GruProg DD1331

Övning 1

Anna Karlhede
karlhed@kth.se

Repo: https://github.com/TypAnna/GruProgDD1331

Idag

- Kursens mål och upplägg
- Att lära sig programmera
- Repetition
- Uppgifter - komma igång
  - Git
  - Python
  - Idle
  - Bash

Kursens mål och upplägg

Mål

- Grunderna till programmering (såklart)
- Tänka som en programmerare
- Att ni ska lära er att studera på högskolenivå på ett hållbart sätt
  - Kontinuerligt
- Kunna skriva bra kod
  - Variabelnamn
  - Strukturering
  - Uppdelning

Att lära sig programmera

Att lära sig att programmera handlar mycket om att lära sig ett nytt sätt att tänka och lösa uppgifter på.

Några gyllene principer att förhålla sig till:

- Gör en övergripande plan för vad du behöver göra - rita gärna!
  - Observera att du inte behöver veta exakt *hur* du ska göra allt från början
- Bryt ner problemet i sina minsta beståndsdelar
  - Vad är det absolut första/simplaste du behöver göra?
- Skriv programmet så att det är lätt för andra att förstå!
  - Bra och tydliga variabelnamn
  - Kan du inte sammanfatta en metod i en mening så gör den mindre
- Lämna koden i ett tillstånd som är lätt att komma tillbaka till
- Om du sitter fast - beskriv vad programmet gör steg för steg, antingen för en kompis, eller för dig själv (högt!). Fantastiskt hur många problem/buggar som kan lösas på detta sätt.

Repetition

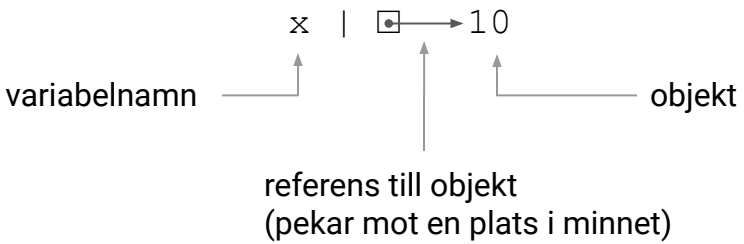
Tilldelning

Vad händer egentligen när vi skriver `x = 10`?

1. Ett objekt med typen `int` skapas
2. Detta objekt sätts till värdet `10`
3. Variabeln `x` binds till detta objekt

Kan tyckas självklart men det är viktigt att kunna vad som händer i bakgrunden.

Med bokens beteckningar:



Git & GitHub

Git

- Ett versionshanteringssystem för att hålla koll på, och på ett strukturerat sätt arbeta på sina filer.

Ditt lokala repository består av tre stycken olika “träd” som Git hanterar.

1. Working directory - det träd som håller alla dina filer och alla dina ändringar. Här “är” du när du sitter och kodar.
2. Index - det träd där du temporärt sparar alla dina ändringar som du sedan vill commita.
3. HEAD - det träd som innehåller din senaste commit.

GitHub

- Där du lagrar din kod (jmf Google Drive) och kan se historiken för alla filer (tillsammans med mycket mer…).

Git vs GitHub

- Git är ett verktyg du kan använda lokalt på din dator
- GitHub är ett onlineverktyg som bygger på Git, och som på ett smidigt sätt visualiserar bla historiken på filerna och tillhandahåller ett s.k. remote repository så du på ett smidigt sätt kan arbeta flera stycken på ett projekt.

Git & GitHub

När du har gjort ändringar till en fil, och är nöjd och vill “ladda upp” den till GitHub behöver du använda följande kommandon:

1. git add filnamn
    a. Med detta kommando så läggs filen till till vårt Index-träd.
    b. Säger åt Git att “denna fil har vi ändrat på och vill ladda upp”. Behöver göras för alla filer vi har ändrat på och vill ladda upp.
2. git commit -m “meddelande om ändringarna som gjorts”
    a. Med detta kommando läggs filen till till vårt HEAD-träd.
    b. Förbereder Git på att ladda upp alla filer vi har addat, och sparar ändringarna lokalt.
3. git push origin branch
    a. Först nu trycker vi faktiskt ut alla våra ändringar till våran remote repository (på GitHub) och gör ändringarna tillgängliga till andra. Branchen ni arbetar på kommer i regel vara master.

Hett tips: arbeta på små komponenter/delar av uppgiften i taget och när du vet att en del fungerar som den ska - add + commit + pusha den komponenten/delen!! Kommer leda till bättre kod, mer lättförståelig kod och mindre buggar 🦄

Uppgifter

Uppgift 1 - Grunderna i Git

Uppgiften:

- Skapa en utvecklingskatalog (= repository).
- Lägg till en Pythonfil i katalogen med ett enkelt program som skriver ut värdet av ett uttryck.
- Gör add, commit och push för att spara ändringarna inte bara lokalt utan på Git.

Bra praxis: ha ALLA dina programmeringsprojekt samlade i en och samma mapp i er hemkatalog (home directory, den kommer ni till genom att skriva `cd ~`).

Uppgift 2 & 3 - Idle och NameError

Uppgift 2: Starta Idle och skriv ett uttryck som ger NameError.

Ett NameError uppstår i python när vi försöker använda en variabel som ännu inte har blivit deklarerad.

Uppgift 3: Skriv kod i Idle som ser till att föregående uttryck inte längre ger name error. Gå tillbaka i kommandohistoriken för att köra uttrycket igen som nu ska fungera.

Vi kan gå tillbaka i historiken antingen med pil upp/ner, eller med ctrl+p/ctrl+n

Uppgift 4 - Python & Bash

Uppgiften: skriv ett Pythonprogram för att beräkna differensen mellan två tal som läses in från Bash.

Använd någon editor, tex Atom eller PyCharm.

En editor är som en ordbehandlare för programkod. Den förenklar helt enkelt ditt kodande med massa inbyggda funktioner (auto-tabbing, highlightning, auto-filling, git- och github-interactions, …)

Atom kan inte köra program (likt IDLE) så för det behöver du använda terminalen och Bash. Bash är programmet som körs i er terminal när ni öppnar den (ifall datorn är unix-baserad (linux och mac-datorer tex)). Kallas ett unix shell. När ni ex skriver `cd Documents` i er terminal så är det ett kommando som tolkas av bash och sen körs.

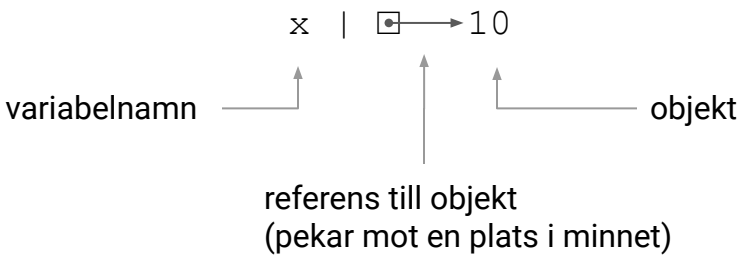
Mer info om Python, PyCharm och Atom hittar ni på kurswebben här. Mer info om Atom och Bash hittar ni på kurswebben här.

Uppgift 5 - två variabler, ett objekt

Uppgiften: Skriv ett Pythonprogram som får två variabelnamn att peka på samma objekt i minnet. Rita låd- och pildiagram över minnet.

Låd- och pildiagram!!!!!!!!!!! Viktigt!!!!!!!!!!!

Kom ihåg:

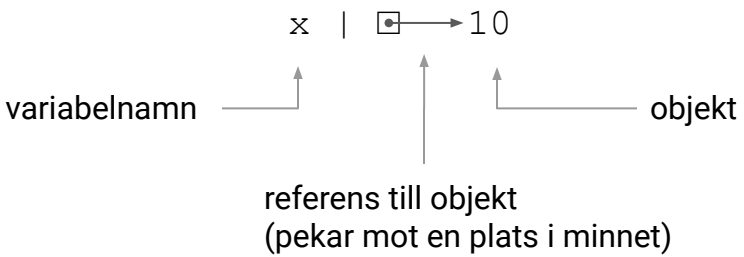


Uppgift 6 - två variabler, två likadana objekt

Uppgiften: Skriv ett Pythonprogram som får två variabelnamn att peka på likadana objekt i minnet. Rita låd- och pildiagram över minnet.

Låd- och pildiagram!!!!!!!!!!! Viktigt!!!!!!!!!!!!

Kom ihåg:



Uppgift 7 - reactiontimer.py

Uppgiften: skriv ett program, reaction timer som väntar en slumpmässig tid, presenterar texten "tryck enter" på skärmen, inväntar att användaren trycker enter och anger reaktionstiden. Eventuell bildfördröjning från skärmen (lag) ignoreras.

Vad behöver vi kunna göra?

- ta fram en randomiserat tal…
- mäta tiden…
- beräkna tidsfördröjningen och printa denna

Behöver vi skriva egna metoder för att ta fram ett random tal och mäta tid?! ;_;

- Nej - för det använder vi existerande moduler! 🌞😉