TDP002 - Imperativ programmering

Pontus Haglund Emma Enocksson

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

1 Programmering - en översikt
2 Python
  Inbyggda typer
  Satser och uttryck

Vad är programmering?

Programmeringsspråkens nivåer

- **Maskinspråk**: datorns eget språk, binärkod (1 eller 0)
- **Assemblerspråk**: mnemoniskt maskinspråk, t.ex. `"MOV A.#0FFH"`
- **Högnivåspråk**: språk utvecklade för att passa mänskliga programmerare

1943 Einac Code
1950 Short Code
1954 FORTRAN
1958 Algol 58, Lisp
1959 COBOL
1964 Basic
1967 Simula
1971 Pascal
1972 **C**, Prolog, ML
1976 Smalltalk
1977 **Bourne Shell**
1983 **C++**, Ada
1987 Perl
1991 **Python**, VB
1993 **Ruby**
1994 PHP
1995 **Java**
2000 **C#**

Syntax och semantik

- **Syntax**: *formatet* på språkets konstruktioner
- **Semantik**: *betydelsen* hos språkets konstruktioner
- Varje språk har **unik syntax**
- Språket ingår ofta i en specifik **paradigm** - en gemensam semantisk kärna med vissa variationer
- I våra kurser: vi fokuserar på principerna så att ni kan paradigmen
  - Språket väljs som exempel av paradigmen
  - Viktigt att se principerna i exemplet

Programmering - en introduktion

Python och ada - en skillnad i syntax...

Python:

```
if x == 2 and y == 3:
  print("Hej")
```

Ada:

```
if x = 2 and y = 3 then
  Put_Line("Hej");
end if;
```

Villkor

```
for x in range(1,6):
  print(x)
```

```
for x in 1..5 loop
  Put(x);
  New_Line;
end loop;
```

Iteration

... men väldigt lika semantik!

Tips för att bli en bra programmerare

- Att lösa problemet är intressant
- Ju svårare desto bättre
- Fokusera på ett (del-)problem i taget
- Klienterna kommer till dig för att det *är* svårt
- Arbeta systematiskt
- Välj dina uppdrag med omsorg
- **Övning ger färdighet!**

1 Programmering - en översikt

2 Python

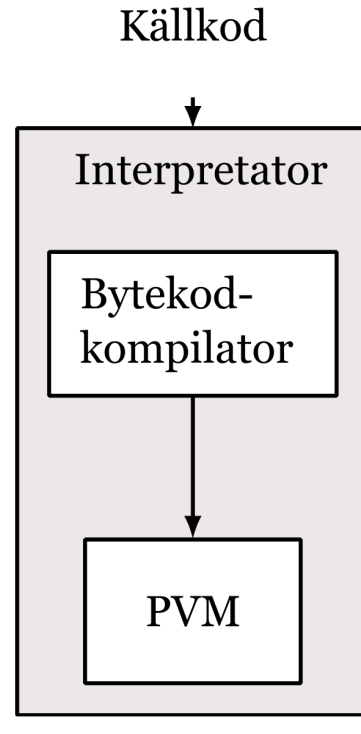
Inbyggda typer

Satser och uttryck

LINKÖPING UNIVERSITY

Python - att köra och skriva program

- Pythonkod skrivs i en texteditor och sparas med filändelsen .py
- Koden matas sedan in i interpretatorn som tolkar och kör koden
- Interpretatorn har en inbyggd bytekod-kompilator
- Python-bytekod exekveras (körs) av ”Python virtual machine”(PVM)



Tre olika sätt att exekvera Pythonkod

- Interaktivt via kommandorad
  - DOS, IDLE, terminal
  - print onödig, möjligt att ladda fil, inga blanka rader
  - Bra vid experimenterande, lätt att testa kod interaktivt

Tre olika sätt att exekvera Pythonkod

- Anorpa fil från kommandorad
  - `$ python3 program.py`
  - Linux/UNIX:
    överst i filen:
    ```
    #! /usr/bin/env python3
    ```
    ```
    $ chmod u+x program.py
    $ ./program.py
    ```
- Generell editor/IDE: t.ex. emacs med C-c C-c

Python - Historia

- Version 0.9 lanserades 1991
  - Grunder från språket ABC (liknar Basic)
  - Python 2.7 och 3.7 nuvarande stabila versioner
  - Öppen källkod
- Programming for everybody
  - Multiparadigm: Imperativt, funktionellt, objektorienterat
  - Skriptspråksfamiljen: Perl, Python, Tcl, BASH, etc.
  - Rapid development: bra både för nybörjare och erfarna
  - Batteries included: standardbibliotek som kan göra det mesta.

Python - Grunderna

- Kraftfullt skriptspråk
  - script = manus, manuscript
  - Kommandospråk, dvs används även interaktivt
  - Bra både för enradare och större program
  - Går att använda för att sammanfoga olika system - har bra stöd för filer, strängar och operativsystemsanrop
  - Går att bädda in i andra språk, t.ex. C och Java
- Generellt lättanvänt språk
  - Enkla konstruktioner: typsystem, modulhantering, exekvering
  - Korta program med stort stöd från ett stort standardbibliotek

Vårt första Pythonprogram

hello.py:

```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

print('Hello World!')
```

```
$ chmod u+x hello.py
$ ./hello.py
Hello World!
$
```

Enkla Pythonprogram fungerar som skript, dvs kommandon på fil

Lab 0 - filer och moduler

```
>>> import imp, os
>>> help(os)
  (Hjälp för modulen os)
>>> import hello
Hello World!
>>> imp.reload(hello)
Hello World!
>>>
```

- Python Standard Library
  - Utvidgar kärnan med kommandon och funktioner som är bra att ha
  - `http://docs.python.org/3/`
  - Källkod: `/usr/lib/python`
- `import` laddar in en modul
- `help` interaktiv hjälp (jmf `man`)
- `imp.reload` laddar om en modul

Typer - Tal

```
>>> 20 + 22
42
>>> 3 * 4
12
>>> 3 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> max(3,7)
7
>>> 5 // 3
1
>>> 5 % 3
2
```

- Icke-förändliga (immutable)
- Tre typer - heltal, flytttal och komplexa

Typer - Strängar

```
>>> msg = 'Hej'
>>> len(msg)
3
>>> msg[1]
'e'
>>> msg[1:3]
'ej'
>>> msg[-2]
'e'
>>> msg + msg
'HejHej'
>>> msg * 3
'HejHejHej'
>>> msg.split('e')
['H', 'j']
```

- Icke-förändliga (immutable)
- Sekvenstyp
- Egna funktioner med strängspecifika operationer

Typer - Listor

```
>>> mylist = ['A', 'B', 'C']
>>> mylist[1]
'B'
>>> mylist[1:3]
['B', 'C']
>>> mylist[0] = 'D'
>>> del mylist[1]
>>> mylist
['D', 'C']
>>> mylist.append('X')
['D', 'C', 'X']
>>> list2 = [4,6,2]
>>> list2.sort()
>>> list2
[2, 4, 6]
```

- Förändliga (mutable), både vad gällande storlek och innehåll
- Sekvenstyp - ”dynamiska arrayer”
- Elementen kan vara vilken typ som helst, även olika typer
- Egna funktioner med operationer för listor

Typer - Dictionary (tabell)

```
>>> words = {'GUL' : 'YELLOW',
             'VIT' : 'WHITE'}
>>> words['GUL']
'YELLOW'
>>> 'GUL' in words
True
>>> words['GUL'] = 'RED'
>>> words.values()
['WHITE', 'RED']
```

- Hashtabell / map
  - Lagrar nyckel-värde associationer
- Nycklarna är icke-förändliga

Typer - Tupler

```
>>> ('A', 'B', 'C')
('A', 'B', 'C')
>>> tup = ('A', ('B', 'C'))
>>> tup[1]
('B', 'C')
>>> ('A','B','C') + ('D', 'E')
('A', 'B', 'C', 'D', 'E')
```

- Sekvenser av objekt
- Icke-Förändliga
  - Fixerad längd
  - Statiska värden

Syntax och struktur

1. Program består av moduler
2. Moduler består av satser
3. Satser innehåller uttryck
4. Uttryck skapar och beräknar värden

Satser - Tilldelning

```
A = 4
summa = 3.2 + 5
res = 'En vacker sträng'
print(A)
print(res)
print(summa/2)
```

```
4
En vacker sträng
4.1
```

- Variabler lagrar värden av uttryck
- Uttryck ”räknar ut något” i språket
  - Jmf matematiska uttryck
  - Även struktur- uttryck som listor
- En variabel *tilldelas* ett värde;
  `variabel = uttryck`

Satser - Villkor

```
import random

c_die = random.randint(1,6)
u_die = random.randint(1,6)

if c_die > u_die:
  print('Datorn vann!')
elif c_die < u_die:
  print('Du vann!')
else:
  print('Lika')
```

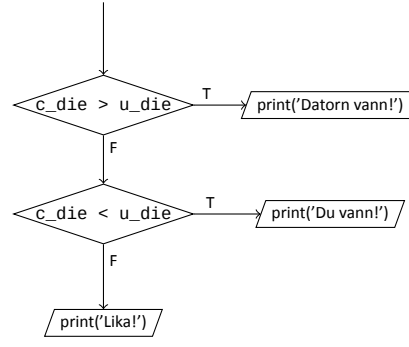
- `if`-satsen: ”om sant, gör följande”
- `elif`: ”annars om detta, gör detta”
- `else`: ”i övriga fall, gör detta”
- elif och else frivilliga
- kan ha flera elif-grenar
- **OBS!** kolon och indentering krävs

Satser - Villkor

```
import random

c_die = random.randint(1,6)
u_die = random.randint(1,6)

if c_die > u_die:
  print('Datorn vann!')
elif c_die < u_die:
  print('Du vann!')
else:
  print('Lika')
```



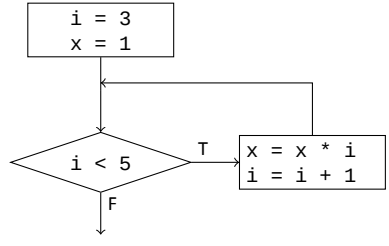
Satser - Upprepning (while)

```
i = 3
x = 1
while i < 5:
  x = x * i
  i = i + 1
```

- `while`-loopen, ”utför sålänge detta stämmer”
- Villkoret testas inför varje nytt varv
- Bra för obestämda loopar
- **OBS!** kolon och indentering krävs

Satser - Upprepning (while)

```
i = 3
x = 1
while i < 5:
  x = x * i
  i = i + 1
```



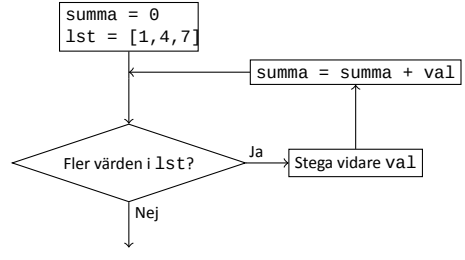
Satser - Upprepning (for)

```
summa = 0
for val in [1,4,7]:
  summa = summa + val
```

- `for`-loopen, ”upprepa för varje element”
- Bra för förutbestämda loopar
- **OBS!** kolon och indentering krävs

Satser - Upprepning (for)

```
summa = 0
lst = [1, 4, 7]
for val in lst:
  summa = summa + val
```



Satser - Upprepning (for)

```
for x in [1, 2, 3, 4, 5]:
  print(x)
for x in range(1,6):
  print(x)
```

- for-loopen *itererar över* varje element i en samling.
- Om man vill upprepa ett visst antal gånger (eller med vissa värden) är *range* bra.
- `range(a,b)` ger värden i intervallet $[a, b[$.
- `range(x)` ger $[0, x[$.

www.liu.se

LINKÖPING UNIVERSITY