Technical University of Denmark

Course name	Introduction to programming and data processing
Course number	02631, 02632, 02692
Aids allowed	All Aids
Exam duration	2 hours
Weighting	All exercises have equal weight

Contents

Assignment A: Sale price 2

Assignment B: ISBN-10 check digit 3

Assignment C: Partial correlation 4

Assignment D: Zero crossing 5

Assignment E: Ranking from pairwise comparison 6

Submission details

You must hand in your solution electronically:

1. You can upload your solutions individually on CodeJudge (`dtu.codejudge.net/prog-f17/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) `salePrice.py`

   (b) `ISBNCheckDigit.py`

   (c) `partialCorrelation.py`

   (d) `zeroCrossing.py`

   (e) `pairwiseRank.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 Sale price

A store is having a sale and makes an offer: "Buy two and get the cheapest one for free." You are given the task to implement the offer in the store's computer system. When a customer wishes to purchase some number of items, the method for computing the total sale price is the following: The customer pays for the most expensive item, gets the second-most expensive item for free, pays for the third-most expensive item, gets the fourth-most expensive item for free, and so on.

Problem definition

Create a function named `salePrice` that takes as input a vector with the prices of the items a customer has chosen, and computes the total sale price.

Solution template

```
def salePrice(prices):
  #insert your code
  return total
```

Input	
`prices`	Prices of the chosen items (vector of decimal numbers).
Output	
`total`	Total sale price (decimal number).

Example

Consider a customer who has chosen 5 items with the following prices:

$$9.95,\ 129.50,\ 9.95,\ 40.00,\ 17.75$$

If we sort the prices,

$$129.50,\ \cancel{40.00},\ 17.75,\ \cancel{9.95},\ 9.95$$

we see that the total sale price can be computed as:

$$129.50 + 17.75 + 9.95 = \underline{157.20}$$

Assignment B ISBN-10 check digit

The final character of a ten-digit International Standard Book Number (ISBN) is a check digit computed from the first 9 digits. It is computed by multiplying each digit by its position in the number (starting with position one, counting from the left) and taking the sum of these products modulo 11. If we denote the 10 digits as *abcdefghij* where j is the check digit, the check digit can be computed as:

$$j = (1 \cdot a + 2 \cdot b + 3 \cdot c + 4 \cdot d + 5 \cdot e + 6 \cdot f + 7 \cdot g + 8 \cdot h + 9 \cdot i) \bmod 11$$

Note that the check digit is a number between 0 and 10, where the value 10 is represented as the character `X`.

Problem definition

Create a function named `ISBNCheckDigit` that takes as input a vector with the first 9 digits in an ISBN number and computes the check-digit as a string containing either a numeric digit `0`–`9` or `X`.

Solution template

```
def ISBNCheckDigit(isbn):
  #insert your code
  return checkDigit
```

Input	
`isbn`	First 9 digits in an ISBN number (vector of whole numbers 0–9).
Output	
`checkDigit`	Check digit (text-string, `0`–`9` or `X`).

Example

Example 1: Consider the following first 9 digits of an ISBN number: [1, 4, 9, 1, 9, 3, 9, 3, 6]. The check digit can be computed as:

$$\begin{aligned} j &= (1 \cdot 1 + 2 \cdot 4 + 3 \cdot 9 + 4 \cdot 1 + 5 \cdot 9 + 6 \cdot 3 + 7 \cdot 9 + 8 \cdot 3 + 9 \cdot 6) \bmod 11 \\ &= (1 + 8 + 27 + 4 + 45 + 18 + 63 + 24 + 54) \bmod 11 \\ &= 244 \bmod 11 = 2 \end{aligned}$$

Thus, the string `2` must be returned.

Example 2: Consider the follwing first 9 digits of an ISBN number: [1, 5, 8, 4, 8, 8, 3, 8, 8]. The check digit can be computed as:

$$\begin{aligned} j &= (1 \cdot 1 + 2 \cdot 5 + 3 \cdot 8 + 4 \cdot 4 + 5 \cdot 8 + 6 \cdot 8 + 7 \cdot 3 + 8 \cdot 8 + 9 \cdot 8) \bmod 11 \\ &= (1 + 10 + 24 + 16 + 40 + 48 + 21 + 64 + 72) \bmod 11 \\ &= 296 \bmod 11 = 10 \end{aligned}$$

Thus, the string `X` must be returned.

Assignment C Partial correlation

In statistics, *partial correlation* is a measure of the degree of association between random variables with the effect of a set of other random variables removed. Given an $N \times K$ matrix X containing N observations of K random variables, the $K \times K$ matrix P of partial correlations between each pair of variables can be computed as follows:

$$C = \left(\frac{1}{N}X^\top X\right)^{-1}, \qquad p_{i,j} = \begin{cases} \dfrac{-c_{i,j}}{\sqrt{c_{i,i} \cdot c_{j,j}}} & i \neq j \\ 1 & i = j \end{cases}$$

where $c_{i,j}$ denotes the i, j'th element of the matrix C and $p_{i,j}$ denotes the i, j'th element of the matrix P.

Problem definition

Create a function named `partialCorrelation` that takes as input a matrix of observations X and computes the matrix of partial correlations P.

Solution template

```
def partialCorrelation(X):
  #insert your code
  return P
```

Input	
`X`	Observations ($N \times K$ matrix).
Output	
`P`	Partial correlations ($K \times K$ matrix).

Example

Consider the following input x, from which the matrix C can be computed:

$$X = \begin{bmatrix} -0.1 & 0.4 & -1.0 \\ 3.4 & 1.8 & -1.0 \\ -1.9 & -1.5 & 0.7 \\ 0.8 & 1.6 & 4.4 \\ 0.7 & 1.0 & 3.0 \end{bmatrix}, \qquad C \approx \begin{bmatrix} 3.422 & -4.742 & 0.934 \\ -4.742 & 7.219 & -1.437 \\ 0.934 & -1.437 & 0.448 \end{bmatrix},$$

The matrix of partial correlations P can then be computed as:

$$P \approx \begin{bmatrix} 1 & 0.954 & -0.754 \\ 0.954 & 1 & 0.799 \\ -0.754 & 0.799 & 1 \end{bmatrix},$$

which is the final result (here shown with three decimals' precision). Notice that the matrix is symmetric and that the diagonal contains ones.

C

Assignment D Zero crossing

In digital audio signal analysis, many different features can be used to characterize a signal. One such feature is *zero crossing*, i.e. the number of times the signal changes sign.

Problem definition

Create a function named `zeroCrossing` that takes as input a signal (as a vector) and returns the number of times the signal changes sign.

Solution template

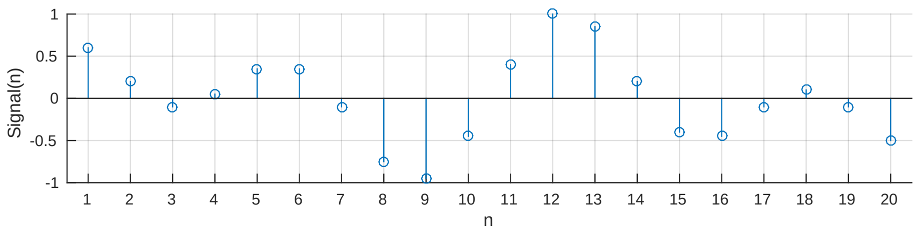
```
def zeroCrossing(signal):
  #insert your code
  return count
```

Input	
`signal`	Audio signal (vector of decimal numbers).
Output	
`count`	Number of zero crossings (whole number).

Example

Consider the following signal as input to the function (negative numbers shown in red color):

[0.6, 0.2, −0.1, 0.1, 0.3, 0.3, −0.1, −0.8, −0.9, −0.5, 0.4, 1.0, 0.8, 0.2, −0.4, −0.5, −0.1, 0.1, −0.1, −0.5]



Since the signal changes sign 7 times, the function must return the number 7.

Assignment E Ranking from pairwise comparison

A number of people are shown two different products (chosen from K products) and are asked which they prefer. This results in a data set which can be stored as a $K \times K$ matrix P of pairwise preferences, where each element $p_{i,j}$ denotes the number of times product i was preferred over product j.

Let $N_{i,j}$ denote the total number of comparisons between product i and j (collected in the matrix N), and let w_i denote the total number of comparisons "won" by product i (collected in the vector w). We can compute these from the matrix P as follows:

$$n_{i,j} = p_{i,j} + p_{j,i}, \qquad w_i = \sum_{j=1}^{K} p_{i,j}.$$

We now wish to rank the products according to the preferences. We will use the following algorithm:

```
Initialize $r_i = 1$ for $i \in 1, \dots, K$
Repeat 100 times
    Compute $\rho_i = \frac{w_i}{\sum_{j=1}^{K} \frac{n_{i,j}}{r_i + r_j}}$   for $i \in 1, \dots, K$
    Compute $r_i = \frac{\rho_i}{\sum_{j=1}^{K} \rho_j}$   for $i \in 1, \dots, K$
•
```

After running the algorithm, r_i will contain a "score" for each product

Problem definition

Create a function named `pairwiseRank` that takes as input a matrix P of pairwise comparisons and computes the ranking scores computed using the algorithm above.

Solution template

```python
def pairwiseRank(P):
  #insert your code
  return score
```

Input	
`P`	Matrix of pair-wise preferences.

Output	
`score`	Ranking score (vector with K elements).

Example

Consider the following matrix of pairwise preferences, from which the total number of comparisons and total number of "won" comparisons can be computed:

$$P = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 5 & 0 & 1 & 6 \\ 8 & 8 & 0 & 5 \\ 7 & 3 & 1 & 0 \end{bmatrix}, \qquad N = \begin{bmatrix} 0 & 6 & 8 & 8 \\ 6 & 0 & 9 & 9 \\ 8 & 9 & 0 & 6 \\ 8 & 9 & 6 & 0 \end{bmatrix}, \qquad w = [2,\ 12,\ 21,\ 11].$$

After running the algorithm we get the numbers $r \approx [0.0150,\ 0.1258,\ 0.7746,\ 0.0845]$ which is the final result.