Danmarks Tekniske Universitet

Kursusnavn	Introduktion til programmering og databehandling
Kursusnummer	02631, 02632, 02633, 02634, 02692
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 (dtu.codejudge.net/prog-aug18/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) `nextTriangular.py`
   (b) `tictactoe.py`
   (c) `blackjack.py`
   (d) `hypergeo.py`
   (e) `formatName.py`

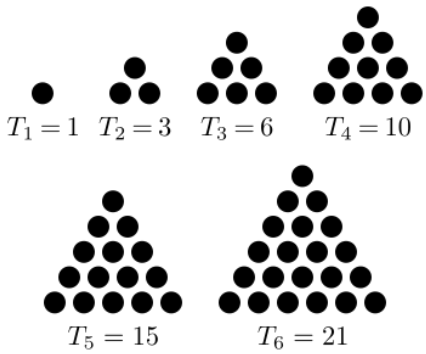
   Filerne skal afleveres separat (*ikke* som en zip-fil) og skal have præcis disse filnavne.

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 udelukkende 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 Triangular numbers

Et *triangulært tal* svarer til antallet af objekter der kan arrangeres i en trekant, som vist nedenfor.



De første 6 triangulære tal er 1, 3, 6, 10, 15, 21. Alle triangulære tal kan beregnes via formlen:

$$T_n = \frac{n(n+1)}{2}$$

Problemdefinition

Skriv en funktion med navnet `nextTriangular` der tager som input et tal x og returnerer det mindste triangulære tal som er større end eller lig x.

Løsningsskabelon

```
def nextTriangular(x):
  #insert your code
  return triangular
```

Input	
`x`	Start-tal, x (heltal.)

Output	
`triangular`	Mindste triangulære tal større end eller lig x (positivt heltal.)

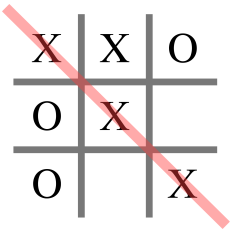
Eksempel

Forestil dig input-tallet $x = 17$. Det mindste triangulære tal som er større end eller lig x er 21, hvilket funktionen skal returnere.

A

Assignment B Tic-tac-toe

Tic-tac-toe (kryds og bolle) er et spil for to spillere, X og O. Brættet har 3-gange-3 felter, og spillerne skiftes til at markere et felt. Den spiller som først får tre markeringer i en horisontal, vertikal eller diagnoal række vinder spillet.



I denne opgave repræsenterer vi spillebrættet med en 3-gange-3 matrix. Et felt kan være tomt eller markeret med X eller O, hvilket vi repræsenterer med henholdsvis 0, 1, og -1. For eksempel er brættet i figuren repræsenteret ved følgende matrix:

$$\begin{bmatrix} 1 & 1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Problemdefinition

Skriv en funktion med navnet `tictactoe` der tager som input en 3×3 matrix der repræsenterer et spillebræt. Hvis en af spillerne har tre på stribe, skal funktionen sætte alle andre felter på brættet til nul, og returnere brættet som nu kun indeholder de tre "vindende" markeringer. Hvis ingen af spillerne har tre på stribe skal funktionen returnere det oprindelige spillebræt. Du kan antage at der ikke er mere end en vindende række på brættet.

Løsningsskabelon

```
def tictactoe(board):
  #insert your code
  return newBoard
```

Input	
`board`	Spillebræt (3×3 matrix indeholdende værdierne 0, 1, and -1.)

Output	
`newBoard`	Opdateret spillebræt (3×3 matrix indeholdende værdierne 0, 1, and -1.)

Eksempel

Forestil dig det spillebræt som ses i figuren ovenfor som input. Da X har en vindende diagonal række, skal alle andre felter på brættet sættes til nul, og funktionen skal returnere

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

B

Assignment C Blackjack

Blackjack er et kortspil som kan spilles mod en dealer på et kasino. Spilleren får et antal kort, og målet er at opnå en højere score end dealeren, uden at overstige 21 point. Scoren for kortene 2–10 er deres tal-værdi (2–10), og billedkort har en score på 10 point. Et es har en score på enten 1 point eller 11 point, således at den samlede score bliver så høj som muligt uden at overstige 21. Den samlede score for en spiller er summen af scorerne for de kort han har på hånden.

Problemdefinition

Skriv en funktion med navnet `blackjack` der tager som input et array med tal, som repræsenterer de kort som spilleren har. Kortene er repræsenteret i arrayet med tallene 1–10, svarende til deres point-værdi. Et es er repræsenteret ved værdien 1. Funktionen skal returnere den samlede score (mindre end eller lig 21). Hvis scoren overstiger 21 skal funktionen returnere 0.

Løsningsskabelon

```
def blackjack(hand):
  #insert your code
  return score
```

Input

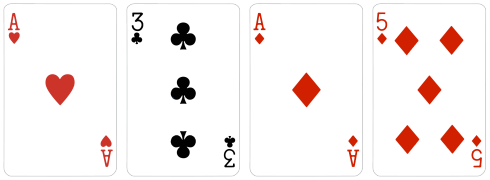
`hand`	Kort på hånden (array med tal 1–10.)

Output

`score`	Håndens score (heltal 0–21.)

Eksempel

Forestil dig følgende hånd, som repræsenteres som 1,3,1,5.



Hvis begge esser tæller som 1 point, er den totale score $1+3+1+5=10$. Hvis ét af esserne tæller som 11 er scoren $11+3+1+5=20$. Hvis begge esser tæller som 11 er scoren $11+3+11+5=30$. Da scoren 20 ikke overstiger 21 er den (bedste) samlede score for denne hånd 20 hvilket funktionen skal returnere.

C

Assignment D Hypergeometric function approximation

Den såkaldte hypergeometriske funktion kan defineres ved følgende uendelige række

$$F(a,b;c;z) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \cdot \frac{z^n}{n!},$$

hvor parametrene a, b og c er tal som kan antage vilkårlige værdier. Rækken er defineret for argumenter $|z| < 1$. Her betegner $n!$ fakultets-funktionen, og notationen $(a)_n$ betegner det stigende Pochhammer symbol, defineret som

$$(a)_n = \begin{cases} 1, & n = 0 \\ a(a+1)\cdots(a+n-1), & n = 1,2,3,\ldots \end{cases}$$

eller ækvivalent defineret som

$$(a)_n = \frac{\Gamma(a+n)}{\Gamma(a)},$$

hvor $\Gamma(\cdot)$ er gamma-funktionen.

I denne opgave vil vi lave en funktion der approksimerer den hypergeometriske funktion ved at summe et endeligt antal led.

$$\tilde{F}(a,b;c;z) = \sum_{n=0}^{N} \frac{(a)_n (b)_n}{(c)_n} \cdot \frac{z^n}{n!}, \tag{1}$$

Problemdefinition

Skriv en funktion med navnet `hypergeo` der tager som input tallene a, b, c, z and N og beregner den endelige sum i ligning (1).

Løsningsskabelon

```
def hypergeo(a, b, c, z, N):
  #insert your code
  return F
```

Input	
`a`, `b`, `c`	Parametre til den hypergeometriske funktion (decimaltal).
`z`	Funktionsargument (decimaltal).
`N`	Indeks for sidste led (positivt heltal).

Output	
`F`	Approksimation af den hypergeometriske funktion (decimaltal).

Eksempel

Forestil dig følgende input: $a = 1.1$, $b = 2.2$, $c = 3.3$, $z = 0.5$ og $N = 3$. Vi skal altså summe de følgende 4 led:

$$\begin{aligned}
\tilde{F} &= \frac{1\cdot 1}{1}\cdot\frac{z^0}{0!} + \frac{a\cdot b}{c}\cdot\frac{z^1}{1!} + \frac{a(a+1)\cdot b(b+1)}{c(c+1)}\cdot\frac{z^2}{2!} + \frac{a(a+1)(a+2)\cdot b(b+1)(b+2)}{c(c+1)(c+2)}\cdot\frac{z^3}{3!} \\
&= \frac{1\cdot 1}{1}\cdot\frac{0.5^0}{0!} + \frac{1.1\cdot 2.2}{3.3}\cdot\frac{0.5^1}{1!} + \frac{1.1(1.1+1)\cdot 2.2(2.2+1)}{3.3(3.3+1)}\cdot\frac{0.5^2}{2!} + \frac{1.1(1.1+1)(1.1+2)\cdot 2.2(2.2+1)(2.2+2)}{3.3(3.3+1)(3.3+2)}\cdot\frac{0.5^3}{3!} \\
&\approx 1 + 0.3666667 + 0.1432558 + 0.0586538 \approx \underline{1.5685763}
\end{aligned}$$

Assignment E Name formatting

Når man sammensætter en litteraturliste er det ofte påkrævet at forfatternavnene optræder på en konsistent måde. I denne opgave vil vi lave en funktion som tager et fuldt navn skrevet på en af de måder som er specificeret nedenfor, og returnerer navnet skrevet på en bestemt måde.

Vi antager at navnet givet som input enten skrives med efternavnet først, efterfulgt af et komma, og dernæst fornavn(e), eller som fornavn(e) efterfulgt af efternavnet. Fornavne kan enten skrive fuldt ud eller forkortes med deres begyndelsesbogstav. Vi kan antage at der enten er et eller to fornavne. Hvis et fornavn er forkortet kan der muligvis være et punktum efter bogstavet. For eksempel kan navnet `Evelyn Boyd Granville` skrives på en af følgende 14 måder.

Efternavn+Komma+Fornavn(e)	Fornavn(e)+Efternavn
`Granville, Evelyn Boyd`	`Evelyn Boyd Granville`
`Granville, E. Boyd`	`E. Boyd Granville`
`Granville, Evelyn B.`	`Evelyn B. Granville`
`Granville, E. B.`	`E. B. Granville`
`Granville, E Boyd`	`E Boyd Granville`
`Granville, Evelyn B`	`Evelyn B Granville`
`Granville, E B`	`E B Granville`

Outputtet fra funktionen skal være i form af efternavnet, efterfulgt af komma, efterfulgt af fornavn(e) forkortet til deres begyndelsesbogstav og uden punktum. I ovenstående eksempel skal output altså være:

`Granville, E B`

Bemærk at i input og output er navnene adskilt af et mellemrums-tegn, og at der er et mellemrum efter kommaet.

Problemdefinition

Skriv en funktion med navnet `formatName` der tager som input en tekst-streng der indeholder et navn skrevet i stil med et af navnene i tabellen ovenfor, og returnerer navnet skrevet som specificeret ovenfor.

Løsningsskabelon

```
def formatName(nameIn):
  #insert your code
  return nameOut
```

Input

`nameIn`	Input navn (tekst-streng).

Output

`nameOut`	Output navn (tekst-streng).

Eksempel

Givet navnet `Charles Babbage` som input, skal funktionen returnere strengen

`Babbage, C`

Givet navnet `Lovelace, A. Ada` som input, skal funktionen returnere strengen

`Lovelace, A A`

E