```python
# Assessed exercises 4
# As before, each question has an associated function, with input arguements
# matching those specified in the question. Your functions will be test for a
# range of different input values, against a model solution, to see if they
# produce the same answers.
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import numpy.random as npr
import math

# What things have we learnt this week? Series and DataFrames: using indices,
# indexing and slicing; boolean indexing, simple functions on series and data frames
# Pass-by-reference, Numpy and DataFrames, The in operator

# You may find it useful to test your functions on the Diamonds dataset from Week 1.
# Locate it on your computer and copy it into your current working directory
diamonds = pd.read_csv('./Diamonds.csv')
# You don't need to include the output of your tests in your PDF.

# Q1 Write a function that takes a DataFrame 'df' and returns a subset of this
# DataFrame. The function inputs should be the DataFrame 'df', and two numerical
# arrays 'rowinds' and 'colinds', which specify the rows and columns you wish to
# be includes in your new DataFrame.

def exercise1(df,rowinds,colinds):
    df_temp = df.iloc[rowinds,colinds]
    return df_temp

# Suggested test
exercise1(diamonds,np.arange(12),np.array([1,4,5,8]))
# This should return a DataFrame with 12 rows and 5 columns, where the rows are
# the 1st to 12th row of diamonds and the columns are cut, depth, table and y.

# Q2 This question is similar to Q1, but instead of using numerical indices
# we're going to specify a boolean condition for selecting the data for our
# subset. Your inputs should include a DataFrame 'df', a column of that DataFrame
# 'col', the label of another column 'label' and two values 'val1' and 'val2'.
# The function should output the entries of the column labelled 'label' for
# which the entries of the column 'col' are greater than the number 'val1' and
# less than 'val2'.
def exercise2(df,col,output_label,val1,val2):
    flag = (col > val1) & (col <val2)    #set a flag for boolean operators to figure o
    df_temp = df.loc[flag,[output_label]]
    return df_temp

# Suggested test
test_df = exercise1(diamonds,np.arange(500),np.arange(10))
exercise2(test_df,test_df.carat,'price',1.1,1.4)
# This should return a Series with the price of diamond number 172 and 376.
# Note here that 'col' is in the form test_df.carat, whereas 'label' is the
# column name in quotation marks, this is because one refers to data and the
# other a label.

# Q3 We define a distance measure for the distance between observations i and j
# as dist = ((carat_i - carat_j)/0.8)^2 + ((table_i - table_j)/57)^2. Write a
# function that takes a DataFrame 'df' as its input and computes the distance
# between each of the observations in 'df'. The output should be a nxn matrix,
# where n is the number of rows in 'df'. The entry in the ith row and jth column
# of this matrix should be the distance between the ith and jth measurements
# (i.e. ith and jth row of 'df'). You can assume that 'df' has columns 'carat'
# and 'table' and df.carat and df.table will work inside your function.
def exercise3(df):
    n = df.shape[0]      #get the number of rows in df
    m = np.eye(n)        #create a n*n matrix for initialization
    for i in range(n):
        for j in range(n):
            m[i][j] = math.pow(((df['carat'][i] - df['carat'][j])/0.8),2) + math.pow(
    return m
# Suggested test
test_df_2 = exercise1(diamonds,np.arange(0,10),np.arange(10))
dist = exercise3(test_df_2)
dist.max()
np.where(dist == dist.max())
# dist should be a 10x10 matrix, dist.max() (largeest entry) should be 0.03218
# and np.where(dist == dist.max()) (the location of the max) should give [2,7].

# Q4 The dissimilarity score is the sum of all the distances for a particular
# measurement, i.e. the sum of each row of the distance matrix. Write a function
# which takes a DataFrame 'df' as an input and computes the dissimilarity score
# for each measurement and add this as an extra column called 'Dissimilarity' to
# the DataFrame 'df'. This extended DataFrame should be returned by the function.
# Note: You can call your function from Q3 inside the exercise4 function.
def exercise4(df):
    temp_list = list()
    n = df.shape[0]
    distance = exercise3(test_df_2)
    for i in range(n):
        temp_list.append(sum(distance[i]))   #use sum() function to calculate the sum
    df['Dissimilarity'] = temp_list
    return df

# Suggested test
exercise4(test_df_2)
# this should return the DataFrame test_df_2, with and additional column for the
# dissimilarity of each diamond. The values in this column should be between
# 0.05 and 0.17
```