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

   https://dtu.codejudge.net/prog-f20/assignments

   under *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 DTU's Onlineeksamen site. Hver assignment skal uploades som en separat .py fil med samme navn som funktionen i opgaven:

   (a) `sequence_cut.py`
   (b) `geometry.py`
   (c) `count_words.py`
   (d) `maximum_pair.py`
   (e) `peak.py`

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

Efter eksamen vil dine løsninger fra DTU Inside 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, dog kan `import`-sætninger inkluderes.
- 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 Sequence cut

Vi vil gerne dele en sekvens af tal i to dele. Sekvensen skal deles ved punktet hvor summen af den første del af sekvensen er nærmest ved halvdelen af summen af den fulde sekvens. Den nærmeste måles som den mindste absolutværdi.

Problemdefinition

Skriv en funktion med navnet `sequence_cut` som tager som input: $\mathbf{x}$ som er en vektor der repræsenterer sekvensen af tal. Funktionen skal returnere en ny sekvens, $\mathbf{y}$, der starter med startelementet af $\mathbf{x}$ og tager elementer from $\mathbf{x}$ i den samme rækkefølge, så summen af $\mathbf{y}$ er nærmest til halvdelen af summen af $\mathbf{x}$. Hvis to dele er lige tætte så skal den længste $\mathbf{y}$ vælges.

Løsningsskabelon

```
def sequence_cut(x):
  #insert your code
  return y
```

Input	
`x`	Vektor med en eller flere tal.

Output	
`y`	Vektor med en eller flere tal.

Eksempel

Betragt sekvensen

$$\mathbf{x} = [2, 3, 1, 1, 4, 5, 23, 6, 3, 40]$$

Summen er 88 og halvdelen af det er 44. Vi betragter den akkummulerede sum af sekvensen

$$\begin{aligned}\tilde{\mathbf{x}} &= [x_1,\ x_1 + x_2,\ x_1 + x_2 + x_3,\ \ldots] \\ &= [2, 5, 6, 7, 11, 16, 39, 45, 48, 88]\end{aligned}$$

Her er 45 den værdi der er nærmest til halvdelen af summen (44). Summen ved 45 svarer til et deling mellem det 8. og 9. element i sekvensen $\mathbf{x}$. Således returneres $\mathbf{y}$:

$$\mathbf{y} = \underline{[2, 3, 1, 1, 4, 5, 23, 6]}$$

A

Assignment B Geometry selection

Vi ønsker at vælge den største to-dimensionale geometriske form med hensyn til areal blandt fire givne former. Arealerne er beregnet som:

$$
\begin{aligned}
A_1 &= ab && \text{Rektangel} \\
A_2 &= \pi r^2 && \text{Cirkel} \\
A_3 &= \frac{1}{2} n r^2 \sin\frac{2\pi}{n} && \text{Indskreven (regulær polygon)} \\
A_4 &= n r^2 \tan\frac{\pi}{n} && \text{Omskreven (regulær polygon)}
\end{aligned}
$$

Problemdefinition

Skriv en funktion med navnet `geometry` som tager fire input: parametre til et rektangel, cirkel, polygon indskrevet i en cirkel og polygon omskrevet om en cirkel. Funktionen skal returnere arealet af den største geometriske form.

Løsningsskabelon

```
def geometry(rectangle, circle, inscribed, circumscribed):
  # insert your code
  return area
```

Input	
`rectangle`	2-element vektor med $[a, b]$ for rektangel.
`circle`	Skalar med radius r for cirkel.
`inscribed`	2-element vektor med $[n, r]$ for n sider og radius r for den indskrevne polygon.
`circumscribed`	2-element vektor med $[n, r]$ for n sider og radius r for den omskrevne polygon.

Output	
`area`	Areal af den geometriske form med det største areal.

Eksempel

Betragt rektanglet $a = 3$ og $b = 5$, en cirkel $r = 2$, en indskreven polygon $n = 4$ og $r = 2,5$, og en omskreven polygon $n = 6$ og $r = 2.2$

$$
\begin{aligned}
A_1 &= ab = 3 \cdot 5 = 15 && \text{Rektangel} \\
A_2 &= \pi r^2 = \pi 2^2 = 4\pi = 12.57 && \text{Cirkel} \\
A_3 &= \frac{1}{2} n r^2 \sin\frac{2\pi}{n} = \frac{1}{2} 4 \cdot 2.5^2 \sin\frac{2\pi}{4} = 12.5 \cdot 1 = 12.5 && \text{Indskreven} \\
A_4 &= n r^2 \tan\frac{\pi}{n} = 6 \cdot 2.2^2 \tan\frac{\pi}{6} = 29.04 \cdot 0.577 = 16.77 && \text{Omskreven}
\end{aligned}
$$

Her har polygonen omkrevet i en cirkel det største areal med 16.77, eller mere præcist 16.7662518, og dette er returneret som arealet.

Assignment C Count frequent word

Givet en tekst med ord separeret af mellemrum vil vi tælle antallet af gange det oftest forekommende ord forekommer. Ord der starter med `no` skal ignoreres.

Problemdefinition

Skriv en funktion med navnet `count_words` der tager en tekst som en streng input. Funktionen skal returnere optællingen af det oftest forekommende ord. Ord der begynder med `no` skal ignoreres. En streng med ingen ord, dvs. en tom streng skal returnere tallet nul. Det kan antages at teksten består af små bogstaver fra `a` til `z` og det eneste andet tegn er mellemrumstegnet.

Løsningsskabelon

```
def count_words(text):
  # insert your code
  return count
```

Input	
`text`	Streng med ord separeret med et mellemrum.

Output	
`count`	Skalar med antallet af gange det oftest forekommen ord forekommer, hvor ord der begynder med `no` er ignoreret.

Eksempel

Betragt teksten

```
none none is here but none is coming here again
```

Her forekommer `none` 3 gange men siden det begynder med `no` skal det ignoreres. De næstoftest forekommende ord er `here` og `is` som begge forekommer to gange, så tallet 2 skal returneres.

C

Assignment D Maximum pair

Givet en matrix ønsker vi at finde to nabo-tal der har den højeste sum. Nabo-tal er defineret som de elementer der er umiddelbare naboer i vandret og lodret retning. Sættet af naboer for matrix-elementet x_{ij} er

$$\mathcal{N}(x_{i,j}) = \{x_{i-1,j}, x_{i+1,j}, x_{i,j-1}, x_{i,j+1}\}$$

Problemdefinition

Skriv en funktion `maximum_pair` som tager en matrix som input og returnerer som output det par af naboer, hvis sum er maximum. Parret skal returneres sorteret så elementet med det laveste indeks er først. Du kan antage at parret af naboer med den højeste sum er unikt.

Løsningsskabelon

```
def maximum_pair(x):
    #insert your code
    return pair
```

Input

`x`	2D-array der repræsenterer en matrix med mindst to elementer.

Output

`pair`	1D-array med to elementer der repræsenterer parret.

Eksempel

Betragt matricen

$$\mathbf{X} = \begin{bmatrix} 2 & 2 & 29 & -4 \\ 1 & -6 & 7 & 20 \\ 9 & -37 & 8 & 15 \end{bmatrix}$$

Nogle af de mulige par er $2 + 29 = 31$ fra den første række, $7 + 20 = 27$ fra den anden række, $29 + 7 = 36$ fra den tredje kolonne og $20 + 15 = 35$ fra den fjerde kolonne. Når vi betragter samtlige mulige par er 36 den maximale værdi, så $\underline{[29, 7]}$ skal returneres.

D

Assignment E Convolution

Givet et signal repræsenteret af vektoren $\mathbf{x}$ og et filter repræsenteret af vektoren $\mathbf{h}$ med M elementer, ønsker vi at finde peak-amplituden af *foldningen*, hvor peak-amplituden er maximum af den absolutte værdi af foldningen. Foldningen er givet som:

$$y[n] = \sum_{m=0}^{M-1} h[m]\, x[n-m]$$

hvor indeksene m og n begynder ved nul, og hvor

$$x[n] = 0 \qquad \text{if } n < 0$$

og hvor $\mathbf{y}$ har den samme længde som $\mathbf{x}$.

Problemdefinition

Skriv en funktion med navnet `peak` som tager en signal-vektor $\mathbf{x}$ og en filter-vektor $\mathbf{h}$, beregner foldningen $\mathbf{y}$ med den samme længde som $\mathbf{x}$ og returnerer maximum af de absolutte værdier af foldningen.

Løsningsskabelon

```
def peak(x, h):
  # insert your code
  return p
```

Input	
`x`	Vektor med signalet $\mathbf{x}$
`h`	Vektor med filtret $\mathbf{h}$

Output	
`p`	Peak-amplitude som maximum af de absolutte værdier af foldningen.

Eksempel

Betragt signalet og filtret givet ved

$$\mathbf{x} = [1, 2, -1, -3, -4] \qquad \mathbf{h} = [0, 1, 3]$$

Foldningen, $\mathbf{y}$, (når der indekseret fra nul) er

$$y[0] = \sum_{m=0}^{M-1} h[m]\, x[0-m] = h_0x_0 + h_1x_{(-1)} + h_2x_{(-2)} = 0\cdot 1 + 1\cdot 0 + 3\cdot 0 = 0$$
$$y[1] = \sum_{m=0}^{M-1} h[m]\, x[1-m] = h_0x_1 + h_1x_0 + h_2x_{(-1)} = 0\cdot 2 + 1\cdot 1 + 3\cdot 0 = 1$$
$$y[2] = \sum_{m=0}^{M-1} h[m]\, x[2-m] = h_0x_2 + h_1x_1 + h_2x_0 = 0\cdot (-1) + 1\cdot 2 + 3\cdot 1 = 5$$
$$y[3] = \sum_{m=0}^{M-1} h[m]\, x[3-m] = h_0x_3 + h_1x_2 + h_2x_1 = 0\cdot (-3) + 1\cdot (-1) + 3\cdot 2 = 5$$
$$y[4] = \sum_{m=0}^{M-1} h[m]\, x[4-m] = h_0x_4 + h_1x_3 + h_2x_2 = 0\cdot (-4) + 1\cdot (-3) + 3\cdot (-1) = -6$$

Hvilket giver den fulde folding

$$\mathbf{y} = [0, 1, 5, 5, -6]$$

Her er peak-amplituden $|-6| = 6$, så $p = \underline{6}$ skal returneres.