Övningar för Python, del 2

Daniel Bosk[1]

Avdelningen för informations- och kommunikationssytem (IKS),
Mittuniversitetet, Sundsvall.

python2.tex 1348 2013-09-30 08:52:32Z danbos

Mittuniversitetet
MID SWEDEN UNIVERSITY

Övningar

1. Utöka speed1.py så att användaren kan ange enhet på inmatat data istället för att måsta ange med en given enhet.
2. Utöka programmet vidare med en slinga som låter användaren köra fler än en konvertering per körning.
3. Utöka programmet med en loggningsfunktion som skriver alla konverteringar till en fil på disk.

Lösningsförslag I

```
1  #encoding: utf8
2
3  import sys
4
5  def Mbps2Mibps( speed ):
6    # 1 Mbit/s = 1000 kbit/s = 1000000 bit/s
7    # 1 Mibit/s = 1024 Kibit/s = 1024^2 bit/s =
       1048576 bit/s
8    return speed * ( 1000000 / float(1048576) )
9
10 # konverteringsfunktioner till och från bit/s
     för olika storhetsprefix
11 def Gbps2bps( speed ):
12   return speed * 1000000000
13
14 def bps2Gbps( speed ):
15   return float( speed ) / 1000000000
```

Lösningsförslag II

```
16
17 def Mbps2bps( speed ):
18   return speed * 1000000
19
20 def bps2Mbps( speed ):
21   return float( speed ) / 1000000
22
23 def kbps2bps( speed ):
24   return speed * 1000
25
26 def bps2kbps( speed ):
27   return float( speed ) / 1000
28
29 def Mibps2bps( speed ):
30   return speed * 1048576
31
32 def bps2Mibps( speed ):
33   return float( speed ) / ( 1024*1024 )
```

Lösningsförslag III

```
34
35 def kibps2bps( speed ):
36   return speed * 1024
37
38 def bps2kibps( speed ):
39   return float( speed ) / 1024
40
41 # returnera antalet bytes (utan enhetsprefix)
42 def parse_data( s ):
43   # 1 MB = 1000 kB = 1000000 B
44   # 1 MiB = 1024 KiB = 1024^2 B = 1048576 B
45   s = s.split() # "123 m" -> ["123", "m"]
46   data = float( s[0] )
47   # kontrollera att listan faktiskt innehåller
         två element
48   if ( len( s ) < 2 ):
49     raise Exception( "Enhetsprefix␣saknas" )
50   prefix = s[1]
```

Lösningsförslag IV

```
51
52      # kolla enhetsprefix och multiplicera med
            lämpligt tal
53      if ( prefix == "k" ):
54        #data = data * 1000
55        data *= 1000
56      elif ( prefix == "Ki" ):
57        data *= 1024
58      elif ( prefix == "M" ):
59        data *= 1000*1000
60      elif ( prefix == "m" ):
61        # för millibytes
62        data /= 1000
63      elif ( prefix == "Mi" ):
64        data *= 1024*1024
65      elif ( prefix == "G" ):
66        data *= 1000*1000*1000
67      elif ( prefix == "Gi" ):
```

Lösningsförslag V

```
68      data *= 1024**3
69    elif ( prefix == "T" ):
70      data *= 1000**4
71    elif ( prefix == "Ti" ):
72      data *= 1024**4
73    else:
74      # ge ett särfall med tillhörande
            felmeddelande
75      raise Exception( "Enhetsprefix ’" + prefix
            + "’ finns inte" )
76
77    return data
78
79  # returnera tiden i sekunder
80  def parse_time( s ):
81    s = s.split()
82    time = float( s[0] )
```

Lösningsförslag VI

```
83    # kontrollera att listan faktiskt innehåller
          två element
84    if ( len( s ) < 2 ):
85      raise Exception( "Tidsenhet saknas" )
86    # ta ut tidsenheten
87    unit = s[1]
88
89    # undersök vilken tidsenhet som användaren
          vill ha
90    if ( unit == "min" ):
91      time *= 60
92    elif ( unit == "h" ):
93      time *= 60*60
94    elif ( unit == "d" ):
95      time *= 24*60*60
96    elif ( unit == "y" ):
97      time *= 365*24*60*60
98    # men enbart "s" är SI-enhet, så undvik "sek"
```

Lösningsförslag VII

```
 99    elif ( unit == "s" or unit == "sek" ):
100      time *= 1
101    else:
102      # ge ett särfall med tillhörande
             felmeddelande
103      raise Exception( "Tidsenheten␣'" + unit +
             "'␣finns␣inte" )
104
105    return time
106
107  # gör om huvudprogrammet till en separat
       funktion för att underlätta
108  # för while-slingan (mindre kod där, mer
       modulär kod generellt också).
109  # funktionen tar ett argument logfile som är en
       öppen fil som vi kan
110  # skriva loggen till.
111  def interactive_convert( logfile ):
```

Lösningsförslag VIII

```
112   # använd raw_input för python2 och input för
          python3
113   data = ""
114   while ( data == "" ):
115     data = input("Ange␣datamängd␣i␣bytes␣(ange␣
            enhetsprefix␣efter" \
116           "mellanslag,␣0␣för␣avslut):␣")
117   if ( data == "0" ):
118     # alternativt sys.exit( 0 ), båda fungerar
119     return True
120   # skriv till loggen
121   logfile.write( data + "\n" )
122   try:
123     data = parse_data( data )
124   except Exception as e:
125     print( e )
126     sys.exit( 1 )
127
```

Lösningsförslag IX

```
128   print( "Datamängden␣är␣" + str( data ) + "␣
          bytes." )
129   logfile.write( str( data ) + "␣bytes\n" )
130
131   # använd raw_input för python2 och input för
          python3
132   time = input("Ange␣tid␣(enhet␣efter␣
          mellanslag):␣")
133   logfile.write( time + "\n" )
134   try:
135     time = parse_time( time )
136   except Exception as ex:
137     print( ex )
138     sys.exit( 1 )
139
140   print( "Tiden␣är␣" + str( time ) + "␣
          sekunder." )
141   logfile.write( str( time ) + "␣sekunder\n" )
```

Lösningsförslag X

```
142
143     # vill ha hastigheten i bit/s (8 bitar per
            byte)
144     speed = 8 * data / time
145     logfile.write( str( speed ) + " bit/s\n" )
146
147     print( "Det ger en hastighet på ")
148
149     # vi behöver en logfile.write för varje sak
            vi vill skriva till loggen,
150     # många blir det.  för att underlätta skapar
            vi en ny variabel newspeed
151     # som kan agera mellanlagring.
152     newspeed = ""
153     if ( speed > 1000000000 ):
154       newspeed = str( bps2Gbps( speed ) ) + "
              Gbit/s"
155     elif ( speed > 1000000 ):
```

Lösningsförslag XI

```
156       newspeed = str( bps2Mbps( speed ) ) + "␣
              Mbit/s"
157     elif ( speed > 1000 ):
158       newspeed = str( bps2kbps( speed ) ) + "␣
              kbit/s"
159     else:
160       newspeed = str( speed ) + "␣bit/s"
161
162     print( newspeed )
163     logfile.write( newspeed + "\n" )
164
165     return False
166
167 ################################
168 # Huvudprogrammet börjar nu här
169 ################################
170
171 # försök att öppna loggfilen för skrivning
```

Lösningsförslag XII

```
172 try:
173   logfile = open( "convert.log", "w" )
174 # meddela eventuella fel och avsluta
175 except Exception as e:
176   print( e )
177   sys.exit( 1 )
178
179 # kör konverteringen så länge användaren inte
      vill avsluta
180 quit = False
181 while ( not quit ):
182   # vi skickar med loggfilen och får tillbaka
        om användaren vill
183   # avsluta eller ej
184   quit = interactive_convert( logfile )
185
186 # glöm inte att stänga loggfilen
187 logfile.close()
```

Lösningsförslag XIII