Labbkompendium för kursen FYD095

Programmering med Python

Allmän information

Labbkompendiet består av fyra stora kapitel:

1. ”Grundläggande Python, del 1”,
2. ”Grundläggande Python, del 2”,
3. ”NumPy, SciPy, och matplotlib modulerna”
4. ”Objektorienterad programmering”.

Kapitlen innehåller delkapitel som består av tre uppgifter vardera. Uppgifter har olika antal poäng beroende på svårighetsgrad. För att bli godkänd på uppgiften ska en student presentera ett körbart Pythonskript och svara på frågor från en labbassistent. **Det är obligatoriskt att redovisa åtminstone 1 uppgift från varje delkapitel.**

För att bli godkänd på labbdelen av kursen med betyget **G** ska studenten få åtminstone 30 poäng. För att bli godkänd på labbdelen av kursen med betyget **VG** ska studenten få åtminstone 45 poäng.

Lycka till med laborationerna!

Kapitel 1: Grundläggande Python, del 1

Delkapitel 1.1. Printing

Uppgift 1.1.1 (1p): Skapa ett program som hittar och skriver ut lösningen till andragradsekvationen $ax^2 + bx + c = 0$. a, b, och c är koefficienter som anges av användaren. Vi utgår från att ekvationen har två reella lösningar.

Uppgift 1.1.2 (1p): Skapa två heltal av datatypen `int`. Skriv ett program som skriver ut

a) Dessa tal formaterade som hexadecimala tal.
b) Kvoten av de två talen. Resultatet ska skrivas ut som ett flyttal med 3 siffror efter decimalpunkten.
c) Resten efter division mellan de två talen.

Uppgift 1.1.3 (1p): Skriv ett program som läser in information om människor, så som namn, efternamn, och födelseår, angett av användaren och skriver ut en formaterad tabell med denna information. Man får inte använda `tabulate` modulen.

Delkapitel 1.2. Arbete med strängar

Uppgift 1.2.1 (1p): Beräkna och skriv ut medelvärde, högsta och lägsta tal från strängen inmatad av användaren. Talen är separerade med mellanslag i strängen.

Uppgift 1.2.2 (1p): Programmet beräknar antal a) siffror och b) bokstäver i strängen angiven av användaren.

Uppgift 1.2.3 (1p): Programmet kommer att hitta alla positioner av ett mönster i en sträng. Mönstret och strängen inmatas av användaren. Med mönstret menas en annan sträng. Till exempel, för strängen ”123123” och mönstret ”123” skall programmet returnera två index 0 och 3.

Delkapitel 1.3. If-satser

Uppgift 1.3.1 (1p): Skapa ett program som avgör huruvida ett tal är reellt eller komplext. Om talet är reellt avgör programmet huruvida talet är ett heltal. Talet inmatas av användaren.

Uppgift 1.3.2 (1p): Skapa flera objekt och flera referenser. Programmet ska avgöra huruvida två referenser pekar på samma objekt. Om de är olika, kollar programmet om de två objekten har samma värde.

Uppgift 1.3.3 (1p): Skapa ett program som avgör huruvida en sträng är en palindrom. Om strängen innehåller andra tecken än bokstäver, ignoreras de andra tecknen.

Delkapitel 1.4. Repetitionssatser, del 1

Uppgift 1.4.1 (1p): Skriv ett program i vilket en användare anger ett namn och ålder för en person. Efter varje inmatning skriver ut programmet medelålder för alla inmatade personer tillsammans med den äldsta personens namn och ålder. Användaren har en möjligt att avsluta inmatningen på ett lämpligt sätt.

Uppgift 1.4.2 (2p): Skriv ett program som bestämmer veckans dag för datum angett av användaren i formatet yyyy-mm-dd. Det tidigaste datumet är 2001-01-01 och det senaste datumet 2099-01-01. Man får inte använda `datetime` modulen. Denna uppgift får lösas utan repetitionssatser.

Uppgift 1.4.3 (1p): Programmet skriver ut multiplikationstabell från 1 till 9 i formen av en formaterad tabell. Man får inte använda `tabulate` modulen.

Delkapitel 1.5. Repetitionssatser, del 2

Uppgift 1.5.1 (1p): Skriv ett program som skriver ut alla positiva delare till ett positivt heltal.

Uppgift 1.5.2 (1p): En geometrisk serie $S = a + aq + aq^2 + aq^3 + \ldots$ konvergerar om $|q| < 1$. Hitta en summa av en geometrisk serie med N stycken termer. Antalet termer (N), q, och det begynnelse elementet, a, inmatas i programmet. Kontrollera svaret med formeln:

$$S = \frac{a}{1 - q}$$

Uppgift 1.5.3 (1p): Skriv ut en formaterad tabell med åtminstone 20 värden för $z = \sin(x)\sin(y)/(xy)$ funktionen. Tabellen innehåller tre kolumner med namn x, y, och z. Användaren anger intervall för x och y. Programmet skall hantera punkterna med x = 0 och/eller y = 0 på ett lämpligt sätt.

Delkapitel 1.6. Tupler och listor

Uppgift 1.6.1 (1p): Skapa en lista med *N* ekvidistanta punkter från *a* till *b*. *a* och *b* skall ingå i listan. *N*, *a* och *b* anges av användaren. Man får inte använda **`NumPy`** modulen.

Uppgift 1.6.2 (1p): Användaren matar in flera tal (åtminstone 5) separerade med mellanslag. Efter inmatningen ställer programmet tre frågor: 1) om de inmatade numeriska talen ska lagras i en lista eller en tupel, 2) om sekvensen ska sorteras i växande eller avtagande ordning, och 3) om den sorterade sekvensen ska returneras i en lista eller tupel.

Uppgift 1.6.3 (1p): Givet listan A = [5,4,1,0,10,12,34] och tupeln A = (5,10,3,20,2), bestäm utan dator följande uttryck:

1) **`A[1:3]`**
2) **`A[3:1:-1]`**
3) **`A[-1::-1]`**
4) **`A[len(A):-1:1]`**
5) **`A[1:10]`**
6) **`A.append(10)`**

med **`A`** som både listan eller tupeln.

Delkapitel 1.7. Tvådimensionella listor

I detta kapitel får man inte använda **`NumPy`** biblioteket.

Uppgift 1.7.1 (1p): Användaren matar in två matriser som behållas i programmet som tvådimensionella listor. Programmet kollar om matriserna har samma storlek. Om storlekerna är samma kan användaren från meny välja att 1) summera matriserna, 2) subtrahera matriserna, och 3) utföra elementvis multiplikation. Resultatet sparas i en ny tvådimensionella lista

Uppgift 1.7.2 (1p): Användaren matar in två matriser som behålls i programmet som tvådimensionella listor. Programmet kollar om matrisstorlekarna tillåter matrismultiplikation och utför isåfall matrismultiplikationen. Resultatet sparas i en ny tvådimensionell lista.

Uppgift 1.7.3 (2p): Skapa en enkel databas med personliga uppgifter. Databasen innehålls i en tvådimensionell lista där varje rad motsvarar en person och varje kolumn (totalt 4) beskriver namn, efternamn, födelseår och födelseort. Programmet innehåller meny så att användaren kan

1) lägga till en ny person till databasen,

2) ta bort en person från databasen,

3) sortera databasen efter efternamn,

4) sortera basadaten efter födelseår,

5) skriva ut medelåldern av människorna från databasen,

6) skriva ut databasen som en formaterad tabell,

7) lämna programmet.

Delkapitel 1.8. *Comprehension listor*

Uppgift 1.8.1 (1p): Skriv ett program som skapar en lista med 100 slumpmässiga heltal mellan 1 och 100. Med hjälp av *comprehension listor* skapa en ny lista som plockar från den slumpmässiga listan element som är delbart med 7, 11, eller 13.

Uppgift 1.8.2 (2p): a) En 3×3-matris av enbart nollor skapas som

```
M = [[0]*3]*3
```

Om man tilldelar elementet med indexen 0 och 1 till 2 som

```
M[0][1]=2
```

blir matrisen

```
[[0, 2, 0], [0, 2, 0], [0, 2, 0]]
```

Förklara varför ändras alla elementen i kolumnen med indexet 1 och hur man kan överkomma detta problem.

b) Skapa en endimensionell lista med hjälp av *comprehension listor* som innehåller diagonalelementen av en given tvådimensionell lista.

Uppgift 1.8.3 (2p): Med hjälp av *comprehension listor* skapa en kvadratisk matris med kryss-mönster:

```
1 0 0 0 1

0 1 0 1 0

0 0 1 0 0

0 1 0 1 0

1 0 0 0 1
```

Användaren ska kunna välja antal rader (kolumner).

Delkapitel 1.9. Mängder (`set`) och associativa tabeller (`dict`), del 1

Uppgift 1.9.1 (1p): Användaren anger en ordföljd där orden är separerade med komma. Med hjälp av **`set`**, skapa en tupel med de inmatade orden. Orden skall inte upprepas i tupeln.

Uppgift 1.9.2 (1p): Skriv ett program som omvandlar en lista till en associativ tabell (**`dict`**) där **`key`** är elementets index från listan och **`value`** är listans element själv.

Uppgift 1.9.3 (1p): Skapa en lista med 20 ord, sådan att några ord upprepas. Skriv ett program som returnerar en associativ tabell där söknycklarna är orden från listan, och de motsvarande värdena är antalet gånger de förekommer i listan.

Delkapitel 1.10. Mängder (`set`) och associativa tabeller (`dict`), del 2

Uppgift 1.10.1 (1p): Skapar en tvådimensionell lista bestående av ord. Med hjälp av **`set`**, ta bort duplikat från varje rad.

Uppgift 1.10.2 (1p): Skapa en associativ tabell med 10 element. Skapa även en lista vars element är söknycklarna i tabellen. Skriv ett program där användaren anger ett index för ett element i listan och det motsvarande värdet returneras från tabellen.

Uppgift 1.10.3 (2p): Skriv ett program som skapar en associativ tabell, T, från två andra tabeller, T1 och T2, på tre olika sätt: 1) den nya tabellen innehåller elementen (key-value paren) från T1 och T2. Om T1 och T2 innehåller elementen med samma söknyckel elementet från T2

bortses; 2) den nya tabellen innehåller de gemensamma elementen (key-value paren) från T1 och T2; 3) Elementen (key-value paren) i den skapade tabellen skall vara den exklusiva differensen av elementen (key-value paren) från T1 och T2. Programmet ska innehålla en meny där användaren kan välja mellan dessa tre olika sätt.

Kapitel 2: Grundläggande Python, del 2

Delkapitel 2.1. Funktioner, del 1

Uppgift 2.1.1 (2p): Skapa en funktion som omvandlar ett heltal (`int`) till andra talsystem. Funktionen tar in ett heltal och en bas, och skriver sedan ut talet i talsystemet med den basen. Basen kan vara mellan 1-16.

Uppgift 2.1.2 (1p): Skapa en funktion som tar in ett komplext tal på formen $a + bj$ (ett objekt av klassen `Complex`) och returnerar absolutbeloppet och vinkeln. Man får inte använda `cmath` modulen.

Uppgift 2.1.3 (1p): Skapa en funktion som avgör huruvida ett angivet tal är ett primtal. Funktionen skall returnera ett värde av datatypen `bool`.

Delkapitel 2.2. Funktioner, del 2

Uppgift 2.2.1 (1p): Skriv en rekursiv funktion som returnerar n:te elementet i en geometrisk följd. Funktionen antar tre argument, a_1, q, och n. Elementet med ordningsnumret n beräknas som $a_n = a_1 \cdot q^{n-1}$.

Uppgift 2.2.2 (1p): Värdet på cosinus och sinusfunktioner i punkten x kan beräknas numeriskt enligt Taylorutvecklingen som

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

Skapta två funktioner som beräknar cosinus och sinusfunktioner med en särskild noggrannhet. *x* värdet och noggrannheten är ingående parametrar.

Uppgift 2.2.3 (1p): Skapa en funktion som returnerar en lista med *N* ekvidistanta punkter från *a* till *b*. *a* och *b* ingår i listan. Listan skall skapas med hjälp av *comprehension listor*. Om *N* inte anges i funktionsanrop, tar det värdet som är lika med 100.

Delkapitel 2.3. Funktioner, del 3

Uppgift 2.3.1 (1p): Skapa en funktion som kan anropas med ett variabelt antal argument. Funktionen ska om ingående parametrar är numeriska returnera summan av de kvadrerade elementen. Om inga argument anges eller någon parameter är icke numerisk, returnerar funktionen ett meddelande om detta.

Uppgift 2.3.2 (1p): Skapa ett `lambda` uttryck som returnerar en tupel med lösningen till andragradsekvationen $ax^2 + bx + c = 0$. *a*, *b*, och *c* är funktionens ingående argument. Det är garanterat att ekvationen har två lösningar.

Uppgift 2.3.3 (1p): Skapa en lista med tre `lambda` uttryck som alla tar in ett argument. Skriv en funktion *F* som kan ta in denna lista och returnera en tupel med värden av dessa tre `lambda` uttryck i punkten *x*. *x* är ett annat ingående argument till *F*.

Delkapitel 2.4: Enkla spel

Uppgift 2.4.1 (2p): Skriv ett program med ett lämpligt användargränssnitt som en elev kan använda för att öva multiplikationstabellen mellan 1 och 9. Eleven skall ha möjligheten att välja svårighetsnivå mellan 1 och 5. Svårighetsnivån 1 innebär att eleven har 1 sekund på sig att svara o.s.v. Programmet ska kontrollera om svaret inmatat av eleven är rätt och om det är givet under tiden bestämd av svårighetsnivån. I användargränssnittet finns det en möjlighet att välja antal övningar. Frågor, t.ex. `7 × 9 = ?`, förbereds med hjälp av slumpmässig talgenerator.

Tips: Använd funktionen `time` från modulen `time` för att returnera tiden.

Uppgift 2.4.2 (1p): Skapa ett lämpligt användarsnitt för spelen Sten, Sax, Påse. Spelaren spelar mot datorn vars val genereras slumpmässigt.

Uppgift 2.4.3 (2p): Skapa ett lämpligt användarsnitt för spelen Tre i rad (eng. Tic-Tac-Toe). Ett bräde är sparat i en tvådimensionell lista 3×3 som skrivs ut efter varje steg. Spelaren väljer en ruta i brädet genom att ange två index för den tvådimensionella listan. Spelaren spelar mot datorn vars drag genereras slumpmässigt.

Delkapitel 2.5: Arbete med filer, del 1

I detta kapitel får man inte använda `OS` biblioteket.

Uppgift 2.5.1 (1p): Skriv ett program med en meny sådan att användaren kan väja att 1) skriva ut en fil, 2) lägga till en rad till filen, 3) skriva över filen, och 4) lämna programmet.

Uppgift 2.5.2 (1p): Skapa en funktion som har två ingående argument: filnamn och ett ord. Funktionen returnerar antal förekomster av detta ord i filen.

Uppgift 2.5.3 (1p): Skriv ett program som räknar antal tecken, antal ord, och antal rader i en fil.

Delkapitel 2.6: Arbete med filer, del 2

I detta kapitel får man inte använda `OS` biblioteket.

Uppgift 2.6.1 (1p): Skapa två funktioner som har filnamn som ett ingående argument. Den första funktionen lägger till en matris (tvådimensionell lista) till filen och den andra skriver ut matriserna som finns i filen. Matriser skall skrivas ut som formaterade tabeller. Demonstrera användning av funktionerna.

Uppgift 2.6.2 (1p): Denna uppgift använder sig av en fil som ni laddar ner från Canvas. Filen innehåller en rad av namn. Skriv ett program som läser filen och räknar hur många gånger varje namn förekommer i filen. Resultatet presenteras som en associativ tabell där söknyckel är namn och det motsvarande värdet är förekomsten av namnet i filen.

Uppgift 2.6.3 (2p): Skapa en funktion som kodar en text till Morsealfabetet. Skapa även en funktion som avkodar filen med Morsealfabetet. Demonstrera funktionalitet av funktionerna.

Kapitel 3: `NumPy`, `SciPy`, och `matplotlib` modulerna

Delkapitel 3.1: Vektorer och matriser från modulen `NumPy`, del 1

I detta delkapitel ska **`NumPy`** modulen användas för att skapa vektorer och matriser samt som utföra olika operationer med dem. Man får inte använda **`for`**-loopar.

Uppgift 3.1.1 (1p): Skapa en vektor bestående av 200 ekvidistanta punkter från och med 12, till och med 18. Beräkna summan av

1) alla element med jämna index,

2) alla element med udda index,

3) alla element vars index är delbart med 10,

4) elementen med indexen 2, 5, 19, 92.

Uppgift 3.1.2 (1p): Skapa en radvektor *R* med element 1,2, och 3, och en kolumnvektor *C* med samma element. Bestäm utan dator följande uttryck:

1) **`R - C.T`**
2) **`R.T - C`**
3) **`R - C`**
4) **`dot(R,C)`**
5) **`dot(C,R)`**

Uppgift 3.1.3 (1p): Skapa en 4 × 5-matris vars element är slumpmässiga heltal mellan 0 och 11. Beräkna för den här matrisen:

1) summor av element i varje rad

2) summor av element i varje kolumn

3) summor av element i varje kolumn med ett jämnt index

4) medelvärden för varje kolumn

5) ett medelvärde för alla element i matrisen

6) en standardavvikelse för alla element i matrisen

Delkapitel 3.2: Vektorer och matriser från modulen `NumPy`, del 2

I detta delkapitel ska modulen **`NumPy`** användas för att skapa vektorer och matriser samt att utföra olika operationer med dem.

Uppgift 3.2.1 (1p): Skapa två 4 × 4-matriser *A* och *B* vars element är slumpmässiga tal. Skapa sedan en matris *C* som ser ut som

$$C = \begin{bmatrix} A & B \\ B & A \end{bmatrix}$$

Hitta diagonalen för matrisen *C*. Diagonalelementen ska presenteras i en 4 × 2-matris.

Uppgift 3.2.2 (2p): Bestäm utan dator storleken för matrisen A i följande uttryck:

1) `A = vstack([zeros((2,2)),zeros((2,2)),zeros((2,2))])`

2) `A = hstack([A,zeros((6,4))])`

3) `A = concatenate([A,zeros((6,4))],axis=1)`

4) `A = concatenate([A,zeros((2,10))],axis=0)`

5) `A = delete(A,[8,9],axis = 1)`

6) `A = diag(A).reshape(4,-1)`

Uppgift 3.2.3 (1p): Skapa en 4 × 5 matris med slumpmässiga heltal mellan -10 och 10. Med hjälp av *logisk indexering*, 1) hitta index för element som är större än noll, 2) skriv ut negativa heltal, och 3) ersätta negativa heltal med 0.

Delkapitel 3.3: Grafik från `matplotlib.pyplot` modulen

Uppgift 3.3.1 (1p): Skapa ett program som ritar de 5 olympiska ringarna.

Uppgift 3.3.2 (1p): Skriv ett program som plottar tre funktioner sin(x), cos(x), sin(x)/x för x från intervallet mellan -6π till 6π. Programmet skall ha en interaktiv meny sådan att användaren kan välja att plotta 1) tre funktioner på samma figur, 2) varje funktion i egen figur, och 3) tre funktioner på samma figur men i varsin subfigur.

Uppgift 3.3.3 (1p): Skapa ett program som ritar en godtycklig kurva. Programmet har en meny sådant att användaren kan välja: 1) linjestil, 2) linjefärg, 3) markör, 4) linjetjocklek, 5) gränser för *x*- och *y*-axel, och 6) figurtitel.

Delkapitel 3.4: Numeriska metoder och modulen `SciPy`

I detta delkapitel skall funktionaliteten för varje program kontrolleras genom att jämföra resultatet med en liknande funktion/metod från modulen **`SciPy`.**

Uppgift 3.4.1 (1p): Skapa en funktion som löser ekvationen $f(x) = 0$ med bisektionsmetoden. Funktionen har 3 ingående argument: 1) en funktion för vilken en rot letas efter, 2) en tupel med *a* och *b,* som är gränserna för intervallet innehållande nollställe så att $f(a)f(b)<0$, och 3) lösningstolerans.

https://en.wikipedia.org/wiki/Bisection_method

Uppgift 3.4.2 (1p): Skapa en funktion som utför numerisk integration med hjälp av trapetsregeln. Funktionen har 3 ingående argument: 1) en funktion som skall integreras, 2) en tupel med *a* och *b* som är gränserna för integrationsområdet, och 3) antalet intervall.

https://en.wikipedia.org/wiki/Trapezoidal_rule

Uppgift 3.4.3 (2p): Skapa en funktion som löser första ordningens differentialekvationer med givet begynnelsevärdes, med Heuns metod. Begynnelsevärdesproblemet är

$y'=f(y(x),x)$ och $y(a)=C$.

Funktionen har 4 ingående argument: 1) *f*, 2) en tupel med *a* och *b* som är gränserna på intervallet där lösningen skall hittas, 3) *C*, och 4) steglängden. Funktionen returnerar två listor X och Y som innehåller lösningen. Plotta lösningen för att se om du har gjort rätt.

https://en.wikipedia.org/wiki/Heun%27s_method

Kapitel 4: Objektorienterad programmering

Delkapitel 4.1. Klasser och objekt

Uppgift 4.1.1 (1 p). Skapa en klass `Point` som representerar punkter i den tredimensionella rymden. Varje objekt har 3 instansvariabler, `x`, `y`, och `z`, som är punktens Kartesiska koordinater. Klassen har 3 instansmetoder: `Sfäriska()`, `__str__()`, och `Avstånd()`. Metoden `Sfäriska()` returnerar en tupel med koordinater av punkten i det sfäriska koordinatsystemet. `__str__()` presenterar informationen om objektet. Metoden `Avstånd()` beräknar avståndet mellan origo och punkten.

https://sv.wikipedia.org/wiki/Sf%C3%A4riska_koordinater

Tips: använd atan2() för att beräkna en vinkel.

Uppgift 4.1.2 (1 p). Skapa en klass `Matris`. Klassen har en instansvariabel som är en tvådimensionell lista med numeriska värden. När ett nytt objekt är skapat, har en användare möjlighet att ange värden för matrisen. Klassen skall ha 4 instansmetoder `min()`, `max()`, `mean()` och `std()` som hittar för matrisen det minsta elementet, det största elementet, medelvärdet och standardavvikelse för alla element som ingår i matrisen.

Uppgift 4.1.3 (1 p). Skapa en klass `Fil`. Klassen har en instansvariabel som innehåller filnamn. Klassen skall ha 3 metoder för att skriva till filen, läsa från filen och metod `__str__()` som sammanfattar informationen om filen såsom antal siffror, antal bokstäver, och antal rader.

Delkapitel 4.2. Klasser och objekt, del 2

Uppgift 4.2.1 (1 p). Skapa en klass `Student` med tre privata instansvariabler som innehåller studentens namn, efternamn, och ålder. Objekt av klassen `Student` behålls i en lista `L`. Skapa en meny så att användaren kan 1) Lägga till ett nytt objekt till listan, 2) Ta bort objekt med visst namn, 3) sortera objekten i listan med namn, efternamn, eller ålder, och 4) skriva ut en formaterad tabell med information om studenterna.

Uppgift 4.2.2 (1 p). Skapa en klass `Klass` var objekt instansieras med en associativ tabell (`dict`) så att tabellens `key` är instansvariabler och `value` är instansvariablers värden. Klassen innehåller 3 instansmetoder för att ta bort ett attribut hos objektet, lägga till ett nytt attribut och

byta värdet på ett befintligt attribut. Klassen skall ha en metod för att printa ut objektets attribut och motsvarande värden som en formaterad tabell.

Uppgift 4.2.3 (3 p). Skriv ett användargränssnitt för spelet *Sänka skepp*. Man spelar mot datorn. Programmet innehåller en klass `SpelPlan` som är spelplan för varje spelare. En spelare kan välja huruvida fartyg placeras på planen slumpmässigt eller det är gjort av spelaren. Klassen ska ha tillämpliga privata och publika instansvariabler och instansmetoder. Efter varje drag skall programmet printa ut spelplanen.

https://sv.wikipedia.org/wiki/S%C3%A4nka_skepp

Delkapitel 4.3. Klassmetoder och klassvariabler

Uppgift 4.3.1 (1 p). Skapa en klass `Person` med fyra instansvariabler: `namn`, `efternamn`, `id`, och `ålder`. Klassen innehåller även en klassvariabel `antal` som håller reda på hur många personer finns. Instansvariabeln `id` initialiseras med närvarande antal personer. Klassen innehåller instansmetoden `__str__()` som presenterar informationen om personen på ett formaterat sätt. Dessutom innehåller klassen en klassmetod som tar reda på hur många `Person` objekt som har instansierats.

Uppgift 4.3.2 (1 p). Skapa en klass `Person` med tre instansvariabler: `namn`, `efternamn`, och `ålder`. Klassen ska ha instansmetoden `__str__()` som presenterar informationen om personen på ett formaterat sätt. Klassen innehåller en klassvariabel `OBJ` som är en lista. När ett nytt objekt instansieras, läggs till en referens till objektet till listan `OBJ`. Klassen innehåller även en klassmetod som printer ut information om alla personer som står i `OBJ` i form av en formaterad tabell. Denna klassmetod skall anropa instansmetoden `__str__()`.

Uppgift 4.3.3 (1 p). Skapa en klass `Vektor` som representerar en vektor i ett tre-dimensionellt rum. Klassen har instansvariabler för vektorns tre koordinater: `x`, `y`, och `z`. Klassen har metoden `R()` som returnerar vektorns längd. Klassen har även en klassmetod som skapar en ny vektor från en annan vektor. Klassen skall ha metoden `__str__()` som returnerar en strängrepresentation.

Delkapitel 4.4. Arv och polymorfism

Uppgift 4.4.1 (1 p). Först, skapa en klass `Person` innehållandes instansvariablerna `namn` och `personnummer,` samt implementationer av metoderna `__init__` och `__str__`. Sedan, skapa en underklass `Bilist` som ärver `Person` och lägger till instansvariabeln `körkortsnivå` (=”B”,”C”, etc). Metoderna `__init__` och `__str__` skall överlagras på lämpligt sätt. Sist, definiera en funktion/metod `harKörkort` som kan särskilja objekt av typen `Person` och `Bilist`, utan att använda `type` eller `if`.

Uppgift 4.4.2 (1 p). Skapa två klasser `Bil` och `Flygplan`. Den första klassen representerar bilar och den andra klassen representerar flygplan. Studenten väljer själv vilka attribut två klasserna skall ha. Både klasserna har metoden `__str__()` som sammanfattar informationen om objektet och skall ser ut på annorlunda sätt för klasserna. Utanför klassen definierar funktionen `Info()` som har ett objekt av `Bil` eller `Flygplan` som ett ingående argument. Funktionen `Info()` skall anropa instansmetoden `__str__()` för att printa ut information om objektet som passas till funktionen. Skapa flera objekt av klasserna `Bil` och `Flygplan` och lägg till deras referenser till en lista. Använd en repetitionssats för att löpa igenom listans objekt och anropa funktionen `Info()` för varje objekt. Hur manifesterar sig polymorfism i denna uppgift?

Uppgift 4.4.3 (1 p). Skapa en klass `Student` som har fyra instansvariabler: `namn`, `efternamn`, `ålder`, och `studieår`. Skapa tre klasser som ärver klassen student: `Elev,` `Kandidat`, och `Master`. Varje klass (`Elev`, `Kandidat`, och `Master`) har egna unika attribut som du kan välja själv. Alla fyra klasserna innehåller funktion `Info()` som printer ut informationen om studeranden. Skapa flera objekt av dessa fyra klasser och lägg till deras referenser till en lista. Använd en repetitionssats för att löpa igenom listans objekt och anropa instansmetoden `Info()` för varje objekt. Hur manifesterar sig polymorfism i denna uppgift?

Delkapitel 4.5. Överlagrade operatorer

Uppgift 4.5.1 (1 p). Skapa en klass `Bråk` som representerar matematiska bråk. Klassen har två instansvariabler: `nämnare` och `täljare`. Klassen innehåller fyra överlagrade operatorer för att summera(`+`), subtrahera(`-`), multiplicera(`*`) och dividera(`/`) två bråk. Klassen innehåller även en metod `Förkorta()` som reducerar bråket så att nämnare och täljare inte har

gemensamma delare utom 1. `Förkorta()` skall anropas i metoden `__init__()` så att alla bråk är i en förkortad form. Klassen skall ha en lämplig implementation `__str__` för att printa ut bråket.

Uppgift 4.5.2 (1 p). Skapa en klass `Vektor` som presenterar vektorer i tre-dimensionellt rum. Varje vektor presenteras av en tupel med 3 vektors koordinater. Klassen har 4 operatorer: `+`, `-`, `*` och `@` för att summera, subtrahera, skalärmultiplicera och kryssmultiplicera två vektorer. Klassen skall ha en lämplig metod för att printa ut vektorn.

Uppgift 4.5.3 (2 p). Skap en klass `Matris` som presenterar 4×4 matriser. Matriselementen lagras i en tvådimensionell lista. Klassen innehåller metoden `Det` som beräknar determinanten av matrisen. Klassen innehåller även metoden `T` som transponerar den nuvarande matrisen, d.v.s. den skapar inte en ny matris. Klassen har 3 överlagrade operatorer för att summera(`+`), subtrahera(`-`) och matrismultiplicera(`@`) två matriser. Klassen skall ha en lämplig metod för att printa ut matrisen.