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) `nBest.py`
   (b) `timeDilation.py`
   (c) `derangements.py`
   (d) `industry.py`
   (e) `taskDispatch.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 N-best scores

Du arbejder på et anbefalings-system til en webshop, som skal anbefale N produkter til kunden. Du får givet en liste med $K > N$ match-scorer (decimaltal) som indikerer hvor interesseret kunden er i hvert af K produkter.

Problemdefinition

Skriv en funktion med navnet `nBest` der tager som input en vektor med længden K samt et heltal N, og returnerer en vektor der indeholder de N største værdier fra den originale vektor i samme rækkefølge som de optræder. Du kan antage at alle værdier er unikke (ingen duplikater).

Løsningsskabelon

```
def nBest(allValues, N):
  #insert your code
  return bestValues
```

Input	
`allValues`	Match-scorer (vektor med decimaltal).
`N`	Antal produkter der skal anbefales (positivt heltal).

Output	
`bestValues`	N bedste match-scorer (vektor med decimaltal).

Eksempel

Forestil dig følgende input match-scorer:

$$[12.5,\ 13.6,\ -9.1,\ 17.5,\ 15.3,\ 10.5]$$

Hvis det ønskede antal anbefalinger er $N = 3$ skal resultatet være vektoren:

$$[13.6,\ 17.5,\ 15.3]$$

A

Assignment B Time dilation

Ifølge relativitetsteorien vil tiden gå langsommere på en rumskib i høj fart end på jorden, ifølge følgende formel:

$$T(t,v) = \frac{t}{\sqrt{1-\frac{v^2}{c^2}}},$$

hvor T er tiden på jorden, t er tiden på rumskibet, v er rumskibets hastighed (relativt til jorden) og c er lysets hastighed. Vi benytter $c = 299\,792\,458$ [meter/sekund].

Problemdefinition

Skriv en funktion med navnet `timeDilation` der tager som input en vektor med N_t værdier for tiden (i sekunder) på rumskibet samt en vektor med N_v værdier for hastigheden (i meter/sekund) og beregner en matrix med størrelse $N_t \times N_v$ som indeholder de tilsvarende tider (i sekunder) på jorden for hver kombination af de givne tider og hastigheder for rumskibet.

Løsningsskabelon

```
def timeDilation(t, v):
  #insert your code
  return T
```

Input	
`t`	Tid på rumskibet, t (vektor).
`v`	Hastighed af rumskibet, v (vektor).

Output	
`T`	Tid på jorden, T (matrix).

Eksempel

Antag at vi får oplyst følgende tider og hastigheder for rumskibet:

$$t = [5,\ 10.5,\ 16], \qquad v = [1.1 \cdot 10^8,\ 2.2 \cdot 10^8]$$

Vi har således $N_t = 3$ and $N_v = 2$, og den resulterende matrix skal da være:

$$\overline{\overline{T}} = \begin{bmatrix} T(t_1,v_1) & \cdots & T(t_1,v_{N_v}) \\ \vdots & & \vdots \\ T(t_{N_t},v_1) & \cdots & T(t_{N_t},v_{N_v}) \end{bmatrix} = \begin{bmatrix} 5.375 & 7.360 \\ 11.287 & 15.457 \\ 17.200 & 23.553 \end{bmatrix}$$

hvor, for eksempel, element (3,1) er beregnet som:

$$T(t_3,v_1) = \frac{16}{\sqrt{1-\frac{(1.1\cdot 10^8)^2}{299\,792\,458^2}}} \approx 17.200,$$

Assignment C Derangements

Indenfor kombinatorik er en "derangement" en permutation af et sæt således at intet element står på sin oprindelige position. Antallet af mulige derangements af et set med størrelse n er givet ved:

$$k = \text{round}\left(\frac{n!}{e}\right),$$

hvor funktionen round$(\cdot)$ returnerer det nærmeste heltal, $n!$ er fakultet-funktionen af n, og e er den matematiske konstant kendt som Eulers tal.

Problemdefinition

Skriv en funktion med navnet `derangements` der tager som input størrelsen af et sæt og beregner som output antallet af derangements ifølge ovenstående formel.

Løsningsskabelon

```
def derangements(n):
  #insert your code
  return k
```

Input	
`n`	Størrelsen af et sæt (positivt heltal).

Output	
`k`	Antal derangements (positivt heltal).

Eksempel

Hvis vi er givet et sæt med størrelse $n = 5$ kan antallet af mulige derangements beregnes som:

$$k = \text{round}\left(\frac{5!}{e}\right) = \text{round}\left(\frac{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{e}\right) = 44$$

C

Assignment D Industrial classification

En virksomhed kan klassificeres som en af følgende industrier.

Kode	Industri-navn
0000–0999	`Agriculture`
1000–1499	`Mining`
1500–1999	`Construction`
2000–3999	`Manufacturing`
4000–4999	`Transportation`
5000–5999	`Trade`
6000–6999	`Finance`
7000–8999	`Services`
9000–9999	`Public`

Problemdefinition

Skriv en funktion med navnet `industry` der tager som input en tekst-streng som repræsenterer en industri-kode og returnerer det tilsvarende industri-navn som en tekst streng (skrevet præcis som i tabellen ovenfor). Input indeholder 4 karakterer som kan være cifre (0–9) eller bogstavet `x`. Bogstavet `x` fungerer som et "wild card" som kan stå for et vilkårligt ciffer. Du kan antage at den angivne kode altid matcher præcis et af kode-intervallerne.

Løsningsskabelon

```
def industry(industryCode):
  #insert your code
  return industryName
```

Input

`industryCode`

Output

`industryName`

Eksempel

Forestil dig input-strengen `35xx`. Da de to `x`-karakterer kan repræsentere ethvert ciffer kan dette input repræsentere et tal mellem 3500 og 3599. Da disse koder ligger inden for 2000–3999: Manufacturing, skal strengen `Manufacturing` returneres.

D

Assignment E Task dispatch

Du arbejder på et produktions-system hvor et antal opgaver skal behandles på et antal processerings-enheder. Hver opgave kræver et vist antal resurser, og hver processerings-enhed har et vist antal resurser til rådighed. Du skal lave et opgave-fordelings-system som placerer hver opgave på en processerings-enhed under hensyntagen til antallet af resurser. Opgaverne skal fordeles via følgende algoritme:

Lad $t(n)$ angive antallet af resurser krævet for opgave n.
Lad $u(k)$ angive antallet af resurser til rådighed på processerings-enhed k.
Lad $a(n)$ angive nummeret på den processering-enhed som opgave n placeres.
Gennemløb *alle processerings-enheder (lad k angive processerings-enheden).*
  Gennemløb *alle opgaver som endnu ikke er placeret (lad n angive opgaven).*
    Hvis *der er plads på processerings-enhed k til opgave n, dvs. hvis* $u(k) \geq t(n)$.
      Placer opgave n på processerings-enhed k: $a(n) \leftarrow k$.
      Opdater antallet af resurser på processerings-enhed k: $u(k) \leftarrow u(k) - t(n)$.

Problemdefinition

Skriv en funktion med navnet `taskDispatch` der tager som input en vektor der indeholder antallet af krævede resurser for N opgaver samt en vektor der indeholder antallet af resurser til rådighed for K processerings-enheder. Funktionen skal returnere en vektor med længde N som indeholder nummeret på den processerings-enhed (tal mellem 1 og K) hvor hver opgave er placeret ifølge ovenstående algoritme. Opgaver som ikke bliver placeret på nogen processerings-enhed (pga. mangel på resurser) skal markeres med nul.

Løsningsskabelon

```
def taskDispatch(tasks, units):
  #insert your code
  return assignment
```

Input	
`tasks`	Antal resurser krævet for hver opgave (vektor).
`units`	Det totale antal resurser til rådighed for hver processerings-enhed (vektor).

Output	
`assignment`	Placering (processerings-enhedens nummer) for hver opgave (vektor).

Eksempel

Forestil dig at vi har 6 opgaver og to processerings-enheder med følgende antal resurser krævet/til rådighed:

$$t = [5,\ 7,\ 4,\ 2,\ 3,\ 5], \quad u = [11,\ 13].$$

Den første processeringsenhed har $u(1) = 11$ resurser. Vi placerer her den første opgave og opdaterer $u(1) = 11 - 5 = 6$. Der er ikke plads til den anden opgave, så vi går videre og placerer den tredje opgave og opdaterer $u(1) = 6 - 4 = 2$. Der er også plads til den fjerde opgave så vi placerer den og opdaterer $u(1) = 2 - 2 = 0$. Der er ikke plads til nogen af de følgende opgaver, så vi går videre til den anden processerings-enhed. Den første opgave er allerede placeres, så vi placerer den anden opgaver og opdaterer $u(2) = 13 - 7 = 6$. Den tredje og fjerde opgave er allerede placeret, så vi går videre og placerer den femte opgave og opdaterer $u(2) = 6 - 3 = 3$. Der er ikke plads til den sjette opgave, så den bliver ikke placeret på nogen processerings-enhed, og algorithm afsluttes. Den endelige opgave-placering skal således være:

$$a = [1,\ 2,\ 1,\ 1,\ 2,\ 0].$$