Tentamen för TDP002 Imperativ programmering
2010-01-10 08-12

Instruktioner:

- Skriv svar på varje fråga i en separat fil i hemkatalogen (~) namngiven enligt mönstret `login-uppgift.(txt)|(py)` exempelvis `olale507-1.txt` eller `fregy123-2.py`.
- Uppge namn och personnummer i varje inlämnad fil.
- Uppgifterna löses enskilt.
- Kursboken *Learning Python* är tillåten.
- Kommunikation med andra under tentamenstillfället är förbjudet.

Fråga:	1	2	3	4	5	6	7	8	9	10	11	12	Totalt
Poäng:	3	3	2	2	1	1	2	2	4	3	2	3	28

För betyg 3 gäller 50% rätt, för betyg 4 65% och för betyg 5 80%, avrundat till närmaste hela poäng.
Både rätt innehåll och välformulerad text krävs för full poäng på uppgifterna.
Det totala betyget på tentamen är detsamma som det lägsta betyget av teoridel och problemdel.

Teori

1. (3 poäng) Förklara vad kostnaden för att använda ett programspråk är och vilka egenskaper hos programspråk som bidrar till kostnaden. I kursboken Concepts of Programming Languages anges sex olika egenskaper. Varje egenskap som anges och förklaras ger 0.5 poäng.

2. (3 poäng) Vilket enskilt koncept hör begreppen nedan till (på engelska inom parentes)? Ange konceptet och förklara begreppen.

   - namn (name)
   - adress (address)
   - värde (value)
   - typ (type)
   - livslängd (lifetime)
   - räckvidd (scope)

3. (2 poäng) Förklara begreppet delad referens (shared reference) med hjälp av körexemplet nedan. Förklara allt som händer med avseende på begreppen namn, adress och värde enligt ovan.

```
>>> a=[1,2,3]
>>> b=a
>>> c=100
>>> d=c
>>> c=200
>>> d
100
>>> b[2]
3
>>> b[2]+=2
>>> b
[1, 2, 5]
>>> a
[1, 2, 5]
```

4. (2 poäng) Beskriv vad som menas med stark respektive svag typning av variabler. Är Python ett exempel på ett starkt eller svagt typat språk? Ange exempel med (pseudo-)kod på vad begreppet innebär.

5. (1 poäng) Förklara reglerna för *operator precedence* med hjälp av ett eget kodexempel som visar när det spelar roll.

6. (1 poäng) Ange hur man räknar ut värdet av det oktadecimala talet 31 på decimalform.

7. (2 poäng) Förklara vad som menas med att skapa en egen datatyp i Python och ge ett kort exempel på en sådan definition.

Python

8. (2 poäng) Implementera en funktion `my_reduce` som accepterar tre in-argument:

   - en funktion som tar två argument
   - en lista
   - ett första-värde

Funktionen som utgör den första parametern ska sedan appliceras på föregående delresultat och nästa element i listan. Första gången hämtas det föregående resultatet från den trede parametern. Resultat av att evaluera uttrycken `my_reduce(alpha, [1,2,3], "hej")` ska alltså vara likvärdigt med `alpha(alpha(alpha("hej", 1), 2), 3)`.

9. (4 poäng) Skriv en funktion som läser in rader från en textfil (skapa en textfil som exempel), sorterar raderna efter längd och skriver dem till en ny fil. Gör din funktion till ett program som kan köras direkt från kommandoraden med namn på filen att läsa och filen att skriva till som argument.

10. (3 poäng) Givet koden i `~/given_files/demo-test.py`, skapa ett program som du kan verifiera fungerar med hjälp av testerna i `demo-test.py` och visa att programmet fungerar genom att kopiera in en testkörning.

11. (2 poäng) Förklara vad som är skillnaden mellan en funktion och en procedur i Python och ge exempel på båda.

12. (3 poäng) Gör en funktion som fungerar som översättare. Funktionen ska översätta siffror mellan 0-99 till motsvarande svenska ord. Till exempel ska argumentet `17` göra att översättaren returnerar `"sjutton"`. Ange också tydligt hur den skulle kunna generaliseras till godtyckligt stora tal.