Danmarks Tekniske Universitet

Kursusnavn	Introduktion til programmering og databehandling
Kursusnummer	02631, 02692, 02632
Hjælpemidler	Alle hjælpemidler er tilladt
Varighed	2 timer
Vægtning	Opgaverne vægtes ens

Indhold

Aflevering

Du skal aflevere dine løsninger elektronisk:

1. Du kan uploade dine løsninger individuelt på CodeJudge (http://dtu.codejudge.net/prog-e16/assignment) under *Afleveringer/Exam*. Når du afleverer en løsning på CodeJudge bliver det første test-eksempel som fremgår i opgavebeskrivelsen kørt på din løsning. Hvis din løsning består denne ene test, vil den fremgå som *Submitted*. Det betyder at din løsning består på dette ene test-eksempel. Du kan uploade til CodeJudge så mange gange som du ønsker under eksamen.

2. Du skal uploade alle dine løsninger på CampusNet. Hver assignment skal uploades som en separat .py fil med samme navn som funktionen i opgaven:

   (a) `cvrchecksum.py`
   (b) `nextleapyear.py`
   (c) `capitalize.py`
   (d) `submatrix.py`
   (e) `approximatepi.py`

   Filerne skal afleveres separat (*ikke* som en zip-fil) og skal have præcis disse filnavne.

Efter eksamen vil dine løsninger blive automatisk evalueret på CodeJudge på en række forskellige tests for at undersøge om de generelt fungerer korrekt. Bedømmelsen af din aflevering foretages udeklukkende på baggrund af hvor mange af de automatiserede test der bestås.

- Du skal sikre dig at din kode følger specifikationerne nøje.
- Hver løsning må ikke indeholde nogen yderligere kode udover den specificerede funktion.
- Husk at du kan tjekke om dine løsninger følger specifikationerne ved at uploade dem til CodeJudge.
- Bemærk at alle vektorer og matricer anvendt som input eller output skal være numpy arrays.

Assignment A CVR Number Checksum

Alle danske virksomheder skal registreres med et 8-cifret identifikationsnummer, det såkaldte *CVR nummer* (Det Centrale Virksomhedsregister). Det sidste tal er et kontrolciffer, som følger den såkaldte *modulus 11* regel med vægtene 2-7-6-5-4-3-2. Kontrolcifferet bliver udregnet på følgende måde:

- De første syv cifre bliver ganget elementvis med vægtene 2-7-6-5-4-3-2.
- Summen af alle elementvise produkter bliver udregnet.
- Modulus (resten efter heltals-division) af denne sum og 11 bliver fundet.
- Kontrolcifferet er lig med 11 minus denne modulus.

Problemdefinition

Skriv en funktion med navnet `cvrchecksum` der tager de første syv cifre af et CVR nummer som input og beregner det korrekte kontrolciffer.

Løsningsskabelon

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

Input

`cvr`	7-cifret heltal for hvilket kontrolcifferet skal udregnes.

Output

`checksum`	Et heltal $(0-9)$ som er kontrolcifferet til `cvr`.

Eksempel

Vi ønsker at udregne kontrolcifferet, x, for CVR nummeret $3006094x$. Vi regner kontrolcifferet ud på følgende måde:

Første syv cifre	3	0	0	6	0	9	4
Vægte	2	7	6	5	4	3	2
Elementvise produkter	6	0	0	30	0	27	8

Summen af elementvise produkter:

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

Tallet kan divideres med 11 seks gange med rest:

$$71 \bmod 11 = 5$$

Det korrekte kontrolciffer er $x = 11-5 = 6$.
Andre CVR numre og deres kontrolcifre:

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

A

Assignment B Next Leap Year

I den almindelige Gregorianske kalender er der i gennemsnit 365.2425 dage per år. De fleste år har 365 dage, hvor februar har 28 dage.

Nogle år er såkaldte *skudår*, hvor februar har 29 dage. Skudår forekommer efter følgende regler:

- Hvis årstallet er deleligt med 4 (uden rest), er året et skudår.
- Dog: hvis årstallet er deleligt med 100 så er året ikke et skudår, medmindre det er deleligt med 400.

Problemdefinition

Skriv en funktion med navnet `nextleapyear` der tager et årstal som input, og returnerer det første følgende skudår.

Løsningsskabelon

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

Input	
`y`	Positivt heltal som repræsenterer årstallet.

Output	
`l`	Positivt heltal som repræsenterer det første følgende skudår. Bemærk at der altid gælder: `l` > `y`.

Eksempel

Bruger vi 1896 som input, er det næste skudår 1904. 1900 er delelig med 4, men også med 100 og ikke med 400.
Andre eksempler:

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

Assignment C Capitalization

En udbredt grammatisk fejl i skriftsprog er manglende stort begyndelsesbogstav. Mange gange kan denne fejl rettes automatisk ved at anvende følgende regler:

- Det første ord i teksten skal have stort begyndelsesbogstav.
- Det første ord efter tegnene punktum (.), udråbstegn (!) og spørgsmålstegn (?) skal have stort begyndelsesbogstav.

Problemdefinition

Skriv en funktion med navnet `capitalize` der tager en streng som input og returnerer strengen med store begyndelsesbogstaver iflg. reglerne. Indeholder input strengen store bogstaver skal de beholdes, selv hvis de overtræder reglerne og du kan antage højest et mellemrum ad gangen. De bogstaver der skal kunne håndteres er:

Bogstaver	`abcdefghijklmnopqrstuvwxyz`
Store bogstaver	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Løsningsskabelon

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

Input

`strin`	Input streng som kan have fejl med begyndelsesbogstaver.

Output

`strout`	Den samme streng med rigtige begyndelsesbogstaver.

Eksempel

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

C

Assignment D Submatrix

En *submatrix* fremkommer fra en matrix ved at slette et antal rækker og/eller kolonner (også kaldt søjler). De tilbageblivende elementer samles i deres oprindelige relative positioner og danner sub-matricen.

Problemdefinition

Skriv en funktion med navnet `submatrix` der tager en matrix og et række- og kolonne-indeks som input og returnerer den submatrix som fremkommer ved at fjerne denne række og søjle.

Løsningsskabelon

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

Input

`M`	Input matrix af størrelse $m \times n$.
`r`	Rækkeindeks. Hvis $1 \leq$ `r` $\leq m$, skal den `r`-te række fjernes. Ellers skal ingen rækker fjernes.
`c`	Kolonneindeks. Hvis $1 \leq$ `c` $\leq n$, skal den `c`-te kolonne fjernes. Ellers skal ingen kolonner fjernes.

Output

`S`	Submatrix af `M` efter fjernelse af række `r` og søjle `c`.

Eksempel

I eksemplet nedenunder fjernes tredje række og anden søjle fra en 3×4 matricen, hvilket danner en 2×3 submatrix:

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

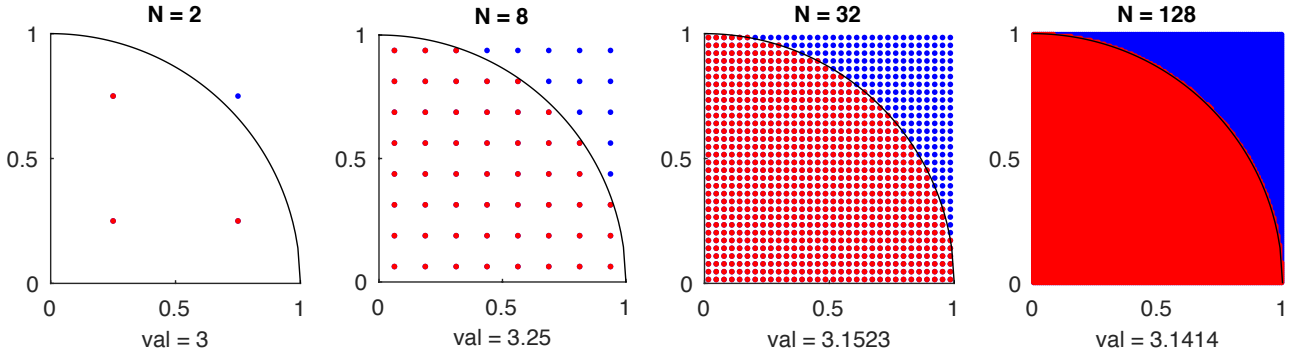
Andre eksempler:

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

D

Assignment E Approximating Pi

En approksimation af π er 3.14159. En geometrisk metode til approksimation af π er at tegne punkter med jævne afstande i et enheds-kvadrat, og tælle hvor stor en del af punkterne har afstand mindre end eller lig 1 til origo. Denne brøkdel nærmer sig $\pi/4$ når antallet af punkter stiger. Afstanden til origo er givet ved: $\sqrt{x^2 + y^2}$.



Problemdefinition

Skriv en funktion med navnet `approximatepi` der tager et positivt heltal N som input, som er antallet af punkter til approksimering af pi langs én dimension. Det samlede antal punkter er N^2. Funktionen skal returnere approksimationen af π given denne værdi af N, beregnet som:

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

hvor K er antallet af punkter med afstand mindre end en til origo. Punkternes koordinater er givet ved:

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

Løsningsskabelon

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

Input	
`N`	Heltal i intervallet $[1; 10000]$.

Output	
`val`	Den approksimerede værdi af π.

Eksempel

Hvis vi vælger $N = 2$, dannes et gitter af $N^2 = 4$ punkter. Punkternes koordinater er:

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

Tre af de fire punkter har afstand mindre end 1 til origo. Approksimationen bliver $\frac{3}{4} \cdot 4 = 3$.
Andre approksimationer:

N	1	10	100	1000	10000
Approksimeret værdi	4	3.16	3.1428	3.141676	3.14159388