```python
# E:\Retour 2023 Exercice Python\Exo Python 2023.py
# Exercice 1
## Q1
def LongCr(L):
    n = len(L)-1
    i = 0
    while L[i] > L[i+1]:
        i += 1
        if i == n:
            return -1,0
    k = 1
    while k < n and L[k] <= L[k+1]:
        k += 1
    return i, k

L = [12,2,4,6,3,5,10,11,8,4]
print(LongCr(L))

## Q2
def SuitesCr(L):
    Liste = []
    while len(L) > 1:
        i, l = LongCr(L)
        suite = [L[k] for k in range(i,i+l)]
        if suite != []:
            Liste.append(suite)
        L = L[i+l:]
    return Liste

L = [12,2,4,6,3,5,10,11,8,4]
print(SuitesCr(L))

# Exercice 2
## Q1
def el_commun(L1,L2):
    for x in L1:
        if x in L2:
            return True
    return False

## Q2
def Liste_commun(L1,L2):
    return [x for x in L1 if x in L2]

# Exercice 3
## Q1
def test_arithmetique(L):
    r = L[1] - L[0]
    for i in range(1,len(L)-1):
        if L[i+1] - L[i] != r:
```

```
050|             return False
051|     return True
052|
053| ## Q2
054| def test_geometrique(L):
055|     if L[0] == 0:
056|         for x in L:
057|             if x != 0:
058|                 return False
059|         return True
060|     q = L[1]/L[0]
061|     for i in range(1,len(L)-1):
062|         if L[i] == 0 or L[i+1]/L[i] != q:
063|             return False
064|     return True
065|
066| # Exercice 4
067| ## Q1
068| def Maxi(Tab):
069|     m = Tab[0]
070|     for x in Tab:
071|         if x > m:
072|             m = x
073|     return m
074|
075| Tab = [4,2,5,4,2,1,2]
076| print(Maxi(Tab))
077|
078| ## Q2
079| def Occur(Tab):
080|     m = Maxi(Tab)
081|     L = [0]*(m+1)
082|     for i in Tab:
083|         L[i] += 1
084|     return L
085|
086| print(Occur(Tab))
087|
088| # Exercice 5
089| ## Q1
090| def Test(ADN):
091|     for x in ADN:
092|         if x not in 'ATCG':
093|             return False
094|     return True
095|
096| ADN = 'ATCGCCT'
097| print(Test(ADN))
098| ADN = 'ABCGCCT'
099| print(Test(ADN))
```

```
100|
101| ## Q2
102| def Mut(ADN):
103|     for i in range(len(ADN)):
104|         if ADN[i] == 'T':
105|             ADN = ADN[:i]+'U'+ADN[i+1:]
106|     return ADN
107|
108| def Mut2(ADN):
109|     return ADN.replace('T','U')
110|
111| print(ADN)
112| print(Mut(ADN))
113| print(Mut2(ADN))
114|
115| ## Q3:
116| def Test_Gene(Gene):
117|     start, stop = ['AUG','GUG','UUG'], ['UAA','UAG','UGA']
118|     return Gene[:3] in start and Gene[-3:] in stop
119|
120| Gene = 'AUGCGAGCAUGA'
121| print(Test_Gene(Gene))
122| Gene = 'AUGCGAGCAUCGA'
123| print(Test_Gene(Gene))
124|
125| # Exercice 6
126| ## Q1
127| def Maxi2(L):
128|     m1, m2 = L[0], L[1]
129|     for x in L:
130|         if x > m1:
131|             m1 = x
132|         elif x > m2:
133|             m2 = x
134|     return m2
135|
136| ## Q2
137| def Indice(L,x):
138|     if x not in L:
139|         return -1
140|     return len([el for el in L if el < x])
141|
142| # Exercice 7
143| ## Q1
144| def Tri(L):
145|     if len(L) <= 1:
146|         return L
147|     else:
148|         p = L[0]
149|         Lgauche = [x for x in L if x < p]
```

```
150|         Ldroite = [x for x in L if x > p]
151|         return Tri(Lgauche)+[p]+Tri(Ldroite)
152|
153| L = [4,2,4,5,8,5,1,3]
154| print(Tri(L))
155|
156| ## Q2
157| def El_Oc(L):
158|     E = [L[0]]
159|     for x in L:
160|         if x not in E:
161|             E.append(x)
162|     E = Tri(E)
163|     O = []
164|     for x in E:
165|         c = 0
166|         for e in L:
167|             if e == x:
168|                 c += 1
169|         O.append(c)
170|     return E, O
171|
172| print(El_Oc(L))
173|
174| # Exercice 8
175| ## Q1
176| def Liste_multi(n):
177|     L = []
178|     for k in range(1,n+1):
179|         L += k*[k]
180|     return L
181|
182| print(Liste_multi(3))
183|
184| ## Q2
185| import random as rd
186| def tirage():
187|     n = rd.randint(1,5)
188|     Urne2 = Liste_multi(n)
189|     return rd.choice(Urne2)
190|
191| print(tirage())
192|
193| ## Q3
194| def proba1():
195|     N =10000
196|     return sum([tirage() == 1 for _ in range(N)])/N
197|
198| print(proba1())
199|
```

```python
# Soit p la probabilité de tirer le 1 dans la 2e urne.
# Si le 1er tirage a donné le numéro n (avec une probabilité de 1/5)
# alors la 2e urne contient n(n+1)/2 termes dont un seul 1.
# la formule des probas totales donne :
# p = (2/5)*somme(1/(k(k+1)) pour k de 1 à 5).
# Comme 1/(k(k+1)) = 1/k- 1/(k+1), par télescopage, p= (2/5)*(1-1/6)=1/3.

# Exercice 9
## Q1
def Entre(L):
    n = len(L)
    for x in L:
        if x not in list(range(n+1)):
            return False
    return True

## Q2
def Recherche(x,L):
    Lindx = []
    for i in range(len(L)):
        if L[i] == x:
            Lindx.append(i)
    return Lindx

# Exercice 10
## Q1
def Freq(ADN):
    n = len(ADN)
    L = [0]*4
    for x in ADN:
        if x == 'A' or x == 'a':
            L[0] += 1/n
        elif x == 'T' or x == 't':
            L[1] += 1/n
        elif x == 'C' or x == 'c':
            L[2] += 1/n
        elif x == 'G' or x == 'g':
            L[3] += 1/n
    return L
## Q2
def Long1(ADN):
    nu = ADN[0]
    i = 0
    while i < len(ADN) and ADN[i] == nu:
        i += 1
    return i, nu

```

```python
## Q3
def Brin(ADN):
    L = []
    New = ADN[:]
    while len(New) > 0:
        l, nu = Long1(New)
        L += [l, nu]
        New = New[l:]
    return L

# Exercice 11
## Q1
def Eval(P,a):
    return sum([P[k]*a**k for k in range(len(P))])

def Eval2(P,a):
    x = 1
    S = P[0]
    for k in range(1,len(P)):
        x *= a
        S += P[k]*x
    return S

P = [1,3,2,-1,2]
a = 2
print(Eval(P,a))

## Q2
def Deriv(P):
    n = len(P)
    return [k*P[k] for k in range(1,n)]

P = [1,3,2,-1,2]
print(Deriv(P))

## Q3
def f(P):
    n = len(P)
    L2XP = [0]+[2*P[k-1] for k in range(1,n+1)]
    Pprime = Deriv(P) + 2*[0]
    return [L2XP[k] - Pprime[k] for k in range(n+1)]

P = [1,3,2,-1,2]
print(f(P))

# Exercice 12
## Q1
def listeX(n):
    return [k/n for k in range(n+1)]

```

```
297| print(listeX(5))
298|
299| ## Q2
300| def Euler(n):
301|     X = listeX(n)
302|     Y = [1]
303|     for k in range(n):
304|         Y.append(Y[k]*(1+X[k]/n))
305|     return X, Y
306|
307| print(Euler(5))
308|
309| ## Q3
310| import matplotlib.pyplot as plt
311| import math as m
312| n = 100
313| X, Y = Euler(n)
314| Z = [m.exp((x**2)/2) for x in X]
315| plt.plot(X,Y,'blue')
316| plt.plot(X,Z,'red')
317| plt.show()
318|
319| # Exercice 13
320| ## Q1
321| def R(f,a,b,n):
322|     h = (b-a)/n
323|     return h*sum([f(a+k*h) for k in range(n)])
324|
325| ## Q2
326| import math as m
327| def f(x):
328|     return (m.sqrt(4-x**2))/(2*m.pi)
329| a, b = -2, 2
330| n = 1000
331| print(R(f,a,b,n))
332|
333| # Exercice 14
334| ## Q1
335| def el_com(G,T):
336|     return len([x for x in G if x in T])
337|
338| def el_com2(G,T):
339|     C = []
340|     for x in G:
341|         if x in T and x not in C:
342|             C.append(x)
343|     return len(C)
344|
345| G, T =[1,3,2,4,1,2,1], [3,1,3,4]
346| print(el_com(G,T))
```

```python
print(el_com2(G,T))

## Q2
import random as rd
def loto():
    Urne = list(range(50))
    T = []
    for _ in range(6):
        t = rd.choice(Urne)
        Urne.remove(t)
        T.append(t)
    return T

## Q3
def Nb_Gagnant(T):
    G = [2,6,0,4,9,8]
    return el_com(G,T)

## Q4
def Gagne(n):
    c = 0
    for _ in range(n):
        T = loto()
        if Nb_Gagnant(T) == 6:
            c += 1
    return c

n = 10000000
print(Gagne(n)/n)

# Exercice 15

## Q1
def est_mixte(L):
    return 0 in L and 1 in L

## Q2
def plus_long_suite1(L):
    n = len(L)
    if 1 not in L:
        return 0
    i = 0
    m = 0
    while i < n:
        while i < n and L[i] == 0:
            i += 1
        c = 0
        while i < n and L[i] == 1:
            i += 1
            c += 1
```

```
397|         if c > m:
398|             m = c
399|     return m
400|
401| L = [0,0,1,1,0,1,1,1,0]
402| print(plus_long_suite1(L))
403|
404| # Exercice 16
405| ## Q1
406| import math as m
407| def distance(A,B):
408|     return m.sqrt((B[0]-A[0])**2+(B[1]-A[1])**2)
409|
410| A, B = [1,2], [-1,3]
411| print(distance(A,B))
412|
413| ## Q2
414| def point_moyen(Nuage):
415|     n = len(Nuage)
416|     xG = sum([Nuage[k][0] for k in range(n)])/n
417|     yG = sum([Nuage[k][1] for k in range(n)])/n
418|     return [xG,yG]
419|
420| Nuage = [[1,2],[-1,3],[2,3]]
421| print(point_moyen(Nuage))
422| ## Q3
423| def plus_proche_G(Nuage):
424|     G = point_moyen(Nuage)
425|     P = Nuage[0]
426|     m = distance(P,G)
427|     for M in Nuage:
428|         if distance(M,G) < m:
429|             m = distance(M,G)
430|             P = M
431|     return P
432|
433| Nuage = [[1,2],[-1,3],[2,3]]
434| print(plus_proche_G(Nuage))
435|
436| ## Q4
437| import matplotlib.pyplot as plt
438| def representation(Nuage):
439|     n = len(Nuage)
440|     X = [Nuage[k][0] for k in range(n)]
441|     Y = [Nuage[k][1] for k in range(n)]
442|     plt.plot(X,Y,'o',color = 'black')
443|     G = point_moyen(Nuage)
444|     plt.plot([G[0]],[G[1]],'x',color = 'blue')
445|     P = plus_proche_G(Nuage)
446|     plt.plot([P[0]],[P[1]],'*',color = 'red')
```

```
447|     plt.show()
448|
449| Nuage = [[1,2],[-1,3],[2,3]]
450| representation(Nuage)
451|
452| # Exercice 17
453| ## Q1a
454| import random as rd
455| def Tirage_avec(n,k):
456|     return [rd.randint(1,n) for _ in range(k)]
457|
458| ## Q1b
459| def X(n,k):
460|     L = Tirage_avec(n,k)
461|     return max(L)
462|
463| ## Q1c
464| def espX(n,k):
465|     N = 1000
466|     return sum([X(n,k) for _ in range(N)])/N
467|
468| ## Q2a
469| def Tirage_sans(n,k):
470|     Urne = list(range(1,n+1))
471|     T = []
472|     for _ in range(k):
473|         t = rd.choice(Urne)
474|         Urne.remove(t)
475|         T.append(t)
476|     return T
477|
478| ## Q2b
479| def Y(n,k):
480|     L = Tirage_sans(n,k)
481|     return max(L)
482|
483| ## Q2c
484| def espY(n,k):
485|     N = 1000
486|     return sum([X(n,k) for _ in range(N)])/N
487|
488| # Exercice 18
489| ## Q1
490| def somme(L,i,j):
491|     return sum(L[i:j])
492|
493| L = [3.2,2.1,7.5,2.4]
494| print(somme(L,1,3))
495|
496| ## Q2
```

```python
def sous_liste_long(L,k):
    n = len(L)
    ind = 0
    m = somme(L,0,k)
    for i in range(1,n-k+1):
        s = somme(L,i,i+k)
        if s > m:
            m = s
            ind = i
    return L[ind:ind+k]

L = [3.2,2.1,7.5,2.4]
print(sous_liste_long(L,2))

# Exercice 19
## Q1
import random as rd
def geom(p):
    X = 1
    while rd.random() > p:
        X += 1
    return X

p = 0.3
print(geom(p))

## Q2
def Y(n,p):
    return max([geom(p) for _ in range(n)])

n, p = 10, 0.7
print(Y(n,p))

# Exercice 20
## Q1
def Entre_0_et_k(L,k):
    for n in L:
        if n not in list(range(k+1)):
            return False
    return True

L = [0,2,0]
print(Entre_0_et_k(L,2))
print(Entre_0_et_k(L,1))

## Q2
def plus_frequent(L,k):
    eff = [0]*(k+1)
    for n in L:
        eff[n] += 1
```

```
547|     ind = 0
548|     for i in range(k+1):
549|         if eff[i] > eff[ind]:
550|             ind = i
551|     return ind
552| L = [0,3,2,0,1,3,1,1,1]
553| print(plus_frequent(L,4))
554|
555| # Exercice 21
556| ## Q1
557| import random as rd
558| def gene(n):
559|     chaine = ''
560|     for _ in range(n):
561|         chaine += rd.choice(['A','C','G','T'])
562|     return chaine
563|
564| print(gene(10))
565|
566| ## Q2
567| def nbAC(g):
568|     n = len(g)
569|     nb = 0
570|     for i in range(n-1):
571|         if g[i:i+2] == 'AC':
572|             nb += 1
573|     return nb
574|
575| g = gene(50)
576| print(g)
577| print(nbAC(g))
578|
579| # Exercice 22
580| ## Q1
581| def somme_cumul(L):
582|     Som = [L[0]]
583|     for k in range(1,len(L)):
584|         Som.append(Som[-1]+L[k])
585|     return Som
586|
587| L = [1,2,3]
588| print(somme_cumul(L))
589|
590| ## Q2
591| def repartition(dureesA,dureesB):
592|     SCA, SCB = somme_cumul(dureesA), somme_cumul(dureesB)
593|     A, B = [], []
594|     for i in range(len(SCA)):
595|         if SCA[i] < SCB[i]:
596|             A.append(i)
```

```
597|         elif SCA[i] > SCB[i]:
598|             B.append(i)
599|     return A, B
600|
601| dureesA, dureesB = [7,1,2,1,2], [4,2,4,3,1]
602| print(repartition(dureesA,dureesB))
603|
604| # Exercice 23
605| ## Q1
606| # P(X=k) = (μ**k/k!) exp(-μ)
607| # F(μ,k) = somme pour n de 0 à k de (μ**n/n!) exp(-μ)
608|
609| import math as m
610| def F(μ,k):
611|     p = m.exp(-μ)
612|     S = p
613|     for n in range(1,k+1):
614|         p = p*μ/n
615|         S += p
616|     return S
617|
618| print(F(2,5))
619|
620| ## Q2
621| def G(μ,x):
622|     k = 0
623|     while F(μ,k) <= x:
624|         k += 1
625|     return k
626|
627| print(G(2,0.7))
628|
629| # Exercice 24
630| ## Q1
631| def nb_vrai(L):
632|     return sum(L)
633|
634| L = [True,False,False,True,True]
635| print(nb_vrai(L))
636|
637| def bloc_vrai(L):
638|     i = 0
639|     while i < len(L) and not L[i]:
640|         i += 1
641|     c = 0
642|     while i < len(L) and L[i]:
643|         i += 1
644|         c += 1
645|     return c == nb_vrai(L)
646|
```

```
647| L = [False, True, True, True, False, False]
648| print(bloc_vrai(L))
```