```python
# -*- coding: utf-8 -*-
"""
Created on Tue Mar  1 15:18:49 2016

@author: harora1
"""

import math

print()

print ("-"*11)

# variation 1
for i in range (5):
    print ("hello")

# variation 2
for i in range (5):
    print (i)

# variation 3
for i in range (5):
    print (i)
    print ("hello")

print ("-"*11)

strg = "Sundevils"

for i in range(len(strg)):
    print (i, strg[i])

print ("-"*11)

mylist = [10, 20, 30]

# method 1
for i in range (len(mylist)):
    print (i, mylist[i])

# method 2
for l in mylist:
    print (l)

print ("-"*11)

mylst1 = [10, 20, 30]
mylst2 = [100, 200, 300]
```

```python
# method 1
for i in range (len(mylst1)):
    print (i, mylst1[i], mylst2[i])

# method 2
for l1, l2 in zip (mylst1, mylst2):
    print (l1, l2)

print ("-"*11)

mydictionary = {'A':100, 'B':200, 'C':300}

# variation 1
for k in sorted(mydictionary.keys()):
    print (k, mydictionary[k])

# variation 2
for k in reversed(sorted(mydictionary.keys())):
    print (k, mydictionary[k])

print ("-"*11)

sumi = 0

for i in range (5):
    sumi = sumi + i
    print (i, sumi)

print (sumi)

print ("-"*11)

P = [10, 20, 30]

sumP = 0

for i in range (3):
    sumP = sumP + P[i]
    print (i, P[i], sumP)

print (sumP)

print ("-"*11)

P = [1, 2, 3]
Q = [10, 20, 30]

sumP = 0
sumQ = 0
```

```
for i in range (3):
    sumP = sumP + P[i]
    sumQ = sumQ + Q[i]

print ("P =", P)
print ("Q =", Q)
print ("sumP =", sumP)
print ("sumQ =", sumQ)

print ("-"*11)

P = [1, 2, 3]
Q = [10, 20, 30]

manhattan = 0

for i in range (3):
    manhattan = (manhattan +
                 math.fabs(P[i] - Q[i]))

print ("P =", P)
print ("Q =", Q)
print ("Manhattan Distance =", round(manhattan,2))

print ("-"*11)

P = [1, 2, 3]
Q = [10, 20, 30]

manhattan = 0
euclidean = 0
minkowski = 0

for i in range (len(P)):
    manhattan = (manhattan +
                 math.fabs(P[i] - Q[i]))
    euclidean = (euclidean +
                 pow(math.fabs(P[i] - Q[i]), 2))
    minkowski = (minkowski +
                 pow(math.fabs(P[i] - Q[i]), 3))

manhattan = pow (manhattan, 1/1)
euclidean = pow (euclidean, 1/2)
minkowski = pow (minkowski, 1/3)

print ("P =", P)
print ("Q =", Q)
print ("Manhattan Distance =", round(manhattan,2))
print ("Euclidean Distance =", round(euclidean,2))
print ("Minkowski Distance (r=3) =", round(minkowski,2))
print ("-"*11)

P = [1, 2, 3, 4, 5]
Q = [10, 20, 30, 40, 50]
n = len(P)

# initialize various component sums
sumpq = 0
sump = 0
sumq = 0
sump2 = 0
sumq2 = 0

# calcualte pearson correlation using the
# computationally efficient form
for p, q in zip(P, Q):
    sumpq += p * q
    sump += p
    sumq += q
    sump2 += pow(p, 2)
    sumq2 += pow(q, 2)

# pearson correlation coefficient
nr = (sumpq - (sump * sumq) / n)
dr = (math.sqrt(sump2 - pow(sump, 2) / n) *
        math.sqrt(sumq2 - pow(sumq, 2) / n))
r = nr/dr

print ("P =", P)
print ("Q =", Q)
print ("Pearson Correlation =", round(r,2))

print ("-"*11)

UserXRatingsD = {'A':1, 'B':2, 'C':3, 'D':4, 'E':5}

for k in sorted(UserXRatingsD.keys()):
    print(k, UserXRatingsD[k])

print ()

print ("-"*11)

UserXRatingsD = {'A':1, 'B':2, 'C':3, 'D':4, 'E':5}
UserYRatingsD = {'A':10, 'B':20, 'C':30, 'D':40, 'E':50}

for k in sorted(UserXRatingsD.keys()):
    print(k, UserXRatingsD[k], UserYRatingsD[k])
```

```python
print ("-"*11)

UserXRatingsD = {'A':1, 'B':2, 'C':3, 'D':4, 'E':5}
UserYRatingsD = {'A':10, 'B':20, 'C':30, 'D':40, 'E':50}

md = 0
for k in UserXRatingsD.keys():
    md = md + math.fabs(UserXRatingsD[k] - UserYRatingsD[k])

print ("UserXRatingsD =", UserXRatingsD)
print ("UserYRatingsD =", UserYRatingsD)
print ("Manhattan Distance =", round(md,2))

print ("-"*11)

UserRatingsND = {'X':{'A':10, 'B':20, 'C':30, 'D':40, 'E':50},
                 'Y':{'A':100, 'B':200, 'C':300, 'D':400, 'E':500}}

UserXRatingsD = UserRatingsND['X']
UserYRatingsD = UserRatingsND['Y']

for k in sorted(UserXRatingsD.keys()):
    print(k, UserXRatingsD[k], UserYRatingsD[k])

print ("-"*11)
```