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-aug17/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:

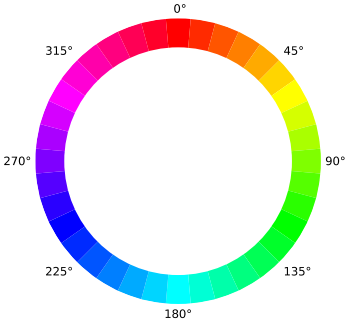
   (a) `rgb2hue.py`
   (b) `cardValidation.py`
   (c) `polygonPerimeter.py`
   (d) `rotateScale.py`
   (e) `costliestCar.py`

   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 Color hue



På en computer repræsenteres farver ofte i RGB-farveskemaet, dvs. som tre tal der indikerer mængden af rødt, grønt og blåt lys. En anden måde man kan repræsentere farver på er HSL-farveskemaet som består af tre tal der indikerer farvetone (hue), mætning (saturation) og lysstyrke (luminance).

Givet en farve i RGB-repræsentation kan farvetonen (H, målt i grader) beregnes som beskrevet nedenfor. Lad R, G og B betegne RGB-værdierne (i procent, repræsenteret med tal mellem 0 og 1). Vi lader Δ betegner variationsbredden af RGB-værdierne (forskellen mellem den største og mindste værdi),

$$\Delta = \max(R, G, B) - \min(R, G, B).$$

1. Beregn først H ud fra følgende formel:

$$H = \begin{cases} 60 \cdot \dfrac{G - B}{\Delta} & \text{Hvis } R \text{ er den største RGB-værdi, } R = \max(R, G, B). \\ 120 + 60 \cdot \dfrac{B - R}{\Delta} & \text{Hvis } G \text{ er den største RGB-værdi, } G = \max(R, G, B). \\ 240 + 60 \cdot \dfrac{R - G}{\Delta} & \text{Hvis } B \text{ er den største RGB-værdi, } B = \max(R, G, B). \end{cases}$$

   Du kan antage at $\Delta \neq 0$. Bemærk at hvis to af RGB-værdierne begge er maksimum vil de tilsvarende linjer i formlen føre til det samme resultat.

2. Hvis den beregnede værdi for H er negativ, skal du lægge 360 til resultatet for at få den korrekte værdi i grader, så vi har

$$0° \leq H < 360°.$$

Problemdefinition

Skriv en funktion med navnet `rgb2hue` der tager som input de tre RGB-værdier og beregner farvetonen i grader.

Løsningsskabelon

```
def rgb2hue(R, G, B):
  #insert your code
  return H
```

Input	
`R`, `G`, `B`	Rød, grøn og blå farve-værdi i procent (decimaltal mellem 0 og 1).

Output	
`H`	Farvetone i grader (decimaltal, $0 \leq H < 360$).

Eksempel

Forestil dig RGB-værdierne $R = 0.6$, $G = 0.2$, $B = 0.3$. Vi kan nu beregne

$$\Delta = \max(R, G, B) - \min(R, G, B) = 0.6 - 0.2 = 0.4.$$

Da den røde farve-værdi er størst beregner vi

$$H = 60 \cdot \frac{G - B}{\Delta} = 60 \cdot \frac{0.2 - 0.3}{0.4} = -15$$

Fordi resultatet er negativt skal vi lægge 360 til, hvilket giver det endelige resultat

$$H = -15 + 360 = \underline{345}$$

A

Assignment B Credit card validation

Udfra et 16-cifret kreditkort-nummer kan en checksum beregnes vha. følgende algoritme (kaldet Luhn's algoritme), som gør det muligt at undersøge om kort-nummeret er gyldigt:

1. Begyndende med første ciffer fra venstre, konverter *hvert andet* ciffer (dvs. ciffer på position 1, 3, 5 osv.) ifølge denne tabel:

Ciffer	0	1	2	3	4	5	6	7	8	9
Konverter til	0	2	4	6	8	1	3	5	7	9

   De resterende cifre beholdes som de var.

2. Beregn checksummen ved at lægge alle cifre sammen.

Hvis checksummen er delelig med 10, er kort-nummeret gyldigt.

Problemdefinition

Skriv en funktion med navnet `cardValidation` der tager som input et 16-cifret kreditkort-nummer som en tekst-streng og returnerer checksummen beregnet som beskrevet ovenfor.

Løsningsskabelon

```
def cardValidation(cardnumber):
  #insert your code
  return checksum
```

Input

`cardnumber`	16-cifret kreditkort-nummer (tekst-streng).

Output

`checksum`	Checksum (heltal).

Eksempel

Forestil dig for eksempel kreditkort-nummeret: `4024007156748096`. Det første ciffer (4) konverteres til 8; det andet ciffer (0) beholdes som det er; det trejde ciffer (2) konverteres til 4; det fjerde ciffer (4) beholdes som det er; og så videre. Checksummen kan så beregnes som:

Kort-nummer:	4	0	2	4	0	0	7	1	5	6	7	4	8	0	9	6
Konverter hvert andet ciffer:	8	↓	4	↓	0	↓	5	↓	1	↓	5	↓	7	↓	9	↓

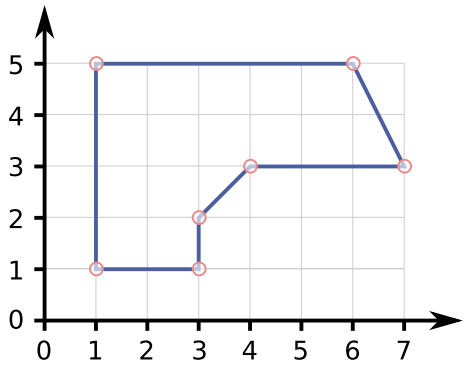
Læg alle cifre sammen: $8+0+4+4+0+0+5+1+1+6+5+4+7+0+9+6 = \underline{60}$

Funktionen skal returnere checksummen 60 (og idet den er delelig med 10 er kort-nummeret gyldigt).

B

Assignment C Perimeter of a polygon

En polygon med n hjørner kan specificeres ved en liste af hjørne-koordinater $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
For eksempel kan polygonen i figuren nedenfor specificeres ved koordinaterne $(1,1), (3,1), (3,2), (4,3), (7,3), (6,5), (1,5$



På baggrund af koordinaterne kan omkredsen (perimeteren) af polygonen beregnes som summen af dens sidelængder vha. følgende formel

$$P = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2} + \sqrt{(x_2-x_3)^2+(y_2-y_3)^2} + \cdots + \sqrt{(x_{n-1}-x_n)^2+(y_{n-1}-y_n)^2} + \sqrt{(x_n-x_1)^2+(y_n-y_1)^2}.$$

Bemærk at formlen for omkredsen indeholder et led (Pythagoras formel) for hver kant i polygonen.

Problemdefinition

Skriv en funktion med navnet `polygonPerimeter` der tager som input to vektorer med henholdsvis x- og y-koordinater, og returnerer omkredsen af polygonen. Funktionen skal fungere for polygoner med et vilkårligt antal hjørner, $n \geq 3$.

Løsningsskabelon

```
def polygonPerimeter(x, y):
  #insert your code
  return P
```

Input

`x, y`	X- og y-koordinater der specificerer en polygon (vektorer med decimaltal).

Output

`P`	Omkreds af polygonen.

Eksempel

Forestil dig følgende input-koordinater som svarer til polygonen i figuren.

$$x = [1,\ 3,\ 3,\ 4,\ 7,\ 6,\ 1], \quad y = [1,\ 1,\ 2,\ 3,\ 3,\ 5,\ 5]$$

Omkredsen kan da beregnes som

$$\begin{aligned} P &= \sqrt{(x_1-x_2)^2+(y_1-y_2)^2} + \sqrt{(x_2-x_3)^2+(y_2-y_3)^2} + \sqrt{(x_3-x_4)^2+(y_3-y_4)^2} + \\ &\quad \sqrt{(x_4-x_5)^2+(y_4-y_5)^2} + \sqrt{(x_5-x_6)^2+(y_5-y_6)^2} + \sqrt{(x_6-x_7)^2+(y_6-y_7)^2} + \sqrt{(x_7-x_1)^2+(y_7-y_1)^2} \\ &= \sqrt{(1-3)^2+(1-1)^2} + \sqrt{(3-3)^2+(1-2)^2} + \sqrt{(3-4)^2+(2-3)^2} + \\ &\quad \sqrt{(4-7)^2+(3-3)^2} + \sqrt{(7-6)^2+(3-5)^2} + \sqrt{(6-1)^2+(5-5)^2} + \sqrt{(1-1)^2+(5-1)^2} \\ &= \sqrt{4} + \sqrt{1} + \sqrt{2} + \sqrt{9} + \sqrt{5} + \sqrt{25} + \sqrt{16} \ = \ 2+1+\sqrt{2}+3+\sqrt{5}+5+4 \ \approx \ \underline{18.650} \end{aligned}$$

C

Assignment D Rotation and scaling

Et punkt (x, y) i et 2-d vektor-rum kan roteres og skaleres om et center-punkt (a, b) ifølge denne matrix/vektor formel:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x-a \\ y-b \end{bmatrix} \cdot s + \begin{bmatrix} a \\ b \end{bmatrix},$$

hvor θ er rotations-vinklen målt i radianer, s er skalerings-faktoren, og (x', y') er de resulterende roterede og skalerede koordinater.

Problemdefinition

Skriv en funktion med navnet `rotateScale` der tager som input en $2 \times N$ matrix der indeholder N par af x- og y-koordinater, en vinkel θ, en skalerings-faktor s, samt et center-punkt (a, b), og returnerer en $2 \times N$ matrix med de roterede og skalerede koordinater. Funktionen skal fungere for et vilkårligt antal koordinater $N \geq 1$.

Løsningsskabelon

```
def rotateScale(coordinates, center, theta, scale):
  #insert your code
  return newCoordinates
```

Input	
`coordinates`	X- og y-koordinater for punkter der skal roteres ($2 \times N$ matrix)
`center`	Center-punkt for rotationen, dvs. (a, b) i formlen (vektor).
`theta`	Rotations-vinkel, dvs. θ i formlen (decimaltal, i radianer).
`scale`	Skaleringsfaktor, dvs. s i formlen (decimaltal).

Output	
`newCoordinates`	X- og y-koordinater for de roterede og skalerede punkter ($2 \times N$ matrix)

Eksempel

Vi vil rotere og skalere de følgende syv punkter $(1,1), (3,1), (3,2), (4,3), (7,3), (6,5), (1,5)$ omkring center-punktet $(a, b) = (3, 1)$ med en vinkel på $\theta = -\frac{\pi}{3}$ radianer og en skalerings-faktor på $s = 2$. Punkterner er samlet i følgende input-matrix:

$$\texttt{coordinates}: \quad \begin{bmatrix} 1 & 3 & 3 & 4 & 7 & 6 & 1 \\ 1 & 1 & 2 & 3 & 3 & 5 & 5 \end{bmatrix}.$$

Efter at have roteret og skaleret hvert punkt ifølge formlen fås følgende resultat (vist afrundet til tre decimaler):

$$\texttt{newCoordinates}: \quad \begin{bmatrix} 1 & 3 & 4.732 & 7.464 & 10.464 & 12.928 & 7.928 \\ 4.464 & 1 & 2 & 1.268 & -3.928 & -0.196 & 8.464 \end{bmatrix},$$

hvor, til eksempel, de roterede koordinater for det sidste (syvende) punkt $(x_7, y_7) = (1, 5)$ er beregnet som:

$$\begin{aligned} \begin{bmatrix} x'_7 \\ y'_7 \end{bmatrix} &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_7 - a \\ y_7 - b \end{bmatrix} \cdot s + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \cos(-\frac{\pi}{3}) & -\sin(-\frac{\pi}{3}) \\ \sin(-\frac{\pi}{3}) & \cos(-\frac{\pi}{3}) \end{bmatrix} \begin{bmatrix} 1-3 \\ 5-1 \end{bmatrix} \cdot 2 + \begin{bmatrix} 3 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} -2 \\ 4 \end{bmatrix} \cdot 2 + \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} (-\frac{1}{2} \cdot 2 + \frac{\sqrt{3}}{2} \cdot 4) \cdot 2 + 3 \\ (\frac{\sqrt{3}}{2} \cdot 2 + \frac{1}{2} \cdot 4) \cdot 2 + 1 \end{bmatrix} \approx \begin{bmatrix} 7.928 \\ 8.464 \end{bmatrix} \end{aligned}$$

D

Assignment E Costliest car within budget

Når du køber en ny bil kan du vælge mellem udstyrsniveauer samt eventuelt tilføje ekstraudstyr.

Grundpris
159 000

Udstyrsniveau	Pris
`Access`	0
`Comfort`	22 000
`Sport`	44 000

Ekstraudstyr	Pris
`Cruise control`	4 000
`Air conditioning`	8 000
`Alloy wheels`	13 000
`Chrome spoiler`	7 000

For at sammensætte en bil skal du betale grundprisen og du skal vælge et udstyrsniveau. Derudover kan du tilføje et vilkårligt antal af ekstraudstyr (hvert ekstraudstyr kan ikke tilføjes mere end en gang).

Vi definerer en bil-specifikation som en streng der indeholder udstyrsniveau og ekstraudstyr skrevet som i tabellerne ovenfor, adskilt af komma og mellemrum, og givet i samme rækkefølge som i tabellerne ovenfor. Et eksempel er følgende specifikations-streng:

`Comfort, Cruise control, Alloy wheels, Chrome spoiler`

Prisen for denne bil vil være: $159\,000 + 22\,000 + 4\,000 + 13\,000 + 7\,000 = \underline{205\,000}$

Problemdefinition

Skriv en funktion med navnet `costliestCar` der tager som input et tal `maxPrice` som specificerer den højest tilladte pris for bilen. Funktionen skal returnere specifikationen for den dyreste bil med en samlet pris der er mindre eller lige maksimumprisen (du kan antage at maksimumprisen er større end grundprisen og at der er en unik dyreste konfiguration). Det kan gøres ved at beregne prisen på samtlige kombinationsmuligheder og så finde den dyreste konfiguration der er inden for budgettet: Med 3 udstyrsniveauer og 4 mulige typer ekstraudstyr er der $3 \cdot 2^4 = 48$ forskellige mulige konfigurationer. Specifikationen skal returneres som en streng som defineret ovenfor.

Løsningsskabelon

```
def costliestCar(maxPrice):
  #insert your code
  return carSpecification
```

Input	
`maxPrice`	Bilens maksimumpris (heltal).

Output	
`carSpecification`	Bil-specifikation (tekst-streng).

Eksempel

Forestil dig en maksimum-pris på 170 500. Når vi fratrækker grundprisen er der 11 500 tilbage. Den dyreste kombination inden for budgettet er udstyrsniveauet `Access` med ekstraudstyret `Cruise control` og `Chrome spoiler`, så outputtet af funktionen skal være:

`Access, Cruise control, Chrome spoiler`

Bilens samlede pris er således $159\,000 + 0 + 4\,000 + 7\,000 = 170\,000$, hvilket ikke overskrider budgettet.

E