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) `nBest.py`

   (b) `timeDilation.py`

   (c) `derangements.py`

   (d) `industry.py`

   (e) `taskDispatch.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 N-best scores

You are creating recommendation system for a webshop, which must recommend N products to the customer. You are given a list of $K > N$ matching scores (decimal numbers) which indicate how interested the customer is in each of K products.

Problem definition

Create a function named `nBest` that takes as input a vector of length K and an integer N, and returns a vector containing the N largest values from the original vector in the same order as they appear. You may assume that all the values are unique (no duplicates).

Solution template

```
def nBest(allValues, N):
  #insert your code
  return bestValues
```

Input	
`allValues`	Matching scores (vector of decimal numbers).
`N`	Number of products to recommend (positive integer).
Output	
`bestValues`	N best matching scores (vector of decimal numbers).

Example

Consider the following input matching scores:

$$[12.5,\ 13.6,\ -9.1,\ 17.5,\ 15.3,\ 10.5]$$

If the desired number of recommendations is $N = 3$, the result should be the vector:

$$[13.6,\ 17.5,\ 15.3]$$

A

Assignment B Time dilation

In the theory of relativity, on a fast moving spaceship time will go slower than on earth according to the following formula:

$$T(t, v) = \frac{t}{\sqrt{1 - \frac{v^2}{c^2}}},$$

where T is the time on earth, t is the time on the spaceship, v is the speed of the spaceship (relative to earth), and c is the speed of light. We will use $c = 299\,792\,458$ [meter/second].

Problem definition

Write a function named `timeDilation` that takes as input a vector of N_t values for the time on the spaceship (in seconds) as well as a vector of N_v values of the velocity (in meters/second) and computes a matrix of size $N_t \times N_v$ that contains the corresponding time (in seconds) on earth for each combination of the given times and velocities of the spaceship.

Solution template

```
def timeDilation(t, v):
  #insert your code
  return T
```

Input

`t`	Time on spaceship, t (vektor).
`v`	Velocity of spaceship, v (vektor).

Output

`T`	Time on earth, T (matrix).

Example

Assume that we are given the following times and velocities for the spaceship:

$$t = [5,\ 10.5,\ 16], \qquad v = [1.1 \cdot 10^8,\ 2.2 \cdot 10^8]$$

We thus have $N_t = 3$ and $N_v = 2$, and the resulting matrix should then be:

$$\overline{\overline{T}} = \begin{bmatrix} T(t_1, v_1) & \cdots & T(t_1, v_{N_v}) \\ \vdots & & \vdots \\ T(t_{N_t}, v_1) & \cdots & T(t_{N_t}, v_{N_v}) \end{bmatrix} = \begin{bmatrix} 5.375 & 7.360 \\ 11.287 & 15.457 \\ 17.200 & 23.553 \end{bmatrix}$$

where, for example, element (3,1) is computed as:

$$T(t_3, v_1) = \frac{16}{\sqrt{1 - \frac{(1.1 \cdot 10^8)^2}{299\,792\,458^2}}} \approx 17.200,$$

B

Assignment C Derangements

In combinatorics, a “derangement” is a permutation of a set such that no element appears in its original position. The number of possible derangements of a set of size n is given by:

$$k = \text{round}\left(\frac{n!}{e}\right),$$

where the function round$(\cdot)$ returns the nearest integer, $n!$ is the factorial of n, and e is the mathematical constant known as Euler’s number.

Problem definition

Create af function named `derangements` that takes as input the size of a set and computes as output the number of derangements according to the formula above.

Solution template

```
def derangements(n):
  #insert your code
  return k
```

Input	
`n`	Size of a set (positive integer).
Output	
`k`	Number of derangements (positive integer).

Example

Say we are given a set of size $n = 5$, the number of possible derangements can be computed as:

$$k = \text{round}\left(\frac{5!}{e}\right) = \text{round}\left(\frac{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{e}\right) = 44$$

C

Assignment D Industrial classification

A company can be classified as one of the following industries.

Code	Industry name
0000–0999	`Agriculture`
1000–1499	`Mining`
1500–1999	`Construction`
2000–3999	`Manufacturing`
4000–4999	`Transportation`
5000–5999	`Trade`
6000–6999	`Finance`
7000–8999	`Services`
9000–9999	`Public`

Problem definition

Create a function named `industry` that takes as input a text string representing an industry code and returns the corresponding industry name as a text string (written exactly as in the table above). The input consists of 4 characters which can be either digits (0–9) or the letter `x`. The `x` works as a "wild card" that can stand for any digit. You can assume that the given code always matches exactly one of the code ranges in the table.

Solution template

```
def industry(industryCode):
  #insert your code
  return industryName
```

Input
`industryCode`

Output
`industryName`

Example

Consider the input string `35xx`. Since the two `x` characters can stand for any digit, this would represent a number between 3500 and 3599. Since these codes fall within 2000–3999: Manufacturing, the string `Manufacturing` must be returned.

D

Assignment E Task dispatch

You are working on a production system in which a number of tasks are to be processed on a number of processing units. Each task requires a certain number of resources, and each processing unit has a certain number of resources available. You must create a task dispatching system which will assigne each task to a processing, taking the number of resources into account. To assign the tasks you must use the following algorithm:

```
Let $t(n)$ denote the number of resources required for task $n$.
Let $u(k)$ denote the number of resources available on processing unit $k$.
Let $a(n)$ denote the number of the processing unit to which task $n$ is assigned.
Loop over all processing units (let $k$ denote the processing unit).
    Loop over all tasks that have not yet been assigned (let $n$ denote the task).
        If there is enough room on processing unit $k$ for task $n$, i.e. if $u(k) \geq t(n)$.
            Assign task $n$ to processing unit $k$: $a(n) \leftarrow k$.
            Update the number of resources on processing unit $k$: $u(k) \leftarrow u(k) - t(n)$.
        •
    •
•
```

Problem definition

Create a function named `taskDispatch` that takes as input a vector containing the number of required resources for N tasks as well as a vector containing the number of available resources for K processing units. The function must return a vector of length N containing the number of the processing unit (number between 1 and K) to which each task was assigned according to the algorithm above. Tasks that have not been assigned to any processing unit (because of lack of resources) should be marked by zero.

Solution template

```
def taskDispatch(tasks, units):
  #insert your code
  return assignment
```

Input	
`tasks`	The number of resources required for each task (vector).
`units`	The total number of available resource for each processing unit (vector).
Output	
`assignment`	Assignment (processing unit number) for each task (vector).

Example

Consider the situation where we have 6 tasks and 2 processing units with the following number of resources required/available:

$$t = [5,\ 7,\ 4,\ 2,\ 3,\ 5], \quad u = [11,\ 13].$$

The first processing unit has $u(1) = 11$ resources. We assign to it the first task and update $u(1) = 11-5 = 6$. There is not room the second task, so we move on and assign the third task and update $u(1) = 6-4 = 2$. There is room for the fourth task also, so we assign that and update $u(1) = 2-2 = 0$. There is not room for any of the following tasks, so we move on to the second proceesing unit. The first task is already assigned, so we assign the second task and update $u(2) = 13-7 = 6$. The third and fourth task are already assigned, so we move on and assign the fifth task and update $u(2) = 6-3 = 3$. There is not room for the sixth task, so that is not assigned to any processing unit, and the algorithm is finished. The final assignment should thus be:

$$a = [1,\ 2,\ 1,\ 1,\ 2,\ 0].$$