Slutuppgift

Observera att det under kursens gång är obligatoriskt att redovisa att man har en utvecklingsmiljö installerad (som man kan använda).

För G på slutuppgiften krävs:

- Koden gör åtminstone det som beskrivs i delarna som inte är utökning
- Koden går att köra
- Ni lämnar in koden tillsammans med en fil vid namn README.md som beskriver hur man gör för att köra erat program.
- Du som student kan redogöra för vad din kod gör
- All kod som är tagen utifrån markeras tydligt med källa, spara undan länkar under uppgiftensgång.
- Den övervägande majoriteten kod är skriven av dig själv.

För VG på slutuppgiften krävs dessutom:

- Koden är uppdelad i funktioner: Varje metod och klass har ett tydligt, isolerat syfte.
- Tydlig ansvarsfördelning: En metod ansvarar för att utföra en sak (och en sak enbart)
- Koden är objektorienterad
- Namnen på variabler har betydelse för en människa som läser koden

Goda råd vid lösning av programmeringsuppgifter

En god idé när man ska sätta sig och lösa en programmeringsuppgift är att förbereda sig innan man börjar hacka på tangenterna. Det är lite som att läsa ett recept och kolla att man har alla nödvändiga ingredienser hemma. INNAN man börjar laga mat.

Steg 1. Förstå kraven

- Läs igenom instruktionerna och vad ditt program ska göra.
- Läs instruktionerna igen – noggrant
- Testa att förklara problemet för någon annan (lämpligen någon som har samma problem som du). Ni kommer att se olika delar av problemet och olika lösningar.

Steg 2. Individuellt konkretisera dina krav:

- Skriv ned minst 3 olika uppsättningar av indata och vad du borde få för utdata av att köra programmet.
- Gå igenom steg för steg hur en dator skulle kunna lösa problemet för dina olika uppsättningar indata. Skriv inte detaljerad kod utan håll dig till att skissa problemet.
- Ta ett steg tillbaka och kolla på ditt program:
    a. Kan programmet förenkla programmet med villkor och loopar?
    b. Finns det onödiga upprepningar av steg?
    c. Förenkla, optimera.

Steg 3. Pseudo kod

- Skriv mer utförlig s.k. pseudo kod med fokus på logik och alla steg (inte på att språket ska vara100% korrekt Python)
- Gå igenom din Pseudokod steg för steg och bekräfta att den löser problemet.

Steg 4. Översätt till riktig kod

- Du kan exempelvis skriva in din pseudo kod som kommentarer i Python.
- Skriv sedan din riktiga kod bit för bit emellan kommentarerna
- Testa och “debugga”

Prova gärna s.k. “Rubber duck debugging” - om du stöter på ett problem medan du kodar, försökförklara ditt problem som om du pratade med en badanka som aldrig hört talas omprogrammering. När man talar högt kommer man ofta plötsligt på vad det är som är problemet.

Steg 5. Försök göra koden mer elegant

- Förenkla och optimera igen.
- Ta bort onödiga saker.
- Försök göra koden mer lättläst.
- Kommentera koden! Det som känns självklart för dig just nu är sällan det en månad senare.

DRY - Don’t repeat yourself. Känner du att du har flera ganska lika metoder i ditt program? Titta på vad metoderna hargemensamt och om du kan isolera den gemensamma delen i en ny, separat metod. Testa och “debugga” igen efter att du gjort koden mer elegant (du ju kan ha råkat ta bort något viktigt)

Steg 6. Innan inlämning

- Kontrollera att alla källor är angivna korrekt
- Kontrollera att programmet kör enligt README.md
- Kontrollera lösningen med olika data/inmatning (tips använd dig av steg 2)