonction s'arrête à cet endroit et renvoie le résultat. Si il n'y est pas, la fonction s'arrêtera à la fin des instructions et renvoie None
- (parametre1, parametre 2, ...) sont les arguments de la fonction. S'il n'y a pas d'arguments, on écrit NomDeLaFonction()
- On a défini la fonction... mais on ne l'a pas exécutée! Pour ça, il faut l'appeler par NomDeLaFonction (valeur1, valeur2 ...). Si on veut récupérer le résultat, il faut prévoir de le stocker dans dans une variable MaVaraible = NomDeLaFonction (valeur1, valeur2 ...).

Exemple. Calculer le périmètre d'un carré

Python

```
def  perimetreCarre(cote):
  perimetre=4*cote
  return perimetre

a= perimetreCarre(5)
print(a)
print("le perimetre est ", perimetreCarre(8.7))
```

Exemple. Définir la fonction f sur $\mathbb{R}$ par $f(x) = 3x^2 + 2x - 4$:

Python

```
def f(x):
  return (3*x**2+2*x -4)
```

Exercice 3

Écrire en Python une fonction aireCarre() avec un argument qui renvoie l'aire d'un carré de côté **cote**.

print() ≠ return () print(Truc) se contente d'afficher Truc à l'écran, mais Truc n'est sauvegardé en mémoire nulle part et on ne peut pas l'utiliser. return(Truc) ne se trouve qu'à l'intérieur d'une fonction. Truc est alors stocké comme résultat de la fonction et utilisable ailleurs. Par exemple, perimetreCarre(5) peut être utilisé dans un calcul par la suite.

Exemple. Une fonction sans return

Python

```
def Rien():
  a=2+4
  b=a*3
  print("il ne se passe rien")

Rien()  # il va s'afficher "il ne se passe rien"
print(Rien())
# il va s'afficher "il ne se passe rien" puis None
```

4 Listes

4.1 Définition de liste

Définition 7.

Une liste (type list en Python) est une collection d'objets que l'on groupe dans une seule variable. Les objets sont séparés par des virgules et l'ensemble est enfermé dans des crochets.

UneListe =[objet0, objet1, objet 2, ..., objetN]

On peut créer une liste vide en faisant UneListe =[ ]

Exemple. maListe=["a", "b", "c", 5, 8.3, "d"]

Propriété 8.

Pour accéder aux objets de la liste, on met le nom de la liste, suivi du numéro de l'objet entre crochet. Attention la numérotation des objets commence à zéro !

Uneliste[0] donne le premier objet.
Uneliste[1] donne le deuxième objet
...
Uneliste[-1] donne le dernier objet

Exemple. maListe[1] renvoie "b", mais maListe[6] renvoie une erreur !

4.2 Opérations avec des liste

- in permet de tester si un objet appartient à une liste : Objet in UneListe renvoie Vrai si l'objet est dans la liste et Faux sinon.
- + permet de concaténer (coller) deux listes
- Uneliste[n] = b met b en position n dans la liste (en effaçant ce qui s'y trouvait)

Exemple. $[1, 2, 3] + [6, 5, 4]$ donne $[1, 2, 3, 6, 5, 4]$.

Python

```
ListeDeNombre=[2,3,6,9]
3 in ListeDeNombre      # donne True
7 in ListeDeNombre    # donne False
ListeDeNombre[3]=10   # donne [2,3,6,10]
```

Fonctions (ne modifient pas la liste passée en paramètre) :

- len(...) donne le nombre d'élément de la liste
- min(...) donne le plus petit élément de la liste
- max(...) donne le plus grand élément de la liste
- mean(...) calcule la moyenne des éléments de la liste. Nécessite from numpy import* auparavant

Exemple.

Python

```
UneListe=[3,2,5,7,5,4]
len(UneListe)     # donne 6
min(UneListe)     # donne 2
max(UneListe)     # donne  7
mean(UneListe)  # donne 4.33333
```

Méthodes (modifient la liste)

- maListe.index(objet) donne la première position de l'objet dans la liste (en commençant à compter à 0)
- maListe.append(objet) rajoute l'objet à la fin de maListe
- maListe.remove(objet) supprime la première apparition de l'objet dans maListe
- uneVariable = maListe.pop(n) retire l'objet en position n de la liste et le stocke dans uneVariable.
- maListe.sort() trie les objets dans l'ordre croissant dans la liste.
- maListe.reverse() inverse l'ordre des éléments dans la liste

Exemple.

Python

```
ListeDeNombre=[2,3,6,9]
ListeDeNombre.index(3)   # donne  1
ListeDeNombre.append(7) # modifie [2,3,6,9,7]
a= ListeDeNombre.pop(2) # modifie [2,3,9] et donne a=6
```

Exercice 4

On considère la liste suivante

maListe1 =[15,12,22,16]

Que renvoie chacune des instructions suivantes ?

- maListe1[2]
- maListe1[4]

Comment la liste est-elle modifié par les instructions suivantes ? (on considérera que chaque instruction est indépendante et que la liste est remise à son état initial entre chaque instruction)

- maListe1.append(11)
- maListe1.remove(16)
- maListe1.sort()

Compréhension de liste Une compréhension de liste permet de créer une nouvelle liste en appliquant une formule sur tous les éléments d'une liste.

nouvelleListe = [formule sur i for i in liste]

Exemple. On considère la fonction $f(x) = 2x^2$ et on veut la liste de l'image par f de tous les entiers de 0 à 4.

Python

```
maListe=[2*k**2 for k in [0,1,2,3,4] ]
# crée la liste [0,2,8,18,32]
```

Si on ne veut faire la formule que sur des éléments remplissant une certaine condition

nouvelleListe = [formule sur i for i in liste if condition sur i]

5 Condition

Définition 9.

Une condition est une opération qui renvoie un booléen True ou False.

Comparaisons

1. == (est égal) renvoie vrai si les deux valeurs sont égales
2. != (est différent) renvoie vrai si les deux valeurs ne sont pas égales
3. > (est strictement supérieur) renvoie vrai si la première valeur est strictement plus grande que la deuxième
4. >= (est supérieur ou égal) renvoie vrai si la première valeur est plus grande ou égale à la deuxième
5. < (est strictement inférieur) renvoie vrai si la première valeur est strictement plus petite que la deuxième
6. <= (est inférieur ou égal) renvoie vrai si la première valeur est plus petite ou égale à la deuxième

Exemple. Pour $x = 3$ et $y = 7$: $x == 2 \rightarrow$ False, $y! = 5 \rightarrow$ True, $x > 1 \rightarrow$ True, $y <= x \rightarrow$ False

Opérations de booléens Si (*Condition*1) et (*Condition*2) sont deux conditions :

- (*Condition*1) and (*Condition*2) est vraie si les deux conditions sont vraies
- (*Condition*1) or (*Condition*2) est vraie si une des deux conditions est vraie
- not (*Condition*1) est vraie si (*Condition*1) est fausse. (not est la négation)

Exemple. Pour $x = 3$ et $y = 7$,

- $(x == 3)$ and $(y == 7) \rightarrow$ True
- $(x > 1)$ and $(y > 10) \rightarrow$ False
- $(x > 1)$ or $(y > 10) \rightarrow$ True
- not$(x == 1) \rightarrow$ True.

Exercice 5

Déterminer la valeur (Vrai ou Faux) de chacun des tests suivants :

- $(1 < 2)$ and $(6 < 5)$
- $(4 == 2 + 2)$ and $(6! = 7)$
- $(3 <= 2 + 1)$ and $(2 > 3)$
- $(3 <= 2 + 1)$ or $(2 > 3)$
- not (6 !=7)

6 Instructions conditionnelles

Définition 10.

Une instruction conditionnelle est une instruction qui dépend d'une condition. Certaines parties du code sont exécutées ou ignorées selon que la condition est vérifiée ou non

Python

```
if condition :
  instruction si la condition est vraie
  avec indentation obligatoire
else :
  instruction si la condition est fausse
  facultatif
  mais indentation toujours obligatoire

le changement d'indentation signale
la fin de la structure conditionnelle
```

Exemple. On veut créer une fonction qui teste si un nombre A est pair, puis qui calcule $A/2$ si c'est le cas et $A + 1$ si A est impaire. Pour savoir si A est pair, il suffit de regarder si le reste de sa division euclidienne par 2 A % 2 est nul.

Python

```
def calculBizarre (A):
  if (A% 2)== 0 :
    A=A/2
  else :
    A= A+1
  return A

calculBizarre(27) # renvoie 82
calculBizarre(26) # renvoie 13.0
```

Exercice 6

Programmer une fonction avec Python qui reçoit en argument la longueur et la largeur d'un rectangle. Cette fonction doit renvoyer True si le rectangle est un carré et False sinon.

Remarque : Si... alors... sinon... permet de séparer deux cas. Si on veut faire plus de cas, on peut imbriquer plusieurs **Si** (ou utiliser d'autres structures).

Exemple. Ecrire une fonction qui compare deux nombres x et y et qui renvoie le plus grand des deux si il sont différents, et un message si il sont égaux. Il y a trois cas : $x < y$, $x = y$ et $x > y$.

Python

```
def PlusGrand (x,y) :
  if (x==y):
    return "Les valeurs sont égales"
  else :
    if (x>y):
      return x
    else :
      return y
PlusGrand(6, 2*3)    # renvoie "Les valeurs sont égales"
PlusGrand(95,76)  # renvoie 95
```

Plutôt que d'utiliser des si-alors-sinon imbriqués, on peut utiliser elif en Python (raccourci de else if)

Python

```
def PlusGrand (x,y) :
  if (x==y):
    return "Les valeurs sont égales"
  elif (x>y):
    return x
  else :
    return y
```

7 Boucles bornées

Définition 11.

Dans un algorithme, une boucle est une suite d'instructions qu'on répète en boucle un certain nombre de fois.

on dit que la boucle est bornée lorsqu'on sait exactement combien de fois la boucle va se répéter. On parle aussi de boucle Pour ou boucle for

Python

```
for variable in liste de valeur :
  instructions à faire avec la variable
  autant qu'on veut
le retour d'indentation signale la fin du for
```

liste de valeur peut-être :

- une liste de nombre. Ou de tout autre objet.
- range(n) avec $n \in \mathbb{N}^*$. correspond à la liste des entiers de 0 à $n-1$
- range(k,n) avec $k, n \in \mathbb{N}$ correspond à la liste des entiers de k à $n-1$.
- range(k,n,p) avec $k, n, p \in \mathbb{N}$ correspond à la liste des entiers de k à $n-1$ avec un pas de p (c'est à dire un entier sur p)
- un string la variable va parcourir les caractères.

Exemple.

Python

```
for i in range(10) :
  print(i)
  # affiche la liste des entiers de 0 à 9
for i in ["a", "e", "r"]:
  print(i)
  # affiche successivement a, e et r
```

Exercice 7

1. Ecrire un programme qui affiche la table de multiplication par 4 de 4×0 à 4×10
2. Modifier le programme pour qu'il affiche la table de multiplication par 4 de 4×3 à 4×8
3. Ecrire une fonction **TableMultiplication(n)** qui affiche la table de multiplication par n de $n\times0$ à $n\times10$

8 Boucle non bornée

Définition 12.

On dit qu'une boucle est non bornée lorsqu'on ne sait pas à l'avance le nombre d'itérations de la boucle nécessaire.

Un test d'arrêt est effectué à chaque passage de la boucle : si la condition est vraie, on continue. Si la condition est fausse, on s'arrête.

Python

```
while condition :
  instructions à faire
  avec indentation
le changement d'indentation indique la fin de la boucle
```

Danger ! le test d'arrêt est obligatoire et il faut s'assurer que la condition d'arrêt soit vérifiée à un moment. Sinon ça crée une boucle infinie qui fait planter le programme.

Exemple. On cherche $x = \sqrt{13}$, c'est-à-dire x tel que $x^2 = 13$. On veut une valeur approchée de x à 10^{-2} près. On commence avec une valeur de $x = 3$ car $x^2 = 9$ est proche de 13 et on teste tous les nombres de 0,01 en 0,01 jusqu'à trouver celui dont le carré dépasse tout juste 13.

Python

```
x=3
while (x**2 <13) :
  x=x+0.01
print(x)     # résultat 3.609999999999987
```

Remarque : On ne peut pas mettre comme condition d'arrêt que $x^2 = 13$ car il n'y a aucune chance de tomber pile dessus !

Exercice 8

Python

```
i=0
while i<10:
  print(i)
  i=i+1
print("fin")
```

1. Qu'est-ce qu'affiche ce programme ?
2. Modifier le programme pour qu'il affiche les nombres de 1 à 15
3. Modifier le programme pour qu'il affiche les nombres pairs de 2 à 18
4. Modifier le programme pour qu'il affiche "fin" après chacun des nombre.

Classe préparatoire ATS **mathématiques**

Fiche d'exercices : Algorithme et Python

Exercice 1

Jean veut faire un programme affichant le résultat de l'opération 732+27. Il propose trois programmes différent :

```
print("732 + 27") #programme 1
print("732" + "27") #programme 2
print(732 + 27) #programme 3
```

1. Quel programme affiche le résultat souhaité ?
2. Donner le résultat affiché par les autres programmes.

Exercice 2

La bibliothèque **math** permet d'accéder à la variable **pi** ($\approx \pi$)

1. Afficher la valeur de la variable pi dans la console Python.
2. Écrire un algorithme en Python qui calcule et affiche le périmètre d'un cercle de rayon donné et l'aire d'un disque de même rayon.

Exercice 3

1. Programmer en Python la fonction g défini sur $\mathbb{R}$ avec $g(x) = -2x^3 + 4x - 7$
2. Afficher l'image des nombres -3 ; 0 ; $\frac{5}{2}$ et $\frac{5}{7}$ par la fonction g.

Exercice 4

Compléter la fonction tri suivante qui prend en paramètre deux listes de nombre et renvoie la liste obtenue en rangeant dans l'ordre décroissant la concaténation des deux listes.

Python

```
def tri(A,B) :
  out=.........     #concatenation des listes
  out .........       # tri des listes
  out .........
  return out
```

Exercice 5

Que renvoie la fonction suivante ?

Python

```
def mystery():
  A=[k-7 for k in [8,9,10]]
  A.append(0)
  A.append(22)
  A.reverse()
  A=A+[3,5,11,9]
  A.sort()
  return A
```

Exercice 6

1. Proposer une fonction **mult** prenant en parametre liste1 (une liste de nombre) et k un réel, et qui renvoie une autre liste contenant les éléments de liste1 multiplié par k.
2. Tester avec $liste1 = [2, 4, 7, 1, 5]$ et $k = 3$. On doit obtenir $[6, 12, 21, 3, 15]$

Exercice 7

1. Ecrire une fonction **EstPaire(nombre)** qui retourne Vrai si le nombre est un multiple de 2 et faux dans le cas contraire.
2. Programmer une fonction **MultipleCinq(Nombre)** qui retourne Vrai si le nombre est un multiple de 5 et faux dans le cas contraire.
3. On considère le programme ci-dessous :

Python

```
def Mystere(Nombre):
  return EstPaire(Nombre) and MultipleCinq(Nombre)
```

Quel est le but de cette fonction ?

Exercice 8

On considère deux points A et B de coordonnées respectives (x_A, y_A) et (x_B, y_B). On recherche la droite passant par ces deux points.

1. Programmer une fonction en Python qui prend argument les coordonnées des deux points, et qui renvoie Vrai si les deux points ont la même abscisses et Faux sinon.
2. Utiliser cette fonction pour créer un programme qui reçoit en argument les coordonnées de deux points et qui renvoie l'équation de la droite (AB) sous la forme $x = k$ ou $y = mx + p$.

Exercice 9

On considère la fonction f définie sur $\mathbb{R}$ par $f(x) = x^2$.

1. Définir cette fonction avec Python.
2. Ecrire un programme qui affiche les images des entiers entre -3 et 3 par la fonction f
3. Modifier ce programme pour n'afficher que les images des entiers pairs entre -3 et 3.

Exercice 10

On considère le programme suivant

Python

```
mot="MATHEMATIQUES"
n=0
for i in mot:
  if i=="A":
    n=n+1
```

1. Quel est le type de la variable **mot** ? Et quelle sont les valeurs successives prises par i ?
2. Que fais ce programme ?
3. Ecrire un programme Python qui affiche le nombre de lettre A et de lettre N dans le mot

INTERGOUVERNEMENTALISATIONS

Exercice 11

On considère le programme suivant

Python

```
for i in range(5):
  for j in range(4):
    print(i+j)
```

1. Quelle est la première valeur prise par i et j ? Quelle est la première valeur affichée ?
2. Après l'affichage de la première valeur, quel compteur va changer en premier : i ou j ? pour prendre quelle valeur ? Quelle est alors la deuxième valeur affichée ?

Exercice 12

On considère la chaine de caractère "AEIOUY" représentant les voyelles de l'alphabet français. Ecrire un programme qui compte le nombre de voyelle dans un mot donné en majuscule. Tester avec MATHEMATIQUES et INTERGOUVERNEMENTALISATIONS.

Exercice 13

La fonction x^3 est continue, strictement croissante et on a $(-3)^3 = -27$ et $(-2)^3 = -8$. Comme $-27 < -20 < -8$, par le théorème des valeurs intermédiaires, il existe $c \in [-3, -2]$ tel que $c^3 = -20$. On veut déterminer une valeur approchée de c à 10^{-2}. Ecrire un algorithme en python permettant de trouver la valeur approchée de c.

Exercice 14

Jonathan dispose de 10 000€ en banque avec un taux d'intérêt composé à 2%. (Il gagne 2% du montant de l'année précédente).
On suppose que 10 000€ correspond à l'année 0.

1. Calculer à la main le montant en banque lors de l'année 2.
2. Ecrire un programme avec Python qui calcule le montant en banque après la dixième année.
3. Jonathan souhaite avoir au moins 15 000€. Ecrire un programme qui utilise une boucle non bornée pour déterminer le nombre d'année à attendre.

Exercice 15

Un entier naturel est dit premier lorsqu'il n'est divisible que par 1 et lui-même. On veut écrire un programme qui déterminer si un entier naturel est un nombre premier. On définit le programme suivant :

Python

```
def EstPremier(n):
  i=2
  while n%i != 0:
    i=i+1
  return i
```

1. Justifier que la boucle n'est pas infinie.
2. Dans la ligne 2, pourquoi ne pas avoir commencé par $i = 0$ ou $i = 1$?
3. Comment utiliser cette fonction pour déterminer si n est premier ou non ? Ecrire une fonction **testPremier(n)** qui renvoie vraie si n est premier et Faux sinon.