Danmarks Tekniske Universitet

Kursusnavn	Introduktion til programmering og databehandling
Kursusnummer	02631, 02633, 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-f18/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) `weightOnPlanet.py`
   (b) `planets.py`
   (c) `inversions.py`
   (d) `polygon.py`
   (e) `medal.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 Space weight

Har du nogensinde tænkt på hvor meget du ville veje på en anden planet? Denne følgende tabel viser vægten på seks planeter, relativt til vægten på jorden.

Nummer	Planet	Relativ vægt (R)
1	Venus	0.91
2	Mars	0.38
3	Jupiter	2.53
4	Saturn	1.06
5	Uranus	0.89
6	Neptune	1.13

For at beregne en persons vægt på en af disse planeter skal du gange personens vægt på jorden med den relative vægt (R) for planeten.

Problemdefinition

Skriv en funktion med navnet `weightOnPlanet` der tager som input vægten af en person på jorden samt et planet-nummer fra tabellen ovenfor. Funktionen skal returnere vægten af personen på den planet.

Løsningsskabelon

```
def weightOnPlanet(weight, planetNumber):
  #insert your code
  return newWeight
```

Input	
`weight`	Vægt på jorden (decimaltal.)
`planetNumber`	Planet-nummer, se tabellen (heltal mellem 1 og 6.)

Output	
`newWeight`	Vægt på planeten (decimaltal.)

Eksempel

Forestil dig en person som vejer 82 kg og rejser til Saturn (planet nummer 4). Hans vægt på saturn vil være:

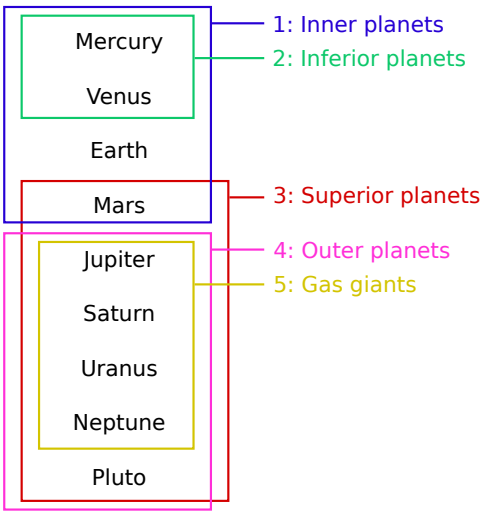
$$82 \cdot 1.06 = \underline{86.92}$$

A

Assignment B Planet classification

Planeterne i solsystemet kan klassificeres som tilhørende en eller flere af følgende kategorier:

1. Inner planets
2. Inferior planets
3. Superior planets
4. Outer planets
5. Gas giants



Figuren viser hvilke planeter der tilhører hver kategori, og vi vil betegne kategorierne ved deres nummer, 1–5

Problemdefinition

Skriv en funktion med navnet `planets` der tager som input en vektor med tal mellem 1 og 5, svarende til en liste af planet-kategorier. Funktionen skal returnere et array af tekst-strenge som indeholder navnene på de planeter der tilhører en eller flere af input-kategorierne. Rækkefølgen af planet-navnene skal være som i figuren, og planet-navnene skal skrives med stort begyndelsesbogstav som i figuren.

Løsningsskabelon

```
def planets(planetCategories):
  #insert your code
  return planetNames
```

Input	
`planetCategories`	Liste af planet-kategori-numre (vektor med heltal 1–5.)

Output	
`planetNames`	Liste af planet-navne (array af tekst-strenge.)

Eksempel

Forestil dig følgene liste af planet-kategorier som input:

4, 5, 2

Det er kategorierne *4: Outer planets*, *5: Gas giants* og *2: Inferior planets*. De planeter der tilhører en eller flere af disse kategorier er (listet i den korrekte rækkefølge):

`Mercury`, `Venus`, `Jupiter`, `Saturn`, `Uranus`, `Neptune`, `Pluto`

Det endelige resultat skal være et array af tekst-strenge som indeholder disse planet-navne.

B

Assignment C 15-puzzle

Målet i 15-puzzle er at flytte 15 brikker nummereret 1–15 på et 4-gange-4 bræt (med en tom plads) således at de kommer til at stå i korrekt numerisk rækkefølge.

Hvis vi er givet en startposition, hvor det tomme felt er i nederste højre hjørne, er spillet kun løsbart hvis antallet af *inversioner* er et lige tal. Hvis vi skriver tallene på pladen ud som en enkelt vektor, række for række, så er antallet af inversioner defineret som antallet af gange et "større" tal efterfølges af et "mindre" tal, dvs. a efterfølges af b hvor $a > b$.

Problemdefinition

Skriv en funktion med navnet `inversions` der tager som input en vektor der indeholder 15 tal (heltal 1–15) svarende til en konfiguration af en spilleplade i et spil 15-puzzle taget række for række. Funktionen skal returnere antallet af inversioner i denne konfiguration.

```
def inversions(board):
  #insert your code
  return inv
```

Input

`board`	Spilleplade (vektor med 15 elementer, tal fra 1–15.)

Output

`inv`	Antal inversioner (heltal).

Eksempel

Forestil dig spillepladen der er vist i figuren, som repræsenteres ved den følgende input vektor:

`board`: 15, 2, 1, 12, 8, 5, 6, 11, 4, 9, 10, 7, 3, 14, 13

Vi kan markere inversionerne (skift fra et større til et mindre tal) med en nedadgående pil, og ikke-inversioner med en opadgående pil.

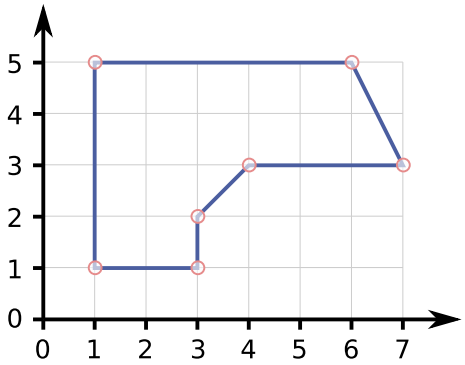
$$15\searrow 2\searrow 1\nearrow 12\searrow 8\searrow 5\nearrow 6\nearrow 11\searrow 4\nearrow 9\nearrow 10\searrow 7\searrow 3\nearrow 14\searrow 13$$

Når vi tæller antallet af inversioner når vi til det endelige resultat: 8.

C

Assignment D Polygon convexity check

En polygon 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 $\{(1,1), (3,1), (3,2), (4,3), (7,3), (6,5), (1,5)\}$.



En polygon siges at være konveks, når intet linje-segment mellem to punkter på polygonens sider nogensinde kan gå uden for polygonen. Vi kan undersøge om en polygon er konveks, ved at gennemløbe hvert knudepunkt i polygonen og tjekke at vi hele tiden "drejer i samme retning."

For det k'te knudepunkt i polygonen kan vi definere følgende krydsprodukt:

$$c_k = (x_k - x_{k-1})(y_{k+1} - y_k) - (x_{k+1} - x_k)(y_k - y_{k-1})$$

hvor vi benytter følgende konvention: $x_0 \equiv x_n$, $y_0 \equiv y_n$, $x_{n+1} \equiv x_1$ og $y_{n+1} \equiv y_1$ (cyklisk indeksering) hvor n betegner antallet af knudepunkter i polygonen. Hvis c_k har samme fortegn for alle værdier af k, er polygonen konveks. I denne opgaver skal vi blot beregne værdierne $c_1, c_2, \ldots, c_n$.

Problemdefinition

Skriv en funktion med navnet `polygon` der tager som input to vektorer med henholdsvis x- og y-koordinater, og returnerer værdierne $c_1, c_2, \ldots, c_n$ som en vektor.

Løsningsskabelon

```
def polygon(x, y):
  #insert your code
  return C
```

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

Output	
`C`	Krydsprodukt for hvert knudepunkt (vektor med decimaltal.)

Eksempel

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

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

Det første krydsprodukt, c_1, kan beregnes som:

$$\begin{aligned} c_1 &= (x_1 - x_0)(y_2 - y_1) - (x_2 - x_1)(y_1 - y_0) \\ &= (x_1 - x_n)(y_2 - y_1) - (x_2 - x_1)(y_1 - y_n) \\ &= (1-1)(1-1) - (3-1)(1-5) = 8 \end{aligned}$$

På samme måde kan resten af krydsprodukterne beregnes, hvilket giver følgende resultat:

$$\texttt{C}\text{: } [8,\ 2,\ -1,\ -3,\ 6,\ 10,\ 20]$$

D

Assignment E Sports medals

I en sports-konkurrence for børn bliver medaljerne tildelt til deltagerne på baggrund af hvor mange point de opnår samt deres aldersgruppe ifølge denne tabel:

Medal	B (Baby)	P (Preschooler)	G (Gradeschooler)	T (Teen)
`gold`	10–20	15–20	17–20	19–20
`silver`	5–9	10–14	14–16	17–18
`bronze`	0–4	5–9	11–13	15–16
ingen medalje	—	0–4	0–10	0–14

Antal point der kræves for at vinde de forskellige medaljer.

Problemdefinition

Skriv en funktion med navnet `medal` der tager som input en streng der repræsenterer en af de fire aldersgrupper, samt en pointscore. Funktionen skal kun se på første bogstav i input-strengen der angiver aldersgruppen (dvs. `B`, `P`, `G`, eller `T`). Funktionen skal returnere en tekst-streng der afhænger af hvilken medalje (om nogen) deltageren har vundet.

Hvis deltageren har vundet en medalje, skal returværdien være på følgende form:

`You got` <pts> `points and won a` <mdl> `medal.`

Ellers, hvis deltageren ikke har vundet en medalje, skal returværdien være på følgende form:

`You got` <pts> `points.`

I disse returværdier skal udtrykkene <pts> og <mdl> erstattes af henholdsvis antallet af point og typen af medalje (`gold`, `silver`, eller `bronze`).

Løsningsskabelon

```
def medal(ageGroup, points):
  #insert your code
  return message
```

Input	
`ageGroup`	Aldersgruppe (tekst-streng.)
`points`	Antal point (heltal mellem 0 og 20.)

Output	
`message`	Besked vedrørende point og medalje (tekst-streng).

Eksempel

Forestil dig følgende input:

`ageGroup: Preschooler` `points: 12.`

Ifølge tabellen skal en "Preschooler" med 10–14 point have en sølv-medalje (silver), så det endelige resultat skal være tekst-strengen

`You got 12 points and won a silver medal.`

E