Danmarks Tekniske Universitet

Kursusnavn	Introduktion til programmering og databehandling
Kursusnummer	02633
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-jan18/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) `health.py`
   (b) `imageCrop.py`
   (c) `chaos.py`
   (d) `threshold.py`
   (e) `checkParentheses.py`

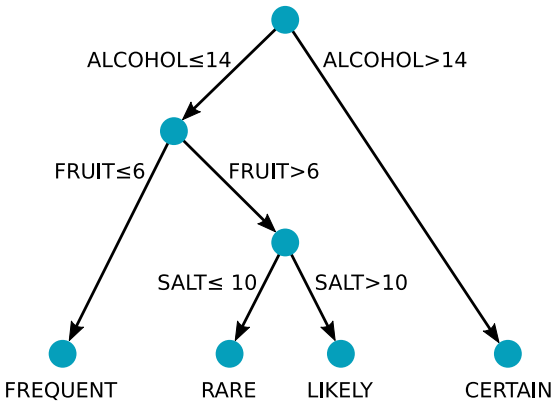
   Filerne skal afleveres separat (*ikke* som en zip-fil) og skal have præcis disse filnavne.

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 Health risk

Som hjælp til at vurdere patienters helbredsrisiko har en læge lavet det beslutningstræ som er vist i figuren. Træet er baseret på 3 risikofaktorer:



ALCOHOL: Antallet af genstande pr. uge.

FRUIT: Antallet af stykker frugt pr. uge

SALT: Mængden af salt i gram per dag.

For at vurdere en patients sandsynlighed for at udvikle en helbredsproblem starter doktoren i toppen af træet. Hvis patienten drikker mere end 14 genstande om ugen går doktoren videre ned af den højre gren i træet; ellers, går hun videre ned ad den venstre gren. Denne proces fortsætter indtil doktoren når en af de nederste knudepunkter i træet. De nederste knudepunkter er betegnet med den tilsvarende sandsynlighed for at udvikle et helbredsproblem.

Problemdefinition

Skriv en funktion med navnet `health` der tager som input de tre risikofaktorer og returnerer den tilsvarende sandsynlighed for et helbredsproblem. Outputtet skal være i form af en tekst-streng, skrevet med store bogstaver som i figuren.

Løsningsskabelon

```
def health(ALCOHOL, FRUIT, SALT):
  #insert your code
  return RISK
```

Input

`ALCOHOL`	Antallet af genstande pr. uge. (heltal)
`FRUIT`	Antallet af stykker frugt pr. uge (heltal).
`SALT`	Mængden af salt i gram per dag (heltal).

Output

`RISK`	Sandsynlighed for at udvikle et helbredsproblem (tekst-streng). Én af følgende værdier: `FREQUENT`, `RARE`, `LIKELY`, `CERTAIN`.

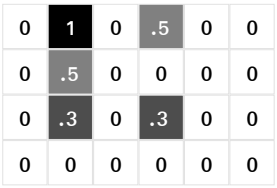
Eksempel

Forestil dig en patient som drikker 8 genstande om ugen, spiser 9 stykker frugt om ugen og indtager 10 gram salt per dag. Når vi følge beslutningstræet går vi først til venstre fordi ALCOHOL ≤ 14; derefter går vi til højre fordi FRUIT > 6; til sidst går vi til venstre fordi SALT ≤ 10. Vi ender dermed i knudepunkter mærket `RARE` hvilket er hvad funktionen skal returnere.

A

Assignment B Image crop

Et sort og hvidt billede kan repræsenteres ved et array, hvor værdierne i arrayet repræsenterer gråtone-niveauer, fx. hvid=0, sort=1, og gråtoner er tal mellem 0 og 1. Når et billede har en fast baggrundsfarve kan det være brugbart at beskære billedet ved at fjerne de ydre dele af billedet som har konstant farveværdi.



Problemdefinition

Skriv en funktion med navnet `imageCrop` der tager som input et array der repræsenterer et sort/hvidt billede, samt et tal mellem 0 og 1 der repræsenterer baggrundsfarven. Funktionen skal returnere et beskåret billeder, hvor de første og sidste rækker og søjler som har konstant værdi lig med baggrundsfarven er fjernet. Du kan antage at billedet indeholder mindst en pixel der har en anden farve end baggrundsfarven.

Løsningsskabelon

```
def imageCrop(img_in, background):
  #insert your code
  return img_out
```

Input

`img_in`	Gråtone input-billede (array med decimalttal mellem 0 og 1.
`background`	Background gråtone-niveau (decimaltal mellem 0 og 1).

Output

`img_out`	Beskåret gråtone billede (array med decimaltal mellem 0 og 1).

Eksempel

Forestil dig det billede der er illustreret i figuren ovenfor. Det kan repræsenteres ved det følgende array:

$$\begin{bmatrix} 0 & 1 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Hvis baggrundsfarven er givet som 0 (hvid) skal den første søjle og de to sidste søjler samt den sidste række fjernes, idet disse rækker og søjler er konstant og lig baggrundsfarven 0. Resultatet skal således være:

$$\begin{bmatrix} 1 & 0 & 0.5 \\ 0.5 & 0 & 0 \\ 0.3 & 0 & 0.3 \end{bmatrix}$$

B

Assignment C Chaos

Den såkaldte *logistiske ligning* er et simpelt eksempel som kan udvise kaotisk opførsel. Den kan udtrykkes ved formlen

$$x^* = k \cdot x \cdot (1 - x) \tag{1}$$

Formlen tager et tal, x (mellem nul og et) og beregner et nyt tal, x^*. Det nye tal kan derefter sættes ind i formlen igen på x's plads, og så videre, og således kan man beregne en sekvens af tal.

Tallet k i formlen kaldes en *parameter*, og den bestemmer sekvensens opførsel. Parameteren k skal være i intervallet $0 \leq k \leq 4$. Efter nogle iterationer falder sekvensen ind i et mønster, som afhængig af k kan være konstant, periodisk eller kaotisk.

Problemdefinition

Skriv en funktion med navnet `chaos` der tager som input en start-værdi for x, en værdi for parameteren k samt et antal iterationer T. Funktionen skal returnere den tal-sekvens der kommer af at iterere den logistiske ligning T gange.

Løsningsskabelon

```
def chaos(x, k, T):
  #insert your code
  return seq
```

Input

`x`	Begyndelses-værdi for x (decimaltal mellem 0 og 1).
`k`	Parameter k for den logistiske ligning (decimaltal mellem 0 og 4).
`T`	Antal iterationer (positivt heltal).

Output

`seq`	Tal-sekvens beregnet ved at iterere den logistiske ligning (vektor af decimaltal.)

Eksempel

Forestil dig følgende input: $x = 0.5$, $k = 3.6$, $T = 10$. Med begyndelsesbetingelsen $x = 0.5$ skal vi nu beregne en sekvens på $T = 10$ tal. Det første tal kan beregnes som:

$$x^* = 3.6 \cdot 0.5 \cdot (1 - 0.5) = 0.9.$$

Dette tal fødes herefter tilbage i formlen som x for at beregne det andet tal:

$$x^* = 3.6 \cdot 0.9 \cdot (1 - 0.9) = 0.324.$$

Den endelige sekvens af 10 tal kan beregnes som:

0.9000 0.3240 0.7885 0.6004 0.8637 0.4238 0.8791 0.3827 0.8505 0.4578

C

Assignment D Fail prediction

En universitetsunderviser ønsker at gøre en indsats for at hjælpe studerende som er i risiko for at dumpe hendes kursus på kandidatniveau. Hun beslutter at analysere hvorledes de studerendes karaktergennemsnit (GPA) på bachelorniveau er relateret til hvorvidt de dumper hendes kandidat-kursus. Hun har adgang til data fra N tidligere studerende: For hver af disse N studerende kender hun deres bachelor-karaktergennemsnit (GPA) samt hvorvidt de dumpede hendes kandidat-kursus. Et eksempel på et sådant datasæt med $N = 7$ studerende er givet her:

Karaktergennemsnit (GPA)	4.20	3.52	9.44	12.00	10.50	7.30	9.44
Dumpet	1	1	0	0	0	0	1

Hun beslutter sig for at gøre en særlig indsats over for studerende med GPA under en vis tærskelværdi, X

$$\text{GPA} \leq X.$$

For at bestemme tærskelværdien, benytter hun følgende tilgang:

1. Hun udtrækker alle unikke GPA-værdier og sorterer dem.
2. Hun tager udelukkende alle midtpunkter (gennemsnit) mellem fortløbende unikke GPA-værdier i betragtning som mulige tærskelværdier.
3. For hver mulig tærskelværdi beregner hun en F-score, definieret som:
$$\text{F-score} = 2 \cdot \frac{[\text{Antal dumpede studerende med GPA} \leq X]}{[\text{Totalt antal studerende med GPA} \leq X] + [\text{Totalt antal dumpede studerende}]}$$
4. Til slut vælger hun den tærskelværdi der giver den højeste F-score. Hvis der er mere end en tærskelværdi der giver samme højeste F-score, vælger hun den laveste af disse tærskelværdier.

Problemdefinition

Skriv en funktion med navnet `threshold` der tager som input en vektor der indeholder GPAer for N studerende samt en vektor med samme længde der indeholder ettaller og nuller som indikerer om de studerende har dumpet eller bestået. Funktionen skal returnere tærskelværdien X beregnet som beskrevet ovenfor.

```
def threshold(GPA, fail):
  #insert your code
  return X
```

Input

`GPA`	Karaktergennemsnit (GPA) for hver studerende (vektor med decimaltal).
`fail`	Dumpet eller bestået (kodet som 1 og 0) for hver studerende (vektor med nuller og ettaller).

Output

`X`	Tærskelværdi (decimaltal).

Eksempel

Betragt det datasæt der er i tabellen ovenfor. Der er seks unikke værdier (9.44 optræder to gange), og efter at have sorteret de unikke værdier har vi listen nedenfor. Herefter kan de fem mulige tærskelværdier beregnes som gennemsnittet mellem de fortløbende unikke GPA-værdier. For hver tærskelværdi beregnes herefter F-scoren.

Sorterede unikke GPA-værdier	3.52	4.20	7.30	9.44	10.50	12.00
Mulig tærskelværdi X	3.86	5.75	8.37	9.97	11.25	
Antal dumpede studerende med GPA $\leq X$	1	2	2	3	3	
Totalt antal studerende med GPA $\leq X$	1	2	3	5	6	
Totalt antal dumpede studerende	3	3	3	3	3	
F-score $\approx$	0.5	**0.8**	0.667	0.75	0.667	

Den højeste F-score er 0.8, og funktionen skal returnere den tilsvarende tærskelværdi som er $X = 5.75$.

Assignment E Matching parentheses

Paranteser kan bruges til at gruppere led i et aritmetisk udtryk, og man kan benytte forskellige typer af paranteser (runde, firkantede og krøllede) for at gøre komplekse aritmetisk udtryk lettere at læse. Et eksempel kunne være `1/(a + 1/[b + 1/{c + 1/d}])`. Algoritmen nedenfor kan tjekke at paranteserne i et udtryk er balancerede, dvs. at hver start-parantes har en tilsvarende slut-parantes af samme type, og at paranteserne er indlejret korrekt.

Algoritme	Beskrivelse

input: En tekst-streng, **str**, med N tegn. Lad **str[i]** betegne det **i**'te tegn i strengen.
Lad **list** betegne en liste af karakterer. Til at begynde med er **list** tom.
for *i=1 til N*
  hvis ***str[i]*** *er en af de tre mulige start-paranteser*
  Tilføj **str[i]** for enden af **list**.
  ellers hvis ***str[i]*** *er en af de tre mulige slut-paranteser*
  hvis ***list*** *ikke er tom og* ***str[i]*** *matcher typen af den sidste karakter i* ***list***
  Fjern den sidste karakter i **list**
  ellers
  returnér ***i***
hvis ***list*** *er tom*
  returnér *0*
ellers
  returnér *N+1*
	Algoritmen læser hvert tegn i strengen (alle tegn ud over `([{)]}` ignoreres). Når algoritmen møder en start-parantes, tilføjes den for enden af en liste. Når algoritmen møder en slut-parantes, tjekker den at slut-parantesen er af samme type som den seneste start-parantes, hvilken den så fjerner fra listen hvis de matcher. Hvis typen af parantes ikke matcher, returnerer algoritmen det indeks i strengen hvor fejlen opstod. Hvis alle paranteser matcher vil listen være tom når alle tegn i strengen er gennemløbet: I dette tilfælde returnerer algoritmen nul. Ellers, hvis listen ikke er tom, returnerer algoritmen N+1.

Problemdefinition

Skriv en funktion med navnet `checkParentheses` der tager en tekst-streng som input og returnerer et indeks forskelligt fra nul hvis en parantes-fejl opdages ifølge ovenstående algoritme, eller returnerer nul hvis paranteserne matcher. Funktionen skal håndtere runde, firkantede og krøllede paranteser.

Løsningsskabelon

```
def checkParentheses(str):
  #insert your code
  return err
```

Input

`str`	Input streng (tekst-streng).

Output

`err`	Indeks for parantes-fejl eller nul hvis ingen fejl detekteret (heltal).

Eksempel

Forestil dig følgende tekst-streng:

```
([a)]{
```

Det fjerde tegn `)` i strengen matcher ikke den seneste start-parantes, som er `[`. Derfor skal algoritmen returnere værdien 4.

E