SAMMANFATTNING ÖVNING 2

DANIEL BOSK

DD13{10,11,14} Programmeringsteknik (prg), Kungliga Tekniska Högskolan*

1. Datatypen bool

bool är en datatyp som, till skillnad från t.ex. int, kan ta enbart två olika värden – True och False (skrivs också vanligtvis som 1 respektive 0)[1].

2. Logiska operatorer

På denna datatyp `bool` finns några operatorer definierade, likt `+ - * /` för flyttal. Dessa operatorer är `and or not`. `not` är en operator som skiljer sig något från de tidigare operatorerna vi tittat på, den är unär, d.v.s. den opererar enbart på en operand, till skillnad från binära operatorer som opererar på två operander. Vi sammanfattar funktionaliteten hos dessa operatorer i tabellerna 2 och 2.

Operation	Resultat
not True	False
not False	True

Table 1. Sanningtabell för Not

and	True	False	or	True	False
True	True	False	True	True	True
False	False	False	False	True	False

Table 2. Sanningstabeller för And och Or

Några exempel från Python-tolken:

```
Python 2.2.3 (#1, Jan  5 2005, 16:36:30)
[GCC 3.4.2] on sunos5
Type "help", "copyright", "credits" or "license" for more information.
>>> True or False
1
>>> True and False
0
>>> not True
0
>>> not False
1
>>> not True and False
0
>>> not (True and False)
1
>>> not False and True
1
```

Date: 21 september 2010.

[1]Detta genom typkonvertering.

```
>>> not (False and True)
1
>>>
```

3. Jämförelseoperatorer

Precis som de aritmetiska och logiska operatorerna finns det operatorer för jämförelse definierade för de vanliga datatyperna i Python. Dessa är `==` `<` `>` `<=` `>=` `!=`. De används på samma sätt som de aritmetiska operatorerna, men skiljer sig genom att de ger upphov till ett värde som är av boolesk typ (`bool`) istället för ett värde av samma typ som operanderna. Några exempel:

```
>>> 5+5
10
>>> 5 < 5
0
>>> 3< 5
1
>>> 5==5
1
>>> 4 <= 5
1
>>> 5 <= 5
1
>>>
```

4. Villkorssatser

Villkorssatser används för att kontrollera programflödet, d.v.s. i vissa fall vill vi att programmet ska göra en sak och i andra fall något annat. För detta behöver vi olika villkor.

Ett villkor är ett uttryck som kan evalueras till antingen True eller False, exempelvis `x < 5` eller `x < 5 and x != 0`.

En typ av villkorssats är `if-elif-else`-satsen. Den fungerar på följande sätt:

```
1 if <villkor>:
2     # kod
3 # en eller flera elif-satser
4 elif <annat_villkor>:
5     # kod
6 else:
7     # kod
```

Notera att man inte nödvändigtvis behöver ha med varken `elif` eller `else`, och man kan ta med hur många `elif` man önskar.

När något villkor är sant ignoreras alla efterföljande `elif` och `else` även om deras villkor skulle vara sanna. Ett exempel:

```
>>> x=5
>>> if x < 5:
...     print "less than 5"
... elif x < 10:
...     print "less than 10"
... else:
...     print "too large"
...
less than 10
>>>
```

5. Slingor

Slingor används för att få dynamiska upprepningar i programmet, t.ex. när vi vill läsa in en fil eller gå igenom en lista. Det finns två olika loop-konstruktioner som tas upp här, de är for- och while-satserna.

for-satsen fungerar på följande sätt:

```
for <variabel> in <lista>:
    # kod
```

Koden i for-loopen kommer att köras en gång för varje element i listan, varje gång kommer variabeln ha värdet av ett element i listan. Ett exempel i python-tolken:

```
>>> for x in [1, 2, 3]:
...     print "hej"*x
...
hej
hejhej
hejhejhej
>>> for i in range(10):
...     print i,
...
0 1 2 3 4 5 6 7 8 9
>>>
```

Den andra loop-konstruktionen är while-satsen. Den kör sin kod så länge ett givet villkor är sant. Den börjar med att kolla villkoret, om det är sant körs koden, annars fortsätter exekveringen direkt efter while-loopen.

```
while <villkor>:
    # kod
```

Notera att den enbart kollar villkoret en gång per varv, så hela koden körs trots att villkorets värde ändras under körningens gång. Exempel:

```
> i=0
>>> while i<10:
...     print i,
...     i += 1
...
0 1 2 3 4 5 6 7 8 9
>>>
```

6. Programexempel

Det första programmet, `02-sten.py`, är ett litet program som tar upp de grundläggande begreppen för denna övning, och däribland också nästlade slingor.

```
def rektangel(rad, kol):
    for i in range(rad):
        s = ''
        for j in range(kol):
            s = s +'*'
        print s

rektangel(2,3)

def palindrom(s):
    langd = len(s)
    p = True
    for i in range(langd/2):
        if s[i] != s[langd-i-1]:
            p = False
```

```
16      return p
17
18
19  ord = 'Vadsomhelst'
20  while ord != '':
21      ord = raw_input('Ange ett ord: ');
22      if palindrom(ord):
23          print ord + ' är ett palindrom'
24      else:
25          print ord + ' är ej ett palindrom'
```

Det andra programmet, `02.py`, är ett större program som beräknar portot för angivna brev- och paketvikter.

```
1   #encoding: latin1
2   # tar upp bregreppen:
3   ## typkonvertering
4   ## funktion (med returvärde)
5   ## aritmetiska operatorer
6   ## jämförelseoperatorer
7   ## logiska operatorer
8   ## while-slinga
9   ## for-slinga
10  ## range()
11  ## if-elif-else-sats
12
13  # weight är vikten, i gram, hos ett brev.
14  # rätt porto returneras.
15  def check_postage(weight):
16      ## POSTENS 1:a KLASS INRIKES BREV
17      if (weight <= 20):
18          return 6
19      elif (weight <= 100):
20          return 12
21      elif (weight <= 250):
22          return 24
23      elif (weight <= 500):
24          return 36
25      elif (weight <= 1000):
26          return 48
27      elif (weight <= 2000):
28          return 72
29      ## POSTENS POSTPAKET
30      elif (weight <= 3000):
31          return 150
32      elif (weight <= 5000):
33          return 175
34      elif (weight <= 10000):
35          return 225
36      elif (weight <= 15000):
37          return 275
38      elif (weight <= 20000):
39          return 320
40      return -1
41
42
43  ########################
44  # Huvudprogrammet
45  ########################
46
47  # Loopa så länge som möjligt
48  while (True):
49      print "Välkommen till Brevvågen"
```

```
50      nletters = raw_input("Hur många brev vill du beräkna porto för (0
            " \
51          "för att avsluta): ")
52      nletters = int(nletters)
53      # 0 för att avsluta (negativa också)
54      if (nletters <= 0):
55          break
56      sum = 0
57      for i in range(nletters):
58          w = raw_input("Hur mycket väger brev " + str(i+1) + ": ")
59          postage = check_postage(float(w))
60          if (postage < 0):
61              print "Den vikten finns inte med i portotabellen."
62          else:
63              print "Du måste betala", postage, "SEK i porto."
64              sum += postage
65      if (nletters > 1):
66          print "Det blir", sum, "SEK för alla brev, tack!"
67      elif (sum > 100 and nletters == 1):
68          print "Mycket pengar för ett postpaket!"
```