Danmarks Tekniske Universitet

Kursusnavn	Introduktion til programmering og databehandling Programmering og databehandling (andet programmeringssprog)
Kursusnummer	02631, 02692
Hjælpemidler	Alle hjælpemidler er tilladt
Varighed	2 timer
Vægtning	Opgaverne vægtes ens

Indhold

Assignment A: Conversion of grades 2

Assignment B: When to hire an employee 3

Assignment C: Average hearing loss 4

Assignment D: Building LEGO bricks 5

Assignment E: Sudoku check 6

Aflevering

Du skal aflevere dine løsninger elektronisk:

1. Du kan uploade dine løsninger individuelt på CodeJudge (dtu.codejudge.net/prog-e15/assignment) under *Afleveringer/Exam*. Når du afleverer en løsning på CodeJudge bliver det 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) `convertGrade.py`
   (b) `hireApplicant.py`
   (c) `averagedB.py`
   (d) `buildLego.py`
   (e) `sudokuCheck.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 automatisered 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 Conversion of grades

En fordel ved den nye danske 7-trins karakterskala, sammenlignet med den gamle 13-skala, er at den nemt kan oversættes til European Credit Transfer Systemet (ECTS). Den følgende tabel viser konvertering imellem de tre forskellige karakterskalaer.

13-scale	7-point	ECTS
13	12	A
11		
10	10	B
9	7	C
8		
7	4	D
6	02	E
5	00	Fx
03		
00	−3	F

Problemdefinition

Skriv en funktion med navnet `convertGrade` som tager en karakter, givet på 7-trins eller 13-skalaen, som numerisk input samt hvilken skala der er benyttet som en streng og returnerer karakteren konverteret til ECTS skalaen som en streng (skrevet præcis som i tabellen ovenfor.)

Løsningsskabelon

```
def convertGrade(grade, scale):
  #insert your code
  return ECTSGrade
```

Input	
grade	Karakter givet på 7-trins skalaen eller 13-skalaen (i heltal).
scale	Skala der skal konverteres fra (streng): enten '13-scale' eller '7-point'.

Output	
ECTSGrade	Karakter på ECTS skalaen (streng).

Eksempel

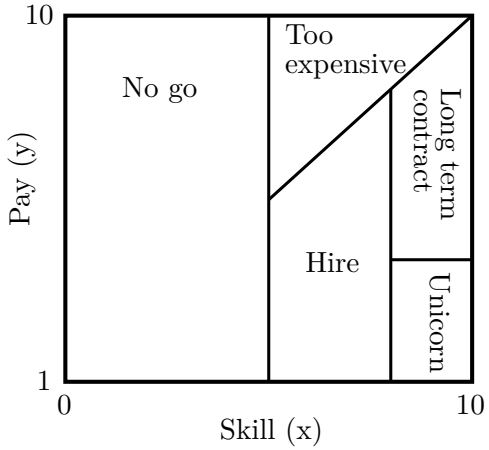
Hvis karakteren er 7 og skalaen der skal konverteres fra er `13-scale`, skal strengen `D` returneres.

A

Assignment B When to hire an employee

En arbejdsgiver har lavet en graf over hvornår en ansøger skal ansættes i hans firma. Han benytter to parametre til at beslutte hvordan ansøgeren skal behandles; hvor kvalificeret ansøgeren er (skill) og hvor høj løn ansøgeren vil have (pay). Evne bedrømmes på en skala fra 0 til 10 og løn bedømmes på en skala fra 1 til 10, da ingen ansøgere vil arbejde gratis. Arbejdsgiveren vil ikke ansætte nogle ansøgere der har en evne på under 5, så disse ansøgere bliver kategoriseret som "No go". Han laver en Evne-Løn linie og ansøgere over denne linie bliver kategoriseret som "Too expensive". Ansøgere under Evne-Løn linien og med en evne på mellem 5 og 8 ender i kategorien "Hire". Ansøgere med en evne på over 8, under Evne-Løn linien og over 4 Løn skal tages i betragtning til en "Long term contract". Ansøgere som har en evne på over 8 og som gerne vil have en løn på under 4 hører til kategorien "Unicorn", da den slags ansøgere ikke eksisterer. Den følgende tabel oplister de forskellige ansøgerzoner:

Applicant zone	Skill (x)	Pay (y)
`No go`	$x < 5$	
`Too expensive`	$x \geq 5$	$y > 0.9x + 1$
`Hire`	$5 \leq x < 8$	$y \leq 0.9x + 1$
`Long term contract`	$x \geq 8$	$4 < y \leq 0.9x + 1$
`Unicorn`	$x \geq 8$	$y \leq 4$



Problemdefinition

Skriv en funktion med navnet `hireApplicant` som tager skill og pay som numerisk input og returnerer applicantZone som en streng (skrevet præcis som i tabellen ovenfor).

Løsningsskabelon

```
def hireApplicant(skill, pay):
  #insert your code
  return applicantZone
```

Input	
skill	Evne givet på en skala fra 0 til 10 (decimaltal).
pay	Løn givet på en skala fra 1 til 10 (decimaltal).

Output	
applicantZone	Ansøger kategori (streng).

Eksempel

Hvis skill er 8 og pay er 7 skal strengen `Long term contract` returneres.

B

Assignment C — Average hearing loss

Høretab evalueres ofte ved brug af et audiogram, hvor høretabet måles for forskellige rentoner som afvigelsen fra 0 dB. Du arbejder med et datasæt hvor høretab er blevet målt for 7 forskellige rentoner og for et antal individer. Data for hvert individ er gemt som rækkerne i matricen M. Matricen indholder derfor 7 kolonner, én for hver målt frekvens. Du vil gerne beregne det gennemsnitslige høretab henover individer for hver frekvens individuelt, dog uden at medtage individer med et svært høretab. Et høretab på over 70 dB betragtes som svært.

Problemdefinition

Skriv en funktion med navnet `averagedB` som tager en matrix som input, fjerner rækker der indeholder værdier over 70 dB og returnerer en vektor der indeholder gennemsnittet af hver kolonne for den reducerede matrice.

Løsningsskabelon

```
def averagedB(M):
  #insert your code
  return averageM
```

Input	
M	Input matrix

Output	
averageM	Output vector

Eksempel

Forestil dig følgende input matrix:

$$M = \begin{bmatrix} 25 & 25 & 30 & 40 & 40 & 40 & 45 \\ 50 & 55 & 55 & 65 & 65 & 70 & 75 \\ 75 & 70 & 70 & 70 & 50 & 50 & 55 \\ 25 & 30 & 35 & 40 & 50 & 55 & 60 \end{bmatrix}$$

Da række 2 og 3 indeholder en værdi over 70 dB skal disse rækker fjernes.

$$M = \begin{bmatrix} 25 & 25 & 30 & 40 & 40 & 40 & 45 \\ \cancel{50} & \cancel{55} & \cancel{55} & \cancel{65} & \cancel{65} & \cancel{70} & \cancel{75} \\ \cancel{75} & \cancel{70} & \cancel{70} & \cancel{70} & \cancel{50} & \cancel{50} & \cancel{55} \\ 25 & 30 & 35 & 40 & 50 & 55 & 60 \end{bmatrix}$$

Gennemsnittet af hver kolonne udregnes for den reducerede matrice for at frembringe nedenstående vektor, som skal være output fra funktionen:

$$averageM = \begin{bmatrix} 25.0 & 27.5 & 32.5 & 40.0 & 45.0 & 47.5 & 52.5 \end{bmatrix}$$

C

Assignment D Building LEGO bricks

Du vil gerne bygge det højeste tårn af legoklodser, således at rumfanget af tårnet ikke er større end 1000cm^3. Det antages at rumfanget af en legoklods beregnes som:

$$V = h \cdot l \cdot w$$

hvor h, l og w er henholdsvis højde, længde og bredde af legoklodsen.

Problemdefinition

Skriv en funktion med navnet `buildLego` der som input tager højde h, længde l og bredde w af en legoklods og beregner det maximale antal legoklodser som du kan benytte til at bygge dit tårn, hvis volumen ikke må overstige 1000cm^3.

Løsningsskabelon

```
def buildLego(h, l, w):
  #insert your code
  return bricks
```

Input

h	Højde af legoklodsen (positivt decimaltal)
l	Længde af legoklodsen (positivt decimaltal)
w	Bredde af legoklodsen (positivt decimaltal)

Output

bricks	Antal legoklodser (heltal)

Eksempel

Forestil dig at vi er givet $h = 1$, $l = 8$, and $b = 1.6$. For forskellige værdier af *bricks* kan volumet beregnes som vist nedenfor:

bricks	75	76	77	78	79	80
volume	960.00	972.80	985.60	998.40	1011.2	1024.0

Det største antal legoklodser for hvilket volumet ikke overstiger $1000cm^3$ er $bricks = 78$. Funktionen skal derfor returnere værdien 78.

D

Assignment E Sudoku check

Sudoku er en logisk opgave med tal, hvor målet er at fylde et 9-gange-9 gitter med tal mellem 1 og 9 således at hver række, hver søjle og hver 3-gange-3 blok indeholder alle tal fra 1 til 9. Hver række, hver søjle og hver 3-gange-3 blok skal derfor summe til 45.

Problemdefinition

Skriv en funktion med navnet `sudokuCheck` der som input tager en 9-gange-9 matrice, sudokuBoard, og tjekker om summen af hver række, hver søjle og hver 3-gange-3 blok er 45. Funktionen skal returnere variablen check, som indeholder det samlede antal rækker, søjler og blokke, der ikke summer til 45.

Løsningsskabelon

```
def sudokuCheck(sudokuBoard):
  #insert your code
  return check
```

Input	
sudokuBoard	9-gange-9 matrice (tal fra 1 til 9)

Output	
check	Resultat af tjek (heltal)

Eksempel

Betragt sudoku pladen nedenfor. Summen af hver række, søjle og 3-gange-3 blok er beregnet (brune og røde tal).

5	3	4	6	7	8	9	1	2	45
6	7	2	1	9	5	3	5	8	46
1	9	8	3	4	2	4	6	7	44
8	5	9	7	6	1	4	2	3	45
4	2	6	8	5	3	7	9	1	45
7	1	3	9	2	4	8	5	6	45
9	6	1	5	3	7	2	8	4	45
2	8	7	4	1	9	6	3	5	45
3	4	5	2	8	6	1	7	9	45
45	45	45	45	45	45	44	46	45	

Summen af række 2 og 3 samt søjle 7 og 8 er forskellig fra 45, og programmet skal derfor returnere tallet 4.

E