2022 sujet 1

Exercice 1

```
def recherche(caractere, mot):
    occurrences = 0
    for c in mot:
        if caractere == c:
            occurrences += 1
    return occurrences
```

Exercice 2

```
pieces = [100,50,20,10,5,2,1] # Erreur de majuscule dans le sujet
def rendu_glouton(arendre, solution=[], i=0):
    if arendre == 0: # Erreur d'indentation dans le sujet
        return solution
    p = pieces[i]
    if p <= arendre :
        solution.append(p)
        return rendu_glouton(arendre - p, solution, i)
    else :
        return rendu_glouton(arendre, solution, i+1)
```

2022 sujet 2

Exercice 1

```
def moyenne(notes):
    somme = 0
    somme_coeff = 0
    for couple in notes:
        somme += couple[1] * couple[0]
        somme_coeff += couple[1]
    return somme / somme_coeff
```

Exercice 2

```
# Erreurs dans l'énoncé :
# Le triangle contient les lignes 1 à n+1
# Ck contient la ligne k+1

def pascal(n):
    C= [[1]]
    for k in range(1,n+1):
        Ck = [1]
        for i in range(1,k):
            Ck.append(C[k-1][i-1]+C[k-1][i] )
        Ck.append(1)
        C.append(Ck)
    return C
```

2022 sujet 3

Exercice 1

```
def delta(tab):
    precedente = 0
    diff = []
    for val in tab:
        diff.append(val-precedente)
        precedente = val
    return diff
```

Exercice 2

```
class Noeud:
    def __init__(self, g, v, d):
        self.gauche = g
        self.valeur = v
        self.droit = d

    def __str__(self):
        return str(self.valeur)

    def est_une_feuille(self):
        '''Renvoie True si et seulement si le noeud est une feuille'''
        return self.gauche is None and self.droit is None


def expression_infixe(e):
    s = ""
    if e.gauche is not None:
        s = s + expression_infixe(e.gauche)
    s = s + str(e.valeur)
    if e.droit is not None:
        s = s + expression_infixe(e.droit)
    if e.gauche is None and e.droit is None:
        return s

    return '('+ s +')'


e = Noeud(Noeud(Noeud(None, 3, None), '*', Noeud(Noeud(None, 8, None),
'+', Noeud(None, 7, None))), '-', Noeud(Noeud(None, 2, None), '+',
Noeud(None, 1, None)))
```

2022 sujet 4

Exercice 1

```
def recherche(tab):
    liste = []
    for i in range(1, len(tab)):
        if tab[i] - tab[i-1] == 1:
            liste.append((tab[i-1], tab[i]))
    return liste
```

Exercice 2

```
def propager(M, i, j, val):
    if M[i][j]== 0:
        return

    M[i][j]=val

    # l'élément en haut fait partie de la composante
    if ((i-1) >= 0 and M[i-1][j] == 1):
        propager(M, i-1, j, val)

    # l'élément en bas fait partie de la composante
    if ((i+1) < len(M) and M[i+1][j] == 1):
        propager(M, i+1, j, val)

    # l'élément à gauche fait partie de la composante
    if ((j-1) >= 0 and M[i][j-1] == 1):
        propager(M, i, j-1, val)

    # l'élément à droite fait partie de la composante
    if ((j+1) < len(M) and M[i][j+1] == 1):
        propager(M, i, j+1, val)


M = [[0,0,1,0],[0,1,0,1],[1,1,1,0],[0,1,1,0]]
```

2022 sujet 5

Exercice 1

```
def rechercheMinMax(tab):
    if tab == []:
        return {'min': None, 'max': None}
    else:
        mini = tab[0]
        maxi = tab[1]
        for i in range(1, len(tab)):
            if tab[i] > maxi:
                maxi = tab[i]
            if tab[i] < mini:
                mini = tab[i]
        return {'min': mini, 'max': maxi}

tableau = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]
```

Exercice 2

```
class Carte:
    """Initialise Couleur (entre 1 à 4), et Valeur (entre 1 à 13)"""
    def __init__(self, c, v):
        assert 1 <= c <= 4, "La couleur n'est pas correcte"
        assert 1 <= v <= 13, "La valeur n'est pas correcte"
        self.Couleur = c
        self.Valeur = v

    """Renvoie le nom de la Carte As, 2, ... 10,
       Valet, Dame, Roi"""
    def getNom(self):
        if ( self.Valeur > 1 and self.Valeur < 11):
            return str( self.Valeur)
        elif self.Valeur == 11:
            return "Valet"
        elif self.Valeur == 12:
            return "Dame"
        elif self.Valeur == 13:
            return "Roi"
        else:
            return "As"
```

```
    """Renvoie la couleur de la Carte (parmi pique, coeur, carreau, trefle"""
    def getCouleur(self):
        return ['pique', 'coeur', 'carreau', 'trefle' ][self.Couleur - 1]

class PaquetDeCarte:
    def __init__(self):
        self.contenu = []

    """Remplit le paquet de cartes"""
    def remplir(self):
        for couleur in range(1, 5):
            for valeur in range(1, 14):
                self.contenu.append(Carte(couleur, valeur))

    """Renvoie la Carte qui se trouve à la position donnée"""
    def getCarteAt(self, pos):
            # On n'obtient pas le 6 de coeur mais cela semble
            # impossible avec un remplissage 'logique'
        assert 1 <= pos <= 52, "Position non valide"
        return self.contenu[pos-1]
```

2022 sujet 6

Exercice 1

```
def maxi(tab):
    maximum = tab[0]
    for i in range(len(tab)):
        if tab[i] > maximum:
            maximum = tab[i]
            indice = i
    return maximum, indice
```

Exercice 2

```
def recherche(gene, seq_adn):
    n = len(seq_adn)
    g = len(gene)
    i = 0
    trouve = False
    while i < n - g + 1 and trouve == False :
        j = 0
        while j < g and gene[j] == seq_adn[i+j]:
            j += 1
        if j == g:
            trouve = True
        i += 1
    return trouve
```

2022 sujet 7

Exercice 1

```
def conv_bin(n):
    b = []
    bit = 0
    while n != 0:
        bit += 1
        b.append(n % 2)
        n = n // 2
    b.reverse()
    return b, bit
```

Exercice 2

```
def tri_bulles(T):
    n = len(T)
    for i in range(n-1, 0, -1):
        for j in range(i):
            if T[j] > T[j+1]:
                temp = T[j]
                T[j] = T[j+1]
                T[j+1] = temp
    return T
```

2022 sujet 8

Exercice 1

```
def recherche(elt, tab):
    indice = -1
    for i in range(len(tab)):
        if tab[i] == elt:
            indice = i
            break
    return indice
```

Exercice 2

```
def insere(a, tab):
    l = list(tab) # l contient les mêmes éléments que tab
    l.append(a)
    i = len(tab) - 1 # On commence à l'avant dernier élément de l
    while a < l[i] and i >= 0:
      l[i+1] = l[i]
      l[i] = a
      i = i - 1
    return l
```

2022 sujet 9

Exercice 1

```
def calcul(k): # Erreur dans l'énoncé
    liste = [k]
    while k > 1:
        if k%2 == 0:
            k = k // 2
        else:
            k = 3*k + 1
        liste.append(k)
    return liste
```

Exercice 2

```
dico = {"A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, \
        "H":8, "I":9, "J":10, "K":11, "L":12, "M":13, \
        "N":14, "O":15, "P":16, "Q":17, "R":18, "S":19, \
        "T":20, "U":21,"V":22, "W":23, "X":24, "Y":25, "Z":26}

def est_parfait(mot) :
    #mot est une chaîne de caractères (en lettres majuscules)
    code_c = ""
    code_a = 0
    for c in mot :
        code_c = code_c + str(dico[c])
        code_a = code_a + dico[c]
    code_c = int(code_c)
    if code_c % code_a == 0 :
        mot_est_parfait = True
    else :
        mot_est_parfait = False
    return [code_a, code_c, mot_est_parfait]
```

2022 sujet 10

Exercice 1

```
def occurrence_lettres(phrase):
    dico = dict()
    for c in phrase:
        if c in dico:
            dico[c] += 1
        else:
            dico[c] = 1
    return dico
```

Exercice 2

```
def fusion(L1,L2):
    n1 = len(L1)
    n2 = len(L2)
    L12 = [0]*(n1+n2)
    i1 = 0
    i2 = 0
    i = 0
    while i1 < n1 and i2 < n2 :
        if L1[i1] < L2[i2]:
            L12[i] = L1[i1]
            i1 = i1 + 1
        else:
            L12[i] = L2[i2]
            i2 = i2 + 1
        i += 1
    while i1 < n1:
        L12[i] = L1[i1]
        i1 = i1 + 1
        i = i + 1
    while i2 < n2:
        L12[i] = L2[i2]
        i2 = i2 + 1
        i = i + 1
    return L12
```

2022 sujet 11

Exercice 1

```
def recherche(tab, n):
    deb = 0
    fin = len(tab) - 1
    mil = (fin + deb) // 2
    while deb <= fin:
        if tab[mil] == n:
            return mil
        elif tab[mil] < n:
            deb = mil + 1
        else:
            fin = mil - 1
        mil = (fin + deb) // 2
    return -1
```

Exercice 2

```
ALPHABET='ABCDEFGHIJKLMNOPQRSTUVWXYZ'

def position_alphabet(lettre):
    return ALPHABET.find(lettre)

def cesar(message, decalage):
    resultat = ''
    for lettre in message :
        if lettre in ALPHABET :
            indice = (position_alphabet(lettre)  + decalage) % 26
            resultat = resultat + ALPHABET[indice]
        else:
            resultat = resultat + lettre
    return resultat
```

2022 sujet 12

Exercice 1

```
def moyenne(tab):
    if tab == []:
        return 'erreur'
    else:
        somme = 0
        for k in tab:
            somme += k
        return somme / len(tab)
```

Exercice 2

```
def tri(tab):
    # i est le premier indice de la zone non triee, j le dernier indice.
    # Au debut, la zone non triee est le tableau entier.
    i = 0
    j = len(tab) - 1
    while i != j :
        if tab[i] == 0:
            i = i + 1
        else :
            valeur = tab[j]
            tab[j] = tab[i]
            tab[i] = valeur
            j = j - 1
    return tab
```

2022 sujet 13

Exercice 1

```
def rendu(somme_a_rendre):
    liste_pieces = [5, 2, 1]
    p = 0
    nb_pieces = [0, 0, 0]
    while somme_a_rendre > 0:
        if somme_a_rendre >= liste_pieces[p]:
            somme_a_rendre = somme_a_rendre - liste_pieces[p]
            nb_pieces[p] += 1
        else:
            p = p + 1
    return nb_pieces
```

Exercice 2

```
class Maillon :
    def __init__(self,v) :
        self.valeur = v
        self.suivant = None

class File :
    def __init__(self) :
        self.dernier_file = None

    def enfile(self,element) :
        nouveau_maillon = Maillon(element)
        nouveau_maillon.suivant = self.dernier_file
        self.dernier_file = nouveau_maillon

    def est_vide(self) :
        return self.dernier_file == None

    def affiche(self) :
        maillon = self.dernier_file
        while maillon != None :
            print(maillon.valeur)
            maillon = maillon.suivant

    def defile(self) :
```

```
if not self.est_vide() :
    if self.dernier_file.suivant == None :
        resultat = self.dernier_file.valeur
        self.dernier_file = None
        return resultat
    maillon = self.dernier_file
    # On travaille sur le maillon suivant
    # pour ne pas devoir modifier le maillon précédent
    # auquel nous n'avons plus accès !
    while maillon.suivant.suivant != None :
        maillon = maillon.suivant
    resultat = maillon.suivant.valeur
    maillon.suivant = None
    return resultat
return None
```

2022 sujet 14

Exercice 1

```
def correspond(mot, mot_a_trous):
    # On écarte tout de suite les mots de tailles différentes
    if len(mot) != len(mot_a_trous):
        return False
    for i in range(len(mot_a_trous)):
        if mot_a_trous[i] != "*" and mot_a_trous[i] != mot[i]:
            return False
    return True
```

Exercice 2

```
def est_cyclique(plan):
    '''
    Prend en paramètre un dictionnaire `plan` correspondant
    à un plan d'envoi de messages entre `N` personnes A, B, C,
    D, E, F ...(avec N <= 26).
    Renvoie True si le plan d'envoi de messages est cyclique
    et False sinon.
    '''
    personne = 'A'
    N = len(plan)
    for i in range(N-1):
        if plan[personne] == 'A':
            return False
        else:
            personne = plan[personne]
    return True
```

2022 sujet 15

Exercice 1

```
def nb_repetitions(elt, tab):
    nb = 0
    for e in tab:
        if e == elt:
            nb += 1
    return nb
```

Exercice 2

```
# L'illustration est trompeuse car elle ne montre pas
# la dernière division !

def binaire(a):
    bin_a = str(a%2)
    a = a // 2
    while a != 0 :
        bin_a = str(a%2) + bin_a
        a = a // 2
    return bin_a
```

2022 sujet 16

Exercice 1

```
def maxi(tab):
    maximum = tab[0]
    indice = 0
    for i in range(len(tab)):
        if tab[i] > maximum:
            indice = i
            maximum = tab[i]
    return maximum, indice
```

Exercice 2

```
def positif(T):
    T2 = list(T)
    T3 = []
    while T2 != []:
        x = T2.pop()
        if x >= 0:
            T3.append(x)
    T2 = []
    while T3 != []:
        x = T3.pop()
        T2.append(x)
    print('T = ',T)
    return T2
```

2022 sujet 17

Exercice 1

```
def nombre_de_mots(phrase):
    compteur = 0
    for c in phrase:
        if c == " ":
            compteur += 1
    if phrase[-1] != ".":
        return compteur
    else:
        return compteur + 1
```

Exercice 2

```
class Noeud:
    '''
    Classe implémentant un noeud d'arbre binaire
    disposant de 3 attributs :
    - valeur : la valeur de l'étiquette,
    - gauche : le sous-arbre gauche.
    - droit : le sous-arbre droit.
    '''
    def __init__(self, v, g, d):
        self.valeur = v
        self.gauche = g
        self.droite = d

class ABR:
    '''
    Classe implémentant une structure
    d'arbre binaire de recherche.
    '''

    def __init__(self):
        '''Crée un arbre binaire de recherche vide'''
        self.racine = None

    def est_vide(self):
        '''Renvoie True si l'ABR est vide et False sinon.'''
        return self.racine is None
```

```
    def parcours(self, tab = []):
        '''
        Renvoie la liste tab complétée avec tous les éléments de
        l'ABR triés par ordre croissant.
        '''
        if self.est_vide():
            return tab
        else:
            self.racine.gauche.parcours(tab)
            tab.append(self.racine.valeur)
            self.racine.droite.parcours(tab)
            return tab

    def insere(self, element):
        '''Insère un élément dans l'arbre binaire de recherche.'''
        if self.est_vide():
            self.racine = Noeud(element, ABR(), ABR())
        else:
            if element < self.racine.valeur:
                self.racine.gauche.insere(element)
            else :
                self.racine.droite.insere(element)

    def recherche(self, element):
        '''
        Renvoie True si element est présent dans l'arbre
        binaire et False sinon.
         '''
        if self.est_vide():
            return False
        else:
            if element < self.racine.valeur:
                return self.racine.gauche.recherche(element)
            elif element > self.racine.valeur:
                return self.racine.droite.recherche(element)
            else:
                return True

a = ABR()
a.insere(7)
a.insere(3)
a.insere(9)
a.insere(1)
a.insere(9)
```

2022 sujet 18

Exercice 1

```
t_moy = [14.9, 13.3, 13.1, 12.5, 13.0, 13.6, 13.7]
annees = [2013, 2014, 2015, 2016, 2017, 2018, 2019]

def mini(releve, date):
    d = date[0]
    r = releve[0]
    for i in range(len(releve)):
        if releve[i] < r:
            r = releve[i]
            d = date[i]
    return r, d
```

Exercice 2

```
def inverse_chaine(chaine):
    result = ""
    for caractere in chaine:
       result = caractere + result
    return result

def est_palindrome(chaine):
    inverse = inverse_chaine(chaine)
    return chaine == inverse

def est_nbre_palindrome(nbre):
    chaine = str(nbre)
    return est_palindrome(chaine)
```

2022 sujet 19

Exercice 1

```
def multiplication(n1, n2):
    res = 0
    if n1 > 0:
        for i in range(n1):
            res += n2
    elif n1 < 0:
        for i in range(-n1):
            res -= n2
    return res
```

Exercice 2

```
def chercher(T,n,i,j):
    if i < 0 or j >= len(T) :
        print("Erreur")
        return None
    if i > j :
        return None
    m = (i+j) // 2
    if T[m] < n :
        return chercher(T, n, m+1 , j)
    elif T[m] > n :
        return chercher(T, n, i , m-1 )
    else :
        return m
```

2022 sujet 20

Exercice 1

```
def xor(tab1, tab2):
    res = []
    for i in range(len(tab1)):
        if tab1[i] == tab2[i]:
            res.append(0)
        else:
            res.append(1)
    return res


# Tests

a = [1,0,1,0,1,1,0,1]
b = [0,1,1,1,0,1,0,0]
c = [1,1,0,1]
d = [0,0,1,1]

assert(xor(a, b) == [1, 1, 0, 1, 1, 0, 0, 1])
assert(xor(c, d) == [1, 1, 1, 0])
```

Exercice 2

```
class Carre:
    def __init__(self, tableau = [[]]):
        self.ordre = len(tableau)
        self.valeurs = tableau

    def affiche(self):
        '''Affiche un carré'''
        for i in range(self.ordre):
            print(self.valeurs[i])

    def somme_ligne(self, i):
        '''Calcule la somme des valeurs de la ligne i'''
        return sum(self.valeurs[i])

    def somme_col(self, j):
        '''calcule la somme des valeurs de la colonne j'''
```

```
        return sum([self.valeurs[i][j] for i in range(self.ordre)])

def est_magique(carre):
    n = carre.ordre
    s = carre.somme_ligne(0)

    #test de la somme de chaque ligne
    for i in range(1, n):
        if carre.somme_ligne(i) != s:
            return False

    #test de la somme de chaque colonne
    for j in range(n):
        if carre.somme_col(j) != s:
            return False

    #test de la somme de chaque diagonale
    if sum([carre.valeurs[k][k] for k in range(n)]) != s:
            return False
    if sum([carre.valeurs[k][n-1-k] for k in range(n)]) != s:
            return False

    return s

# Tests

c2 = Carre([[1,1],[1,1]])
c3 = Carre([[2,9,4],[7,5,3],[6,1,8]])
c4 = Carre([[4,5,16,9],[14,7,2,11],[3,10,15,6],[13,12,8,1]])

assert(est_magique(c2) == 2)
assert(est_magique(c3) == 15)
assert(est_magique(c4) == False)
```

2022 sujet 21

Exercice 1

```
def multiplication(n1, n2):
    res = 0
    if n1 > 0:
        for i in range(n1):
            res += n2
    elif n1 < 0:
        for i in range(-n1):
            res -= n2
    return res
```

Exercice 2

```
def dichotomie(tab, x):
    """
        tab : tableau d'entiers trié dans l'ordre croissant
        x : nombre entier
        La fonction renvoie True si tab contient x et False sinon
    """

    debut = 0
    fin = len(tab) - 1
    while debut <= fin:
        m = (debut + fin) // 2
        if x == tab[m]:
            return True
        if x > tab[m]:
            debut = m + 1
        else:
             fin = m - 1
    return False
```

2022 sujet 22

Exercice 1

```
def renverse(chaine):
    s = ""
    for c in chaine:
        s = c + s
    return s
```

Exercice 2

```
# Erreur dans l'énoncé tab[0] = False et tab[1] = False

def crible(N):
    """renvoie un tableau contenant tous les nombres premiers plus petit que N"""
    premiers = []
    tab = [True] * N
    tab[0], tab[1] = False, False
    for i in range(2, N):
        if tab[i] == True:
            premiers.append(i)
            for multiple in range(2*i, N, i):
                tab[multiple] = False
    return premiers

assert crible(40) == [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]
```

2022 sujet 23

Exercice 1

```
def max_dico(dico):
    valeur_max = -1
    for cle, valeur in dico.items():
        if valeur > valeur_max:
            valeur_max = valeur
            cle_max = cle
    return cle_max, valeur_max
```

Exercice 2

```
class Pile:
    """Classe définissant une structure de pile."""
    def __init__(self):
        self.contenu = []

    def est_vide(self):
        """Renvoie le booléen True si la pile est vide, False sinon."""
        return self.contenu == []

    def empiler(self, v):
        """Place l'élément v au sommet de la pile"""
        self.contenu.append(v)

    def depiler(self):
        """
        Retire et renvoie l'élément placé au sommet de la pile,
        si la pile n'est pas vide.
        """
        if not self.est_vide():
            return self.contenu.pop()


def eval_expression(tab):
    p = Pile()
    for element in tab:
        if element != '+' and element != '*':
            p.empiler(element)
        else:
```

```
        if element == '+':
            resultat = p.depiler() + p.depiler()
        else:
            resultat = p.depiler() * p.depiler()
        p.empiler(resultat)
return p.depiler()
```

2022 sujet 24

Exercice 1

```python
def maxliste(tab):
    maxi = tab[0]
    for i in range(1, len(tab)):
        if tab[i] > maxi:
            maxi = tab[i]
    return maxi
```

Exercice 2

```python
# Attention !!!
# Ça fonctionne car les assert sont placés dans le bon ordre.
# Il ne faut jamais mettre un objet mutable comme valeur par défaut
# d'un paramètre de fonction car il est évalué une seule fois
# lorsque la fonction est définie.
# Ainsi, dans ce programme le tableau valeur par défaut se conserve
# entre chaque instance !
# À cause de cela un assert parenthesage("((()())(()))") == True
# placé à la fin ne fonctionnera pas car le tableau valeurs n'est
# pas vide au début.

class Pile:
    """ Classe définissant une pile """
    def __init__(self, valeurs=[]): # Grosse erreur !!! Il vaut mieux
        # ne pas mettre de valeur par défaut dans ce cas
        self.valeurs = valeurs

    def est_vide(self):
        """Renvoie True si la pile est vide, False sinon"""
        return self.valeurs == []

    def empiler(self, c):
        """Place l'élément c au sommet de la pile"""
        self.valeurs.append(c)

    def depiler(self):
        """
            Supprime l'élément placé au sommet de la pile, à condition
            qu'elle soit non vide
```

```
        """
        if self.est_vide() == False:
            self.valeurs.pop()


def parenthesage (ch):
    """Renvoie True si la chaîne ch est bien parenthésée et False sinon"""
    p = Pile() # Pour éviter tout problème : p = Pile([])
    for c in ch:
        if c == "(":
            p.empiler(c)
        elif c == ")":
            if p.est_vide():
                return False
            else:
                p.depiler()
    return p.est_vide()


assert parenthesage("((()())(()))") == True
assert parenthesage("())(()") == False
assert parenthesage("(())(()") == False
```

2022 sujet 25

Exercice 1

```
def selection_enclos(animaux, num_enclos):
    tab = []
    for animal in animaux:
        if animal[enclos] == num_enclos:
            tab.append(animal)
    return tab

# Tests
animaux = [ {'nom':'Medor','espece':'chien','age':5,'enclos':2},
{'nom':'Titine','espece':'chat','age':2,'enclos':5},
{'nom':'Tom','espece':'chat','age':7,'enclos':4},
{'nom':'Belle','espece':'chien','age':6,'enclos':3},
{'nom':'Mirza','espece':'chat','age':6,'enclos':5}]
```

Exercice 2

```
def trouver_intrus(tab, g, d):
    '''
    Renvoie la valeur de l'intrus situé entre les indices g et d
    dans la liste tab où
    tab vérifie les conditions de l'exercice,
        g et d sont des multiples de 3.
    '''
    if g == d:
        return tab[g]

    else:
        nombre_de_triplets = (d - g)// 3
        indice = g + 3 * (nombre_de_triplets // 2)
        if tab[indice] == tab[indice+1] :
            return trouver_intrus(tab, indice+3, d)
        else:
            return trouver_intrus(tab, g, indice)
```

2022 sujet 26

Exercice 1

```
def RechercheMin(tab):
    indice = 0
    mini = tab[0]
    for i in range(len(tab)):
        if tab[i] < mini:
            indice = i
            mini = tab[i]
    return indice
```

Exercice 2

```
def separe(tab):
    i = 0
    j = len(tab) - 1
    while i < j :
        if tab[i] == 0 :
            i = i + 1
        else :
            tab[i], tab[j] = tab[j], tab[i]
            j = j - 1
    return tab
```

2022 sujet 27

Exercice 1

```
a = {'F':['B','G'], 'B':['A','D'], 'A':['',''], 'D':['C','E'], \
'C':['',''], 'E':['',''], 'G':['','I'], 'I':['','H'], \
'H':['','']}

def taille(arbre, lettre):
    if arbre[lettre][0] == '' and arbre[lettre][1] == '':
        return 1
    elif arbre[lettre][0] == '' and arbre[lettre][1] != '':
        return 1 + taille(arbre, arbre[lettre][1])
    elif arbre[lettre][0] != '' and arbre[lettre][1] == '':
        return 1 + taille(arbre, arbre[lettre][0])
    elif arbre[lettre][0] != '' and arbre[lettre][1] != '':
        return 1 + taille(arbre, arbre[lettre][0]) \
               + taille(arbre, arbre[lettre][1])
```

Exercice 2

```
def tri_iteratif(tab):
    for k in range( len(tab) - 1 , 0, -1):
        imax = k - 1 # On ignore le dernier pour le max
        for i in range(0, k - 1):
            if tab[i] > tab[imax] :
                imax = i
        if tab[imax] > tab[k] : # On compare ici au dernier
            tab[k] , tab[imax] = tab[imax] , tab[k]
    return tab

# Cela fonctionne aussi en prennant le dernier pour le max
# mais il y a une itération supplémentaire dans la boucle
def tri_iteratif2(tab):
    for k in range( len(tab) - 1 , 0, -1):
        imax = k
        for i in range(0, k):
            if tab[i] > tab[imax] :
                imax = i
        # On pourrait alors remplacer le if par imax != k:
        if tab[imax] > tab[k] :
            tab[k] , tab[imax] = tab[imax] , tab[k]
    return tab
```

2022 sujet 28

Exercice 1

```
def moyenne(tab):
    s = 0
    for e in tab:
        s += e
    return s / len(tab)
```

Exercice 2

```
def dec_to_bin(a):
    bin_a = str(a%2)
    a = a//2
    while a > 0 :
        bin_a = str(a%2) + bin_a
        a = a//2
    return bin_a
```

2022 sujet 29

Exercice 1

```
def fibonacci(n):
    if n < 3:
        return 1
    else:
        ukmoins2 = 1
        ukmoins1 = 1
        k = 3
        while k <= n:
            uk = ukmoins2 + ukmoins1
            ukmoins2 = ukmoins1 # On décale
            ukmoins1 = uk # On décale
            k = k + 1
        return uk
```

Exercice 2

```
liste_eleves = ['a','b','c','d','e','f','g','h','i','j']
liste_notes = [1, 40, 80, 60, 58, 80, 75, 80, 60, 24]

def meilleures_notes():
    note_maxi = 0
    nb_eleves_note_maxi = 0
    liste_maxi = []

    for compteur in range(len(liste_eleves)):
        if liste_notes[compteur] == note_maxi:
            nb_eleves_note_maxi = nb_eleves_note_maxi + 1
            liste_maxi.append(liste_eleves[compteur])
        if liste_notes[compteur] > note_maxi:
            note_maxi = liste_notes[compteur]
            nb_eleves_note_maxi = 1
            liste_maxi = [liste_eleves[compteur]]

    return (note_maxi,nb_eleves_note_maxi,liste_maxi)
```

2022 sujet 30

Exercice 1

```python
def fusion(tab1, tab2):
    tab = []
    i1 = 0 # Index de tab1
    i2 = 0 # Index de tab2
    while i1 < len(tab1) and i2 < len(tab2):
        if tab1[i1] < tab2[i2]:
            tab.append(tab1[i1])
            i1 = i1 + 1
        else:
            tab.append(tab2[i2])
            i2 = i2 + 1
    # On utilise des slices ici car ils permettent de renvoyer la fin des
    # tableaux sans erreur, même si l'indice est à la fin.
    return tab + tab1[i1:] + tab2[i2:]
```

Exercice 2

```python
def rom_to_dec (nombre):

    """ Renvoie l'écriture décimale du nombre donné en chiffres romains """

    dico = {"I":1, "V":5, "X":10, "L":50, "C":100, "D":500, "M":1000}
    if len(nombre) == 1:
        return dico[nombre]

    else:
        """ On supprime le premier caractère de la chaîne contenue dans
        la variable nombre et cette nouvelle chaîne est enregistrée
        dans la variable nombre_droite
    """
        nombre_droite = nombre[1:]


        if dico[nombre[0]] >= dico[nombre[1]]:
            return dico[nombre[0]] + rom_to_dec(nombre_droite)
        else:
            return rom_to_dec(nombre_droite) - dico[nombre[0]]

assert rom_to_dec("CXLII") == 142
```

2022 sujet 31

Exercice 1

```
def recherche(a, t):
    c = 0
    for e in t:
        if a == e:
            c += 1
    return c
```

Exercice 2

```
def rendu_monnaie_centimes(s_due, s_versee):
    pieces = [1, 2, 5, 10, 20, 50, 100, 200]
    rendu = []
    a_rendre = s_versee - s_due
    i = len(pieces) - 1
    while a_rendre > 0 :
        if pieces[i] <= a_rendre :
            rendu.append(pieces[i])
            a_rendre = a_rendre - pieces[i]
        else :
            i = i - 1
    return rendu
```

2022 sujet 32

Exercice 1

```
def recherche(elt, tab):
    indice = -1
    for i in range(len(tab)):
        if elt == tab[i]:
            indice = i
    return indice
```

Exercice 2

```
class AdresseIP:

    def __init__(self, adresse):
        self.adresse = adresse

    def liste_octet(self):
        """renvoie une liste de nombres entiers,
           la liste des octets de l'adresse IP"""
        return [int(i) for i in self.adresse.split(".")]

    def est_reservee(self):
        """renvoie True si l'adresse IP est une adresse
           réservée, False sinon"""
        return self.adresse == "192.168.0.0" or self.adresse == "192.168.0.255"

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
           IP qui suit l'adresse self
           si elle existe et False sinon"""
        if self.liste_octet()[3] < 254:
            octet_nouveau = self.liste_octet()[3] + 1
            return AdresseIP('192.168.0.' + str(octet_nouveau))
        else:
            return False

adresse1 = AdresseIP('192.168.0.1')
adresse2 = AdresseIP('192.168.0.2')
adresse3 = AdresseIP('192.168.0.0')
```

2022 sujet 33

Exercice 1

```
def convertir(T):
    """
    T est un tableau d'entiers, dont les éléments sont 0 ou 1 et
    représentant un entier écrit en binaire. Renvoie l'écriture
    décimale de l'entier positif dont la représentation binaire
    est donnée par le tableau T
    """
    decimal = 0
    for i in range(len(T)):
        decimal += T[i] * 2**(len(T) - 1 - i)
    return decimal
```

Exercice 2

```
def tri_insertion(L):
    n = len(L)

    # cas du tableau vide
    if L == []:
        return L

    for j in range(1,n):
        e = L[j]
        i = j

    # A l'etape j, le sous-tableau L[0,j-1] est trie
    # et on insere L[j] dans ce sous-tableau en determinant
    # le plus petit i tel que 0 <= i <= j et L[i-1] > L[j].
        while  i > 0 and L[i-1] > L[j]:
            i = i - 1

        # si i != j, on decale le sous tableau L[i,j-1] d'un cran
        # vers la droite et on place L[j] en position i
        if i != j:
            for k in range(j,i,-1):
                L[k] = L[k-1]
            L[i] = e
    return L
```

2022 sujet 34

Exercice 1

```
def occurrence_max(ch):
    alphabet=['a','b','c','d','e','f','g','h','i','j','k','l','m',
              'n','o,','p','q','r','s','t','u','v','w','x','y','z']
    occurrence = [0]*26
    for c in ch:
        if c in alphabet:
            occurrence[alphabet.index(c)] += 1
    nb_max = 0
    for i in range(26):
        if occurrence[i] > nb_max:
            nb_max = occurrence[i]
            c_max = alphabet[i]
    return c_max


# Voici une solution qui n'utilise pas la méthode index

def occurrence_max_sans_index(ch):
    alphabet=['a','b','c','d','e','f','g','h','i','j','k','l','m',
              'n','o,','p','q','r','s','t','u','v','w','x','y','z']
    occurrence = [0]*26
    for c in ch:
        if c in alphabet:
            for i in range(len(alphabet)):
                if c == alphabet[i]:
                    occurrence[i] += 1
                    break
    nb_max = 0
    for i in range(26):
        if occurrence[i] > nb_max:
            nb_max = occurrence[i]
            c_max = alphabet[i]
    return c_max



# Test
ch=('je suis en terminale et je passe le bac et je souhaite poursuivre'
    'des etudes pour devenir expert en informatique')
```

Exercice 2

```
def nbLig(image):
    '''renvoie le nombre de lignes de l'image'''
    return len(image)

def nbCol(image):
    '''renvoie la largeur de l'image'''
    return len(image[0])

def negatif(image):
    '''renvoie le negatif de l'image sous la forme
       d'une liste de listes'''
    L = [[0 for k in range(nbCol(image))] for i in range(nbLig(image))]
# on cree une image de 0 aux memes dimensions que le parametre image
    for i in range(len(image)):
        for j in range(len(image[0])):
            L[i][j] = 255 - image[i][j]
    return L

def binaire(image, seuil):
    '''renvoie une image binarisee de l'image sous la forme
       d'une liste de listes contenant des 0 si la valeur
       du pixel est strictement inferieure au seuil
       et 1 sinon'''
    # on cree une image de 0 aux memes dimensions que le parametre image
    L = [[0 for k in range(nbCol(image))] for i in range(nbLig(image))]
    for i in range(len(image)):
        for j in range(len(image[0])):
            if image[i][j] < seuil :
                L[i][j] = 0
            else:
                L[i][j] = 1
    return L

# Test
img=[[20, 34, 254, 145, 6], [23, 124, 287, 225, 69],
     [197,174, 207, 25, 87], [255, 0, 24, 197, 189]]
```

2022 sujet 35

Exercice 1

```
def moyenne (tab):
    '''
    moyenne(list) -> float
    Entrée : un tableau non vide d'entiers
    Sortie : nombre de type float
    Correspondant à la moyenne des valeurs présentes dans le
    tableau
    '''
    s = 0
    for e in tab:
        s += e
    return s / len(tab)

assert moyenne([1]) == 1
assert moyenne([1,2,3,4,5,6,7]) == 4
assert moyenne([1,2]) == 1.5
```

Exercice 2

```
def dichotomie(tab, x):
    """
        tab : tableau trie dans l'ordre croissant
        x : nombre entier
        La fonction renvoie True si tab contient x et False sinon
    """
    # cas du tableau vide
    if tab == []:
        return False,1

    # cas ou x n'est pas compris entre les valeurs extremes
    if (x < tab[0]) or (tab[-1] < x):
        return False,2

    debut = 0
    fin = len(tab) - 1
    while debut <= fin:
        m = (debut + fin) // 2
        if x == tab[m]:
```

```
            return True
        if x > tab[m]:
            debut = m + 1
        else:
            fin = m - 1
    return False, 3
```

2022 sujet 36

Exercice 1

```
def recherche(tab, n):
    indice = len(tab)
    for i in range(len(tab)):
        if n == tab[i]:
            indice = i
    return indice
```

Exercice 2

```
from math import sqrt   # import de la fonction racine carree

def distance(point1, point2):
    """ Calcule et renvoie la distance entre deux points. """
    # La nature des préconditions n'est pas claire !
    # Faut-il aussi tester le type ?
    assert len(point1) == 2 and len(point2) == 2, "Mauvaises coordonnées"
    return sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)

assert distance((1, 0), (5, 3)) == 5.0, "erreur de calcul"

# Erreur dans l'énoncé qui demande la distance et non le point
def plus_courte_distance(tab, depart):
    """ Renvoie le point du tableau tab se trouvant a la plus
    courte distance du point depart."""
    point = tab[0]
    min_dist = distance(point, depart)
    for i in range (1, len(tab)):
        if distance(tab[i], depart) < min_dist:
            point = tab[i]
            min_dist = distance(tab[i], depart)
    return point

assert plus_courte_distance([(7, 9), (2, 5), (5, 2)], (0, 0)) == (2, 5), "erreur"
```

2022 sujet 37

Exercice 1

```
def verifie(tab):
    for i in range(1,len(tab)):
        if tab[i-1] > tab[i]:
            return False
    return True
```

Exercice 2

```
urne = ['Oreilles sales', 'Oreilles sales', 'Oreilles sales',
      'Extra Vomit', 'Lady Baba', 'Extra Vomit', 'Lady Baba',
      'Extra Vomit', 'Lady Baba', 'Extra Vomit']

def depouille(urne):
    resultat = dict()
    for bulletin in urne:
        if bulletin in resultat:
            resultat[bulletin] = resultat[bulletin] + 1
        else:
            resultat[bulletin] = 1
    return resultat

def vainqueur(election):
    vainqueur = ''
    nmax = 0
    for candidat in election:
        if election[candidat] > nmax :
            nmax = election[candidat]
            vainqueur = candidat
    liste_finale = [nom for nom in election if election[nom] == nmax]
    return liste_finale
```

2022 sujet 38

Exercice 1

```
def tri_selection(tab):
    for i in range(len(tab) - 1):
        e = tab[i] # L'élément qu'on va echanger
        # On initialise la recherche du minimum
        mini = tab[i]
        mini_indice = i
        # On cherche le minimum dans le rest du tableau
        for j in range(i + 1, len(tab)):
            if tab[j] < mini:
                mini = tab[j]
                mini_indice = j
        # si le minimum n'est pas déjà en i, on fait l'échange
        tab[i] = tab[mini_indice]
        tab[mini_indice] = e
    return tab
```

Exercice 2

```
from random import randint

def plus_ou_moins():
    nb_mystere = randint(1,99)
    nb_test = int(input("Proposez un nombre entre 1 et 99 : "))
    compteur = 1

    # Quand le dixième essai est fait on arrête
    while nb_mystere != nb_test and compteur < 10 :
        compteur = compteur + 1
        if nb_mystere > nb_test:
            nb_test = int(input("Trop petit ! Testez encore : "))
        else:
            nb_test = int(input("Trop grand ! Testez encore : "))

    if nb_mystere == nb_test:
        print ("Bravo ! Le nombre etait ", nb_mystere)
        print("Nombre d'essais: ", compteur)
    else:
        print ("Perdu ! Le nombre etait ", nb_mystere)
```

2022 sujet 39

Exercice 1

```
def moyenne(tab):
    somme = 0
    for e in tab:
        somme += e
    return somme / len(tab)
```

Exercice 2

```
coeur = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], \
         [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0], \
         [0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0], \
         [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0], \
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], \
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], \
         [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], \
         [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0], \
         [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

def affiche(dessin):
    ''' affichage d'une grille : les 1 sont repreente par
        des "*" , les 0 par deux espaces "  " '''
    for ligne in dessin:
        for col in ligne:
            if col == 1:
                print(" *",end="")
            else:
                print("  ",end="")
        print()


def zoomListe(liste_depart,k):
    '''renvoie une liste contenant k fois chaque
       element de liste_depart'''
    liste_zoom = []
    for elt in liste_depart :
```

```
        for i in range(k):
            liste_zoom.append(elt)
    return liste_zoom

def zoomDessin(grille,k):
    '''renvoie une grille ou les lignes sont zoomees k fois
       ET repetees k fois'''
    grille_zoom=[]
    for elt in grille:
        liste_zoom = zoomListe(elt,k)
        for i in range(k):
            grille_zoom.append(liste_zoom)
    return grille_zoom
```

2022 sujet 40

Exercice 1

```
def recherche(elt, tab):
    s = []
    for i in range(len(tab)):
        if elt == tab[i]:
            s.append(i)
    return s
```

Exercice 2

```
resultats = {'Dupont':{'DS1' : [15.5, 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [13, 4],
                       'PROJET1' : [16, 3],
                       'DS3' : [14, 4]},
             'Durand':{'DS1' : [6 , 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [8, 4],
                       'PROJET1' : [9, 3],
                       'IE1' : [7, 2],
                       'DS3' : [8, 4],
                       'DS4' :[15, 4]}}


def moyenne(nom):
    if nom in resultats:
        notes = resultats[nom]
        total_points = 0
        total_coefficients = 0
        for valeurs  in notes.values():
            note , coefficient = valeurs
            total_points = total_points + note * coefficient
            total_coefficients = total_coefficients + coefficient
        return round( total_points / total_coefficients , 1 )
    else:
        return -1
```