Introduktion till programmering i Python

DAT455 (Chalmers), DIT001 GU)
Tentamen 2020-08-28 kl 8:30-10:30 på distans
Examinator Aarne Ranta, email aarne@chalmers.se

Tentan består av tre frågor. **Du måste svara på alla frågor**. Skalan är G/U. Du måste få minst 50% rätt för varje fråga för sig för att få ett G.

Svaret ges i form av en fil som ska heta `python_exam.py` och kunna importeras i Python3. Filen ska innehålla funktioner som specificeras i frågorna. Filen kan också innehålla kommentarer som Python3 ignorerar men lärarna kan läsa. Filen får inte importera några bibliotek.

Alla hjälpmedel är tillåtna, men man får inte kommunicera med andra människor under tentans gång, med undantag av email till aarne@chalmers.se ifall någonting är oklart.

Innan du börjar med frågorna: skriv följande kommentarer överst i din fil

```
# Jag bekräftar härmed att jag inte kommunicerar med andra personer än kursens lärare under tentans gång.

# Jag är medveten om att fusk i tentan kan leda till disciplinåtgärder.
```

Fråga 1. Interaktiva program: biobiljetter

Skriv ett interaktivt program där man köper biobiljetter. Programmet ska fråga antalet gäster, hur många som är barn, och vilken föreställning man vill ha. Efter att ha fått svar till dessa frågor beräknar programmet det totala priset enligt följande:

- en normal biljett kostar 100 kr
- för ungdomar under 18 år är priset 50 kr
- om föreställningen börjar före kl 18 får man 10% rabatt på dessa priser

Programmet ska vara en funktion som heter `movieTickets()`. Nedan är ett exempel på hur en körning ska se ut?

```
>>> movieTickets()
Hur många biljetter vill du köpa? 5
Hur många av er är under 18 år? 3
Vilken föreställning (ange klockslag i hela timmar)? 17
Biljetterna kostar sammanlagt 315 kr.
```

Fråga 2. Analys av textfiler: varna för långa rader

Rekommendationerna i PEP-8 säger att en rad i en Python-fil får vara maximalt 79 tecken lång. Skriv en funktion pepLineLength(filename) som gör följande:

- för varje rad som är längre en 79 tecken, generera en varning som anger radnummer och antalet tecken. **Obs**. i varningarna ska du ange radnummer som motsvarar till numrering som börjar från 1, inte från 0.
- avsluta med en sammanfattning som anger antalet för långa rader

Nedan är ett exempel på körning av funktionen.

```
>>> pepLineLength('divisibility.py')
line 8 too long: 91
line 10 too long: 81
2 lines are too long
```

Fråga 3. Objekt: träd och rekursion

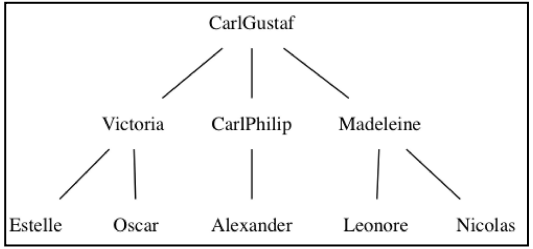
Träd (i datalogisk bemärkelse) kan definieras som en klass med koden bredvid. Kopiera den här koden till din fil och definiera sedan tre saker:

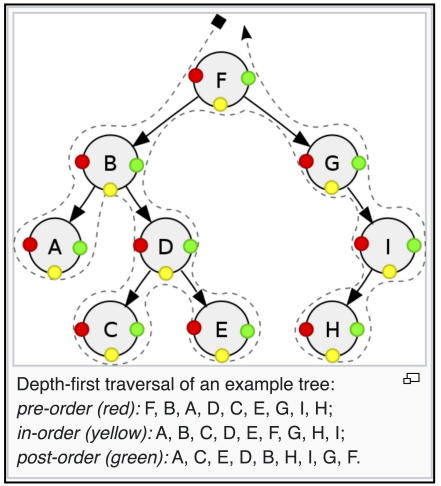
- variabeln `royal`, som representerar släktträdet av Sveriges kungliga familj år 2016 som ett objekt av klass Tree; se bilden
- funktionen `preorder(tree)`, som bygger en lista av alla noder i ett träd så att toppnoden alltid kommer före listorna för delträden. Bilden längst till höger (från Wikipedia) illustrerar begreppet
- funktionen `postorder(tree)`, som bygger en lista av alla noder i ett träd så att toppnoden alltid kommer efter listorna för delträden

Nedersta bilden visar vad dessa funktioner ska returnera för den kungliga familjen.

```
class Tree:
    def __init__(self,node,trees):
        self.root = node
        self.subtrees = trees

    def getParts(self):
        return self.root, self.subtrees
```





Depth-first traversal of an example tree:
pre-order (red): F, B, A, D, C, E, G, I, H;
in-order (yellow): A, B, C, D, E, F, G, H, I;
post-order (green): A, C, E, D, B, H, I, G, F.

```
>>> preorder(royal)
['CarlGustaf', 'Victoria', 'Estelle', 'Oscar', 'CarlPhilip',
'Alexander', 'Madeleine', 'Leonore', 'Nicolas']
>>> postorder(royal)
['Estelle', 'Oscar', 'Victoria', 'Alexander', 'CarlPhilip',
'Leonore', 'Nicolas', 'Madeleine', 'CarlGustaf']
```