UppLYSning om Python

Rickard Westman

ricwe@ida.liu.se

Disposition

- Vad är Python?
- Grunderna i Python
- Avancerade och exotiska features

  15 minuters paus

- Standardmoduler
- Nuvarande implementation & verktyg
- Jämförelser med näraliggande språk

Vad är Python?

Nisch

- Överbryggar gapet mellan C och shell-programmering
- Ett riktigt högnivåspråk
- Kan användas som utökningsspråk
- Okomplicerat – lätt att lära, lätt att läsa
- Dynamiskt – lämpligt för prototyping
- Praktiskt – bra externa gränssnitt, portabelt

Historik

- Skapades av Guido van Rossum 1989
- Döpt efter "Monty Python's Flying Circus"
- Avsett för multimedia-experiment på CWI
- Influerat av ABC, Amoeba och Modula-3
- Släpptes fritt utanför CWI 1991
- Version 1.0 i januari 1994 – senare versioner i stort sett bakåtkompatibla
- Senaste version: 1.3
- 1995: Grail – Pythons "killer app"?

Grunderna i Python

Kan användas interaktivt, som kalkylator, t.ex:

```
sen21-31% python
Python 1.3 (Oct 22 1995) [C]
Copyright 1991-1995 Stichting Mathematisch
Centrum, Amsterdam
>>> 370*7*1.25
3237.5
>>> _+10*7*1.25
3325.0
>>> def USA_pris(x):
...   return x*7*1.25
...
>>> USA_pris(370+10)
3325.0
```

Aritmetik

```
>>> 1+2
3
>>> 1+2.0
3.0
>>> 1+2L
3L
>>> pow(17,17L)
827240261886336764177L
>>> -1/2
-1
>>> -1/2.0
-0.5
```

Tilldelningar

```
>>> x=y=z=7
>>> print x,y,z
7 7 7
>>> (x,y,z)=(2*3,3*4,4*5)
>>> print x,y,z
6 12 20
```

Relationer

```
>>> x<y<z
1
>>> x<>y and x!=y and not x==y
1
```

Strängar

```
>>> word = 'Help' + 'A'
>>> word
'HelpA'
>>> '<' + word*5 + '>'
'<HelpAHelpAHelpAHelpAHelpA>'
>>> print '"Isn\'t," she said.'
"Isn't," she said.
>>> "\"Isn't,\"" == '"Isn\'t,"'
1
>>> print "foo \n \' \" \\ bar \*"
foo
 ' " \ bar \*
```

Strängar (forts.)

```
>>> word
'HelpA'
>>> word[4]
'A'
>>> word[0:2]
'He'
>>> word[2:4]
'lp'
>>> word[:2]
'He'
>>> word[2:]
'lpA'
>>> word[:2] + word[2:]
'HelpA'
```

Strängar (forts.)

```
>>> word[4]='a'
Traceback (innermost last):
  File "<stdin>", line 1, in ?
TypeError: can't assign to this subscripted
object
>>> word[:1]+'a'+word[2:]
'HalpA'
>>> ord('a')
97
>>> chr(97)
'a'
```

Listor

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> a
['spam', 'eggs', 100, 1234]
>>> a[0]
'spam'
>>> a[3]
1234
>>> a[-2]
100
>>> a[1:-1]
['eggs', 100]
>>> a[:2] + ['bacon', 2*2]
['spam', 'eggs', 'bacon', 4]
```

Listor (forts.)

```
>>> a
['spam', 'eggs', 100, 1234]
>>> a[2] = a[2] + 23
>>> a
['spam', 'eggs', 123, 1234]
>>> a[0:2] = [1, 12]
>>> a
[1, 12, 123, 1234]
>>> a[0:2] = []
>>> a
[123, 1234]
>>> a[1:1] = ['bletch', 'xyzzy']
>>> a
[123, 'bletch', 'xyzzy', 1234]
```

Listor (forts.)

```
>>> q = [2, 3]
>>> p = [1, q, 4]
>>> len(p)
3
>>> p[1]
[2, 3]
>>> p[1][0]
2
>>> p[1].append('xtra')
>>> p
[1, [2, 3, 'xtra'], 4]
>>> q
[2, 3, 'xtra']
```

Kontrollstrukturer

```
>>> if x < 0:
...      x = 0
...      print 'Negative changed to zero'
... elif x == 0:
...      print 'Zero'
... elif x == 1:
...      print 'Single'
... else:
...      print 'More'
```

Kontrollstrukturer (forts.)

```
>>> # Measure some strings:
... a = ['cat', 'window', 'defenestrate']
>>> for x in a:
...     print x, len(x)
cat 3
window 6
defenestrate 12

>>> # Fibonacci series:
... # the sum of two elements defines the next
... (a, b) = (0, 1)
>>> while b < 10:
...       print b,
...       (a, b) = (b, a+b)
1 1 2 3 5 8
```

Kontrollstrukturer (forts.)

break och continue som i C.

Dessutom:

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...            print n, 'equals', x, '*', n/x
...            break
...     else:
...           print n, 'is a prime number'
```

Tupler

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
>>> empty = ()
>>> singleton = ('hello',)
>>> x, y, z = t
```

Tabeller

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> tel.has_key('guido')
1
```

Mer om jämförelser

```
(1, 2, 3)                 < (1, 2, 4)
[1, 2, 3]                 < [1, 2, 4]
'ABC' < 'C' < 'Pascal' < 'Python'
(1, 2, 3, 4)              < (1, 2, 4)
(1, 2)                    < (1, 2, -1)
(1, 2, 3)                 = (1.0, 2.0, 3.0)
(1, 2, ('aa', 'ab')) < (1, 2, ('abc', 'a'), 4)
```

men även

```
[1,2,3] < 'Pascal'
{'foo':0} < 2
{'bar':0} < {'foo':-1}
```

Moduler

I filen "fibo.py":

```
def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
          print b,
          a, b = b, a+b

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
          result.append(b)
          a, b = b, a+b
    return result
```

Moduler (forts.)

```
>>> import fibo
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'

>>> fib = fibo.fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377

>>> from fibo import fib, fib2
>>> fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Klasser och objekt

```
class ClassName:
                <statement-1>
                .
                .
                .
                <statement-N>


class MyClass:
                i = 12345
                def f(x):
                        return 'hello world'


>>> x=MyClass()
```

Klasser och objekt

```
# Function defined outside the class
        def f1(self, x, y):
                return min(x, x+y)

        class C:
                f = f1
                def g(self):
                        return 'hello world'
                h = g
```

Klasser och objekt

```
class Bag:
        def empty(self):
                self.data = []
        def add(self, x):
                self.data.append(x)
        def addtwice(self, x):
                self.add(x)
                self.add(x)
```

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0,-4.5)
>>> x.r, x.i
(3.0, -4.5)

class DerivedClassName(BaseClassName1, BaseClassName2):
                <statement-1>
                .
                .
                .
                <statement-N>
```

Felhantering

```
>>> numbers = [0.3333, 2.5, 0, 10]
>>> for x in numbers:
...     print x,
...     try:
...         print 1.0 / x
...     except ZeroDivisionError:
...         print '*** has no inverse ***'
0.3333 3.00030003
2.5 0.4
0 *** has no inverse ***
10 0.1
```

Felhantering (forts.)

```
>>> def this_fails():
...     x = 1/0
>>> try:
...     this_fails()
... except ZeroDivisionError, d:
...     print 'Handling run-time error:', d
Handling run-time error: integer division or
modulo

>>> raise NameError, 'HiThere'
Traceback (innermost last):
  File "<stdin>", line 1
NameError: HiThere
```

Felhantering (forts.)

```
>>> my_exc = 'my_exc'
>>> try:
...     raise my_exc, 2*2
... except my_exc, val:
...     print 'My exception occurred, value:',
val
My exception occurred, value: 4


>>> try:
...     raise KeyboardInterrupt
... finally:
...     print 'Goodbye, world!'
...
```

Standardmoduler

Allmänt

- sys
- types
- pickle
- shelve

Strängar

- string
- regex
- regsub

Matematik

- math
- rand
- array
- md5

Operativsystem

- os
- getopt
- signal
- socket
- select
- thread

UNIX-specifikt

- posix
- pwd
- grp
- gdbm
- termios
- fcntl

Fönstersystem

- Tk
- Stdwin
- wxWindows
- m.fl

Nuvarande implementation & verktyg

- Python 1.3 (interpretator och kompilator)
- Debugger
- Profiler
- Emacs-mode

Jämförelser med näraliggande språk

- Perl
- Tcl
- Awk
- shell-språk