Technical University of Denmark

Course name	Introduction to programming and data processing
Course number	02633 (02631, 02632, 02634, 02692)
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 test your solutions individually on CodeJudge

   https://dtu.codejudge.net/prog-jun20/assignments

   under *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 all your solutions at DTU's Online exam site. Each assignment must be uploaded as one separate .py file, given the same name as the function in the assignment:

   (a) `stable_measurements.py`
   (b) `smooth_curve.py`
   (c) `slide_row.py`
   (d) `punctuation_check.py`
   (e) `stock_status.py`

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

After the exam, your solutions submitted to DTU Inside will be automatically evaluated on CodeJudge on a range of different tests, to check that they work correctly in general. The assessment of your 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, though `import` statements can be included.
- 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 1 Stable measurements

An experiment is set up for measuring the growth of microorganisms. It is known, that the measuring equipment gives unstable values in the initial part of the experiment. Therefor, the measurements need to be split into the unstable and stable interval. The unstable measurements need to be removed from the data, keeping only the interval containing the stable measurements. For this, it is decided that the stable intervals starts with the first measurement which is strictly larger than all previous measurement, and strictly smaller than all later measurements. The start of the stable interval is never the first, or the last, of the original values. Also, it is safe to assume that at least three measurements are taken.

Problem definition

Create a function `stable_measurements` which takes a vector with all measurements as input, and returns a vector with only stable measurements. If the stable interval is not found (for example, if all measurements are equal), an empty vector is to be returned.

Solution template

```
def stable_measurements(m):
  #insert your code
  return s
```

Input	
`m`	Original measurements, a vector containing three or more numbers.

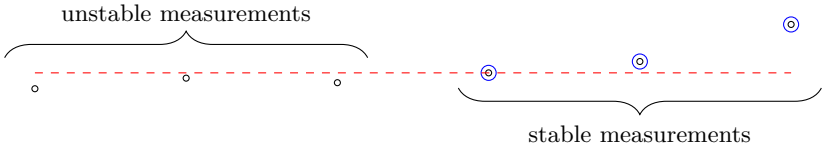
Output	
`s`	A vector with stable measurements.

Example

Consider the sequence

$$\mathbf{m} = [\,4.3\,,\quad 5.7\,,\quad 5.1\,,\quad 6.4\,,\quad 7.9\,,\quad 12.8\,]$$

also shown in the illustration.



Here, the measurement 5.7 is not starting the stable interval, since it is larger than a later measurement 5.1. The measurement 5.1 is not starting the stable interval, since is smaller than a previous measurement 5.7. The measurement 6.4 is strictly larger than all previous measurements (5.7 and 5.1) and strictly smaller than all later measurements (7.9 and 12.8), so 6.4 is starting the stable interval. Therefore

$$\mathbf{s} = \underline{[\,6.4\,,\quad 7.9\,,\quad 12.8\,]}\,.$$

1

Assignment 2 Smooth curve

A closed curve is represented using a sequence of 2D points $p_i = (x_i, y_i)$, $i = 1, \ldots, N$ connected by the line segments, where it is assumed that the last point is connected to the first point. The curve is smoothed by moving every curve point p_i slightly in the direction of a position midways between its two neighbors p_{i-1} and p_{i+1}, where one also needs to make sure that the first and the last point of the curve are correctly displaced. The new coordinates for the curve points where $i = 2, \ldots, N-1$ can be computed as

$$x_i^{\text{new}} = (1-\alpha)x_i + \alpha\frac{x_{i-1} + x_{i+1}}{2}, \qquad y_i^{\text{new}} = (1-\alpha)y_i + \alpha\frac{y_{i-1} + y_{i+1}}{2}$$

For the first and the last point we have

$$x_1^{\text{new}} = (1-\alpha)x_1 + \alpha\frac{x_N + x_2}{2}, \qquad y_1^{\text{new}} = (1-\alpha)y_1 + \alpha\frac{y_N + y_2}{2}.$$

and

$$x_N^{\text{new}} = (1-\alpha)x_N + \alpha\frac{x_{N-1} + x_1}{2}, \qquad y_N^{\text{new}} = (1-\alpha)y_N + \alpha\frac{y_{N-1} + y_1}{2}.$$

The parameter α controls the strength of the smoothing and is usually set between 0.1 and 0.5.

Problem definition

Create a function `smooth_curve` that takes as input: a matrix $\mathbf{C}$ of size $2 \times N$ containing coordinates of curve points (x_i in the first row, y_i in the second row), and a scalar α. The function should return a matrix of the same size as $\mathbf{C}$, containing coordinates of the smoothed points.

Solution template

```
def smooth_curve(C, alpha):
  # insert your code
  return S
```

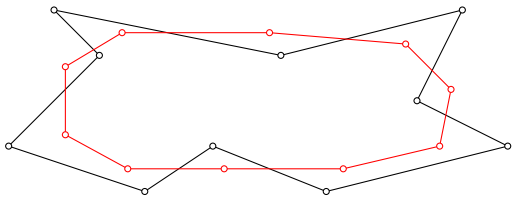
Input	
`C`	A matrix of size $2 \times N$ containing coordinates of curve points, $N \geq 3$
`alpha`	Scalar with the smoothing parameter α (float).

Output	
`S`	A matrix of size $2 \times N$ containing coordinates of the smoothed curve.

Example

Consider a curve, also shown in the illustration in black, given by

$$\mathbf{C} = \begin{bmatrix} 24 & 40 & 36 & 44 & 28 & 18 & 12 & 0 & 8 & 4 \\ 12 & 16 & 8 & 4 & 0 & 4 & 0 & 4 & 12 & 16 \end{bmatrix},$$



and a smoothing parameter

$$\alpha = 0.5\,.$$

The new coordinates of the first two points can be computed as

$$x_1^{\text{new}} = (1-0.5)24 + 0.5\frac{40+4}{2} = 23, \qquad y_1^{\text{new}} = (1-0.5)12 + 0.5\frac{16+16}{2} = 14\,,$$

$$x_2^{\text{new}} = (1-0.5)40 + 0.5\frac{24+36}{2} = 35, \qquad y_2^{\text{new}} = (1-0.5)16 + 0.5\frac{12+8}{2} = 13\,,$$

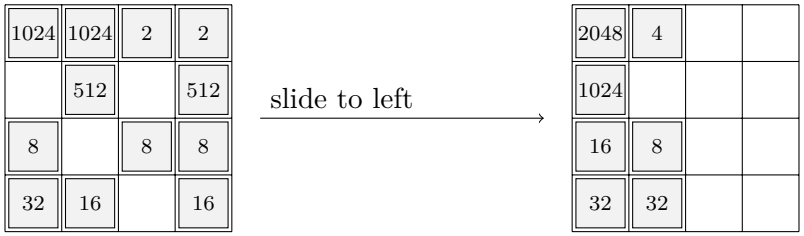
and similar computation is performed for the remaining coordinates. The smoothed curve, shown in the illustration in red, is

$$\mathbf{S} = \begin{bmatrix} 23 & 35 & 39 & 38 & 29.5 & 19 & 10.5 & 5 & 5 & 10 \\ 14 & 13 & 9 & 4 & 2 & 2 & 2 & 5 & 11 & 14 \end{bmatrix}.$$

2

Assignment 3 Slide row

2048 is a single-player game based on a 4×4 grid, where some grid elements contain numbered tiles that can be slided by the player. When slided, the tiles with same values merge and their numbers sum up, as shown below.



Here, we will consider only one horizontal row of the grid, and a slide of the tiles in the left direction. For this, the rules are:

- Tiles move as far left as possible until they are stopped by another tile or the edge of the grid.
- If two neighboring tiles have the same value, they merge into a tile with the total value of the two tiles that merged. The resulting tile cannot merge with another tile.
- If three consecutive tiles have the same value, only the two leftmost tiles merge. If all four spaces in the row are filled with tiles of the same value, the first two merge and the last two merge.

Problem definition

Write a function `slide_row` that takes a vector with 4 elements as input. Elements of the vector are whole numbers representing the tiles, and zero represents an empty grid space. The function should return a vector of 4 elements representing the tiles after a slide in the left direction. The values of the input vector may be any integer numbers, not only powers of 2 as in the original 2048 game.

Solution template

```
def slide_row(R):
  # insert your code
  return S
```

Input	
`R`	A vector with 4 elements representing a row of the grid before the slide.

Output	
`S`	A vector with 4 elements representing a row of tiles after the slide.

Example

Consider a row

$$\mathbf{R} = [8,\ 13,\ 0,\ 13]\,.$$

Only rightmost tile can move left, so row becomes $[8, 13, 13, 0]$. Now, two tiles with values 13 are neighboring and merge into a tile with a value 26. With no further move to left possible, the function should return

$$\mathbf{S} = \underline{[8,\ 26,\ 0,\ 0]}\,.$$

Assignment 4 Punctuation check

In typed text, some punctuation marks should always be followed by an empty space. These include:

- Punctuation used for ending an sentence: period (.), question mark (?) and exclamation point (!).
- Punctuation indicating a pause: comma (,), semicolon (;) og colon(:).

For an automatic text evaluation, we need to count the number of punctuation errors, where punctuation marks (.), (?), (!), (,), (;) and (:) are *not* followed by an empty space. If the last character in the text is a punctuation, it should not count as an error.

Problem definition

Write a function `punctuation_check` which takes as an input a string representing a typed text and returns a number of punctuation errors.

Solution template

```
def punctuation_check(text):
    #insert your code
    return n
```

Input	
`text`	String representing a text.

Output	
`n`	Skalar with the number of punctuation errors in the text (integer).

Example

Consider the text given as a string

```
text = 'This is:a text, a text with many,many errors!  But,who cares?'
```

which has following punctuation errors: `':a'`, `',m'` and `',w'`. So the function should return 3.

4

Assignment 5 Stock status

For every product they sell, a web shop has the following information: x is the number of items currently in stock; and d is the number of days until delivery of new items, where 0 indicates that new items are not expected. To inform the customers about their stock status they use the rule

Message	Stock status
`In stock`	5 or more items are in stock.
`Only <x> items in stock`	1, 2, 3 or 4 items are in stock, `<x>` being the number of items in stock, x.
`In stock again in <d> days`	No items are in stock, but new items are expected, `<d>` being the number of days until new delivery, d.
`Out of stock`	No items are in stock, and new items are not expected.
`Unknown stock status`	Either x or d are negative.

Problem definition

Write a function `stock_status` which takes as input a number x representing the number of items in stock, and a number d representing the number of days until next delivery. The function should return a string with the message corresponding to the stock status.

Solution template

```
def stock_status(x, d):
  # insert your code
  return s
```

Input

`x`	Number of items in stock (integer).
`d`	Number of days until next delivery, 0 for no expected delivery (integer).

Output

`s`	Message with a stock status (string).

Example

Consider $x = 3$ and $d = 0$. Since beween 1 and 4 items are in stock, the string

`Only 3 items in stock`

should be returned by the function.

5