```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.
import pandas as pd
import numpy as np

# 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):
    return df.iloc[rowinds, colinds]

```

```
27
28 # Suggested test
29 exercise1(diamonds, np.arange(12), np.array([1, 4, 5, 8]))
30
31
32 # This should return a DataFrame with 12 rows and 5 columns, where the rows are
33 # the 1st to 12th row of diamonds and the columns are cut, depth, table and y.
34
35 # Q2 This question is similar to Q1, but instead of using numerical indices
36 # we're going to specify a boolean condition for selecting the data for our
37 # subset. Your inputs should include a DataFrame 'df', a column of that DataFrame
38 # 'col', the label of another column 'label' and two values 'val1' and 'val2'.
39 # The function should output the entries of the column labelled 'label' for
40 # which the entries of the column 'col' are greater than the number 'val1' and
41 # less than 'val2'.
42 def exercise2(df, col, output_label, val1, val2):
43     return df.loc[(col[val1 < col.values] < val2).index, output_label]
44
45
46 # Suggested test
47 test_df = exercise1(diamonds, np.arange(500), np.arange(10))
48 exercise2(test_df, test_df.carat, 'price', 1.1, 1.4)
49
50
51 # This should return a Series with the price of diamond number 172 and 376.
52 # Note here that 'col' is in the form test_df.carat, whereas 'label' is the
53 # column name in quotation marks, this is because one refers to data and the
54 # other a label.
```

```
55
56 # Q3 We define a distance measure for the distance between observations i and j
57 # as dist = ((carat_i - carat_j)/0.8)^2 + ((table_i - table_j)/57)^2. Write a
58 # function that takes a DataFrame 'df' as its input and computes the distance
59 # between each of the observations in 'df'. The output should be a nxn matrix,
60 # where n is the number of rows in 'df'. The entry in the ith row and jth column
61 # of this matrix should be the distance between the ith and jth measurements
62 # (i.e. ith and jth row of 'df'). You can assume that 'df' has columns 'carat'
63 # and 'table' and df.carat and df.table will work inside your function.
64 def exercise3(df):
65     n = df.shape[0]
66     lists = []
67     for i in range(n):
68         list = []
69         for j in range(n):
70             list.append(((df.carat[i] - df.carat[j]) / 0.8) ** 2 + ((df.table[i
   ] - df.table[j]) / 57) ** 2)
71         lists.append(list)
72     lists = np.array(lists)
73     return lists
74
75
76 # Suggested test
77 test_df_2 = exercise1(diamonds, np.arange(0, 10), np.arange(10))
78 dist = exercise3(test_df_2)
79 dist.max()
80 np.where(dist == dist.max())
81
```

```
 82
 83 # dist should be a 10x10 matrix, dist.max() (largeest entry) should be 0.03218
 84 # and np.where(dist == dist.max()) (the location of the max) should give [2,7].
 85
 86 # Q4 The dissimilarity score is the sum of all the distances for a particular
 87 # measurement, i.e. the sum of each row of the distance matrix. Write a function
 88 # which takes a DataFrame 'df' as an input and computes the dissimilarity score
 89 # for each measurement and add this as an extra column called 'Dissimilarity' to
 90 # the DataFrame 'df'. This extended DataFrame should be returned by the function
    .
 91 # Note: You can call your function from Q3 inside the exercise4 function.
 92 def exercise4(df):
 93     new_df = pd.DataFrame(exercise3(df))
 94     new_df['Dissimilarity'] = new_df.sum(axis=1)
 95     return new_df
 96
 97
 98 # Suggested test
 99 exercise4(test_df_2)
100 # this should return the DataFrame test_df_2, with and additional column for the
101 # dissimilarity of each diamond. The values in this column should be between
102 # 0.05 and 0.17
103
```