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 (`dtu.codejudge.net/prog-dec17/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) `normrange.py`
   (b) `readability.py`
   (c) `dataparser.py`
   (d) `temperaturemonitor.py`
   (e) `gamecalculator.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 Normalize to Range

En matrix eller en vektor kan normaliseres til et givent interval på følgende måde. Første skridt er at normalisere alle værdier imellem 0 og 1:

$$n_i = \frac{x_i - \min(x)}{\max(x) - \min(x)},$$

hvor x_i er det i. element af datapunkterne, organiseret i enten en vektor eller en matrix, og n_i er det tilsvarende i. element i det normaliserede data. Andet skridt er at skalere det normaliserede data til et givent interval $[a, b]$:

$$s_i = n_i \cdot (b - a) + a,$$

hvor s_i er det i. element af det nu skalerede data.

Problemdefinition

Skriv en funktion med navnet `normrange`, der som input tager en vektor eller en matrix `x`, samt en vektor `r` med to elementer $[a, b]$ der definerer intervallet. Funktionen skal returnere vektoren eller matricen `s` normaliseret til det givne interval. Hvis alle elementer i `x` er ens således at $\max(x) = \min(x)$ skal funktionen returnere `x`.

Løsningsskabelon

```
def normrange(x, r):
  #insert your code
  return s
```

Input	
`x`	Data der skal normaliseres til et interval (vektor eller matrix med decimaltal).
`r`	Interval (vektor med 2 elementer).

Output	
`s`	Input `x` normaliseret til interval `r` (vektor eller matrix med decimaltal).

Eksempel

Forestil dig følgende input vektor $x = [2,\ 4,\ 6,\ 8,\ 10]$ skaleret til $r = [-1,\ 1]$. Først normaliserer vi x fra 0 til 1:

$$n_i = \frac{x_i - 2}{10 - 2}$$

$$n = [0,\ 0.25,\ 0.5,\ 0.75,\ 1]$$

Så skalerer vi data i det givne interval (fra -1 til 1):

$$s_i = n_i \cdot (1 - (-1)) + (-1)$$

$$s = [-1,\ -0.5,\ 0,\ 0.5,\ 1]$$

Eksempel

Eksempel med matrix input:

$$x = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix},\ r = [5,\ 20]$$

resulterer i at funktionen skal returnere følgende:

$$s = \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}.$$

A

Assignment B Text Readability

LIX er en skala der indikerer hvor svær en tekst kan være at læse. Et læsbarhedsindex kan udregnes ved brug af ord og tegnsætning i teksten, som stiger med hvor svær læsbarheden af teksten regnes for at være. Læsbarhedsindekset LIX af en tekst er beregnet på følgende måde:

$$\text{LIX} = \frac{\text{W}}{\text{P}} + \frac{\text{L} \cdot 100}{\text{W}}.$$

Hvor

W er antallet af ord.

P er antallet af punktummer.

L er antallet af lange ord (flere end 6 bogstaver).

Problemdefinition

Skriv en funktion med navnet `readability`, der tager en tekst streng som input og returnerer LIX. Du kan antage, at tekststrengen kun indeholder bogstaver (`a-z` og `A-Z`), mellemrum og punktummer (`.`), dvs. der vil ikke forekomme andre symboler såsom komma, tal, spørgsmålstegn, bindestreg eller lignende. Du kan antage at hvis du tæller antallet af punktummer (`.`) i tekststrengen giver det værdien `P`. Funktionen skal returnere 0, hvis der ikke er nogle punktummer i teksten.

Løsningsskabelon

```
def readability(text):
  #insert your code
  return LIX
```

Input	
`text`	En tekststreng (streng kun indeholdende bogstaver, punktummer(`.`) og mellemrum).

Output	
`LIX`	Læsbarhedsindekset LIX (decimaltal)

Eksempel

Forestil dig følgende input tekststreng:

str = 'The Technical University of Denmark is located in Lyngby. Lyngby is a city in Denmark.'

Strengen indeholder **15 ord**, **2 punktummer** og **5 lange ord** (ord med flere end 6 bogstaver: Technical, University, Denmark, located, Denmark). Derfor,

$$\text{W} = 15, \;\; \text{P} = 2, \;\; \text{L} = 5$$

$$\text{LIX} = \frac{15}{2} + \frac{5 \cdot 100}{15} = 40.83$$

B

Assignment C Data Parser

Du skal fortolke data fra en ekstern kilde. Det data du modtager er altid en vektor af vilkårlig længde indeholdende heltal, som skal fortolkes og konverteres til enten en matrix eller en vektor. Forestil dig inputvektoren har følgende format, hvor A,B,C,D,E,F,... er heltal:

$$\text{data} = [\text{A, B, C, D, E, F, ...}]$$

Data vektoren ovenfor skal fortolkes ved brug af følgende regler:

hvis A=0: Data skal fortolkes som en vektor,

- B er længden af vektoren.
- C,D,E,F,... er indholdet af vektoren (begrænset af længden B).

hvis A=1: Data skal fortolkes som en matrix,

- B er antallet af rækker i matricen,
- C er antallet af søjler i matricen,
- D,E,F,... er indholdet af matricen (begrænset af antal rækker B og søjler C)*.

*) Bemærk, at matricen altid er 'row-major', hvilket betyder, at første (øverste) række fyldes fra venstre mod højre, derefter fyldes anden række fra venstre mod højre, osv.

Problemdefinition

Skriv en funktion med navnet `dataparser`, der som input tager en vektor indeholdende data, som skal fortolkes. Funktionen skal returnere enten en vektor eller en matrix med det fortolkede data. Du kan antage, at vektoren har minimumslængden 2, hvis A = 0, og minimumslængden 3, hvis A = 1. Hvis der ikke er nok data til at konstruere vektoren eller matricen, skal funktionen returnere en tom vektor `[]`. Hvis A hverken er 0 eller 1, skal funktionen returnere en tom vektor `[]`. Hvis der er for meget data, skal funktionen kun fortolke den nødvendige mængde data.

Løsningsskabelon

```
def dataparser(data):
  #insert your code
  return parsed
```

Input	
`data`	Data der skal fortolkes (vektor med heltal).

Output	
`parsed`	Det fortolkede data (vektor eller matrix med heltal).

Eksempel

Forestil dig følgende input

$$\text{data} = [1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10].$$

Det 1. element er `1`, hvilket betyder at data skal fortolkes som en matrix. Det 2. element er `2` hvilket betyder, at matricen har 2 rækker, og det 3. element er `3`, hvilket betyder, at matricen har 3 søjler. Vi skal konstruere en matrix med $2 \cdot 3 = 6$ elementer ved brug af det resterende data: `[4,5,6,7,8,9,10]` (7 elementer). Vi skal derfor forkaste det sidste element 10. Funktionen skal da returnere:

$$\text{parsed} = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Assignment D Temperature Monitor

Du skal skrive et overvågningssystem, der kan advare brugeren, hvis noget uønsket hænder med temperaturen i et område. Systemet måler allerede temperaturen i faste intervaller og gemmer resultatet. Baseret på disse temperaturer skal systemet returnere advarselskoder for følgende kriterier:

advarselskriterie	**kode**
Ingen advarsel.	`0`
Temperaturen er 2 enheder højere eller lavere end den tidligere måling.	`1`
Temperaturen er under 0 for anden gang i træk.	`2`
Temperaturen er mere end 5 for anden gang i træk.	`3`

Problemdefinition

Skriv en funktion med navnet `temperaturemonitor`, der tager en sekvens `T` af temperaturmålinger som input, og returnerer advarselskoderne `W` for hver temperaturmåling ved brug af kriterierne listet ovenfor. Den højeste advarselskode skal altid returneres, hvis flere kriterier er opfyldt samtidigt. Bemærk, at advarselskoder 1-3 aldrig kan forekomme ved første måling.

Løsningsskabelon

```
def temperaturemonitor(T):
  #insert your code
  return W
```

Input

`T`	Sekvens af temperaturer (vektor med decimaltal).

Output

`W`	Advarselskoder (vektor med heltal $[0-3]$).

Eksempel

Forestil dig følgende sekvens af temperaturer

$$T = [1.8,\ 2.1,\ -0.2,\ -2.3,\ 1.1,\ 5.1,\ 5.5].$$

Første advarselskode er altid `0`. Der er færre end 2 enheder fra 1.8 til 2.1, temperaturen er ikke lavere end 0 eller højere end 5, så 2. advarselskode er `0`. Der er flere end 2 enheder fra 2.1 til -0.2, så 3. advarselskode er `1`. Der er flere end 2 enheder fra -0.2 til -2.3 og temperaturen har været under 0 to gange i træk. Derfor gælder både advarselskode `1` og `2`, så vi vælger den højeste af dem som 4. advarselskode. 5. og 6. advarselskode er begge `1`, fordi temperaturen ændrer sig mere end 2 enheder. Den 7. advarselskode er `3`, fordi temperaturen har været højere end 5 to gange i træk. Dermed er

$$W = [0,\ 0,\ 1,\ 2,\ 1,\ 1,\ 3].$$

Assignment E Game Point Calculator

I en sportsturnering kan man bestemme et holds placering udfra antal point og deres mål difference. For et hold beregnes antal point udfra følgende udfald:

udfald	point
Holdet scorer flere mål end deres modstander.	`3`
Holdet scorer samme antal mål som deres modstander.	`1`
Holdet scorer færre mål end deres modstander.	`0`

Måldifferencen `GD` er beregnet udfra differencen imellem antal mål holdet har scoret `GF`, og antal mål modstanderen har scoret imod dem `GA`.

$$\mathrm{GD} = \mathrm{GF} - \mathrm{GA}$$

Problemdefinition

Skriv en funktion med navnet `gamecalculator`, der som input tager en matrix med mål scoret i en serie af kampe imellem et hold og deres modstandere i N kampe. Første række i matricen `goals` er mål scoret af referenceholdet, og anden række i matricen er mål scoret af referenceholdets modstandere. Funktionen skal returnere en vektor med summen af det antal point referenceholdet har fået, samt deres måldifference.

Løsningsskabelon

```
def gamecalculator(goals):
  #insert your code
  return result
```

Input

`goals`	Mål scoret ($2 \times N$ matrix med heltal).

Output

`result`	Point og mål difference for reference holdet `[points, GD]` (vektor med heltal).

Eksempel

Forestil dig input matricen

$$\mathrm{goals} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}.$$

I den 1. kamp vinder reference holdet `1-0`, hvilket giver 3 point. 2. kamp ender uafgjort, hvilket giver 1 point til referenceholdet. 3. kamp ender også uafgjort, hvilket giver 1 point til referenceholdet. I 4. kamp taber referenceholdet `0-2`, hvilket giver 0 point. Det totale antal point er

$$\mathrm{points} = 3 + 1 + 1 + 0 = 5.$$

Måldifferencen for referenceholdet er

$$\mathrm{GD} = (1 + 1 + 0 + 0) - (0 + 1 + 0 + 2) = -1.$$

Funktionen skal da returnere

$$\mathrm{result} = [5, -1]$$

E