Inlämningsuppgift och rapport versionshanteringssystem

**Beskrivning av inlämningsuppgiften:**

* I inlämningsuppgiften ska du visa att du har förstått hur man versionshanterar med Git, både i ett av dina egna projekt och en annan kursdeltagares projekt.
* Fokus ligger på versionshantering vilket innebär att du inte behöver programmera något stort eller lägga för mycket fokus på kod.

**Beskrivning av rapporten:**

* Du ska även skriva och lämna in en rapport (ett textdokument).  
  Detta dokument ska finnas i ditt git repo som du länkar vid inlämning.  
  ( Mer information om detta längst ned i detta dokument )

**Information om betyg:**

* På inlämningsuppgiften som består av praktiska uppgifter och att skriva kortfattade svar på några frågor kan du få något av följande betyg: **IG** och **G**.
* På rapporten som innebär att du beskriver lite mer om det du gjorde i inlämningsuppgiften och hur du använde dig av Git kan du få något av följande betyg: **IG, G och VG.**

**Principer för betygssättning:**

För betyget **Godkänd** ska den studerande

Ha nått samtliga läranderesultat för kursen

För betyget **Väl Godkänd** ska den studerande:

Uppnått kraven för betyget Godkänd

I sin rapport detaljerat förklara hur de olika koncepten hör ihop med den

inlämnade uppgiften

**Icke Godkänd** ges till studerande som har fullföljt kursen men inte nått alla mål för

kursen.

**Information om inlämning:**

* När du är klar med inlämningsuppgiften och rapporten ska du ladda upp en länk till ditt github repo.  
  Repot ska innehålla:
  + Ett textdokument med svaren på inlämningsuppgiftens frågor.
  + Ett separat textdokument med din rapport.

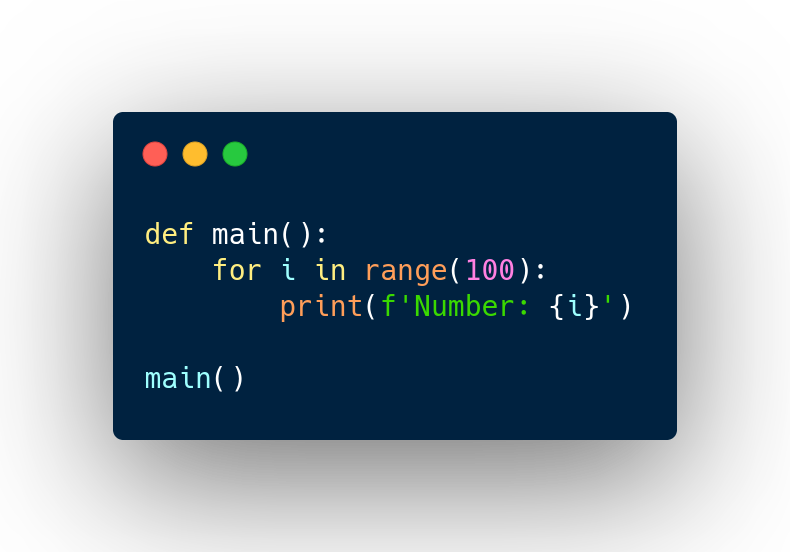
# Inlämningsuppgift

Genomför uppgifterna här nedanför och besvara kortfattat frågorna som står i anslutning till uppgifterna.

# Del 1.

### Steg 1

Först skapar du ett nytt Github repo.

I repot ska det finnas en \_\_main\_\_.py som ser ut så här:  
  


Det ska även finnas en “report.txt” och en “questions.txt” i ditt repo.  
(Du kan lämna dessa tomma just nu, men vid inlämning skall du som sagt populera dessa med din rapport, och frågor)  
  
Efter att \_\_main\_\_.py, report.txt och questions.txt existerar i din mapp, så ska du nu commita och pusha detta till Github (master).  
*“git push -u origin master”*

### Steg 2

* 1. Skapa en ny branch i ditt projekt (detta gör du via terminalen)  
     Döp branchen till “important-changes”  
       
     Nu, i denna nya branch så ska vi ändra på \_\_main\_\_.py.  
     Vi vill ändra i loopen så att den loopar 50 gånger istället för 100.  
       
     När du gjort ändringen, så commitar du och pushar dessa ändringar (stå kvar i branchen du skapade när du gör detta).  
       
     **Steg 3**
  2. Nu skall du göra en pull-request till ditt egna repo. (Via Githubs webbgränssnitt)  
     Skapa en pull-request från din nya branch “important-branch” in till master branchen.  
     Tryck sedan på “merge” i Githubs webbgränssnitt.

## Del 2

I den här delen av inlämningsuppgiften ska du göra en ändring i en kurskamrats projekt och lägga upp en pull request. Du ska också ta emot en pull request från en kurskamrat och merga den.  
(Du behöver inte lägga till kamraten som “collaborator” här, man kan göra pull-requests utan att vara “collaborator”)

### Steg 1

* 1. Be din kamrat om en länk till dennes Github repo.  
     Använd Githubs “fork” funktion för att “forka” din kamrats repo.  
     Klona sedan ned forken som hamnat under ditt namn på Github.  
       
     (Länk för hur man forkar: <https://docs.github.com/en/free-pro-team@latest/github/getting-started-with-github/fork-a-repo>)
  2. Skapa en ny branch via terminalen i det nyligen klonade repot.  
     Döp branchen till *“improvements”.*  
     Öppna upp projektet på din dator i valfri editor, gör en valfri ändring i \_\_main\_\_.py  
     Se till att du inte förstör något för din kamrat.  
     Pusha upp dina ändringar i din nya branch upp till Github.  
     Sedan gå in på Githubs webbgränssnitt, gå till din väns repo (inte din fork)  
     Sedan skapa en pull-request.  
     Nu är det upp till din kamrat att godkänna och merga din pull-request.

### Steg 2

1. Nu är det dags att **ta emot** ändringar från en kamrat som gjort en pull-request till ditt repo.
2. Invänta pull-requests från din kamrat, godkänn och “merga” pull-requesten från Githubs webbgränssnitt.

# Frågor

1. 1. Vad menas med versionshantering?
2. 2. Vad menas med en “merge” ?
3. 3. Om du vill merga ihop två brancher från terminalen, hur gör du då?
4. 4. Nämn ett alternativ till Git.
5. 5. Är Git ett verktyg som erbjuds av Github, eller kan man använda Git på andra ställen?
6. 6. Vad menas med en “merge conflict” och när kan det uppstå?
7. 7. Om du vill se en lista med tidigare ändringar / commits i ett Git repo via terminalen, vad skriver du?
8. 8. När du skriver “git init” i en mapp, på vilket sätt ändras då mappen?
9. 9. Varför vill man dela ibland upp sina ändringar i separata “brancher”?
10. 10. Nämn ett Git command som **inte** gör några förändringar på remote servern (Github är tex en remote server), Nämn även ett Git command som **gör** förändringar på remote servern.

# 

# Rapport

Skriv en sammanhängande text där du beskriver hur och varför du använde olika Git-kommandon i din inlämningsuppgift. Nedan punkter ska du använda dig av för att skriva din rapport.

* Beskriv hur du använde dig av Git i inlämningsuppgiften. Vilka Git-kommandon använde du och varför?
* Vad är fördelen med att använda sig av pull requests när man arbetar i ett team av systemutvecklare?
* Varför är det viktigt att använda sig av versionshantering? Vad var fördelarna i förhållande till inlämningsuppgiften? Vad är fördelarna med versionshantering och Git mer generellt sett?

Ju mer utförlig din rapport är, ju lättare är det för läraren att ge dig de betyg du önskar.