3. Demonstrate the following similarity and dis similarity measures using python.

A) Pearson’s correlation B) Cosine similarity C) Jaccard similarity D) Euclidean Distance E) Manhattan Distance

In [5]:
```
## 3.1 Pearson’s Correlation
from numpy.random import randn
from numpy.random import seed
from scipy.stats import pearsonr

# seed random number generator
seed(1)

# prepare data
data1 =20*randn(1000)+100
data2 =data1+(10*randn(1000)+50)

# calculate Pearson 's correlation
corr,_=pearsonr(data1,data2)
print ('Pearsons correlation:%.3f' % corr)
```

```
Pearsons correlation:0.888
```

In [ ]:
```
#### We can see that the two variables are positively correlated and that the correlation is 0.8.
#### This suggests a high level of correlation, e.g. a value above 0.5 and close to 1.0.
```

In [ ]:

```
#### 3.2 Cosine similarity
The cosine similarity metric finds the normalized dot product of the two attributes. By
determining the cosine similarity, we would effectively try to find the cosine of the angle
between the two objects. The cosine of 0° is 1, and it is less than 1 for any other angle.

Cosine similarity is particularly used in positive space, where the outcome is neatly
bounded in [0,1].
One of the reasons for the popularity of cosine similarity is that it
is very efficient to evaluate, especially for sparse vectors.
```

In [8]:

```
from math import *
def square_rooted ( x ):
    return round ( sqrt (sum ([ a * a for a in x ]) ) ,3)
def cosine_similarity (x , y ):
    numerator = sum ( a * b for a , b in zip (x , y ) )
    denominator = square_rooted ( x ) * square_rooted ( y )
    return round ( numerator / float ( denominator ) ,3)
print (cosine_similarity([3,45,7,2] , [2,54,13,15]))
```

```
0.972
```

In [ ]:

```
#### 3.3 Jaccard similarity

So far discussed some metrics to find the similarity between objects. where the objects
are points or vectors. When we consider Jaccard similarity these objects will be sets.

Suppose you want to find Jaccard similarity between two sets A and B it is the ration of
the cardinality of A ∪ B and A ∩ B

JaccardSimilarity(A, B) = ||Intersection(A, B)||/||Union(A, B)||
```

In [11]:
```
from math import *
def jaccard_similarity (x , y ) :
    intersection_cardinality = len (set . intersection(*[ set ( x ) , set ( y ) ]) )
    union_cardinality = len (set . union (*[ set ( x ) , set (y) ]))
    return intersection_cardinality / float (union_cardinality)
print (jaccard_similarity ([0 ,1 ,2 ,5 ,6] ,[0 ,2 ,3 ,5 ,7 ,9]))
```

```
0.375
```

In [ ]:
```
### 3.4 Euclidean Distance

Euclidean distance is the most common use of distance measure.
The Euclidean distance between two points is the length of the path connecting them.
The Pythagorean theorem gives this distance between two points.
```

In [13]:
```
from math import *
def euclidean_distance (x , y ) :
    return sqrt ( sum (pow (a -b ,2) for a , b in zip(x , y )) )
print (euclidean_distance ([0 ,3 ,4 ,5] ,[7 ,6 ,3 , -1]))
```

```
9.746794344808963
```

In [ ]:
```
### 3.5 Manhattan Distance
Manhattan distance is a metric in which the distance between two points is calculated
as the sum of the absolute differences of their Cartesian coordinates.
In a simple way of saying it is the total sum of the difference between
the x-coordinates and y-coordinates.

In a plane with p1 at (x1, y1) and p2 at (x2, y2).
Manhattan distance = |x1 – x2| + |y1 – y2|

This Manhattan distance metric is also known as Manhattan length, rectilinear distance,
L1 distance or L1 norm, city block distance, Minkowski’s L1 distance, taxi-cab metric, or
city block distance.###
```

In [14]:

```
from math import *
def manhattan_distance (x , y ) :
    return sum (abs (a - b ) for a , b in zip (x , y ) )
print (manhattan_distance ([10 ,20 ,10] ,[10 ,20 ,20]))
```

```
10
```

In [ ]: