```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 os

# 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(os.path.join(os.path.dirname(__file__), 'data/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
```

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

```
54 # This should return a Series with the price of diamond number 172 and 376.
55 # Note here that 'col' is in the form test_df.carat, whereas 'label' is the
56 # column name in quotation marks, this is because one refers to data and the
57 # other a label.
58
59 # Q3 We define a distance measure for the distance between observations i and j
60 # as dist = ((carat_i - carat_j)/0.8)^2 + ((table_i - table_j)/57)^2. Write a
61 # function that takes a DataFrame 'df' as its input and computes the distance
62 # between each of the observations in 'df'. The output should be a nxn matrix,
63 # where n is the number of rows in 'df'. The entry in the ith row and jth column
64 # of this matrix should be the distance between the ith and jth measurements
65 # (i.e. ith and jth row of 'df'). You can assume that 'df' has columns 'carat'
66 # and 'table' and df.carat and df.table will work inside your function.
67 def exercise3(df):
68     n = df.shape[0]
69     lists = []
70     for i in range(n):
71         list = []
72         for j in range(n):
73             list.append(((df.carat[i] - df.carat[j]) / 0.8) ** 2 + ((df.table[i
   ] - df.table[j]) / 57) ** 2)
74         lists.append(list)
75     lists = np.array(lists)
76     return lists
77
78
79 # Suggested test
80 test_df_2 = exercise1(diamonds, np.arange(0, 10), np.arange(10))
```

```
 81 dist = exercise3(test_df_2)
 82 dist.max()
 83 np.where(dist == dist.max())
 84
 85
 86 # dist should be a 10x10 matrix, dist.max() (largeest entry) should be 0.03218
 87 # and np.where(dist == dist.max()) (the location of the max) should give [2,7].
 88
 89 # Q4 The dissimilarity score is the sum of all the distances for a particular
 90 # measurement, i.e. the sum of each row of the distance matrix. Write a function
 91 # which takes a DataFrame 'df' as an input and computes the dissimilarity score
 92 # for each measurement and add this as an extra column called 'Dissimilarity' to
 93 # the DataFrame 'df'. This extended DataFrame should be returned by the function
    .
 94 # Note: You can call your function from Q3 inside the exercise4 function.
 95 def exercise4(df):
 96     new_df = pd.DataFrame(exercise3(df))
 97     new_df['Dissimilarity'] = new_df.sum(axis=1)
 98     return new_df
 99
100
101 # Suggested test
102 exercise4(test_df_2)
103 # this should return the DataFrame test_df_2, with and additional column for the
104 # dissimilarity of each diamond. The values in this column should be between
105 # 0.05 and 0.17
106
```