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 (`http://dtu.codejudge.net/prog-e16/assignment`) under *Afleveringer/Exam*. When you hand in a solution on CodeJudge, the first 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) `cvrchecksum.py`

   (b) `nextleapyear.py`

   (c) `capitalize.py`

   (d) `submatrix.py`

   (e) `approximatepi.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 CVR Number Checksum

It is mandatory for Danish companies to register for a 8 digit identification number called *CVR number* (Det Centrale Virksomhedsregister). The last digit is a check-sum which is computed according to the so-called *modulus 11* rule with the weights 2-7-6-5-4-3-2. The checksum is computed in the following way:

- The seven first digits are multiplied elementwise with the weights 2-7-6-5-4-3-2.
- The sum of all elementwise-products is computed.
- The modulus (remainder after integer division) of this sum and 11 is determined.
- The checksum is given by the modulo subtracted from 11.

Problem definition

Create a function named `cvrchecksum` that takes as input the seven first digits of a CVR number and computes the correct checksum.

Solution template

```python
def cvrchecksum(cvr):
  #insert your code
  return checksum
```

Input	
`cvr`	7-digit integer for which the checksum is calculated.
Output	
`checksum`	An integer $(0 - 9)$ which is the checksum for `cvr`.

Example

We want to calculate the checksum, x, for CVR number $3006094x$. We compute the checksum in the following way:

First seven digits	3	0	0	6	0	9	4
Weight-vector	2	7	6	5	4	3	2
Element-wise products	6	0	0	30	0	27	8

Sum of element-wise products:

$$6 + 0 + 0 + 30 + 0 + 27 + 8 = 71$$

This number divides by 11 six times with remainder:

$$71 \bmod 11 = 5$$

The correct checksum is $x = 11 - 5 = 6$.
Other CVR numbers and their checksums:

$$1234567 : 4 \qquad 9876543 : 3 \qquad 1111111 : 4$$

Assignment B Next Leap Year

In the common Gregorian calendar a year has on average 365.2425 days. Most years have 365 days, with February having 28 days.

Occasionally, a *leap year* occurs, in which February has 29 days. Leap years are determined by the following rules:

- If the year is divisable by 4 (without remainder), the year is a leap year.
- However: if the year is divisable by 100 the year is not a leap year, unless it is divisable by 400.

Problem definition

Create a function named `nextleapyear` that takes as input a year, and returns the first following leap year.

Solution template

```
def nextleapyear(y):
  #insert your code
  return l
```

Input	
y	Positive integer number, which represents the year.
Output	
l	Positive integer, which represents the first following leap year. Note that always: `l` $>$ `y`.

Example

With year 1896 as the input, the next leap year would be 1904. While 1900 is divisible by 4, it is also divisible by 100, and not by 400.
Other examples:

y	96	396	1999	2000	2099
l	104	400	2000	2004	2104

Assignment C Capitalization

A common grammatical error in writing is the lack of capitalization (upper case first letters). Many instances of this mistanke can be automatically corrected by applying the following rules:

- The first word in the text must be capitalized.
- The first word after the punctuation marks period (.), exclamation mark (!) and question mark (?) must be capitalized.

Problem definition

Create a function named `capitalize` that takes as input a string and returns the string with the capitalization rules applied. Capital letters in the input string must be kept even if they voilate the capitalization rule and you can assume no more than one subsequent space. The letters that need to be handled are:

Letters	`abcdefghijklmnopqrstuvwxyz`
Capitals	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Solution template

```
def capitalize(strin):
  #insert your code
  return strout
```

Input

`strin`	Input string which might contain capitalization mistakes.

Output

`strout`	The same string with capitalization rules applied.

Example

`strin`	hello! how are you? please remember capitalization. EVERY time.
`strout`	Hello! How are you? Please remember capitalization. EVERY time.

C

Assignment D Submatrix

A *submatrix* is obtained from a matrix by deleting a number of rows and/or colums. The remaining entries are collected in their original relative positions to form the submatrix.

Problem definition

Create a function named `submatrix` that takes as input a matrix and a row and column index, and returns the submatrix formed by removing the row and column.

Solution template

```python
def submatrix(M, r, c):
  #insert your code
  return S
```

Input

`M`	Input matrix of size $m \times n$.
`r`	Row index. If $1 \le$ `r` $\le m$, the `r`th row shall be removed. Otherwise, no row is removed.
`c`	Column index. If $1 \le$ `c` $\le n$, the `c`th column shall be removed. Otherwise, no column is removed.

Output

`S`	Submatrix of `M` after removing row `r` and column `c`.

Example

In the example shown below, the third row and the second colum are removed from a 3×4 matrix to form a 2×3 submatrix:

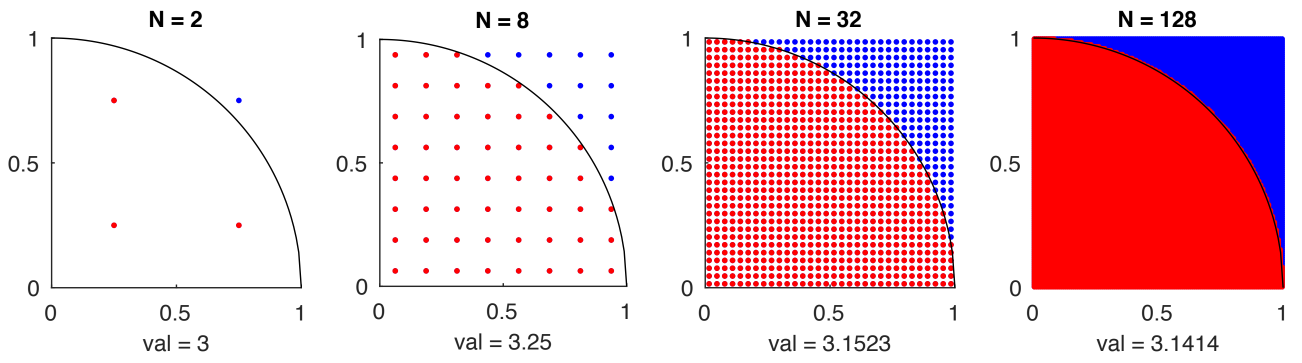
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \to \begin{bmatrix} 1 & 3 & 4 \\ 5 & 7 & 8 \end{bmatrix}$$

Other examples:

`M`	$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$
`r`	2	1	4	5
`c`	1	0	2	1
`S`	$\begin{bmatrix} 2 & 3 & 4 \\ 10 & 11 & 12 \end{bmatrix}$	$\begin{bmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 3 & 4 \\ 6 & 7 & 8 \\ 10 & 11 & 12 \end{bmatrix}$

Assignment E Approximating Pi

The value of π is approximately 3.14159. A geometrical method of approximating the value of π is by drawing points in a regular grid, equally spaced in a unit square, and counting the fraction of points which have a distance less than or equal to 1 from the origin. This fraction approaches $\pi/4$ as the number of points increases. The distance from origin is given by: $\sqrt{x^2+y^2}$.



Problem definition

Create a function named `approximatepi` that takes as input the positive integer number N, which is the number of points to use in the pi approximation along one dimension. The function shall return the approximation of π given this value of N computed as:

$$\frac{K}{N^2}\cdot 4$$

Where K is the number of points with distance less than one to the origin. The point coordinates are given by:

$$\begin{bmatrix} x_{i,j} \\ y_{i,j} \end{bmatrix} = \begin{bmatrix} \frac{1}{2N}+\frac{j}{N} \\ \frac{1}{2N}+\frac{i}{N} \end{bmatrix}, \quad i \in 0,1,2,...,N-1, \quad j \in 0,1,2,...,N-1$$

Solution template

```
def approximatepi(N):
  #insert your code
  return val
```

Input	
`N`	Integer number in the range $[1; 10000]$.

Output	
`val`	The approximated value of π.

Example

If we choose $N = 2$, a grid of $N^2 = 4$ points is formed. The point coordinates are:

$$(0.25; 0.25), (0.75; 0.25), (0.25; 0.75), (0.75; 0.75).$$

Three of the four points have a distance smaller than 1 to the origin. The approximation becomes $\frac{3}{4}\cdot 4 = 3$. Other approximations:

N	1	10	100	1000	10000
Approximate Value	4	3.16	3.1428	3.141676	3.14159388