✓Andreas
✓Ingela
✓Alan

57 VG

Anonym kod: 0030-LJC

1) Träd (4 + 6p)

Antag följande klass Tree:

```
class Tree:
    def __init__(self,data):
        self.data = data
        self.left = None
        self.right = None
```

4+5,5 | 9,5

a) Placera följande heltal i en trädstruktur: 34, 1, 100, 43, 22, 5, 93, 12. 3 i sorted ordning genom att rita ett träd med noder som har dessa heltal som dataelement (dvs. rita trädet på papper).

b) Skriv sedan en rekursiv funktion (**hitta_vaerde**) i python som söker ut ett visst värde i trädet i a-uppgiften. Anta att variabeln **root** pekar ut ingången (roten) till trädstrukturen. Funktionen skall lämna tillbaka sant om värdet finns, annars falskt:

```
def hitta_vaerde(root,soek_nr):
     ## din kod här...
```

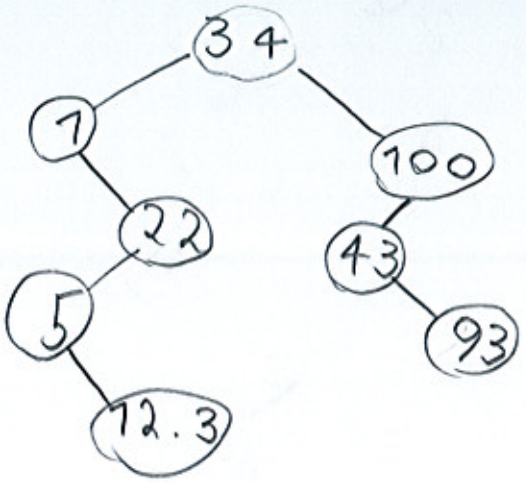
Exempel på hur funktionen kan användas:

```
soek_nr = 43
finns = hitta_vaerde(root,soek_nr)
if finns:
    print("Värdet: ", soek_nr, "finns i trädet")
else:
    print("Värdet: ", soek_nr, "finns inte trädet")
```

Anonym kod: 0030-LJC

1)

a)



Left, mindre
Right, mer

b)

```
def hitta_vaerde(root, soek_nr)
    if root is None:
        return False
    temp = root.data
    if soek_nr < temp:
        return hitta_vaerde(root.left, soek_nr)
    elif soek_nr > temp:
        return hitta_vaerde(root.right, soek_nr)
    elif soek_nr == temp:
        return True
```

5,5

rätt!

testa först (efter test för None)

Anonym kod: 0030-LJC

Anonym kod: 0030 - LJC

2) Rekursion (5p)

Förklara hur följande kod fungerar. Visa vad som skrivs ut.

```
def A (i):
    print(i, end=' ')
    if i == 10:
        return 0
    else:
        return 1 + B(i+1)

def B (j):
    print(j, end=' ')
    if j == 10:
        return j
    else:
        return 1 + A(j+1)

x = A(2)
print()
print (x)
```

5

Först printas "2 " alltså "2" och ett blanksteg. Condition därefter uppfylls inte utan koden utför istället "else:" satsen.
För att komma fram till vad som ska returneras så skickas i+1 (just nu 3) in i funktionen B. Då printas "3 " alltså "3" och ett blanksteg. Condition j==10 uppfylls inte utan vi går vidare till else som nu kallar på funktionen A.

Vi räknar alltså upp till dess att vi når 10, eftersom vi också har med "1 + ..." i funktionernas returns. Rekursionen når sitt slut när antingen i eller j är 10.

Vad som gör kodens användningsområde knapp givet är att beroende på om talet är

Anonym kod: 0030-LJC

Jämt eller udda så kommer antingen 0 eller 10 "adderas" på slutet av return-kedjan, och vi får därför bara tillbaks differensen mellan 2 och 10 om anrops-kedjan botthar med A-funktionen.

Såhär kommer det se ut:

```
2 3 4 5 6 7 8 9 10
                  # newline från tomt print()
8
```

(Observera att sista print riskerar att tolkas som ett Syntax error p.g.a mellanslaget... det följer i alla fall inte pep-8...)

Anonym kod: 0030-LJC

3) SQL och Python (5p)

Antag att följande relationer som finns i en SQLite3 databasfil som heter **books.db**

Böcker

Boknamn	Författare	Utgivningsår
Emma	Jane Austen	1815
Två städer	Charles Dickens	1859
Alice i Underlandet	Lewis Caroll	1865
Världarnas krig	H G Wells	1898
Robinson Crusoe	Daniel Defoe	1719
Gullivers resor	Jonathan Swift	1726

Läsare

Namn	Boknamn
Alan	Emma
Alan	Världarnas krig
Alan	Robinson Crusoe
Andreas	Två städer
Lisa	Alice i Underlandet
Lisa	Gullivers resor
Jenny	Emma

Skriv ett pythonprogram som använder SQL för att ta fram namn på läsare som läst en bok från 1700-talet. Programmet bör skriva ut:

```
Namn
-----------
Alan
Lisa
```

Inledningen av programmet:

```
import sqlite3
conn = sqlite3.connect('books.db')
## Din kod kommer här:
```

Anonym kod: 0030-LJC

#(Inledningen i uppgiften:)

```
import sqlite3
conn = sqlite3.connect('books.db')
```

#(Min kod:)

```
my_cursor = conn.cursor()

my_cursor.execute(""" Select distinct namn
  from läsare, böcker where
  läsare.boknamn = böcker.boknamn and
  utgivningsår >= 1700 and utgivningsår < 1800; """)

my_list = my_cursor.fetchall()

print("Namn")
print("--------")
for i in my_list:
    print(i)
```

Anonym kod: 0030-LJC

Anonym kod: 0030-LJC

4) Utveckling och debugging (3 + 2p)

```
 1  def funky(n):
 2      svar = 3 * n
 3      print(n, '* 3 =', svar)
 4
 5  tal = float(input('Skriv en siffra: '))
 6
 7  funky(tal)
 8
 9  if tal > 10000:
10      print('Wow, stort tal!')
```

a) Vi ska köra debugging på koden i bilden. När vi kommer till första breakpoint, på rad 7, kan vi välja hur vi stegar vidare. Förklara vad de olika alternativen gör med en mening var, samt vilken rad du kommer till när du väljer att stega vidare med detta alternativ.

- continue
- step in
- step over

b) Peka ut två fel i koden nedan och berätta om det är ett logiskt fel, kompileringsfel eller exekveringsfel:

```
tal = float(ipnut('Skriv in ett tal'))
hälften = talet * 2
print('Hälften av detta är', hälften)
```

a) Continue innebär att vi fortsätter enligt programets normala flöde, alltså att vi i detta fall går in i funktionen på rad 1.
Step in innebär att vi går in i funktionen. Vi fortsätter på rad 1.
Step over stegar "över" funktionen. Vi fortsätter på rad 8.

Anonym kod: 0030-LJC

b) Fel 1:
Input är felstavat.
Det är ett kompilerings fel.

Fel 2:

Variabeln "talet" är ej definierad.
(Borde stå "tal")
Det är ett exe~~kevering~~sfel. 1.5
kompilerings fel

3.5 / 5p

Anonym kod: 0030-LJC

5) Matplotlib (3 + 3p)

Vi har data hur långa 10000 olika personer är för åldern 0-18år. Det finns alltså mätvärden för varje person och år. Vi vet också om de har ätit ett fantastiskt piller dagligen under dessa år (ja/nej).

a) Vad kan passa för att visa upp hur åtta av dessa personer vuxit genom åren? Förklara vad vi behöver skicka in till funktionen som ritar upp denna typ av graf samt vilken information är lämplig att skriva ut.

b) Vad kan passa för att visa upp fördelningen av längd på barn i åldern 14 år? Förklara vad vi behöver skicka in till funktionen som ritar upp denna typ av graf samt vilken information är lämplig att skriva ut.

1p ges för lämpligt val av graf, 1p för hur korrekt input ges och 1p för vilken info som är relevant att skriva ut till grafen.

a) En linjegraf där x-axeln är ålder och y-axeln längd. (Med 8 linjer)
Vi behöver skicka in en lista med åldrar, t.ex. list(range(18)) och en lista med 18 motsvarande längder i samma kronologiska ordning.
Dessa sätter vi som x, respektive y i ovan nämnda ordning.
Detta gör vi för var och en av de 8 personerna.
Vi kan sätta följande:
plt.title("Tillväxt")
plt.xlabel("Ålder")
plt.ylabel("Längd")

3

Anonym kod: 0030-LJC

b) Ett histogram med längd på x-axeln och antal barn på y-axeln.

Vi behöver skicka in en lista med alla 14-åringars längder, samt antal bins. Bins är hur många behållare (skilda stolpar) vi vill dela upp längderna i. Förslagsvis 100 bins

Vi kan sätta:

```
plt.title("14-åringars längd")
plt.xtitle("Längd")
plt.ytitle("Antal barn")
```

3

6/6p

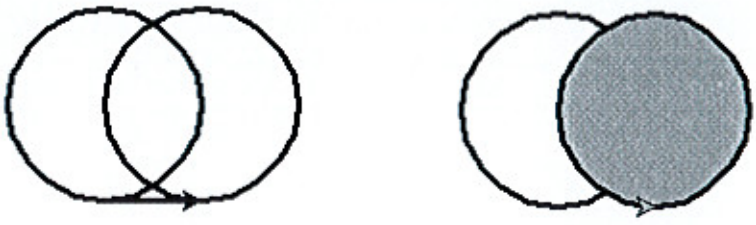
Anonym kod: 0030-LJC

6) Turtles (3 + 3 + 3p)

För samtliga exempel kan du anta att turtle importeras, att vi skapat en turtle med namnet **t** och att vi avslutar med **turtle.done()**.

a) Koden nedan skapar formen i den vänstra bilden. Utöka koden så att den skapar bilden till höger.

```
t.circle(50)
t.forward(50)
t.circle(50)
```



b) Måla ut formen som turtle **t** skapar:

```
for i in range(4):
    t.forward(100*i)
    t.left(90)
```

c) Måla ut formen som turtle **t** skapar:

```
t.circle(100)   # tar in radie
t.right(180)
t.circle(100)
t.dot(200)      # tar in diameter
```

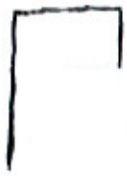
Anonym kod: 0030 - LJC

a)

```
t.circle(50)
t.pen_up()
t.forward(50)
t.pen_down()
t.fill_color("gray")
t.begin_fill()
t.circle(50)
t.end_fill()
```

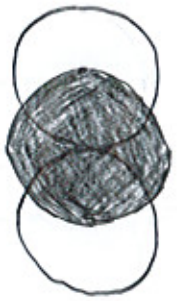
3

b)



3

c)



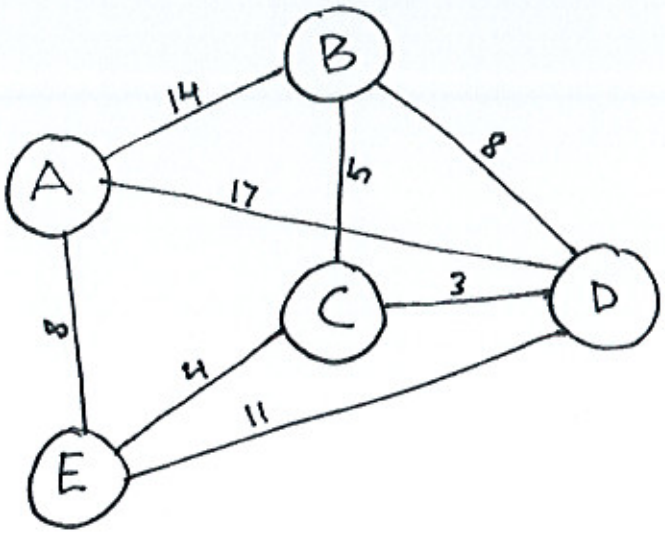
3

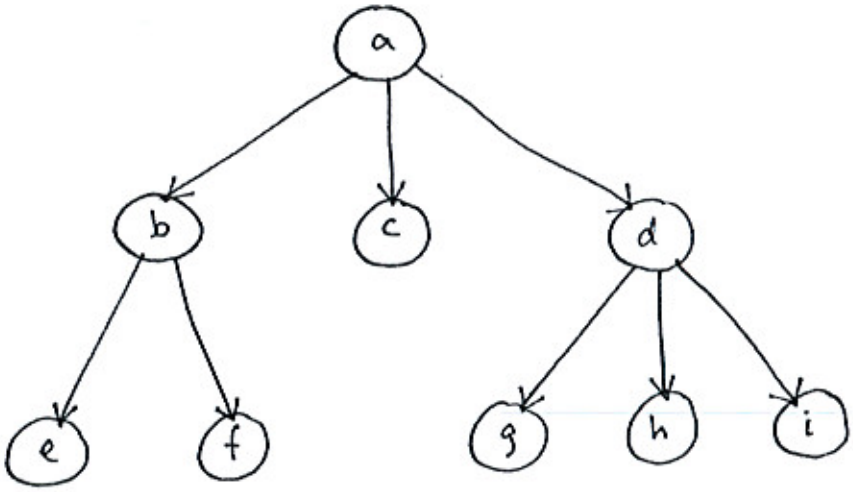
9 / 9p

Anonym kod: 0030-LJC

7) Grafer (2p)

Vilka egenskaper har övre respektive undre graf? Två korrekta termer per graf ska ges för full poäng, längre svar ger inte mer poäng.





Anonym kod: 0030-LJC

Graf 1

Viktad

Graf 2

Hierarkisk

1 / 2p

Anonym kod: 0030-LJC

8) Objektorientering (2 + 2 + 1 + 2 + 3 p)

a) I vilka situationer kan det vara bra med objektorienterad kod?

Tänk dig ett anteckningsprogram där varje anteckning har rubrik (**title**) och innehåll (**content**). Om vi vill vara objektorienterande så skulle vi kunna koda detta på följande vis:

```
class note:
    def __init__(self, title, content):
        self.__title = title
        self.__content = content

title = input("Skriv in rubrik: ")
content = input("Skriv in innehåll: ")

my_note = note(title, content)
```

b) I koden ovan, vad är det för skillnad på variablerna **title** och **content** jämfört med variablerna **self.__title** och **self.__content**?

c) Varför innehåller variablerna **self.__title** och **self.__content** dubbla understreck?

d) **__str__** kan användas för att skriva ut innehållet hos ett objekt. Skriv en sådan metod för ovanstående klass.

e) Skriv ytterligare en metod för ovanstående klass som låter dig ändra anteckningens innehåll.

a) Då vi vill dela upp koden och strukturera den utefter dess objekt. (samt tillhörande metoder) Vilket oftast är fallet med större program som har både många och avancerade klasser. Objekt kan vara t.ex. koordinater, virtuella bilar eller bankkonton. /2

Anonym kod: 0030-LJC

b) "title" och "content" är lokala parametrar som endast tillfälligt antar de inkomna värden. (när ett nytt objekt instantiate:as så tilldelas de de nya inkomna värdena tillfälligt)

Self._title och Self._content är den nuvarande instansens instansvariabler. Dessa tilldelas de även inkomna parametrarnas värden. Instansvariablerna gör att komma åt även senare, till skillnad från parametrarna. /2

c) För att de inte ska användas utifrån av personer som importerar klassen. De anses vara privata enligt vissa python-användare. Metoden kallas name-mangling och "byter namn" till _Class__varname, där Class är den givna klassen och varname namnet på variabeln utan underscores. /1

d)
```
def __str__(self):
    return f"Title: {self.__title}, Content: {self.__content}"
```
/2

e)
```
def set_content(self, new_content):
    self.__content = new_content
```
/3

Anonym kod: 0030-LJC

Anonym kod: 0030-LJC

9) Numpy (1 + 1p)

a) Vad beskrivs med NumPys **shape**?

b) I vilka situationer är det en fördel att använda NumPy-arrayer istället för Pythonlistor?

a) Antalet element i en array. /1
+ Antal dimensioner

b) Då man vill utföra operationer på samtliga element utan att stega igenom en motsvarande lista.
T.ex. dela alla element med 3 eller addera två arrayer med varandra. /1

Anonym kod: 0030 - LJC

10) GUI och grafik (1 + 2 + 3p)

a) Vad är ett GUI?

b) Vad innebär det att en widget är "native"?

c) Om du vill bygga ett program som har ett GUI, i vilka situationer är det en fördel att använda ett bibliotek med native widgets och när är det bättre att ha en egen widgetdesign?

a) GUI = graphical user interface.
Det är ett grafiskt gränssnitt, alltså det som användaren ser och interagerar med. /1

b) Det innebär att widgeten tagit sitt utseende från det givna operativsystemets egna bibliotek, och därmed ser ut som de andra native-apparna på samma OS. /2

c) Om ditt program ska finnas på flera plattformar så är egen design bättre för att användarna ska känna igen sig överallt. (T.ex. Spotify)

Om ditt program t.ex. riktar in sig på oerfarna användare så kan native vara bättre för att slippa lära sig ett nytt gränssnitt.

om du vill ha en "unik" look till t.ex. ett spel så är egen design att föredra. /3