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-jan16/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) `weeklyAverage.py`
   (b) `carsAvailable.py`
   (c) `speedOfLight.py`
   (d) `similarityMatrix.py`
   (e) `coinReturn.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 Robust temperature average

For at beregne den gennemsnitlige temperatur hen over en uge kunne man måle temperaturen hver dag og så beregne gennemsnittet over de syv dage i ugen. Den metode vil dog være meget sensitiv: Hvis, for eksempel, en af dagene i ugen har en ekstrem temperatur, vil det have en meget stor indflydelse på det ugentlige gennemsnit. For at beregne et mindre sensitivt (mere robust) temperatur-gennemnist beslutter du dig for at ekskludere de to højeste og de to laveste temperaturer, og beregne gennemsnittet af de resterende mellemliggende 3 temperaturer.

Problemdefinition

Skriv en funktion med navnet `weeklyAverage` der tager som input en vektor indeholdende 7 temperatur-målinger og returnerer gennemsnittet når de to højeste og de to laveste temperaturer ekskluderes.

Løsningsskabelon

```
def weeklyAverage(dayTemp):
  #insert your code
  return weekTemp
```

Input

`dayTemp`	Temperatur-målinger (vektor med længde 7).

Output

`weekTemp`	Gennemsnitstemperatur (decimaltal).

Eksempel

Hvis du fx. har følgende temperatur-målinger:

$$[17.3,\ 18.2,\ 31.2,\ 14.2,\ -12.5,\ 16.5,\ 14.2]$$

er de to højeste temperaturer 31.2 og 18.2 og de to laveste er -12.5 og 14.2. De resterende temperaturer er 17.3, 16.5, og 14.2, og gennemsnittet kan beregnes som:

$$\frac{17.3 + 16.5 + 14.2}{3} = 16.0$$

A

Assignment B Car rental

Et biludlejningsfirma har 6 forskellige bil-kategorier i deres flåde:

`Mini` `Economy` `Compact` `Standard` `Fullsize` `Luxury`

Antallet af biler til rådighed i hver af de 6 kategorier kan repræsenteres af en vektor med længde 6 hvor rækkefølgen af kategorierne er som i listen ovenfor.

Problemdefinition

Skriv en funktion med navnet `carsAvailable` som tager en vektor indeholdende antallet af biler til rådighed og en streng der repræsenterer en af bil-kategorierne og returnerer antallet af biler til rådighed i den kategori. Funktionen skal kun sammenligne det første bogstav i den givne kategori-streng og skal ikke skelne mellem store og små bogstaver. Hvis den angivne kategori for eksempel er `s`, `Stnd` eller `st` skal funktionen returnere antallet af biler i kategorien `Standard`. Hvis den angivne kategori ikke matcher første bogstav i nogen af de 6 bil-kategorier, skal funktionen returnere -1 (minus et).

Løsningsskabelon

```
def carsAvailable(fleet, category):
  #insert your code
  return n
```

Input

`fleet`	Antal biler til rådighed i hver kategori (vektor med længde 6).
`category`	Bil-kategori (streng).

Output

`n`	Antal biler til rådighed i den valgte kategori, eller -1 (heltal).

Eksempel

Forestil dig følgende vektor med antallet af biler til rådighed: [5, 0, 17, 13, 11, 1]. Hvis den ønskede kategori er givet ved strengen `Lux` er det første bogstav `L` hvilket matcher kategorien `Luxury` hvoraf der er 1 bil i flåden. Funktionen skal altså returnere værdien 1.

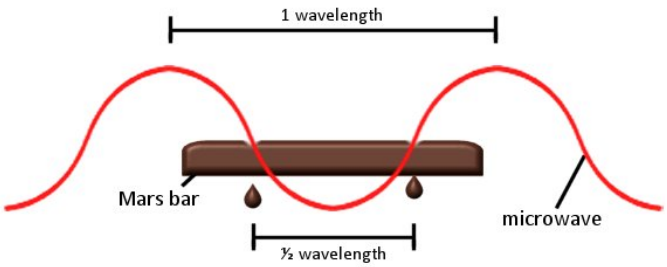
B

Assignment C Speed of light

Lysets hastighed kan måles ved at opvarme en chokolade-bar i en mikrobølgeovn. En mikrobølgeovn udsender elektromagnetisk stråling ved en given frekvens f. På grund af stående bølger inde i mikrobølgeovnen vil chokoladen ikke opvarmes ensartet, og man kan måle afstanden mellem de smeltede områder. Denne afstand vil være lig en halv bølgelængde, og kan derfor benyttes til at beregne lysets hastighed ifølge formlen:

$$c = f \cdot \lambda,$$

hvor f er frekvensen, λ er bølgelængden og c er lysets hastighed.



Kilde: canteengirl.org

For at få et mere præcist estimat kan man udregne et gennemsnit over flere målinger af bølgelængden.

Problemdefinition

Skriv en funktion med navnet `speedOfLight` der tager som input frekvensen og en vektor med målte bølgelængder. Funktionen skal beregne lysets hastighed for hver bølgelængde ifølge ovenstående formel, og returnere gennemsnittet af de beregnede værdier.

Løsningsskabelon

```
def speedOfLight(f, wavelengths):
  #insert your code
  return c
```

Input	
`f`	Frekvensen (f) i Hz
`wavelengths`	Bølgelængder (λ) i meter (vektor)

Output	
`c`	Estimat af lysets hastighed.

Eksempel

Antag at frekvensen er $f = 2.45 \cdot 10^9$ [Hz] og der er 3 målte bølgelænger: 0.122, 0.125 og 0.123 [m]. Vi kan så beregne

$$c_1 = 2.45 \cdot 10^9 \cdot 0.122 = 298.9 \cdot 10^6 \text{ [m/s]} \quad (1)$$
$$c_2 = 2.45 \cdot 10^9 \cdot 0.125 = 306.25 \cdot 10^6 \text{ [m/s]} \quad (2)$$
$$c_3 = 2.45 \cdot 10^9 \cdot 0.123 = 301.35 \cdot 10^6 \text{ [m/s]} \quad (3)$$

Gennemsnittet kan beregnes som $c = \frac{c_1+c_2+c_3}{3} = 302.17 \cdot 10^6$ hvilket er det endlige resultat.

C

Assignment D Similarity matrix

En similaritets-matrix indeholder de parvise similariteter mellem hvert element i to vektorer. Vi har vektorerne $x = [x_1, x_2, \ldots, x_N]$ og $y = [y_1, y_2, \ldots, y_M]$ og definerer følgende similaritets-mål:

$$\mathrm{s}(x_i, y_j) = \exp\left(-\delta(x_i - y_j)^2\right),$$

hvor δ er en parameter kaldet længde-skalaen. Similaritets-matricen S er defineret som:

$$S = \begin{bmatrix} \mathrm{s}(x_1, y_1) & \mathrm{s}(x_1, y_2) & \cdots & \mathrm{s}(x_1, y_M) \\ \mathrm{s}(x_2, y_1) & \mathrm{s}(x_2, y_2) & \cdots & \mathrm{s}(x_2, y_M) \\ \vdots & \vdots & & \vdots \\ \mathrm{s}(x_N, y_1) & \mathrm{s}(x_N, y_2) & \cdots & \mathrm{s}(x_N, y_M) \end{bmatrix}.$$

Matricen S er således en $N \times M$ matrix, hvor element (i, j) er similariteten mellem x_i og y_j.

Problemdefinition

Skriv en funktion med navnet `similarityMatrix` der tager som input de to vektorer x og y samt længde-skalaen δ og returnerer similaritets-matricen S.

Løsningsskabelon

```
def similarityMatrix(x, y, delta):
  #insert your code
  return S
```

Input	
`x`, `y`	Input-data (vektorer).
`delta`	Længdeskala (decimaltal).

Output	
`S`	Similaritets-matrix.

Eksempel

Forestil dig følgende input-vektorer:

$$x = [1.1,\ 1.2], \quad y = [1.3,\ 1.4,\ 1.5],$$

og antag at vi har $\delta = 2$. Similaritets-matricen kan da beregnes som:

$$S = \begin{bmatrix} 0.9231 & 0.8353 & 0.7261 \\ 0.9802 & 0.9231 & 0.8353 \end{bmatrix}$$

hvor, for eksempel, element (1,2) er beregnet som:

$$\mathrm{s}(x_1, y_2) = \exp\left(-2 \cdot (1.1 - 1.4)^2\right) = 0.8353$$

D

Assignment E Coin return

Du er i gang med at lave en maskine til at give vekselpenge (mønter) til kunder i et supermarked. Givet at et bestemt beløb skal udbetales skal du beslutte hvilke mønter der skal udleveres. Du kender værdierne af de forskellige slags mønter der er til rådighed i maskinen og kan antage at der er tilstrækkeligt mange mønter til rådighed af hver slags. Du skal benytte den følgende algoritme til at vælge mønterne:

1. Lad x være det beløb der skal udbetales.
2. Hvis x er mindre end halvdelen af værdien af den mindste mønt: Stop.
3. Udbetal den største tilgængelige mønt der har en værdi mindre end eller lig x.
4. Træk værdien af den udbetalte mønt fra x.
5. Gentag fra 2 .

Problemdefinition

Skriv en funktion med navnet `coinReturn` der tager som input en vektor der indeholder værdierne af de mønter der er tilgængelige i maskinen (sorteret efter værdi med den mindste først) samt samt det beløb der skal udbetales. Funktionen skal returnere en vektor der indeholder værdierne af de mønter der udbetales (i den rækkefølge som algoritmen definerer).

Løsningsskabelon

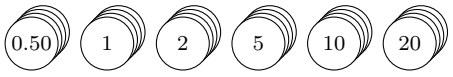
```
def coinReturn(coinsInMachine, amount):
  #insert your code
  return coinsToReturn
```

Input	
`coinsInMachine`	Værdierne af de slags mønter der er tilgængelige i maskinen (vektor).
`amount`	Beløb der skal udbetales (decimaltal)

Output	
`coinsToReturn`	Værdierne af de mønter der udbetales (vektor).

Eksempel

Forestil dig at udbetale beløbet 34.60, når følgende slags mønter er tilgængelige i maskinen:



Inputtet `coinsInMachine` vil være vektoren [0.50, 1, 2, 5, 10, 20]. Algoritmen skal udføres således:

	Iteration 1	Iteration 2	Iteration3	Iteration 4	Iteration 5	Iteration 6
Resterende beløb (x)	34.60	14.60	4.60	2.60	0.60	0.10
Mønt udbetalt	20	10	2	2	0.5	Stop

Derfor vil de udbetalte mønter være $[20, 10, 2, 2, 0.5]$ hvilket skal være output fra funktionen.

E