Technical University of Denmark

Course name	Introduction to programming and data processing
Course number	02631, 02692, 02632
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-dec17/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) `normrange.py`

   (b) `readability.py`

   (c) `dataparser.py`

   (d) `temperaturemonitor.py`

   (e) `gamecalculator.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 Normalize to Range

A matrix or vector can be normalized to a given range using the following method. First step is to normalize all values between 0 and 1:

$$n_i = \frac{x_i - \min(x)}{\max(x) - \min(x)},$$

where x_i is the i^{th} element of the input data, organized as either a vector or matrix, and n_i is the corresponding i^{th} element of the normalized data. Second step is to scale the normalized data to a given range $[a, b]$:

$$s_i = n_i \cdot (b - a) + a,$$

where s_i is the i^{th} element of the now scaled data.

Problem definition

Create a function named `normrange` that takes as input a vector or a matrix `x`, and a vector `r` with two elements $[a, b]$ defining the range. The function should return the vector or matrix `s` normalized to the given range. If all elements of `x` are equal such that $\max(x) = \min(x)$ the function must return `x`.

Solution template

```
def normrange(x, r):
  #insert your code
  return s
```

Input	
`x`	Data that should be normalized to range (vector or matrix of decimal numbers).
`r`	Range (vector with 2 elements, decimal numbers).

Output	
`s`	The input `x` normalized to range `r` (vector or matrix of decimal numbers).

Example

Consider the following input vector $x = [2,\ 4,\ 6,\ 8,\ 10]$ scaled to $r = [-1,\ 1]$. First, we normalize x from 0 to 1:

$$n_i = \frac{x_i - 2}{10 - 2}$$

$$n = [0,\ 0.25,\ 0.5,\ 0.75,\ 1]$$

Then we scale the data in the given range (from -1 to 1):

$$s_i = n_i \cdot (1 - (-1)) + (-1)$$

$$s = [-1,\ -0.5,\ 0,\ 0.5,\ 1]$$

Example

Example with matrix input:

$$x = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix},\ r = [5,\ 20]$$

results in the output:

$$s = \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}.$$

A

Assignment B Text Readability

LIX is a scale that indicates the difficulty of reading a text. Using the words and punctuation in the text, a score can be calculated, where a higher number means more difficult to read. The readability measure LIX of a text is computed the following way:

$$\text{LIX} = \frac{\text{W}}{\text{P}} + \frac{\text{L} \cdot 100}{\text{W}}.$$

Where

W is the number of words.

P is the number of full stops.

L is the number of long words (more than 6 letters).

Problem definition

Create a function named `readability` that takes as input a string containing a text, and returns LIX. You can assume that the text string only contains letters (`a-z` and `A-Z`), spaces and full stops (`.`), i.e. there will not be other symbols present such as commas, numbers, question marks, hyphens or similar. You can assume that counting the number of full stops/dots/periods (`.`) in the text gives the value `P`. The function should return 0 if there are no full stops in the text.

Solution template

```
def readability(text):
  #insert your code
  return LIX
```

Input	
`text`	A text string (string containing only letters, full stops (`.`) and spaces).

Output	
`LIX`	The readability score LIX (decimal number)

Example

Consider the following input text string:

str = 'The Technical University of Denmark is located in Lyngby. Lyngby is a city in Denmark.'

The string contains **15 words**, **2 full stops** and **5 long words** (words with more than 6 letters: Technical, University, Denmark, located, Denmark). Thus,

$$\text{W} = 15, \quad \text{P} = 2, \quad \text{L} = 5$$

$$\text{LIX} = \frac{15}{2} + \frac{5 \cdot 100}{15} = 40.83$$

B

Assignment C Data Parser

You need to parse data from an external source. The data you receive is always a vector of an arbitrary length containing integers, which should be parsed and converted to either a matrix or a vector. Consider the input vector with the following format, where A,B,C,D,E,F,... are integers:

$$\text{data} = [\text{A, B, C, D, E, F, ...}]$$

The datavector from above should be parsed using the following rules:

if A=0: The data should be parsed as a vector,

- B is the length of the vector.
- C,D,E,F,... is the content of the vector (constrained by the length B).

if A=1: The data should be parsed as a matrix,

- B is the number of rows in the matrix,
- C is the number of columns in the matrix.
- D,E,F,... is the content of the matrix (constrained by the number of rows B and colums C)*.

*) Note that the matrix is always row-major, meaning that the first (top) row is filled from left to rigth, then the second row is filled from left to right, and so on.

Problem definition

Create a function named `dataparser` that takes as input a vector containing the data to be parsed. The function must return either a vector or a matrix with the parsed data. You can assume that the vector has a minimum length of 2 if A = 0, and minimum length of 3 if A = 1. If there is not enough data to construct the vector or matrix, the function should return an empty vector `[]`. If A is neither 0 or 1, the function should return an empty vector `[]`. If there is too much input data, only the required data should be parsed.

Solution template

```
def dataparser(data):
  #insert your code
  return parsed
```

Input	
`data`	Data to be parsed (vector of integers).

Output	
`parsed`	The parsed data (vector or matrix of integers).

Example

Consider the following input

$$\text{data} = [1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10].$$

The 1^{st} element is `1`, meaning the data should be parsed as a matrix. The 2^{nd} element is `2` meaning the matrix has 2 rows, and the 3^{rd} element is `3` meaning the matrix has 3 columns. We need to construct a matrix with $2 \cdot 3 = 6$ elements using the remaining data: `[4,5,6,7,8,9,10]` (7 elements). This means we have more than enough data to construct the matrix, and have to discard the last element [10]. The output of the function is then:

$$\text{parsed} = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Assignment D Temperature Monitor

You need to write a monitoring system that can warn the user if anything unwanted is happening with the temperature in an area. The system already samples the temperature at a fixed rate, and logs the result. Based on these temperatures, the system needs to return warning codes for the following conditions:

warning condition	code
No warning.	`0`
Temperature is 2 units higher or lower than the previous measurement.	`1`
Temperature is below 0 for the second time in a row.	`2`
Temperature is more than 5 for the second time in a row.	`3`

Problem definition

Create a function named `temperaturemonitor` that takes a sequence `T` of temperature measurements as input and returns the warnings codes `W` for each temperature measurement using the conditions listed above. The highest warning code should be returned if multiple conditions are met simultaneously. Note that warning codes 1-3 never can occur at the first measurement.

Solution template

```
def temperaturemonitor(T):
  #insert your code
  return W
```

Input	
`T`	Temperature sequence (vector of decimal numbers).

Output	
`W`	Warning sequence (vector of integers $[0-3]$).

Example

Consider the following sequence of temperatures

$$T = [1.8,\ 2.1,\ -0.2,\ -2.3,\ 1.1,\ 5.1,\ 5.5].$$

The first warning code is always `0`. There are less than 2 units from 1.8 to 2.1, the temperature is not lower than 0 or higher than 5, so the 2^{nd} warning code is `0`. There are more than 2 units from 2.1 to -0.2, so the 3^{rd} warning code is `1`. There are more than 2 units from -0.2 to -2.3 and the temperature has been below 0 two times in a row. Both warning code `1` and `2` apply, so we pick the highest of them for the 4^{th} warning code. The 5^{th} and the 6^{th} warning code are both `1`, because the temperature changes more than 2 units. The 7^{th} warning code is `3` because the temperature has been more than 5 two times in a row. Thus,

$$W = [0,\ 0,\ 1,\ 2,\ 1,\ 1,\ 3].$$

Assignment E Game Point Calculator

In a sport tournament, a team's rank or place can be determined by the number of points they have and their goal difference. For a team, the points are calculated using the following outcomes:

outcome	**points**
The team has more goals than their opponent.	`3`
The team and their opponent have the same number of goals.	`1`
The team scores less goals than their opponent.	`0`

The goal difference `GD` is calculated by the difference between the number of goals a team has scored `GF`, and the number of goals an opponent team has scored against them `GA`.

$$\text{GD} = \text{GF} - \text{GA}$$

Problem definition

Create a function named `gamecalculator` that takes as input a matrix `goals` of goals scored in a series of games between a team and their opponents in N matches. The first row in the matrix `goals` is the goals scored by the reference team, and second row in the matrix is goals scored by their opponents. The function should return a vector `result` with the total number of points the reference team has scored, and their goal difference `GD`.

Solution template

```
def gamecalculator(goals):
  #insert your code
  return result
```

Input

`goals`	Goals scored ($2 \times N$ matrix of integers).

Output

`result`	Points and goal difference for the reference team `[points, GD]` (vector of integers).

Example

Consider the input matrix

$$\text{goals} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}.$$

In the 1^{st} match, the reference team wins `1-0`, that gives 3 points. The 2^{nd} match is a draw, that gives 1 point for the reference team. The 3^{rd} match is also a draw, that gives 1 point for the reference team. In the 4^{th} match, the reference team looses `0-2`, that gives 0 points. The total score is

$$\text{points} = 3 + 1 + 1 + 0 = 5.$$

The goal difference of the reference team is

$$\text{GD} = (1 + 1 + 0 + 0) - (0 + 1 + 0 + 2) = -1.$$

The ouput of the function should be:

$$\text{result} = [5, -1]$$

E