SAMMANFATTNING ÖVNING 1

DANIEL BOSK

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

1. Datatyper

Man kan säga att nästan allt i Python är av någon datatyp. Dessa datatyper håller Python själv reda på, men man måste även som programmerare hålla koll på dessa för att programmet ska fungera som man vill.

De datatyper som Python definierar är bl.a.

int: Heltal (Integers),
float: Flyttal (Floating point numbers),
str: Strängar (Strings).

2. Aritmetiska operatorer

På dessa datatyper finns operatorer definierade, t.ex. `+` är en operator definierad för heltal. Andra operatorer som finns är `+ - * / // %`. Några exempel (som körs i Pythons tolk från terminalen):

```
\$ python
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.
>>> 5+5
10
>>> 3*2
6
>>> 5//3
1
>>> 5\%3
2
>>> "hej"+"svejs"
'hejsvejs'
>>> "hej"*3
'hejhejhej'
>>>
```

Alla operatorer är dock inte definierade för alla datatyper, en mycket kort och ej fullständig sammanfattning är:

Heltal: kan använda operatorerna `+ - * / // %`,
Flyttal: kan använda operatorerna `+ - * /`, heltalsdivision och modulo går inte att beräkna för flyttal,
Strängar: har bara operatorerna `+ *`.

Date: 21 september 2010.

3. Konstanter

För att förenkla beräkningar etc. har vi något som kallas för konstanter. Dessa används för att underlätta när vi har återkommande värden, t.ex. π. En konstant i Python skapas på följande vis PI=3.14, och kan sedan användas genom att man skriver PI istället för 3.14 överallt i sin kod. En exempelkörning i Python kan se ut enligt följande.

```
>>> PI=3.14
>>> 2*PI
6.2800000000000002
>>>
```

Konstanter kan också användas för vissa restriktioner som man kanske har i sin kod, t.ex. att ett namn måste vara 32 tecken. Om man senare kommer på att namnet borde kunna vara 64 tecken långt behöver man bara ändra värdet på konstanten[1] istället för att leta upp alla ställen i koden där det används.

4. Identifierare

Vilka regler gäller då för namnet på konstanterna? De får bestå av bokstäver, siffror och även understreck (_), men de får dock inte börja med siffror. De får heller inte vara något av följande reserverade ord:

```
and assert break class continue def del elif
else except exec finally for from global if
import in is lambda not or pass print raise
return try while
```

Viktigt att tänka på är att man skiljer på gemener och versaler, d.v.s. PI är inte samma sak som Pi.

5. Variabler

En variabel är likt en konstant en identifierare, men istället för att representera ett värde representerar variabeln ett minnesutrymme i vilket man kan lagra data. De fungerar precis som konstanterna, med skillnaden att man faktiskt kan ändra värdet som lagras i variabeln. Namnet uppfyller kravet för en identifierare, men får inte vara enbart versaler (för det tolkar Python som en konstant).

```
>>> x=5
>>> y=3
>>> z=x*y*3
>>> print x, y, z
5 3 45
>>> print x
5
>>> print y
3
>>> print z
45
>>> x=x+1
>>> print x
6
>>> print z
45
>>>
```

[1]Med att ändra värde på konstanten menare jag att man ändrar i koden, värdet på en konstant kan inte ändras under programkörning.

Notera att värdet på z inte ändras när vi ändrar värdet på x, detta för att det är värdet 45 som lagras i z och inte relationen x*y*3.

6. Typkonvertering

Ibland kan man vilja konvertera vissa typer till andra. Det kan vara att man läst in en sträng från tangentbordet och vill konvertera den till ett tag (om användaren matade in ett tal). Ett exempel i Python-tolken:

```
>>> x="3.14"
>>> pi=float(x)
>>> print pi
3.14
>>> print x+2
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  TypeError: cannot concatenate 'str' and 'int' objects
>>> print pi+2
5.14
>>>
```

Ett annat bra exempel är procentberäkningar,

```
>>> 99/100*100
0
>>> float(99)/100*100
99.0
>>>
```

där man får fel svar om man inte explicit typkonverterar.

7. Funktioner

Funktioner utgör en viktig byggsten i programmeringen, liksom inom matematiken kan den användas för att dela upp större problem i mindre och bidra till en bättre ordning. De används dessutom på samma sätt.

$$f(x) = 2 \cdot g(x) + 3$$
$$g(x) = x^2$$

inom matematiken skulle kunna skrivas i Python som

```
1 def g(x):
2     return x*x
3
4 def f(x):
5     return 2*g(x)+3
```

När vi talar om funktioner består de av flera delar, funktionen består av ett funktionshuvud och en funktionskropp. Funktionshuvudet är den första raden i funktionsdefinitionen. Den består av funktionsnamn och formella parametrar. D.v.s.

$$\overbrace{\text{def functionname}(\underbrace{\text{argument1, argument2, ...}}_{\text{formella parametrar}}):}^{\text{funktionshuvud}}$$

Resten av funktionen är funktionskroppen, d.v.s. all kod som hör funktionen till.

8. Ett programexempel

Här kommer ett exempel på ett program, `O1.py`, som gör någonting vettigt. Det beräknar kinetisk och potentiell energi för en boll.

```
#encoding: latin1
# energiprogram
## tar upp följande begrepp:
##reserverade ord
##konstanter
##variabler
##datatyper
##lokal variabel
##global variabel
##formell parameter
##anropsparameter
##typkonvertering
##funktionshuvud
##funktionskropp
##funktion med och utan returvärde
##

import math

def kinetic_energy(m,v):
    return 0.5*m*math.pow(v,2)

def potential_energy(m,h):
    G = 9.82
    return m*G*h

def skriv(e1, e2):
    print "Bollens rörelseenergi = ", e1, "Joule"  # alt + str(e1)
    print "Bollens potentiella energi", e2, "Joule"

svar = raw_input('Ange bollens massa: ')
m = float(svar)

svar = raw_input('Ange bollens höjd: ')
h = float(svar)

svar = raw_input('Ange bollens hastighet: ')
v = float(svar)

e1 = kinetic_energy(m,v);
e2 = potential_energy(m,h);

skriv(e1,e2)
```

En körning av detta program (från en terminal) kan se ut så här:

```
$ python O1.py
Ange bollens massa: 2
Ange bollens höjd: 3
Ange bollens hastighet: 4
Bollens rörelseenergi =  16.0 Joule
Bollens potentiella energi 58.92 Joule
$
```

9. Tips och kommentarer

Några punkter att tänka på:

- Blanda aldrig språk, antingen skriver ni alla variabelnamn och funktioner på engelska eller på svenska – aldrig båda!
- Använd förklarande namn till alla variabler och funktioner!
- Skriv kommentarer!
- Dela upp programmen i mindre delar – funktioner! Det blir lättare att följa och det blir snyggare kod, men framför allt mycket enklare att programmera!