TDP002 - Datortenta (DAT1)

2012-10-25

Regler

- Student får lämna salen tidigast en timme efter tentans start.
- Vid toalettbesök eller rökpaus ska pauslista utanför salen fyllas i.
- All form av kontakt mellan studenter under tentans gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av tentavakt i samband med tentans start samt under tentans gång.
- Frågor om specifika uppgifter eller om tentan i stort ska ställas via tentasystemet.
- Systemfrågor kan ställas till assistent i sal.
- Ingen uppgift rättas efter tentatidens slut.
- Ingen kompletteringsmöjlighet ges de sista tio minuterna.
- Så länge en praktisk uppgift inte har betyget U eller tilldelats maxpoäng för uppgiften kan den kompletteras till högre poäng.
- En teoretisk uppgift kan endast kompletteras efter förfrågan från rättare.
- Inga elektroniska hjälpmedel får medtas. Mobiltelefon ska vara avstängd och ligga i jacka eller väska.
- Inga ytterkläder eller väskor vid skrivplatsen.

Antal uppgifter	9
Totalt antal poäng	20
Hjälpmedel	En pythonbok (tex Learning Python 4 ed.) En A4-sida med egna anteckningar

Betygssättning

Tentan är uppdelad i två delar, en teoretisk och en praktisk. Delarna är värda 9 respektive 11 poäng och för godkänt betyg krävs minst fyra poäng per del. Poäng som krävs för de olika betygen kan ses i tabell 1.

Poängsumma	**Betyg**
1 – 9	U
10 – 12	3
13 – 15	4
16 – 20	5

Tabell 1: Poängfördelning för betygssättning

Bonus från labserien

Avklarade deadlines i kursen ger bonus i form av tillgodoräknade uppgifter i den praktiska delen. Denna bonus ges endast under den första ordinarie tentan i samband med kursen. För denna tentamen får student uppgifter tillgodoräknade enligt tabell 2. Vi kontrollerar labstatus under tentans gång och sätter aktuell uppgift som godkänd. Tentasystemet uppmärksammar student när ändringen sker.

Antal avklarade deadlines	**Tillgodoräknad uppgift**
2 – 3	6
4 – 5	7
6 – 7	6 och 7

Tabell 2: Tillgodoräknade av uppgifter

Information

Inloggning

Logga in på tentakontot med följande användaruppgifter:

```
Användarnamn: examx
Lösenord:     kluring1
```

Följ menyvalen så långt det går tills du ska mata in ett engångslösenord. Tag fram ditt LiU-kort och visa det för tentavakten för att få detta lösenord.

När du är inloggad är det viktigt att du startar tentaklienten genom att högerklicka på bakgrunden och välja “tentaklient” i menyn.

Avslutning

Tryck på knappen märkt “exit” i menyn längst nere på skärmen och välj ok. Vänta ett tag och tryck sedan på knappen “Avsluta tentamen” när det är möjligt. När detta är gjort är det omöjligt att logga in igen.

Teoretisk del

1. Sökning

   (a) Förklara skillnaden mellan linjärsökning och binärsökning, gärna med hjälp av ett exempel. [1 p]

   Svar:
   Linjärsökning är när man jämför sitt värde med varje element i sekvensen tills man hittar värdet. Binärsökning sker istället genom att halvera sökmängden för varje jämförelse. För att söka efter en person i en telefonkatalog är det betydligt lättare att slå upp katalogen på mitten och sedan bara söka i den aktuella halvan istället för att söka sida för sida.

   (b) Vad krävs av den sekvens av data man söker i för att man ska kunna använda binärsökning? [1 p]

   Svar:
   Det som krävs är att datamängden är sorterad samt direkt addresserbar (det går att ta ut ett specifikt index). (Sorterad räcker för poäng)

2. Nämn tre egenskaper en variabel alltid har? [1 p]

   Svar:
   Namn, Datatyp, Värde

3. Vad menas med begreppet styrsats (control statement)? [1 p]

   Svar:
   En styrsats är en sats för att styra programflödet med hjälp av val eller upprepning (se PL 8.1)

4. För varje av följande påståenden ska du svara om det är sant (S) eller falskt (F). För 1-2 korrekta svar ges ett poäng och för 3-4 korrekta svar ges 2 poäng på uppgiften. [2 p]

   (a) Ett nyckelord (keyword) och ett reserverat ord (reserved word) är samma sak.

   Svar:
   F: Ett nyckelord är endast speciellt om det används på ett visst sätt, ett reserverat ord får bara användas på specifika sätt. (se PL 5.2.3)

   (b) Procedurer och funktioner anropas på samma sätt.

   Svar:
   F: funktioner returnerar ett värde och kan (ska) vara del av ett uttryck. Procedurer anropas som egna satser. (PL 9.2.5)

(c) Alla imperativa språk har en switch-sats (eller liknande struktur).

Svar:
F: tex saknar Python en switch-sats

(d) KISS-regeln förespråkar enkelhet.

Svar:
S: **Keep It Simple Stupid**
"The KISS principle states that most systems work best if they are kept simple rather than made complex, therefore simplicity should be a key goal in design and unnecessary complexity should be avoided."
`http://en.wikipedia.org/wiki/KISS_principle`

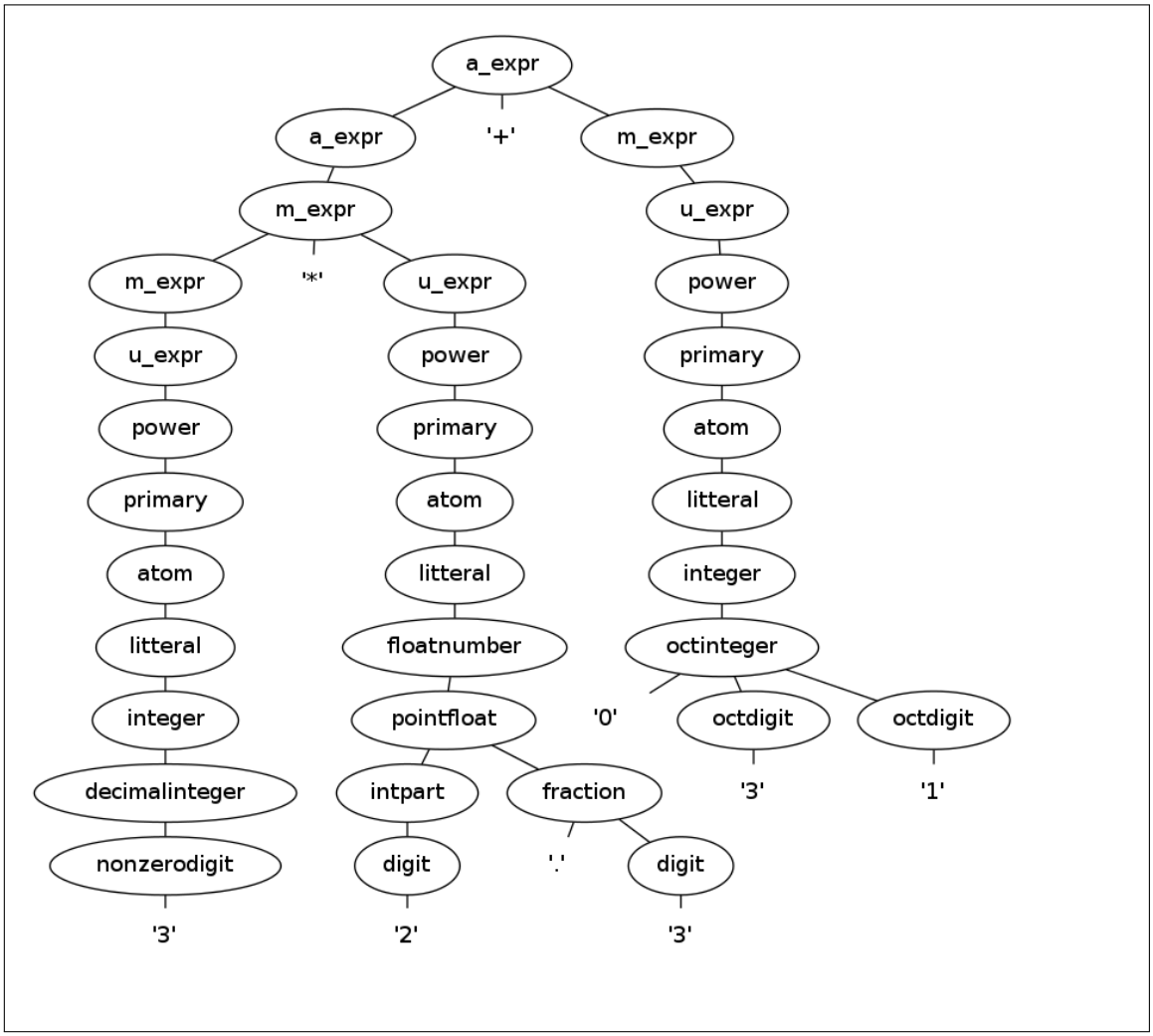
5. Denna uppgift går bra att lämna in på papper för rättning i efterhand. Räck upp handen för att få ett omslag om du vill göra detta. [3 p]

Använd grammatiken i bilaga 1 för att skapa ett parse-träd för uttrycket i Kodruta 1.

```
3*2.3+031
```

Kodruta 1: parse-uttryck

Svar:
Den inledande nollan visar att det är ett oktalt tal och därför måste octinteger användas. 1p avdrag för decimalinteger. Trädet visar det viktiga i parsningen, det går dock bra att börja ännu längre upp i strukturen.


a_expr
a_expr
'+'
m_expr
m_expr
u_expr
m_expr
'*'
u_expr
power
u_expr
power
primary
power
primary
atom
primary
atom
litteral
atom
litteral
integer
litteral
floatnumber
octinteger
integer
pointfloat
'0'
octdigit
octdigit
decimalinteger
intpart
fraction
'3'
'1'
nonzerodigit
digit
'.'
digit
'3'
'2'
'3'

Praktisk del

6. Följande uppgift kan tillgodoräknas med bonus från labserien. [1 p]

   På filen *given_files/comp.py* finns en funktion *is_prime* som kontrollerar om ett tal är ett primtal. Ändra i filen så att variabeln primes innehåller alla primtal i intervallet $[1, 1000]$. Detta ska lösas med ett list-comprehension.

Svar:

```
primes = [x for x in range(1,1000) if is_prime(x)]
```

7. Följande uppgift kan tillgodoräknas med bonus från labserien. [2 p]

   I filen *given_files/accumulate.py* saknas en funktion *accumulate*. accumulate ska ta tre parametrar. En funktion *fn*, en sekvens av data *seq* samt ett startvärde *val*. Resultatet från accumulate ska vara resultatet man får om man först applicerar fn på första elementet i sekvensen och startvärdet och därefter successivt applicerar fn på resultatet av den tidigare beräkningen och nästa element i sekvensen. Skapa accumulate så att det givna programmet fungerar och ger de resultat som står som kommentarer i koden.

Svar:

```
#! /usr/bin/env python3
#-*- coding:utf-8 -*-

def accumulate(fn, seq, val):
    cal = fn(seq[0], val)
    if len(seq) == 1:
        return cal
    return accumulate(fn, seq[1:], cal)

def accumulate2(fn, seq, val):
    res = fn(seq[0], val)
    for v in seq[1:]:
        res = fn(v, res)
    return res

from functools import reduce as accumulate3



mult = lambda x,y: x*y
plus = lambda x,y: x+y

print(accumulate(mult, range(1,6), 1)) #should be 120
print(accumulate(plus, range(1,6), 0)) #should be 15
```

8. Uppdelningen i deluppgifter som följer är endast som grund för betygssättning, det räcker att skicka in en lösning som svar på hela uppgiften som då kan ge fem poäng.

(a) På filen *given_files/befolkning.csv* finns befolkningsdata för alla svenska kommuner för första kvartalet 2012 från SCB. Varje rad i filen har följande kolumner separerade med kommatecken: [3 p]

1. Kommun
2. Befolkningsmängd
3. Antal levande födda
4. Antal döda
5. Antal inflyttade
6. Antal utflyttade

Din uppgift är att skriva ett program som beräknar förändringen i befolkningsmängd över första kvartalet 2012 utifrån givna data, både i antal och procentuell ökning samt skriver ut en tabell sorterad efter befolkningsmängd enligt exempel 1.

```
Kommun          Befolkning  Ökning  Procentuell ökning
Stockholm           871952    3691      0.42%
Göteborg            522275     648      0.12%
Malmö               305033    1122      0.37%
Uppsala             200274    -467     -0.23%
...
Bjurholm              2428       3      0.12%
```

Exempel 1: Programkörning

Svar:

```
#! /usr/bin/env python3
#-*- coding: utf-8 -*-

data = []

def gain(city):
    born, death, move_in, move_out = city[2:]
    return born-death+move_in-move_out


with open('befolkning.csv',encoding='utf-8') as f:
    for line in f:
        city, *int_data = line.strip().split(',')
        cdata = [city.strip('" '),]
        for d in int_data:
            cdata.append(int(d))
        data.append(cdata)

format_str = '{:16s}{:>10d}{:8d}{:10.2%}'
print('Kommun          Befolkning  Ökning  Procentuell ökning')

for c in sorted(data,key=lambda c:c[1], reverse=True):
    g = gain(c)
    print (format_str.format(c[0], c[1], g, g/c[1]))
```

(b) Modifiera ditt program så att användaren kan ändra tabellutskriften med följande flaggor via kommandoraden: [2 p]

- `-n N`: Skriv endast ut de `N` första raderna i tabellen
- `-p`: Sortera efter procentuell ökning
- `-g`: Sortera efter numerär befolkningsökning
- `--asc`: Sortera i ökande ordning

Svar:

Detta går att lösa på huvudsakligen två olika sätt:

- Egenskapad parsning av sys.argv

```
#!/usr/bin/env python3
#-*- coding:utf-8 -*-

data = []

def gain(city):
    born, death, move_in, move_out = city[2:]
    return born-death+move_in-move_out

def percent(city):
    return gain(city)/city[1]

############################################################
#               Argumentparsning med argv                  #
############################################################
import sys
sort_by={'g':gain,
         'p':percent}
sort_fun = None
sort_order = True
lines = 0
for index,arg in enumerate(sys.argv[1:]):
    if arg == '--asc':
        sort_order=False
    elif arg.startswith('-'):
        arg = arg[1:]
        if arg in sort_by:
            sort_fun = sort_by[arg]
        elif arg[0] == 'n':
            lines = int(sys.argv[index+2])

if sort_fun is None:
    sort_fun=lambda c: c[1]
############################################################

with open('befolkning.csv',encoding='utf-8') as f:
    for line in f:
        city, *int_data = line.strip().split(',')
        cdata = [city.strip('" '),]
        for d in int_data:
            cdata.append(int(d))
        data.append(cdata)

if lines == 0 or lines > len(data):
```

```
    lines = len(data)

format_str = '{:16s}{:>10d}{:8d}{:10.2%}'
print('Kommun          Befolkning  Ökning  Procentuell ökning')

for c in sorted(data,key=sort_fun, reverse=sort_order)[:lines]:
    g = gain(c)
    print (format_str.format(c[0], c[1], g, g/c[1]))
```

- argparse

```
############################################################
#               Argumentparsning med argparse              #
############################################################
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('-n', type=int, default=0)
parser.add_argument('-p', dest='sort_fun',
                    action='store_const', const=percent)
parser.add_argument('-g', dest='sort_fun',
                    action='store_const', const=gain,
                    default = lambda c:c[1])
parser.add_argument('--asc', action='store_true')
args = parser.parse_args()

sort_fun=args.sort_fun
sort_order = not args.asc
lines = args.n
############################################################
```

9. På filen *given_files/library.py* finns given kod som hanterar en samling böcker. Funktioner som hanterar en bok finns givna i modulen *given_files/book.py*. Din uppgift är att lägga till de funktioner som krävs i book.py för att library.py ska fungera enligt följande beskrivning. Du får inte ändra något i library.py. [3 p]

   Kravspecifikation för library:

   - Programmet läser in ett antal böcker från filen books.txt till en lista.
   - Programmet skriver ut information om alla inlästa böcker på skärmen på format enligt Kodruta 2.
   - Programmet ska sortera listan med böcker efter i första hand författare och i andra hand titel.
   - Böckerna skrivs ut igen i sorterad ordning.
   - Programmet låter användaren undersöka om en viss titel finns i biblioteket och skriver då ut information om boken.

```
---------------------------------------------
Författare: <bokens författare>
Titel:      <bokens titel>
---------------------------------------------
```

Kodruta 2: Format för utskriften av en bok

Svar:

```
def create_book(title, author, pages, language):
    return {'title' : title,
            'author': author,
            'pages' : pages,
            'lang'  : language}

def title(book):
    """ Returns the title of a book """
    return book['title']

def author(book):
    """ Gets the book's author """
    return book['author']

def compare(book1, book2):
    """ compare two books, return -1 if book1 is "less then" book2,
    zero if they are the same and 1 if book1 is greater then book2 """
    if author(book1) < author(book2):
        return -1
    elif author(book1) == author(book2):
        if title(book1) < title(book2):
            return -1
        elif title(book1) == title(book2):
            return 0
    return 1

def key(book):
```

```
    return (book['author'], book['title'])
    #eller:
    #k = fuctools.cmp_to_key(compare)
    #return k(book)

def display(book):
    """ Prints information about the book """
    info = """
{c:-<{n}s}
Författare: {author}
Titel:      {title}
{c:-<{n}s}""".format(n=45,c='-',**book)
    print(info)

def equal(b, t):
    return title(b) == t
```