Danmarks Tekniske Universitet

Kursusnavn	Introduktion til programmering og databehandling Programmering og databehandling (andet programmeringssprog)
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) `bookPages.py`

   (b) `loadBalance.py`

   (c) `nearestColor.py`

   (d) `climbCategorization.py`

   (e) `romanToValue.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 Pages in a booklet

Antallet af sider i et simpelt foldet hæfte er altid et multiplum af 4 (dvs. 4, 8, 12, 16, 20, ...). Hvis antallet af sider af det indhold du har til folderen ikke er et multiplum af 4, kan du være nødsaget til at have op til 3 tomme sider.

Problemdefinition

Skriv en funktion med navnet `bookPages` der tager som input antallet af sider der er i det indhold du har til folderen og returnerer antallet af sider (et multiplum af 4) i den mindste folder som kan rumme dit indhold.

Løsningsskabelon

```
def bookPages(pagesContent):
  #insert your code
  return pagesBooklet
```

Input	
`pagesContent`	Antal siders indhold (heltal).

Output	
`pagesBooklet`	Antal sider i folder (heltal).

Eksempel

Hvis, for eksempel, du har 17 siders indhold, vil den midste folder som kan rumme dit indhold have 20 sider. Derfor skal funktionen, hvis den modtager 17 som input, returnere tallet 20.

A

Assignment B Load balancing

Du har et antal forskellige workloads (computerprogrammer) som du vil køre på to computere. For hvert workload kender du den forventede køretid, og nu ønsker du at fordele dine workloads så ligeligt som muligt på de to computere. Lad os sige at antallet af workloads er N. Du beslutter at køre de første k workloads på den første computer og de resterende $N - k$ workloads på den anden computer.

Problemdefinition

Skriv en funktion med navnet `loadBalance` som tager som input en vektor indeholdende den forventede køretid for N workloads ($N \geq 2$). Funktionen skal returnere tallet k som opdeler listen af workloads så ligeligt som muligt, dvs. således at den absolutte værdi af forskellen mellem den totale forventede køretid af de første k og de sidste $N - k$ workloads er så lille som muligt. Hvis mere end en opdeling giver samme laveste absolutte forskel skal du returnere den lavest værdi af k.

Løsningsskabelon

```
def loadBalance(runtime):
  #insert your code
  return k
```

Input	
`runtime`	Køretid for N workloads (vektor)

Output	
`k`	Antal workloads der skal køres på computer 1 (heltal)

Eksempel

Forestil dig følgende vektor af køretider: [5, 2.5, 17, 1.5, 22, 3.5]. Vi kan nu beregne summen af køretiden for de to computere og den absolutte forskel for forskellige værdier af k:

k	Computer 1	Computer 2	Absolut forskel
1	5	$2.5 + 17 + 1.5 + 22 + 3.5 = 46.5$	$\|46.5 - 5\| = 41.5$
2	$5 + 2.5 = 7.5$	$17 + 1.5 + 22 + 3.5 = 44$	$\|44 - 7.5\| = 36.5$
3	$5 + 2.5 + 17 = 24.5$	$1.5 + 22 + 3.5 = 27$	$\|27 - 24.5\| = 2.5$
4	$5 + 2.5 + 17 + 1.5 = 26$	$22 + 3.5 = 25.5$	$\|25.5 - 26\| = 0.5$
5	$5 + 2.5 + 17 + 1.5 + 22 = 48$	3.5	$\|3.5 - 48\| = 44.5$

Heraf ses det at $k = 4$ giver den bedste balance (mindste absolutte forskel), og funktionen skal derfor returnere værdien 4.

B

Assignment C Nearest color

En farve kan repræsenteres af tre tal, r, g og b, som svarer til mængden af rødt, grønt og blåt lys. I denne opgave antages det at tallene er angivet i procent, dvs. ligger mellem 0 og 100. Den følgende tabel oplister navnene og rgb-værdierne på nogle forskellige farver:

	White	Grey	Black	Red	Maroon	Yellow	Olive	Lime	Green	Aqua	Teal	Blue	Navy	Fuchsia	Purple
r	100	50	0	100	50	100	50	0	0	0	0	0	0	100	50
g	100	50	0	0	0	100	50	100	50	100	50	0	0	0	0
b	100	50	0	0	0	0	0	0	0	100	50	100	50	100	50

Vi definerer afstanden mellem to farver (r_1, g_1, b_1) og (r_2, g_2, b_2) som den maksimale absolutte forskel:

$$D = \max\Big(|r_2 - r_1|,\ |g_2 - g_1|,\ |b_2 - b_1|\Big) \tag{1}$$

Problemdefinition

Skriv en funktion med navnet `nearestColor` som tager en værdi for r, g og b som input og returnerer navnet på den nærmeste farve som en streng (skrevet præcis som i tabellen ovenfor.) Den nærmeste farve defineres som den farve i tabellen til hvilken afstanden er kortest. Hvis to eller flere farver i tabellen har samme korteste afstand skal navnet på den farve der står først i tabellen returneres.

Løsningsskabelon

```
def nearestColor(r, g, b):
  #insert your code
  return colorName
```

Input

`r`, `g`, `b`	Værdi af r, g og b (decimaltal)

Output

`colorName`	Navn på nærmeste farve (streng).

Eksempel

Hvis input er givet ved $r = 75$, $g = 0$ og $b = 0$ kan afstanden til de 15 farver i tabellen beregnes til:

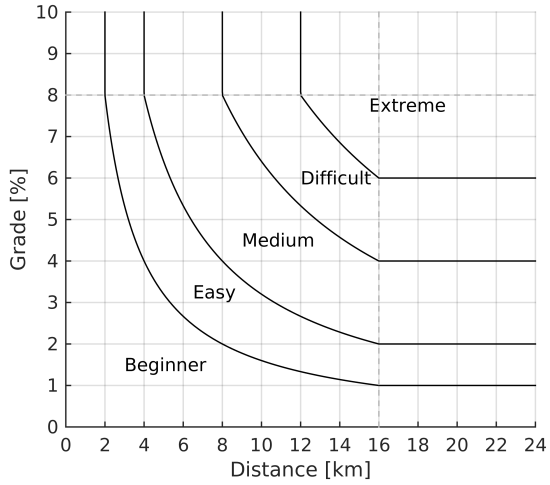
	White	Grey	Black	Red	Maroon	Yellow	Olive	Lime	Green	Aqua	Teal	Blue	Navy	Fuchsia	Purple
D	100	50	75	25	25	100	50	100	75	100	75	100	75	100	50

Da afstanden til farverne Red og Maroon er mindst, og da Red står først i tabellen, skal strengen `Red` returneres.

C

Assignment D Mountain climb categorization

En bjerg-klatretur kan kategoriseres som Beginner, Easy, Medium, Difficult eller Extreme på baggrund af distancen, D (i kilometer) og stigningen, G, ("Grade": gennemsnitlig stigning i procent) ifølge nedenstående figur og tabel.



	`Beginner`	`Easy`	`Medium`	`Difficult`	`Extreme`
Hvis $G > 8$:	$D < 2$	$2 \leq D < 4$	$4 \leq D < 8$	$8 \leq D < 12$	$12 \leq D$
Hvis $D > 16$:	$G < 1$	$1 \leq G < 2$	$2 \leq G < 4$	$4 \leq G < 6$	$6 \leq G$
Ellers:	$DG < 16$	$16 \leq DG < 32$	$32 \leq DG < 64$	$64 \leq DG < 96$	$96 \leq DG$

Problemdefinition

Skriv en funktion med navnet `climbCategorization` der tager distance og grade som input og returnerer kategoriseringen af klatreturen som en streng (skrevet præcis som ovenfor).

Løsningsskabelon

```
def climbCategorization(distance, grade):
  #insert your code
  return categoryName
```

Input

`distance`	Distancen i kilometer (decimaltal)
`grade`	Stigningen i procent (decimaltal)

Output

`categoryName`	Navn på kategoriseringen af bjerg-klatreturen (streng).

Eksempel

Hvis distancen og stigningen er givet ved $D = 8$ og $G = 6$ har vi hverken at $G > 8$ eller $D > 16$. Vi beregner da $DG = 6 \cdot 6 = 48$ og da $32 \leq DG < 64$ skal strengen `Medium` returneres.

D

Assignment E Roman numerals

I det romerske talsystem kan et tal skrives som en sekvens af symboler (bogstaver) som svarer til forskellige værdier.

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Problemdefinition

Skriv en funktion med navnet `romanToValue` der tager et romertal (streng) som input og beregner dets værdi. Funktionen skal benytte følgende algoritme:

1. Omdan hvert symbol til den værdi den repræsenterer.
2. Hold løbende styr på den total værdi samt den maksimale symbol-værdi der er set indtil videre (begge initialiseret til nul). Gennemløb symbolerne et efter et startende fra højre:
   - Hvis symbol-værdien er højere eller lig med maksimum, så adder den til total-værdien og opdater maksimum.
   - Hvis symbol-værdien er mindre end maksimum, så subtraher den fra total-værdien.

Løsningsskabelon

```
def romanToValue(roman):
  #insert your code
  return value
```

Input

`roman`	Romertal (streng)

Output

`value`	Numerisk værdi af romertalet (heltal).

Eksempel

Forestil dig følgende input: `XCIV`. Symbol-værdierne er: 10, 100, 1, 5.

Til at begynde med er maksimum og total-værdien sat til nul. Når vi begynder fra højre er værdien af første symbol lig 5, hvilket adderes til total-værdien som nu er 5 og det nye maksimum er 5. Den næste værdi er 1, som trækkes fra totalen som nu er 4. Herefter kommer 100 som adderes til totalen som nu er 104 og det nye maksimum er 100. Til sidst er 10 som trækkes fra totalen som nu er 94, hvilket er det endelige resultat.

E