TDP007 – Seminarium 1

Hur har ni gått tillväga för att lära er Ruby?
Vi har använt slidesen från föreläsning 1 och 2 i kursen för att lösa vissa uppgifter samt att förstå generell syntax av Ruby. För lite mer avancerade saker samt regex har vi använt oss av Rubys dokumentation samt forum posts på Stackoverflow. För regex har vi använt oss av hemsidan “Rubular” för att testa våra olika mönster. Vi använde oss av slidesen för att skapa en mall för vår testfil.

Vad finns det för nya konstruktioner i Ruby som ni inte sett förut?
I uppgiften för fakultet (uppgift 2) så använde vi en metod som kallas för inject. Denna var helt ny för oss och var lite klurig att lista ut till en början då den använder samma parametrar två gånger i metoden.

Block har vi inte heller sett förut. Till exempel när man ska kalla på ett block i en metod (*block.call()*) osv. Parameterinmatning till en metod med block är också helt nytt.

Vad finns det för konstruktioner som ni känner igen men som ser lite annorlunda ut?
Det mesta är sig likt men har sina skillnader, till exempel konstruktorer och vissa metoder. Det känns som att Ruby har tagit mycket från andra språk och gjort det simplare att förstå. Men kommer man direkt från andra språk är man ju såklart van vid att det funkar annorlunda.

Finns det något som ni irriterar er på eller tycker om i Ruby?
Det är riktigt kul att nästan allt man gör fungerar så som man tror att det ska. Det blir ett väldigt simpelt språk som man kan använda på många olika sätt, till exempel kan man skriva kod ungefär som i C++ men också som i Python.

Uppgift 1:
Hur har ni tolkat uppgiften?
Vi ska skapa en iteratorfunktion som gör någonting ett bestämt antal gånger. Det finns olika varianter på de här iteratorerna – en iteratorfunktion som heter n_times som gör något x antal gånger. N_repeat är en iteratorklass som gör samma sak men görs direkt i klassen med hjälp av medlemsfunktioner och medlemsvariabler istället för att använda en ensamstående funktion.

Vad var svårt eller lätt med uppgiften?
Det var svårt att förstå hur uppbyggnaden i klassen skulle se ut till en början. Vi trodde först att med trean i ”do_three = Repeat.new(3)” att objektet magiskt skulle ha trean i sig, men att vi istället behövde säga till medlemsfunktionen att använda sig av medlemsvariabeln som ”3” i det här fallet faktiskt är.

Uppgift 2:

Hur har ni tolkat uppgiften?

Uppgiften gick ut på att räkna ut fakulteten av 20 på en rad kod. Den andra delen är att göra en mer generell funktion för att räkna ut fakulteten av valfritt tal. Detta skulle göras med metoden range.inject.

Vad var svårt eller lätt med uppgiften?

Det var svårt att tolka hur ”inject” fungerade i början och dess parametrar. I början hade vi svårt att förstå vad:

```
{|result, entry| result * entry}
```

gjorde. Till slut förstod vi att insert uppdaterade result och entry för varje gång uträkningen hade körts, lite som att köra det i en sorts loop.
Sedan hade vi också problem att få output från lösningen, vi hade glömt att printa på rätt sätt. Det hjälpte att ha printen inuti funktionen istället för att printa hela funktionen.

Uppgift 5:

Hur har ni tolkat uppgiften?

Vi har tänkt objektorienterat, med polymorfi i och med att full-name ska vara ett virtuellt attribut. Getter blir därför vårt full-name(returnerar first- och last-name i följd) och i vår setter så sätter vi first- och lastname.

Vad var svårt eller lätt med uppgiften?

Det var svårt att förstå uppgiften och tänka sig ifrån C++ med scopes och liknande aspekter. I denna uppgift verkar inte scope ha någon betydelse utan det funkar ändå.

Uppgift 6:

Hur har ni tolkat uppgiften?

Vi ska skapa en class ”Person” med input av ett PersonName objekt för att skapa namnet, och sedan input av en int för att skapa åldern. Vi ska använda oss av Date eller DateTime för att ”tvåsidigt” kunna ändra birthyear efter åldern, men också kunna ändra åldern efter birthyear om vi sätter det statiskt istället för att köra det nuvarande året.

Vad var svårt eller lätt med uppgiften?

Att fixa den ”tvåsidiga” lösningen med age och birthyear tog rätt lång tid, vi försökte fixa dynamiskt att oavsett vilket år det var man kom tillbaka till lösningen så skulle age ha ändrats men inte birthyear (dock rätt överkurs).

Uppgift 8:

Hur har ni tolkat uppgiften?
Vi ska ta basklassen ”string” och utöka den med en metod som kallas för acronym, som tar första bokstaven i alla orden i strängen, skapar en ny sträng med dem i versaler som då skapar en förkortning, eller akronym, av dessa.

Vad var svårt eller lätt med uppgiften?
Syntaxen för att utöka en basklass samt hur man använde sig av motsvarigheten till ”this” i C++ (self. i Ruby).

Uppgift 9:
Hur har ni tolkat uppgiften?
Vi ska ta basklassen ”array” och utöka den med en metod som kallas för rotate_left, som tar första indexet i arrayn och sätter det sist i arrayen. Detta kan ske x antal gånger baserat på metodparametern som användaren matar in.

Vad var svårt eller lätt med uppgiften?
Det svåra var att lära sig syntax för arrays då den skiljer sig en hel del från tidigare språk vi läst. Här hjälpte Rubys officiella dokumentation en hel del.

Uppgift 10:
Hur har ni tolkat uppgiften?
Vi har tolkat uppgiften som att vi ska skapa en funktion som genomför en regex-sökning på en sträng som liknar ”Username: Brian”” så att allting mellan ett kolon och ett citattecken ska matchas, i detta fall ”Brian”.

Vad var svårt eller lätt med uppgiften?
Regex är alltid svårt, så mönstret var till en början rätt svårt att komma på. Vi använde oss av hemsidan ”Rubular” för att hjälpa oss. Sedan märkte vi att string.match inte returnerar just det vi sökte efter, utan allting i sektioner. Därför var vi tvungna att returnera just den sektionen vi letade efter.

Uppgift 11:
Hur har ni tolkat uppgiften?
Vi ska med hjälp av inläsning av källkod till en hemsida samt Regex ta ut alla HTML taggar som finns på en specifik URL.

Vad var svårt eller lätt med uppgiften?
Att få regex mönstret att ta bort ”<” och ”>”. Detta löste vi med hjälp av .gsub som tar bort ett specifikt mönster och byter ut mot ett annat. Grupper och match var återigen lite förvirrande men det löste sig till slut, internet var en stor hjälpreda här. Inläsning av hemsida och dess taggar var förvånansvärt lätt och felfritt.