Reflektioner kring hur det är att lära sig ett nytt språk (Ruby):

- Hur har ni gått tillväga för att lära er Ruby?
  Vi har använt oss mycket av informationen från föreläsningarna för att kolla upp saker vi var osäkra på. Eftersom vi redan har lite erfarenhet att programmera från de tidigare kurserna var vi mer i behov av att kolla upp strukturer och syntax snarare än exakt dokumentation. Självfallet fick googla en hel del också för att hitta övrig information som inte togs upp i föreläsningarna.

  Vi satte oss direkt med att gå igenom uppgifterna inför seminariet tillsammans och istället för att dela upp arbetet 50/50 diskuterade vi lösningarna med varandra.

- Vilka fel och misstag har ni gjort under tiden?
  Framförallt har det varit mycket syntaxfel nu i början där vi tex glömt att avsluta grejer med end eller satt ett mellanslag på fel ställe så att det inte kompilerar som det ska. Eftersom Ruby visuellt liknar Python väldigt mycket var vi ovana vid att många strukturer här avslutas med end. Exempelvis if-satserna i Ruby liknar väldigt mycket if-satserna i Python men avslutas här med end. Detta misstag ledde till en hel del felmeddelanden.

- Vad finns det för nya konstruktioner i Ruby som ni inte sett förut?

  Något vi märkte direkt var hur kompakt man kan skriva kod i Ruby, direkt uppmärksammade vi att man kan kalla på funktioner utan parenteser.

  Övergången från c++ med dess strikta krav på hur man deklarerar klasser till Rubys mycket mer simpla sätt var något vi reagerade på. I synnerlighet att man i en klass kan använda t.ex. attr_reader för att skapa getters till variabler på en enda rad.

  Att instansvariabler skrivs med ett snabel-a före variabelnamnet och klassvariabler skrivs med två snabel-a före variabelnamnet i klasser var något vi inte hade sett tidigare.

- Vad finns det för konstruktioner som ni känner igen men som ser lite annorlunda ut?

  Sättet man kan skicka en block med kod till en funktion liknar till viss del lambdafunktionerna i python men skiljer sig genom att de är både mer simpla och mer kraftfulla i Ruby. Ett exempel på hur detta yttrar sig är att man i Ruby kan t.ex. kan skicka in ett block i stil med { puts “Hello World” } till en funktion. Detta block går sedan då med enkelhet att manipulera eller loopas.

- Finns det något som ni irriterar er på eller tycker om i Ruby?

  Det är väldigt lätt att ha en åsikt om Ruby då det är mycket som är annorlunda från hur de språk vi jobbat tidigare med är. Till att börja med är nog ingen av oss särskilt

förtjusta i dynamiskt typade språk och att det finns så många sätt att göra saker på att det är svårt att hitta information om hur man gör någonting över huvud taget. I allmänhet tycker vi det är enklare att se vad som händer i tex C++ då allt är väldigt strukturerat medan här känns det som att mycket bara händer automagiskt och att man håller tummarna utan att veta vad som pågår. Sedan är det ju positivt att det är flexibelt, det verkar enkelt att göra klasser, attribute readers var ju en trevlig konstruktion och det går snabbt att skriva koden men när man kommer direkt ifrån C++ känns koden oerhört komprimerad så det nästan gör ont att titta på. Det som har varit jobbigast är dock errormeddelandena man får när man kör koden och något är galet då de är väldigt intetsägande och man famlar i blindo tills man inser att man har ett mellanslag på fel ställe någonstans i koden.

Reflektioner kring de olika tekniker som ni stöter på:

- Har ni hittat alternativa källor för att ta reda på nya saker?

  Vi finner att det i allmänhet är svårt att hitta information angående Ruby i förhållande till hur enkelt det är för Python och C++. Oftast hittar man något om Ruby on rails eller så pass invecklade eller specifika grejer att det är svårt att se hur det är applicerbart med det vi jobbar med för tillfället. Vi borde nog använda oss mer av dokumentationen än vi gör för tillfället samt läsa på i boken i större utsträckning.

Dokumentation av hur ni har tänkt när ni arbetat fram era lösningar

- Vad arbetade ni med (i grova drag) vid varje labbpass?

  Vi påbörjade arbetet på morgonen innan det första labpasset och gjorde egentligen klart det mesta av koden under labpasset. Vi hann inte med testningen och utvecklarbloggen så det tog vi tag i efter det första passet. Under passet ställde vi även en fråga gällande att båda följande är ekvivalenta:

  self.split.map(&:chr).join.upcase

  self.split.map{|word| word.chr}.join.upcase

  Där vi undrade exakt vad “&:chr” gjorde och hur .to_proc fungerar samt vad ett proc objekt är.

- Hur har ni tolkat uppgiften?

  För uppgift 1, 3 och 5 har vi inget närmare att förklara.

  Uppgift nummer 6 har vi tolkat som så att alla inparametrarna ska ha ett default värde så man kan utelämna dem om man vill. Sedan att de setters som finns för @age och @birthyear ska ändra på både @age och @birthyear när man använder dem.

  Uppgift nummer 7 och 8 har vi gjort en liknande tolkning för. Att man här ska kunna

använda dessa metoder direkt ifrån en int/string. Det vill säga att man kan kalla på dem direkt ifrån tex 1.fib eller ”sträng”.acronym och de då inte ska ta några inparametrar.

Uppgift 10 har vi tolkat det som att det kan vara vilka stora eller små bokstäver som helst innan “USERNAME: “ och vi då ska returnera det direkt efterföljande ordet efter matchningen mot “USERNAME: ”.

Uppgift 12 har vi tolkat det som att metoden ska returnera det första regnr som hittas i strängen

- Vad var svårt eller lätt med uppgiften?

  Uppgifterna i sig har inte varit särskilt utmanande än så länge utan det har varit mer att man inte vet hur man gör det man vill göra i själva språket än. När man är så ny på ett språk så blir det väldigt stapplande i början när man måste kolla upp syntaxen för nästan varenda grej man försöker göra.