

```

1
2 #exo1
3 def coorVecteur(xA,yA,xB,yB):
4     x_AB=xB-xA
5     y_AB=yB-yA
6     AB=(x_AB,y_AB)
7     return AB
8
9 #exo2
10 def coorVecteur(xA,yA,xB,yB):
11     x_AB=xB-xA
12     y_AB=yB-yA
13     AB=(x_AB,y_AB)
14     return AB
15 def multVecteurk(k,xA,yA,xB,yB):
16     AB=coorVecteur(xA,yA,xB,yB)
17     x_kAB=AB[0]*k
18     y_kAB=AB[1]*k
19     kAB=(x_kAB,y_kAB)
20     return kAB
21
22
23 #exo3
24 def affichageTuple(monTuple):
25     i=0
26     for elt in monTuple:
27         print(i,elt)
28         i=i+1
29     return None
30
31 def affichageTuple2(monTuple):
32     n = len(monTuple)
33     for i in range(n):
34         print(i," ",monTuple[i])
35     return None
36
37 #exo4
38 def longueurTuple(monTuple):
39     long=0
40     for elt in monTuple:
41         long=long+1
42     return long
43
44
45 #exo5
46 def affichagePair(monTuple):
47     for elt in monTuple:
48         if elt%2==0:
49             print(elt,end=" ")
50     return None
51
52
53 #exo6
54 def verifElement(val,monTuple):
55     n = len(monTuple)
56     for i in range(n):
57         if monTuple[i]==val:
58             return i
59     return False
60
61
62 L = (5,4,8,9,1,4)
63 L1 = (4,3,1)
64 x=9
65 n=3
66
67 print (x in L)
68 print (x in L1)
69
70
71 print (x not in L)
72 print (x not in L1)

```

```

73
74 print(len(L))
75 print(len(L1))
76
77 print(L==L1)
78
79
80 print(L[0])
81 print(L1[0])
82
83 print(L[1:4])
84 print(L1[0:2])
85
86 print(L.index(x))
87 #print(L1.index(x))
88
89 print(L.count(x))
90 print(L1.count(x))
91
92 print(L+L1)
93
94 print(L*n)
95
96 #exo8
97 def maxiTuple(monTuple):
98     maximum = monTuple[0]
99     for i in range(1,len(monTuple)):
100         if monTuple[i] > maximum:
101             maximum = monTuple[i]
102     return maximum
103
104 #def maxiTuple2(monTuple):
105 #return max(monTuple)
106
107 #exo9
108 def comptage(monTuple,valeur):
109     compteur = 0
110     for elt in monTuple:
111         if valeur == elt:
112             compteur = compteur + 1
113     return compteur
114
115 def comptage2(monTuple,n):
116     return monTuple.count(n)
117
118 #Exo10
119 def sommePlus(monTuple):
120     n=len(monTuple)
121     somme=0
122     for i in range(n):
123         if monTuple[i]>0:
124             somme=somme+monTuple[i]
125     return somme
126
127 def sommePlusbis(monTuple):
128     somme = 0
129     for elt in monTuple:
130         if elt > 0:
131             somme = somme + elt
132     return somme
133
134 #Exo11
135 def sommePlus2(monTuple):
136     n=len(monTuple)
137     somme=0
138     for i in range(n):
139         if monTuple[i]>0 and monTuple[i]%2==0:
140             somme=somme+monTuple[i]
141     return somme
142
143 def sommePlus2bis(monTuple):
144     somme = 0

```

```

145     for elt in monTuple:
146         if (elt > 0) and (elt % 2 == 0):
147             somme = somme + elt
148     return somme
149
150 #exo12
151 L=[4,2,1,3,4,6]
152
153 L.insert(4,5)
154 [4, 2, 1, 3, 5, 4, 6]
155
156 del L[3]
157 [4, 2, 1, 5, 4, 6]
158
159 L.pop(4)
160 [4, 2, 1, 5, 6]
161
162 L.insert(2,15)
163 [4, 2, 15, 1, 5, 6]
164
165
166 #exo13
167 L1=[]
168 n=10000
169 for i in range(n+1):
170     L1.append(i)
171
172 L2=[i for i in range(n+1)]
173
174 #exo14
175 n=10000
176 L3=[i for i in range(0,n+1,2)]
177
178 L3bis = [i for i in range(10001) if i%2 == 0]
179
180 #exo15
181 n=10
182 L4=[2**i for i in range(n+1)]
183
184 n=10
185 L4bis=[]
186 for i in range(n+1):
187     L4bis.append(2**i)
188
189 ##exo16
190 from random import *
191
192 def liste5(n):
193     L5=[]
194     L5=[randint(0,20) for i in range(n)]
195     return L5
196
197
198
199 ##exo16bis
200 from random import *
201 def liste5(n):
202     L5 = []
203     i=1
204     a = randint(0,20)
205     while a != 0 and i<=n:
206         L5.append(a)
207         a = randint(0,20)
208         i=i+1
209     return L5
210
211
212 def liste5bis(n):
213     L5bis=[]
214     i=0
215     a=randint(0,20)
216     while a!=0 and i<n:

```

```

217         L5bis.append(a)
218         a=randint(0,20)
219         i=i+1
220     return L5bis
221
222 #exo17
223 L = [[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]]
224
225 L[1]
226 [6, 7, 8, 9, 10]
227
228 L[2][3]
229 14
230
231 len(L)
232 3
233
234 len(L[0])
235 5
236
237 #exo18
238 A = [[1 ,2 ,3] , [4 ,5 ,6] , [7,8,9]]
239
240 m = len(A)
241 n = len(A[0])
242 for j in range(n):
243     for i in range(m):
244         print(A[i][j], end=" ")
245
246 print()
247
248 for sousliste in A:
249     for element in sousliste:
250         print(element,end(" "))
251
252 #exo19
253 B = [[4 ,5] , [2 ,9] , [3 ,7]]
254
255 B[0].reverse()
256 B[1].reverse()
257 B[2].reverse()
258 print(B)
259
260
261
262
263 B1 = [[4 ,5] , [2 ,9] , [3 ,7]]
264 for i in range(3):
265     B1[i][0],B1[i][1] = B1[i][1],B1[i][0]
266
267 #exo20
268 C = [[4 ,5] , [2 ,9] , [3 ,7]]
269 somme=0
270 m=len(C)
271 n=len(C[0])
272 for i in range(m):
273     for j in range(n):
274         somme=somme+C[i][j]
275 print(somme)
276
277 def somme_Termes_Matrice(matrice):
278     somme = 0
279     for ligne in matrice:
280         for elt in ligne:
281             somme = somme + elt
282     return somme
283
284 #exo21
285 D = [[1 ,4] , [9 ,5] , [7 ,2]]
286
287 m=len(D)
288 n=(len(D[0]))

```

```

289 max=D[0][0]
290 for i in range(m):
291     for j in range (n):
292         if D[i][j]>max:
293             max=D[i][j]
294 print(max)
295
296 """
297 La méthode pour trouver le maximum est la suivante :
298 on crée une variable max à laquelle on affecte la valeur du premier élément
299 de la liste. Puis on parcourt le tableau et on compare à chaque fois
300 l'élément atteint avec max. Si l'élément est supérieur ou égal à max,
301 on affecte cet élément à max.
302 """
303
304 D = [[1 ,4] ,[9 ,5] ,[7 ,2]]
305 def plus_grande_valeur_matrice(matrice):
306     maximum = matrice[0][0]
307     for ligne in matrice:
308         for elt in ligne:
309             if elt > maximum:
310                 maximum = elt
311     return maximum
312
313 #exo22
314 E = [[2 ,1 ,3] ,[4 ,8 ,6] ,[7,5,9]]
315
316 m=len(E)
317 n=(len(E[0]))
318
319 for i in range(m):
320     for j in range (n):
321         if E[i][j]%2==0:
322             E[i][j]=True
323         else :
324             E[i][j]=False
325 print(E)
326
327 def matrice_parite(matrice):
328     for i in range(len(matrice)):
329         for j in range(len(matrice[i])):
330             if matrice[i][j] % 2 == 0:
331                 matrice[i][j] = True
332             else:
333                 matrice[i][j] = False
334     return matrice
335
336 def matrice_parite2(matrice):
337     M_partite = [[(matrice[i][j] % 2 == 0) == True for j in range(3)] for i in
338 range(3)]
339     return M_partite
340
341 #exo23
342 def sommeElementCourant(F):
343     m=len(F)
344     n=(len(F[0]))
345     for i in range(m):
346         somme=0
347         for j in range (n):
348             somme=somme+F[i][j]
349             F[i][j]=somme
350     return F
351
352 #exo24
353 #sol non optimale
354 def verifMagique(matrice):
355     # calcul sommes des lignes
356     listeSommeLigne=[] # liste contenant la somme des lignes [somme ligne 0 , somme
357     ligne 1 ,somme ligne2]
358     for i in range(3):
359         sommeLigne=0 # remise à zéro de la somme à chaque ligne
360         for j in range (3):

```

```

359         sommeLigne=sommeLigne+matrice[i][j]
360     listeSommeLigne.append(sommeLigne)
361     print(listeSommeLigne)
362
363     # calcul sommes des colonnes
364     listeSommeColonne=[] # liste contenant la somme des colonnes [somme colonne0 ,
somme colonne1 ,somme colonne2]
365     for j in range(3):
366         sommeColonne=0 # remise à zéro de la somme à chaque ligne
367         for i in range(3):
368             sommeColonne=sommeColonne+matrice[i][j]
369         listeSommeColonne.append(sommeColonne)
370     print(listeSommeColonne)
371
372     # calcul somme diagonale 1 (du haut vers le bas)
373     sommeDiagonale1 =0
374     for k in range(3):
375         sommeDiagonale1=sommeDiagonale1+matrice[k][k]
376     print(sommeDiagonale1)
377
378     # calcul somme diagonale 2 (du bas vers le haut)
379     sommeDiagonale2 =0
380     for k in range(3):
381         sommeDiagonale2=sommeDiagonale2+matrice[k][2-k]
382     print(sommeDiagonale2)
383
384     # vérification carré magique ordre 3 (vérif des sommes)
385     return (listeSommeLigne==listeSommeColonne and sommeDiagonale1==sommeDiagonale2)
386
387 #sol optimale
388 def verifMagique2(matrice):
389     somme_en_liste = []
390     # verif somme des lignes
391     for i in range(3):
392         somme = 0
393         for j in range(3):
394             somme = somme + matrice[i][j]
395         somme_en_liste.append(somme)
396     # verif somme des colonnes
397     for j in range(3):
398         somme = 0
399         for i in range(3):
400             somme = somme + matrice[i][j]
401         somme_en_liste.append(somme)
402     # verif somme diagonales
403     somme1 = 0
404     for k in range(3):
405         somme1 = somme1 + matrice[k][k]
406     somme_en_liste.append(somme1)
407     somme2 = 0
408     for k in range(3):
409         somme2 = somme2 + matrice[k][2-k]
410     somme_en_liste.append(somme2)
411
412     for i in range(len(somme_en_liste)-1):
413         if somme_en_liste[i] != somme_en_liste[i+1]:
414             return False
415     return True
416
417 #exo25
418 from random import *
419 def matriceAlea(m,n):
420     L = [] # Initialisation d'une liste vide
421     for i in range(m): # Balayage des sous listes
422         SL = [] # Creation d'une sous liste vide
423         for j in range(n): # Balayage des elements de la sous liste
424             SL.append(randint(0,100)) # Ajout d'un elemet a droite de la sous
liste
425         L.append(SL)
426     return L
427

```