```
# TIPE.py

0001| import ccxt
0002| from matplotlib.pyplot import *
0003| from math import *
0004|
0005| text_btc = 'D:\\TIPE\\btc.txt'
0006| text_valeurs = 'D:\\TIPE\\valeurs.txt'
0007| text_valeurs_comparaison = 'D:\\TIPE\\valeurs_comparaison.txt'
0008| text_correlation = 'D:\\TIPE\\correlation.txt'
0009| text_correlation_n = 'D:\\TIPE\\correlation_n.txt'
0010| text_correlation_n2 = 'D:\\TIPE\\correlation_n2.txt'
0011|
0012| text_ada_usdt = 'D:\\TIPE\\ada-usdt.txt'
0013| text_btc_usdt = 'D:\\TIPE\\btc-usdt.txt'
0014| text_doge_usdt = 'D:\\TIPE\\doge-usdt.txt'
0015| text_dot_usdt = 'D:\\TIPE\\dot-usdt.txt'
0016| text_eth_usdt = 'D:\\TIPE\\eth-usdt.txt'
0017| text_xrp_usdt = 'D:\\TIPE\\xrp-usdt.txt'
0018| text_comparaison = 'D:\\TIPE\\comparaison.txt'
0019|
0020| text_liste_heures_ada = 'D:\\TIPE\\liste-heures-ada.txt'
0021| text_liste_heures_btc_1 = 'D:\\TIPE\\liste-heures-btc-1.txt'
0022| text_liste_heures_btc_2 = 'D:\\TIPE\\liste-heures-btc-2.txt'
0023| text_liste_heures_doge = 'D:\\TIPE\\liste-heures-doge.txt'
0024| text_liste_heures_dot = 'D:\\TIPE\\liste-heures-dot.txt'
0025| text_liste_heures_eth = 'D:\\TIPE\\liste-heures-eth.txt'
0026| text_liste_heures_xrp = 'D:\\TIPE\\liste-heures-xrp.txt'
0027| text_liste_heures_comparaison = 'D:\\TIPE\\liste-heures-comparaison.txt'
0028|
0029| text_liste_hamming = 'D:\\TIPE\\liste-hamming.txt'
0030| text_liste_hamming_btc = 'D:\\TIPE\\liste-hamming-btc.txt'
0031| text_liste_hamming_comparaison = 'D:\\TIPE\\liste-hamming-comparaison.txt'
0032|
0033| text_liste_octet_ada = 'D:\\TIPE\\liste-octet-ada.txt'
0034| text_liste_octet_btc_1 = 'D:\\TIPE\\liste-octet-btc-1.txt'
0035| text_liste_octet_btc_2 = 'D:\\TIPE\\liste-octet-btc-2.txt'
0036| text_liste_octet_doge = 'D:\\TIPE\\liste-octet-doge.txt'
0037| text_liste_octet_dot = 'D:\\TIPE\\liste-octet-dot.txt'
0038| text_liste_octet_eth_1 = 'D:\\TIPE\\liste-octet-eth-1.txt'
0039| text_liste_octet_eth_2 = 'D:\\TIPE\\liste-octet-eth-2.txt'
0040| text_liste_octet_xrp = 'D:\\TIPE\\liste-octet-xrp.txt'
0041| text_liste_octet_comparaison = 'D:\\TIPE\\liste-octet-comparaison.txt'
0042|
0043| text_liste_octet_1 = 'D:\\TIPE\\liste-octet-1.txt'
0044| text_liste_octet_2 = 'D:\\TIPE\\liste-octet-2.txt'
0045| text_liste_octet_3 = 'D:\\TIPE\\liste-octet-3.txt'
0046| text_liste_octet_4 = 'D:\\TIPE\\liste-octet-4.txt'
0047|
0048| text_liste_octet2_1 = 'D:\\TIPE\\liste-octet2-1.txt'
0049| text_liste_octet2_2 = 'D:\\TIPE\\liste-octet2-2.txt'
0050| text_liste_octet2_3 = 'D:\\TIPE\\liste-octet2-3.txt'
0051| text_liste_octet2_4 = 'D:\\TIPE\\liste-octet2-4.txt'
0052|
0053| text_performance_hamming = 'D:\\TIPE\\performance-hamming.txt'
0054| text_performance_octet = 'D:\\TIPE\\performance-octet.txt'
0055| text_performance_octet2 = 'D:\\TIPE\\performance-octet2.txt'
0056| text_performance_extremum = 'D:\\TIPE\\performance-extremum.txt'
0057| text_performance_extremum_rentabilite = 'D:\\TIPE\\performance-extremum-
rentabilite.txt'
0058|
0059| liste_fichiers =
[text_ada_usdt,text_btc_usdt,text_doge_usdt,text_dot_usdt,text_eth_usdt,text_xrp_usdt
]
0060|
0061| couples_lignes = [(0,1973400),(1973400,4296721),(4296721,5630139),
(5630139,6372604),(6372604,8695951),(8695951,10644388)]
```

```
0062|
0063|
liste_fichiers_heures=[text_liste_heures_ada,text_liste_heures_btc_1,text_liste_heure
s_btc_2,text_liste_heures_doge,text_liste_heures_dot,text_liste_heures_eth,text_liste
_heures_xrp]
0064|
0065|
liste_fichiers_octet=[text_liste_octet_ada,text_liste_octet_btc_1,text_liste_octet_bt
c_2,text_liste_octet_doge,text_liste_octet_dot,text_liste_octet_eth_1,text_liste_octe
t_eth_2,text_liste_octet_xrp]
0066|
0067| exchange = ccxt.binance()
0068|
0069| def plot_unix(unix):
0070|     y=[]
0071|     l = exchange.fetch_ohlcv('ETH/USDT', since=unix, limit=1000)
0072|     for m in l:
0073|         y.append(m[1])
0074|     x=list(range(len(y)))
0075|     figure()
0076|     plot(x,y)
0077|     show()
0078|
0079| def plot_unix_2(m):
0080|     unix=premiere_occurrence(m)
0081|     y=[]
0082|     while unix <= 1642341469000:
0083|         print(unix)
0084|         l = exchange.fetch_ohlcv(m, since=unix, limit=1000)
0085|         for n in l:
0086|             y.append(n[1])
0087|         unix+=60000000
0088|     x=list(range(len(y)))
0089|     figure()
0090|     plot(x,y)
0091|     show()
0092|
0093| def fichier(m):
0094|     btc=open(text_btc, 'w')
0095|     unix=1642374000000
0096|     i=0
0097|     y=[]
0098|     while unix<1653827000000:
0099|         print(unix)
0100|         l = exchange.fetch_ohlcv(m, since=unix, limit=1000)
0101|         for n in l:
0102|             btc.write(str(n[0])+' '+str(n[1])+' '+str(n[2])+' '+str(n[3])+'
'+str(n[4])+' '+str(n[5])+'\n')
0103|         unix+=60000000
0104|         y.append(n[1])
0105|     btc.close()
0106|     x=list(range(len(y)))
0107|     figure()
0108|     plot(x,y)
0109|     show()
0110|
0111| def premiere_occurrence(m):
0112|     l = exchange.fetch_ohlcv('ETH/USDT', since=0, limit=1000)
0113|     return l[0][0]
0114|
0115| def fichier_compilation_valeurs():
0116|     valeurs=open(text_valeurs,'w')
0117|     for fichier in liste_fichiers:
0118|         print(0)
0119|         f=open(fichier,'r')
0120|         c=f.readlines()
0121|         for ligne in c:
```

```
0122|             l=ligne.split(' ')
0123|             valeurs.write(l[1]+"\n")
0124|         f.close()
0125|     valeurs.close()
0126|
0127| def fichier_compilation_valeurs_2():
0128|     valeurs=open(text_valeurs_comparaison,'w')
0129|     f=open(text_comparaison,'r')
0130|     c=f.readlines()
0131|     for ligne in c:
0132|         l=ligne.split(' ')
0133|         valeurs.write(l[1]+"\n")
0134|     f.close()
0135|     valeurs.close()
0136|
0137| def etablissement_liste_heures():
0138|     lh=open(text_liste_heures_comparaison,'w')
0139|     valeurs=open(text_valeurs_comparaison,'r')
0140|     v=valeurs.read().splitlines()
0141|     c=len(v)
0142|     for i in range(c-59):
0143|         if i%10000==0: print(i)
0144|         a=''
0145|         for j in range(60):
0146|             a+=v[i+j]
0147|             a+=' '
0148|         lh.write(a+'\n')
0149|     valeurs.close()
0150|     lh.close()
0151|
0152| def etablissement_liste_hamming():
0153|     lham=open(text_liste_hamming,'w')
0154|     for fichier in liste_fichiers_heures:
0155|         print(0)
0156|         f=open(fichier,'r')
0157|         c=f.readlines()
0158|         for ligne in c:
0159|             s=''
0160|             l=ligne.split(' ')
0161|             for i in range(59):
0162|                 a,b=l[i],l[i+1]
0163|                 if a<b: s+='2'
0164|                 elif a>b: s+='0'
0165|                 else: s+='1'
0166|             lham.write(s+'\n')
0167|         f.close()
0168|     lham.close()
0169|
0170| def etablissement_liste_hamming_2():
0171|     lham=open(text_liste_hamming_comparaison,'w')
0172|     f=open(text_liste_heures_comparaison,'r')
0173|     c=f.readlines()
0174|     for ligne in c:
0175|         s=''
0176|         l=ligne.split(' ')
0177|         for i in range(59):
0178|             a,b=l[i],l[i+1]
0179|             if a<b: s+='2'
0180|             elif a>b: s+='0'
0181|             else: s+='1'
0182|         lham.write(s+'\n')
0183|     f.close()
0184|     lham.close()
0185|
0186| def hamming_1(a,b):
0187|     d=0
0188|     for i in range(len(a)):
```

```
0189|         d+=abs(int(a[i])-int(b[i]))
0190|     return d
0191|
0192| def octet(n):
0193|     a=bin(n)[2:]
0194|     b='0'*(8-len(a))
0195|     return b+a+' '
0196|
0197| def etablissement_liste_octet():
0198|     lo=open(text_liste_octet_2,'w')
0199|     f=open(text_liste_heures_comparaison,'r')
0200|     c=f.readlines()
0201|     for i in range(len(c)):
0202|         ligne=c[i]
0203|         if i%10000==0: print(i)
0204|         l=ligne.split(' ')[:-1]
0205|         m,s=[],''
0206|         for x in l: m.append(float(x))
0207|         a,b=min(m),max(m)
0208|         for fl in m:
0209|             if a==b: d=0
0210|             else: d=int(255*(fl-a)/(b-a))
0211|             s+=str(d)+' '
0212|         lo.write(s+'\n')
0213|     f.close()
0214|     lo.close()
0215|
0216| def etablissement_liste_correlation():
0217|     valeurs=open(text_valeurs,'r')
0218|     tc=open(text_correlation,'w')
0219|     v=valeurs.read().splitlines()
0220|     ada=1231000
0221|     btc=1581000+1973400
0222|     doge=4296721+591000
0223|     dot=5630139
0224|     eth=6372604+1580967
0225|     xrp=8695951+1206000
0226|     for i in range(742321):
0227|         tc.write(v[ada]+' '+v[btc]+' '+v[doge]+' '+v[dot]+' '+v[eth]+' '+v[xrp]
+'\n')
0228|         ada+=1
0229|         btc+=1
0230|         doge+=1
0231|         dot+=1
0232|         eth+=1
0233|         xrp+=1
0234|     valeurs.close()
0235|     tc.close()
0236|
0237| def etablissement_liste_correlation_n():
0238|     tc=open(text_correlation,'r')
0239|     cn=open(text_correlation_n,'w')
0240|     c=tc.read().splitlines()
0241|     m=[[],[],[],[],[],[]]
0242|     p=[]
0243|     for ligne in c:
0244|         l=ligne.split(' ')
0245|         for i in range(6):
0246|             m[i].append(float(l[i]))
0247|     for n in m:
0248|         q=[]
0249|         a,b=min(n),max(n)
0250|         for fl in n:
0251|             q.append(str(int(255*(fl-a)/(b-a))))
0252|         p.append(q)
0253|     for i in range(len(p[0])):
0254|         cn.write(p[0][i]+' '+p[1][i]+' '+p[2][i]+' '+p[3][i]+' '+p[4][i]+'
```

```
'+p[5][i]+'\n')
0255|     tc.close()
0256|     cn.close()
0257|
0258| def moyenne(L):
0259|     s=0
0260|     for i in L: s+=i
0261|     return s/len(L)
0262|
0263| def graphe_correlation():
0264|     cn=open(text_correlation_n,'r')
0265|     c=cn.readlines()
0266|     cn.close()
0267|     x=list(range(len(c)//100))
0268|     y0,y1,y2,y3,y4,y5=[],[],[],[],[],[]
0269|     for i in range(len(c)//100):
0270|         l0,l1,l2,l3,l4,l5=[],[],[],[],[],[]
0271|         for j in range(100):
0272|             ligne=c[100*i+j]
0273|             l=ligne.split(' ')
0274|             l0.append(int(l[0]))
0275|             l1.append(int(l[1]))
0276|             l2.append(int(l[2]))
0277|             l3.append(int(l[3]))
0278|             l4.append(int(l[4]))
0279|             l5.append(int(l[5]))
0280|         y0.append(moyenne(l0))
0281|         y1.append(moyenne(l1))
0282|         y2.append(moyenne(l2))
0283|         y3.append(moyenne(l3))
0284|         y4.append(moyenne(l4))
0285|         y5.append(moyenne(l5))
0286|     figure()
0287|     # plot(x,y0)
0288|     plot(x,y1)
0289|     plot(x,y2)
0290|     # plot(x,y3)
0291|     # plot(x,y4)
0292|     # plot(x,y5)
0293|     show()
0294|
0295| def ecart_type(L):
0296|     a=moyenne(L)**2
0297|     M=[]
0298|     for i in L: M.append(i**2)
0299|     b=moyenne(M)
0300|     return (b-a)**(1/2)
0301|
0302|
0303| def etablissement_liste_correlation_n2():
0304|     tc=open(text_correlation,'r')
0305|     cn=open(text_correlation_n2,'w')
0306|     c=tc.read().splitlines()
0307|     m=[[],[],[],[],[],[]]
0308|     p=[]
0309|     for ligne in c:
0310|         l=ligne.split(' ')
0311|         for i in range(6):
0312|             m[i].append(float(l[i]))
0313|     for n in m:
0314|         q=[]
0315|         a,b=moyenne(n),ecart_type(n)
0316|         for fl in n:
0317|             q.append(str((fl-a)/b))
0318|         p.append(q)
0319|     for i in range(len(p[0])):
0320|         cn.write(p[0][i]+' '+p[1][i]+' '+p[2][i]+' '+p[3][i]+' '+p[4][i]+'
```

```
'+p[5][i]+'\n')
0321|     tc.close()
0322|     cn.close()
0323|
0324| def graphe_correlation_2(n):
0325|     cn=open(text_correlation_n2,'r')
0326|     c=cn.readlines()
0327|     cn.close()
0328|     x=list(range(len(c)//n))
0329|     y0,y1,y2,y3,y4,y5=[],[],[],[],[],[]
0330|     for i in range(len(c)//n):
0331|         l0,l1,l2,l3,l4,l5=[],[],[],[],[],[]
0332|         for j in range(n):
0333|             ligne=c[n*i+j]
0334|             l=ligne.split(' ')
0335|             l0.append(float(l[0]))
0336|             l1.append(float(l[1]))
0337|             l2.append(float(l[2]))
0338|             l3.append(float(l[3]))
0339|             l4.append(float(l[4]))
0340|             l5.append(float(l[5]))
0341|         y0.append(moyenne(l0))
0342|         y1.append(moyenne(l1))
0343|         y2.append(moyenne(l2))
0344|         y3.append(moyenne(l3))
0345|         y4.append(moyenne(l4))
0346|         y5.append(moyenne(l5))
0347|     figure()
0348|     #plot(x,y0)
0349|     plot(x,y1,color='blue',label='Bitcoin')
0350|     # plot(x,y2)
0351|     plot(x,y3,color='red',label='Polkadot')
0352|     #plot(x,y4)
0353|     #plot(x,y5)
0354|     xlabel('Jours')
0355|     legend()
0356|     show()
0357|
0358| def coefficient_de_correlation(L1,L2):
0359|     e1,e2=ecart_type(L1),ecart_type(L2)
0360|     b=moyenne(L1)*moyenne(L2)
0361|     M=[]
0362|     for i in range(len(L1)):
0363|         M.append(L1[i]*L2[i])
0364|     return (moyenne(M)-b)/(e1*e2)
0365|
0366| def liste_coefficients_de_correlation():
0367|     tc=open(text_correlation,'r')
0368|     c=tc.read().splitlines()
0369|     m=[[],[],[],[],[],[]]
0370|     p=[]
0371|     for ligne in c:
0372|         l=ligne.split(' ')
0373|         for i in range(6):
0374|             m[i].append(float(l[i]))
0375|     a=[]
0376|     for i in range(6):
0377|         for j in range(6):
0378|             a.append(coefficient_de_correlation(m[i],m[j]))
0379|     return a
0380|
0381| def couleurs():
0382|     figure()
0383|     for i in range(0,6):
0384|         plot([0,1],[i,i])
0385|     show()
0386|
```

```
0387| def ordre(L): return L[1]
0388|
0389| def comparaison_hamming(s): #Retourne la liste des distances de hamming avec la
base de données
0390|     lh=open(text_liste_hamming_btc,'r')
0391|     L=[]
0392|     c=lh.read().splitlines()
0393|     for i in range(len(c)-60):
0394|         if i%100000==0:print(i//100000)
0395|         l=c[i]
0396|         L.append((l,hamming_1(s,l),i))
0397|     M=sorted(L,key=ordre)
0398|     N=[]
0399|     for i in range(10):
0400|         N.append(M[i][2])
0401|     return N
0402|
0403| def etablissement_liste_hamming_btc():
0404|     lham=open(text_liste_hamming_btc,'w')
0405|     f=open(text_liste_heures_btc_2,'r')
0406|     c=f.readlines()
0407|     for ligne in c:
0408|         s=''
0409|         l=ligne.split(' ')
0410|         for i in range(59):
0411|             a,b=l[i],l[i+1]
0412|             if a<b: s+='2'
0413|             elif a>b: s+='0'
0414|             else: s+='1'
0415|         lham.write(s+'\n')
0416|     f.close()
0417|     lham.close()
0418|
0419| def graphe_cours(A,L):
0420|     figure()
0421|     X=list(range(len(L[0])))
0422|     plot(X,A)
0423|     for Y in L:
0424|         plot(X,Y)
0425|     show()
0426|
0427| def graphe_hamming(n):
0428|     tch=open(text_liste_hamming_comparaison,'r')
0429|     ch=tch.read().splitlines()
0430|     tc=open(text_liste_octet_2,'r')
0431|     c=tc.read().splitlines()
0432|     th=open(text_liste_octet_1,'r')
0433|     h=th.read().splitlines()
0434|     L=comparaison_hamming(ch[n])
0435|     M=[]
0436|     CN=c[n].split(' ')
0437|     A=[]
0438|     for i in range(60): A.append(int(CN[i]))
0439|     for i in L:
0440|         H=h[i].split(' ')
0441|         H1=[]
0442|         for i in range(60): H1.append(int(H[i]))
0443|         M.append(H1)
0444|     graphe_cours(A,M)
0445|
0446| def etablissement_liste_octet_2():
0447|     lo=open(text_liste_octet_4,'w')
0448|     f=open(text_liste_heures_comparaison,'r')
0449|     c=f.readlines()
0450|     for i in range(len(c)-60):
0451|         ligne=c[i]
0452|         ligne2=c[i+60]
```

```
0453|         if i%10000==0: print(i)
0454|         l=ligne.split(' ')[:-1]
0455|         l2=ligne2.split(' ')[:-1]
0456|         m,s=[],''
0457|         for x in l: m.append(float(x))
0458|         a,b=min(m),max(m)
0459|         for x in l2: m.append(float(x))
0460|         for fl in m:
0461|             if a==b: d=0
0462|             else: d=int(255*(fl-a)/(b-a))
0463|             s+=str(d)+' '
0464|         lo.write(s+'\n')
0465|     f.close()
0466|     lo.close()
0467|
0468| def graphe_cours_2(A,L):
0469|     figure()
0470|     X1=list(range(60))
0471|     X2=list(range(59,120))
0472|     A1,A2=[],[A[59]]
0473|     for i in range(60):
0474|         A1.append(A[i])
0475|         A2.append(A[i+60])
0476|     plot(X1,A1,color='blue',label='Cours évalué')
0477|     plot(X2,A2,color='red',label='Évolution réelle')
0478|     B=len(X2)*[A[59]]
0479|     if len(L)!=0:
0480|         Y=L[0]
0481|         Y1,Y2=[],[Y[59]]
0482|         for i in range(60):
0483|             Y1.append(Y[i])
0484|             Y2.append(Y[60+i])
0485|         plot(X1,Y1,color='0.4',label='Prédiction')
0486|         plot(X2,Y2,color='0.4')
0487|     #     for i in range(1,len(L)):
0488|     #         Y=L[i]
0489|     #         Y1,Y2=[],[Y[59]]
0490|     #         for i in range(60):
0491|     #             Y1.append(Y[i])
0492|     #             Y2.append(Y[60+i])
0493|     #         plot(X1,Y1,color='0.7')
0494|     #         plot(X2,Y2,color='0.7')
0495|     plot(X1,A1,color='blue')
0496|     plot(X2,A2,color='red')
0497|     plot(X2,B,linestyle='dashed',color='black')
0498|     legend()
0499|     show()
0500|
0501| def graphe_hamming_2(n):
0502|     tch=open(text_liste_hamming_comparaison,'r')
0503|     ch=tch.read().splitlines()
0504|     tc=open(text_liste_octet_4,'r')
0505|     c=tc.read().splitlines()
0506|     th=open(text_liste_octet_3,'r')
0507|     h=th.read().splitlines()
0508|     L=comparaison_hamming(ch[n])
0509|     M=[]
0510|     CN=c[n].split(' ')
0511|     A=[]
0512|     for i in range(120): A.append(int(CN[i]))
0513|     for i in L:
0514|         H=h[i].split(' ')
0515|         H1=[]
0516|         for i in range(120): H1.append(int(H[i]))
0517|         M.append(H1)
0518|     graphe_cours_2(A,M)
0519|
```

```
0520| def performance_hamming(n):
0521|     tch=open(text_liste_hamming_comparaison,'r')
0522|     ch=tch.read().splitlines()
0523|     tc=open(text_liste_octet_4,'r')
0524|     c=tc.read().splitlines()
0525|     th=open(text_liste_octet_3,'r')
0526|     h=th.read().splitlines()
0527|     L=comparaison_hamming(ch[n])
0528|     p=0
0529|     CN=c[n].split(' ')
0530|     a=int(CN[59])<int(CN[119])
0531|     print(int(CN[59]),int(CN[119]))
0532|     for i in L:
0533|         H=h[i].split(' ')
0534|         b=int(H[59])<int(H[119])
0535|         print(int(H[59]),int(H[119]))
0536|         if a==b: p+=1
0537|     return p
0538|
0539| def performance_hamming_complet():
0540|     tch=open(text_liste_hamming_comparaison,'r')
0541|     ch=tch.read().splitlines()
0542|     tc=open(text_liste_octet_4,'r')
0543|     co=tc.read().splitlines()
0544|     th=open(text_liste_octet_3,'r')
0545|     h=th.read().splitlines()
0546|     lh=open(text_liste_hamming_btc,'r')
0547|     c=lh.read().splitlines()
0548|     for n in range(192,len(ch)):
0549|         ph=open(text_performance_hamming,'r')
0550|         PH=ph.read()
0551|         ph.close()
0552|         L=[]
0553|         for i in range(len(c)-60):
0554|             l=c[i]
0555|             L.append((l,hamming_1(ch[n],l),i))
0556|         M=sorted(L,key=ordre)
0557|         N=[]
0558|         for i in range(10):
0559|             N.append(M[i][2])
0560|         p=0
0561|         CN=co[n].split(' ')
0562|         a=int(CN[59])<int(CN[119])
0563|         if a:
0564|             te='H'
0565|         else:
0566|             te='B'
0567|         for i in N:
0568|             H=h[i].split(' ')
0569|             b=int(H[59])<int(H[119])
0570|             if a==b: p+=1
0571|         print(n,te,p)
0572|         ph=open(text_performance_hamming,'w')
0573|         ph.write(PH+te+' '+str(p)+'\n')
0574|         ph.close()
0575|
0576| def comparaison_octet(s):
0577|     lh=open(text_liste_octet_1,'r')
0578|     L=[]
0579|     c=lh.read().splitlines()
0580|     for i in range(len(c)-60):
0581|         if i%100000==0:print(i//100000)
0582|         l=c[i]
0583|         L.append((l,distance_octet(s,l),i))
0584|     M=sorted(L,key=ordre)
0585|     N=[]
0586|     for i in range(10):
```

```
0587|         N.append(M[i][2])
0588|     return N
0589|
0590| def distance_octet(l,m):
0591|     L=l.split(' ')[:-1]
0592|     M=m.split(' ')[:-1]
0593|     d=0
0594|     for i in range(len(L)):
0595|         d+=abs(int(L[i])-int(M[i]))
0596|     return d
0597|
0598| def graphe_octet(n):
0599|     tch=open(text_liste_octet_2,'r')
0600|     ch=tch.read().splitlines()
0601|     tch.close
0602|     tc=open(text_liste_octet_2,'r')
0603|     c=tc.read().splitlines()
0604|     tc.close()
0605|     th=open(text_liste_octet_1,'r')
0606|     h=th.read().splitlines()
0607|     th.close()
0608|     L=comparaison_octet(ch[n])
0609|     M=[]
0610|     CN=c[n].split(' ')
0611|     A=[]
0612|     for i in range(60): A.append(int(CN[i]))
0613|     for i in L:
0614|         H=h[i].split(' ')
0615|         H1=[]
0616|         for i in range(60): H1.append(int(H[i]))
0617|         M.append(H1)
0618|     graphe_cours(A,M)
0619|
0620| def graphe_octet_2(n):
0621|     tch=open(text_liste_octet_2,'r')
0622|     ch=tch.read().splitlines()
0623|     tch.close()
0624|     tc=open(text_liste_octet_4,'r')
0625|     c=tc.read().splitlines()
0626|     tc.close()
0627|     th=open(text_liste_octet_3,'r')
0628|     h=th.read().splitlines()
0629|     th.close()
0630|     L=comparaison_octet(ch[n])
0631|     M=[]
0632|     CN=c[n].split(' ')
0633|     A=[]
0634|     for i in range(120): A.append(int(CN[i]))
0635|     for i in L:
0636|         H=h[i].split(' ')
0637|         print(H)
0638|         H1=[]
0639|         for i in range(120): H1.append(int(H[i]))
0640|         M.append(H1)
0641|     graphe_cours_2(A,M)
0642|
0643| def performance_octet(n):
0644|     tch=open(text_liste_octet_2,'r')
0645|     ch=tch.read().splitlines()
0646|     tch.close()
0647|     tc=open(text_liste_octet_4,'r')
0648|     c=tc.read().splitlines()
0649|     tc.close()
0650|     th=open(text_liste_octet_3,'r')
0651|     h=th.read().splitlines()
0652|     th.close()
0653|     L=comparaison_octet(ch[n])
```

```
    p=0
    CN=c[n].split(' ')
    a=int(CN[59])<int(CN[119])
    print(int(CN[59]),int(CN[119]))
    for i in L:
        H=h[i].split(' ')
        b=int(H[59])<int(H[119])
        print(int(H[59]),int(H[119]))
        if a==b: p+=1
    return p

def performance_octet_complet():
    tch=open(text_liste_octet_2,'r')
    ch=tch.read().splitlines()
    tch.close()
    tc=open(text_liste_octet_4,'r')
    co=tc.read().splitlines()
    tc.close()
    th=open(text_liste_octet_3,'r')
    h=th.read().splitlines()
    th.close()
    lh=open(text_liste_octet_1,'r')
    c=lh.read().splitlines()
    lh.close()
    for n in range(149,len(ch)):
        ph=open(text_performance_octet,'r')
        PH=ph.read()
        ph.close()
        L=[]
        for i in range(len(c)-60):
            l=c[i]
            L.append((l,distance_octet(ch[n],l),i))
        M=sorted(L,key=ordre)
        N=[]
        for i in range(10):
            N.append(M[i][2])
        p=0
        CN=co[n].split(' ')
        a=int(CN[59])<int(CN[119])
        if a:
            te='H'
        else:
            te='B'
        for i in N:
            H=h[i].split(' ')
            b=int(H[59])<int(H[119])
            if a==b: p+=1
        print(n,te,p)
        ph=open(text_performance_octet,'w')
        ph.write(PH+te+' '+str(p)+'\n')
        ph.close()

def stats():
    tc=open(text_liste_octet_3,'r')
    c=tc.read().splitlines()
    tc.close()
    c1,c2=0,0
    for ligne in c:
        l=ligne.split(' ')
        a1,a2,a3=int(l[0]),int(l[59]),int(l[119])
        if a1>a2:
            c1+=1
            if a2>a3:
                c2+=1
    return c1,c2

def stats_2():
```

```
0721|     tc=open(text_liste_octet_3,'r')
0722|     c=tc.read().splitlines()
0723|     tc.close()
0724|     c1,c2=0,0
0725|     for ligne in c:
0726|         l=ligne.split(' ')
0727|         a1,a2,a3=int(l[0]),int(l[59]),int(l[119])
0728|         if a1<a2:
0729|             c1+=1
0730|             c2+=(a3-a2)/(a2-a1)
0731|     return c1,c2
0732| 
0733| def stats_3():
0734|     tc=open(text_liste_octet_3,'r')
0735|     c=tc.read().splitlines()
0736|     tc.close()
0737|     C=[]
0738|     for i in range(60):C.append([0,0])
0739|     for ligne in c:
0740|         l=ligne.split(' ')
0741|         a1,a2,a3=int(l[0]),int(l[59]),int(l[119])
0742|         k=[]
0743|         for i in range(60):
0744|             k.append(int(l[i]))
0745|         m=min(k)
0746|         if a1>a2:
0747|             pos=k.index(m)
0748|             C[pos][0]+=1
0749|             if a2<a3:
0750|                 C[pos][1]+=1
0751|     return C
0752| 
0753| def graphe_stats_3():
0754|     C=stats_3()
0755|     X=list(range(5,60))
0756|     Y=[]
0757|     for i in range(5,60):
0758|         Y.append(C[i][1]/C[i][0])
0759|     figure()
0760|     Z=55*[0.5]
0761|     plot(X,Y,color='blue')
0762|     plot(X,Z,color='black')
0763|     show()
0764| 
0765| def etude_resultats_hamming():
0766|     ph=open(text_performance_hamming)
0767|     PH=ph.read().splitlines()
0768|     ph.close()
0769|     d,v,f=0,0,0
0770|     L=[]
0771|     for ligne in PH:
0772|         l=ligne.split(' ')
0773|         r=int(l[1])
0774|         d+=r
0775|         if r>5: v+=2
0776|         elif r<5: f+=2
0777|         else:
0778|             v+=1
0779|             f+=1
0780|         L.append((l[0],r))
0781|     return d,len(L),d/len(L),v,f
0782| 
0783| def etude_resultats_octet():
0784|     ph=open(text_performance_octet)
0785|     PH=ph.read().splitlines()
0786|     ph.close()
0787|     d,v,f=0,0,0
```

```
    L=[]
    for ligne in PH:
        l=ligne.split(' ')
        r=int(l[1])
        d+=r
        if r>5: v+=2
        elif r<5: f+=2
        else:
            v+=1
            f+=1
        L.append((l[0],r))
    return d,len(L),d/len(L),v,f

def etablissement_liste_octet_3():
    lo=open(text_liste_octet2_1,'w')
    f=open(text_liste_heures_btc_2,'r')
    c=f.readlines()
    for i in range(len(c)):
        ligne=c[i]
        if i%10000==0: print(i)
        l=ligne.split(' ')[:-1]
        m,s=[],''
        for x in l: m.append(float(x))
        a,b=min(m),max(m)
        for i in range(6):
            e=0
            if a==b: d=0
            else:
                for j in range(10):
                    fl=m[10*i+j]
                    e+=int(255*(fl-a)/(b-a))
                d=e//10
            s+=str(d)+' '
        lo.write(s+'\n')
    f.close()
    lo.close()

def etablissement_liste_octet_4():
    lo=open(text_liste_octet2_3,'w')
    f=open(text_liste_heures_btc_2,'r')
    c=f.readlines()
    for i in range(len(c)-60):
        ligne=c[i]
        ligne2=c[i+60]
        if i%10000==0: print(i)
        l=ligne.split(' ')[:-1]
        l2=ligne2.split(' ')[:-1]
        m,s=[],''
        for x in l: m.append(float(x))
        a,b=min(m),max(m)
        for x in l2: m.append(float(x))
        for i in range(12):
            e=0
            if a==b: d=0
            else:
                for j in range(10):
                    fl=m[10*i+j]
                    e+=int(255*(fl-a)/(b-a))
                d=e//10
            s+=str(d)+' '
        lo.write(s+'\n')
    f.close()
    lo.close()

def comparaison_octet2(s):
    lh=open(text_liste_octet2_1,'r')
    L=[]
```

```
0855|     c=lh.read().splitlines()
0856|     for i in range(len(c)-60):
0857|         if i%100000==0:print(i//100000)
0858|         l=c[i]
0859|         L.append((l,distance_octet(s,l),i))
0860|     M=sorted(L,key=ordre)
0861|     N=[]
0862|     for i in range(10):
0863|         N.append(M[i][2])
0864|     return N
0865|
0866| def graphe_octet2(n):
0867|     tch=open(text_liste_octet2_2,'r')
0868|     ch=tch.read().splitlines()
0869|     tch.close
0870|     tc=open(text_liste_octet2_2,'r')
0871|     c=tc.read().splitlines()
0872|     tc.close()
0873|     th=open(text_liste_octet2_1,'r')
0874|     h=th.read().splitlines()
0875|     th.close()
0876|     L=comparaison_octet2(ch[n])
0877|     M=[]
0878|     CN=c[n].split(' ')
0879|     A=[]
0880|     for i in range(6): A.append(int(CN[i]))
0881|     for i in L:
0882|         H=h[i].split(' ')
0883|         H1=[]
0884|         for i in range(6): H1.append(int(H[i]))
0885|         M.append(H1)
0886|     graphe_cours(A,M)
0887|
0888| def graphe_octet2_2(n):
0889|     tch=open(text_liste_octet2_2,'r')
0890|     ch=tch.read().splitlines()
0891|     tch.close()
0892|     tc=open(text_liste_octet2_4,'r')
0893|     c=tc.read().splitlines()
0894|     tc.close()
0895|     th=open(text_liste_octet2_3,'r')
0896|     h=th.read().splitlines()
0897|     th.close()
0898|     L=comparaison_octet2(ch[n])
0899|     M=[]
0900|     CN=c[n].split(' ')
0901|     A=[]
0902|     for i in range(12): A.append(int(CN[i]))
0903|     for i in L:
0904|         H=h[i].split(' ')
0905|         H1=[]
0906|         for i in range(12): H1.append(int(H[i]))
0907|         M.append(H1)
0908|     graphe_cours_2(A,M)
0909|
0910| def performance_octet2(n):
0911|     tch=open(text_liste_octet2_2,'r')
0912|     ch=tch.read().splitlines()
0913|     tch.close()
0914|     tc=open(text_liste_octet2_4,'r')
0915|     c=tc.read().splitlines()
0916|     tc.close()
0917|     th=open(text_liste_octet2_3,'r')
0918|     h=th.read().splitlines()
0919|     th.close()
0920|     L=comparaison_octet2(ch[n])
0921|     p=0
```

```
0922|      CN=c[n].split(' ')
0923|      a=int(CN[5])<int(CN[11])
0924|      print(int(CN[5]),int(CN[11]))
0925|      for i in L:
0926|          H=h[i].split(' ')
0927|          b=int(H[5])<int(H[11])
0928|          print(int(H[5]),int(H[11]))
0929|          if a==b: p+=1
0930|      return p
0931|
0932|  def performance_octet2_complet():
0933|      tch=open(text_liste_octet2_2,'r')
0934|      ch=tch.read().splitlines()
0935|      tch.close()
0936|      tc=open(text_liste_octet2_4,'r')
0937|      co=tc.read().splitlines()
0938|      tc.close()
0939|      th=open(text_liste_octet2_3,'r')
0940|      h=th.read().splitlines()
0941|      th.close()
0942|      lh=open(text_liste_octet2_1,'r')
0943|      c=lh.read().splitlines()
0944|      lh.close()
0945|      for n in range(1000):
0946|          ph=open(text_performance_octet2,'r')
0947|          PH=ph.read()
0948|          ph.close()
0949|          L=[]
0950|          for i in range(len(c)-60):
0951|              l=c[i]
0952|              L.append((l,distance_octet(ch[n],l),i))
0953|          M=sorted(L,key=ordre)
0954|          N=[]
0955|          for i in range(10):
0956|              N.append(M[i][2])
0957|          p=0
0958|          CN=co[n].split(' ')
0959|          a=int(CN[5])<int(CN[11])
0960|          if a:
0961|              te='H'
0962|          else:
0963|              te='B'
0964|          for i in N:
0965|              H=h[i].split(' ')
0966|              b=int(H[5])<int(H[11])
0967|              if a==b: p+=1
0968|          print(n,te,p)
0969|          ph=open(text_performance_octet2,'w')
0970|          ph.write(PH+te+' '+str(p)+'\n')
0971|          ph.close()
0972|
0973|  def etude_resultats_octet2():
0974|      ph=open(text_performance_octet2)
0975|      PH=ph.read().splitlines()
0976|      ph.close()
0977|      d,v,f=0,0,0
0978|      L=[]
0979|      for ligne in PH:
0980|          l=ligne.split(' ')
0981|          r=int(l[1])
0982|          d+=r
0983|          if r>5: v+=2
0984|          elif r<5: f+=2
0985|          else:
0986|              v+=1
0987|              f+=1
0988|          L.append((l[0],r))
```

```
0989|     return d,len(L),d/len(L),v,f
0990|
0991| def stats_3_h():
0992|     tc=open(text_liste_octet_3,'r')
0993|     c=tc.read().splitlines()
0994|     tc.close()
0995|     C=[]
0996|     for i in range(60):C.append([0,0])
0997|     for ligne in c:
0998|         l=ligne.split(' ')
0999|         a1,a2,a3=int(l[0]),int(l[59]),int(l[119])
1000|         k=[]
1001|         for i in range(60):
1002|             k.append(int(l[i]))
1003|         m=max(k)
1004|         if a1<a2:
1005|             pos=k.index(m)
1006|             C[pos][0]+=1
1007|             if a2<a3:
1008|                 C[pos][1]+=1
1009|     return C
1010|
1011| def stats_3_b():
1012|     tc=open(text_liste_octet_3,'r')
1013|     c=tc.read().splitlines()
1014|     tc.close()
1015|     C=[]
1016|     for i in range(60):C.append([0,0])
1017|     for ligne in c:
1018|         l=ligne.split(' ')
1019|         a1,a2,a3=int(l[0]),int(l[59]),int(l[119])
1020|         k=[]
1021|         for i in range(60):
1022|             k.append(int(l[i]))
1023|         m=min(k)
1024|         if a1>a2:
1025|             pos=k.index(m)
1026|             C[pos][0]+=1
1027|             if a2<a3:
1028|                 C[pos][1]+=1
1029|     return C
1030|
1031| def comparaison_extremum():
1032|     Ch=[[0, 0], [1294, 614], [1937, 947], [2451, 1187], [2913, 1434], [3323,
1671], [3712, 1854], [3913, 1931], [4148, 2082], [4374, 2174], [4520, 2201], [4875,
2385], [5035, 2459], [5241, 2556], [5364, 2642], [5556, 2765], [5759, 2824], [5947,
2949], [6166, 3090], [6377, 3190], [6563, 3288], [6784, 3394], [7027, 3504], [7188,
3571], [7429, 3617], [7548, 3686], [7805, 3842], [8007, 3955], [8139, 4027], [8298,
4103], [8448, 4183], [8642, 4312], [8794, 4403], [9032, 4519], [9183, 4596], [9449,
4753], [9765, 4861], [10009, 4984], [10296, 5146], [10547, 5261], [10945, 5487],
[11323, 5606], [11709, 5819], [12071, 5959], [12527, 6079], [12981, 6305], [13479,
6530], [14023, 6821], [14619, 7049], [15360, 7485], [16134, 7832], [17082, 8183],
[18042, 8615], [19255, 9142], [20662, 9768], [22496, 10537], [25165, 11667], [29317,
13401], [37708, 16965], [71612, 31543]]
1033|     Cb=[[0, 0], [1210, 683], [1915, 1038], [2500, 1346], [2933, 1552], [3317,
1728], [3587, 1857], [3903, 2033], [4117, 2156], [4334, 2282], [4432, 2358], [4659,
2438], [4839, 2553], [5054, 2645], [5275, 2763], [5438, 2878], [5629, 2962], [5815,
3020], [6110, 3143], [6274, 3269], [6417, 3330], [6634, 3443], [6856, 3566], [6968,
3607], [7167, 3744], [7336, 3859], [7529, 3932], [7700, 3993], [7845, 4096], [8021,
4154], [8156, 4222], [8339, 4286], [8479, 4385], [8656, 4470], [8895, 4546], [9134,
4667], [9399, 4833], [9668, 5007], [9962, 5175], [10241, 5295], [10453, 5362],
[10871, 5595], [11197, 5820], [11599, 6078], [11986, 6334], [12463, 6611], [13006,
6906], [13611, 7200], [14220, 7598], [14903, 7974], [15650, 8316], [16501, 8745],
[17502, 9341], [18727, 9996], [20250, 10914], [22062, 12019], [24677, 13594], [28862,
15927], [36617, 20645], [65563, 37980]]
1034|     tc=open(text_liste_octet_4,'r')
1035|     c=tc.read().splitlines()
```

```
    tc.close()
    d,v,f=0,0,0
    for ligne in c:
        d+=1
        l=ligne.split(' ')
        a1,a2,a3=int(l[0]),int(l[59]),int(l[119])
        if a1<=a2:
            k=[]
            for i in range(60):
                k.append(int(l[i]))
            m=max(k)
            pos=k.index(m)
            b1=Ch[pos][1]>0.5*Ch[pos][0]
            b2=a2<=a3
        else:
            k=[]
            for i in range(60):
                k.append(int(l[i]))
            m=min(k)
            pos=k.index(m)
            b1=Cb[pos][1]>0.5*Cb[pos][0]
            b2=a2<a3
        if b1==b2:
            v+=1
        else: f+=1
        if d%1000==0:
            print(d,v,f)
            # ph=open(text_performance_extremum,'r')
            # PH=ph.read()
            # ph.close()
            # ph=open(text_performance_extremum,'w')
            # ph.write(PH+str(d)+' '+str(v)+' '+str(f)+' '+'\n')
            # ph.close()
    print(d,v,f)
    # ph=open(text_performance_extremum,'r')
    # PH=ph.read()
    # ph.close()
    # ph=open(text_performance_extremum,'w')
    # ph.write(PH+str(d)+' '+str(v)+' '+str(f)+' '+'\n')
    # ph.close()

def comparaison_extremum_rentabilite():
    Ch=[[0, 0], [1294, 614], [1937, 947], [2451, 1187], [2913, 1434], [3323,
1671], [3712, 1854], [3913, 1931], [4148, 2082], [4374, 2174], [4520, 2201], [4875,
2385], [5035, 2459], [5241, 2556], [5364, 2642], [5556, 2765], [5759, 2824], [5947,
2949], [6166, 3090], [6377, 3190], [6563, 3288], [6784, 3394], [7027, 3504], [7188,
3571], [7429, 3617], [7548, 3686], [7805, 3842], [8007, 3955], [8139, 4027], [8298,
4103], [8448, 4183], [8642, 4312], [8794, 4403], [9032, 4519], [9183, 4596], [9449,
4753], [9765, 4861], [10009, 4984], [10296, 5146], [10547, 5261], [10945, 5487],
[11323, 5606], [11709, 5819], [12071, 5959], [12527, 6079], [12981, 6305], [13479,
6530], [14023, 6821], [14619, 7049], [15360, 7485], [16134, 7832], [17082, 8183],
[18042, 8615], [19255, 9142], [20662, 9768], [22496, 10537], [25165, 11667], [29317,
13401], [37708, 16965], [71612, 31543]]
    Cb=[[0, 0], [1210, 683], [1915, 1038], [2500, 1346], [2933, 1552], [3317,
1728], [3587, 1857], [3903, 2033], [4117, 2156], [4334, 2282], [4432, 2358], [4659,
2438], [4839, 2553], [5054, 2645], [5275, 2763], [5438, 2878], [5629, 2962], [5815,
3020], [6110, 3143], [6274, 3269], [6417, 3330], [6634, 3443], [6856, 3566], [6968,
3607], [7167, 3744], [7336, 3859], [7529, 3932], [7700, 3993], [7845, 4096], [8021,
4154], [8156, 4222], [8339, 4286], [8479, 4385], [8656, 4470], [8895, 4546], [9134,
4667], [9399, 4833], [9668, 5007], [9962, 5175], [10241, 5295], [10453, 5362],
[10871, 5595], [11197, 5820], [11599, 6078], [11986, 6334], [12463, 6611], [13006,
6906], [13611, 7200], [14220, 7598], [14903, 7974], [15650, 8316], [16501, 8745],
[17502, 9341], [18727, 9996], [20250, 10914], [22062, 12019], [24677, 13594], [28862,
15927], [36617, 20645], [65563, 37980]]
    tc=open(text_liste_octet_4,'r')
    c=tc.read().splitlines()
    tc.close()
```

```
    th=open(text_liste_heures_comparaison,'r')
    h=th.read().splitlines()
    th.close()
    d,r,s=0,0,1
    for ligne in c:
        if d%60==0:
            lh1=h[d].split(' ')
            lh2=h[d+60].split(' ')
            l=ligne.split(' ')

a1,a2,a3,a4,a5=int(l[0]),int(l[59]),int(l[119]),float(lh1[59]),float(lh2[59])
            if a1<=a2:
                k=[]
                for i in range(60):
                    k.append(int(l[i]))
                m=max(k)
                pos=k.index(m)
                b1=Ch[pos][1]>0.5*Ch[pos][0]
                if b1:
                    r+=(a5-a4)/a4
                    s*=a5/a4
                else:
                    r+=(a4-a5)/a4
                    s*=1+(a4-a5)/a4
            else:
                k=[]
                for i in range(60):
                    k.append(int(l[i]))
                m=min(k)
                pos=k.index(m)
                b1=Cb[pos][1]>0.5*Cb[pos][0]
                if b1:
                    r+=(a5-a4)/a4
                    s*=a5/a4
                else:
                    r+=(a4-a5)/a4
                    s*=1+(a4-a5)/a4
            print(d,r,s)
            ph=open(text_performance_extremum_rentabilite,'r')
            PH=ph.read()
            ph.close()
            ph=open(text_performance_extremum_rentabilite,'w')
            ph.write(PH+str(d)+' '+str(r)+' '+str(s)+' '+'\n')
            ph.close()
        d+=1

def graphe_rentabilite():
    tr=open(text_performance_extremum_rentabilite,'r')
    r=tr.read().splitlines()
    tr.close()
    tb=open(text_valeurs_comparaison,'r')
    b=tb.read().splitlines()
    tb.close()
    ini=float(b[0])
    X,Y,Z,W=[],[],[],[]
    for ligne in r:
        l=ligne.split(' ')
        X.append(int(l[0]))
        Y.append(float(l[2]))
        Z.append(1)
    for i in range(len(b)//60-1):
        W.append(float(b[60*i])/ini)
    figure()
    plot(X,W,color='red',linewidth=0.7,label='Cours du Bitcoin')
    plot(X,Y,color='blue',label='Valeur du capital investi',linewidth=0.7)
    plot(X,Z,color='black',linestyle='dotted', linewidth=2)
    ylabel("Valeur de l'actif")
```

```
1149|     legend()
1150|     show()
1151|
1152| def graphe_bitcoin():
1153|     tb=open(text_valeurs_comparaison,'r')
1154|     b=tb.read().splitlines()
1155|     tb.close()
1156|     X,Y=[],[]
1157|     for i in range(len(b)):
1158|         X.append(i)
1159|         Y.append(float(b[i]))
1160|     figure()
1161|     plot(X,Y)
1162|     show()
1163|
1164| def graphe_rentabilite_2():
1165|     tr=open(text_performance_extremum_rentabilite,'r')
1166|     r=tr.read().splitlines()
1167|     tr.close()
1168|     X,Y=[],[]
1169|     for ligne in r:
1170|         l=ligne.split(' ')
1171|         y=int(l[0])
1172|         if y==0:
1173|             X.append(0)
1174|             Y.append(0)
1175|         else:
1176|             X.append(int(l[0]))
1177|             Y.append(exp(525600*log(float(l[2]))/y)-1)
1178|     figure()
1179|     plot(X[400:],Y[400:],color='black')
1180|     show()
1181|
1182| def calcul_rentabilite():
1183|     tr=open(text_performance_extremum_rentabilite,'r')
1184|     r=tr.read().splitlines()
1185|     tr.close()
1186|     X,Y,s=[],[],0
1187|     for ligne in r:
1188|         l=ligne.split(' ')
1189|         y=int(l[0])
1190|         s+=y
1191|         if y==0:
1192|             X.append(0)
1193|             Y.append(0)
1194|         else:
1195|             X.append(int(l[0]))
1196|             Y.append(y*(exp(525600*log(float(l[2]))/y)-1))
1197|     return moyenne(Y)/190740
1198|
1199| def graphe_liste_heure(n):
1200|     tb=open(text_liste_heures_btc_2,'r')
1201|     b=tb.read().splitlines()
1202|     tb.close()
1203|     X=list(range(60))
1204|     Y=[]
1205|     l=b[n].split(' ')
1206|     for i in range(60):
1207|         Y.append(float(l[i]))
1208|     figure()
1209|     plot(X,Y,label='Cours du Bitcoin (en $)')
1210|     plot(X,Y,label='Liste
échantillonnée',marker='o',color='red',linestyle='none')
1211|     legend()
1212|     show()
1213|
1214| def performance_extremum():
```

```
1215|     tp=open(text_performance_extremum,'r')
1216|     p=tp.read().splitlines()
1217|     tp.close
1218|     for ligne in p:
1219|         l=ligne.split(' ')
1220|         a,b=int(l[0]),int(l[1])
1221|         print(a,b/a)
1222|
1223| def graphe_extremum(n):
1224|     tc=open(text_liste_octet_4,'r')
1225|     c=tc.read().splitlines()
1226|     tc.close()
1227|     l=c[n].split(' ')
1228|     A=[]
1229|     for i in range(120):
1230|         A.append(int(l[i]))
1231|     graphe_cours_2(A,[])
```