Technical University of Denmark

Course name	Introduction to programming and data processing
Course number	02633
Aids allowed	All Aids
Exam duration	2 hours
Weighting	All exercises have equal weight

Contents

Submission details

You must hand in your solution electronically:

1. You can upload your solutions individually on CodeJudge (`dtu.codejudge.net/prog-jan16/assignment`) under *Afleveringer/Exam*. When you hand in a solution on CodeJudge, the test example given in the assignment description will be run on your solution. If your solution passes this single test, it will appear as *Submitted*. This means that your solution passes on this single test example. You can upload to CodeJudge as many times as you like during the exam.

2. You must upload your solutions on CampusNet. Each assignment must be uploaded as one separate .py file, given the same name as the function in the assignment:

   (a) `weeklyAverage.py`

   (b) `carsAvailable.py`

   (c) `speedOfLight.py`

   (d) `similarityMatrix.py`

   (e) `coinReturn.py`

   The files must be handed in separately (*not* as a zip-file) and must have these exact filenames.

After the exam, your solutions will be automatically evaluated on CodeJudge on a range of different tests, to check that they work correctly in general. The assessment of you solution is based only on how many of the automated tests it passes.

- Make sure that your code follows the specifications exactly.
- Each solution shall not contain any additional code beyond the specified function.
- Remember, you can check if your solutions follow the specifications by uploading them to CodeJudge
- Note that all vectors and matrices used as input or output must be numpy arrays.

Assignment A Robust temperature average

To compute the weekly average temperature you could simply measure the temperature each day and then compute the average over the seven days of the week. However, that approach would be quite sensitive: If for example a single day in the week has an extreme temperature, that will have a very large influence on the weekly average. To compute a less sensitive (more robust) temperature average, you decide to exclude the two highest and two lowest temperatures, and compute the average of the remaining middle 3 temperatures.

Problem definition

Create a function named `weeklyAverage` that takes as input a vector containing 7 temperature measurements and returns the average when the two highest and the two lowest temperatures are excluded.

Solution template

```
def weeklyAverage(dayTemp):
  #insert your code
  return weekTemp
```

Input	
`dayTemp`	Temperature measurements (vector of length 7).

Output	
`weekTemp`	Average temperature (decimal number)

Example

If, for example, you have the following temperature measurements:

$$[17.3,\ 18.2,\ 31.2,\ 14.2,\ -12.5,\ 16.5,\ 14.2]$$

the two highest temperatures are 31.2 and 18.2 and the two lowest are -12.5 and 14.2. The remaining temperatures are 17.3, 16.5, and 14.2, and the average can be computed as:

$$\frac{17.3 + 16.5 + 14.2}{3} = 16.0$$

A

Assignment B Car rental

A car rental company has 6 different categories of cars in their fleet:

`Mini` `Economy` `Compact` `Standard` `Fullsize` `Luxury`

The number of cars that are available in each of the 6 categories is represented by a vector of length 6, where the order of the categories is as in the list above.

Problem definition

Create a function named `carsAvailable` that takes a vector containing the number of cars available and a string representing one of the categories and returns the number of cars available in that category. The function must compare only the first character of the given category string and must be insensitive to case. If the requested category for example is `s`, `Stnd`, or `st` the function must return the number of cars in the `Standard` category. If the requested category does not match the first character of any of the 6 categories, the function must return -1 (negative one).

Solution template

```
def carsAvailable(fleet, category):
  #insert your code
  return n
```

Input	
`fleet`	Number of cars available in each category (vector of length 6)
`category`	Car category (string).

Output	
`n`	Number of cars available in the chosen category, or -1 (integer).

Example

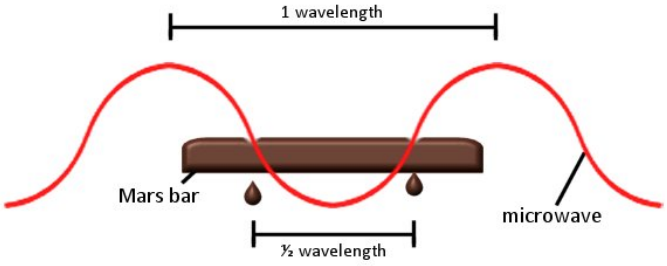
Consider the following vector of cars available: [5, 0, 17, 13, 11, 1]. If the requested category is given by the string `Lux` the first character is `L` which matches the `Luxury` category of which there is 1 car in the fleet. Thus the function must return the value 1.

Assignment C Speed of light

The speed of light can be measured by heating up a chocolate bar using a microwave oven. A microwave oven emits electromagnetic radiation at a given frequency, f. Because of standing waves inside the microwave oven, the chocolate will be unevenly heated, and one can measure the distance between the melted areas. This distance will be equal to one half wavelength and can thus be used to compute the speed of light according to the formula:

$$c = f \cdot \lambda,$$

where f is the frequency, λ is the wavelength, and c is the speed of light.



Source: canteengirl.org

To get a more precise estimate, one can average over several measurements of the wavelength.

Problem definition

Create a function named `speedOfLight` that takes as input the frequency and a vector of measured wavelengths. The function must compute the speed of light for each wavelength according to the formula above, and return the average of the computed values.

Solution template

```
def speedOfLight(f, wavelengths):
  #insert your code
  return c
```

Input	
`f`	Frequency (f) in Hz
`wavelengths`	Wavelengths (λ) in meters (vector)

Output	
`c`	Estimated speed of light.

Example

Say the frequency is $f = 2.45 \cdot 10^9$ [Hz] and there are 3 measured wavelengths: 0.122, 0.125, and 0.123 [m]. We can then compute

$$c_1 = 2.45 \cdot 10^9 \cdot 0.122 = 298.9 \cdot 10^6 \text{ [m/s]} \quad (1)$$

$$c_2 = 2.45 \cdot 10^9 \cdot 0.125 = 306.25 \cdot 10^6 \text{ [m/s]} \quad (2)$$

$$c_3 = 2.45 \cdot 10^9 \cdot 0.123 = 301.35 \cdot 10^6 \text{ [m/s]} \quad (3)$$

The average can the be computed as $c = \frac{c_1+c_2+c_3}{3} = 302.17 \cdot 10^6$ which is the final result.

C

Assignment D Similarity matrix

A similarity matrix contains the pairwise similarities between each element in two vectors. We have the vectors $x = [x_1, x_2, \ldots, x_N]$ and $y = [y_1, y_2, \ldots, y_M]$ and we define the following similarity measure:

$$\mathrm{s}(x_i, y_j) = \exp\left(-\delta(x_i - y_j)^2\right),$$

where δ is a parameter known as the length scale. The similarity matrix S is then defined as:

$$S = \begin{bmatrix} \mathrm{s}(x_1, y_1) & \mathrm{s}(x_1, y_2) & \cdots & \mathrm{s}(x_1, y_M) \\ \mathrm{s}(x_2, y_1) & \mathrm{s}(x_2, y_2) & \cdots & \mathrm{s}(x_2, y_M) \\ \vdots & \vdots & & \vdots \\ \mathrm{s}(x_N, y_1) & \mathrm{s}(x_N, y_2) & \cdots & \mathrm{s}(x_N, y_M) \end{bmatrix}.$$

The matrix S is thus an $N \times M$ matrix, where element (i, j) is the similarity between x_i and y_j.

Problem definition

Write a function named `similarityMatrix` that takes as input the two vectors x and y as well as the length scale δ and returns the similarity matrix S.

Solution template

```
def similarityMatrix(x, y, delta):
  #insert your code
  return S
```

Input	
`x`, `y`	Input data (vectors).
`delta`	Length scale (decimal number).

Output	
`S`	Similarity matrix.

Example

Consider the following input vectors:

$$x = [1.1,\ 1.2], \quad y = [1.3,\ 1.4,\ 1.5],$$

and say we have $\delta = 2$. The similarity matrix can then be computed as:

$$S = \begin{bmatrix} 0.9231 & 0.8353 & 0.7261 \\ 0.9802 & 0.9231 & 0.8353 \end{bmatrix}$$

where, for example, element (1,2) is computed as:

$$\mathrm{s}(x_1, y_2) = \exp\left(-2 \cdot (1.1 - 1.4)^2\right) = 0.8353$$

D

Assignment E Coin return

You are designing a machine for giving change (coins) to customers in a supermarket. Given that a certain amount of money is to be paid out, you need to decide which coins to give. You know the denominations (values) of the different kinds of coins available in the machine, and can assume that there are sufficiently many coins available of each kind. You must use the following algorithm to select the coins:

1. Let x denote the amount to be paid.
2. If x is less than half the value of the smallest coin: Stop.
3. Pay out the largest available coin with a value less than or equal to x.
4. Subtract the value of the paid out coin from x.
5. Repeat from 2 .

Problem definition

Create a function named `coinReturn` that takes as input vector containing the values of coins available in the machine (ordered starting with the smallest value) as well as the amount to be paid out. The function must return a vector with the values of the coins that are paid out (in the order defined by the algorithm).

Solution template

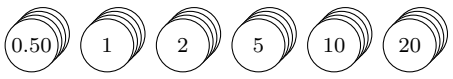
```
def coinReturn(coinsInMachine, amount):
  #insert your code
  return coinsToReturn
```

Input	
`coinsInMachine`	The values of the kinds of coins available in the machine (vector).
`amount`	Amount to be paid out (decimal number).

Output	
`coinsToReturn`	The values of the coins that are paid out (vector).

Example

Consider paying out the amount 34.60, when the following kinds of coins are available:



The input `coinsInMachine` will be the vector [0.50, 1, 2, 5, 10, 20]. The algorithm should proceed as follows:

	Iteration 1	Iteration 2	Iteration3	Iteration 4	Iteration 5	Iteration 6
Remaining amount (x)	34.60	14.60	4.60	2.60	0.60	0.10
Coin paid out	20	10	2	2	0.5	Stop

Thus, the coins paid out will be $[20, 10, 2, 2, 0.5]$ which should be the output of the function.

E