Per Sedholm
DD1320 (tilda11)
2011-08-31

Övning 0

Python för den som kan MATLAB

- Lättare att läsa färdig kod än att skriva själv
- Det krävs övning för att ”automatiskt” få detaljerna rätt:

```
print "a", "b", "c"
print "a" "b", "c"
print "a", ("b", "c")
```

  Vad är rätt? Antal argument till *print*? Datatyp för dem?

- Känna igen felmeddelanden

1. Inmatning, utskrift

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Hello world
#

namn = raw_input("Vad heter du? ")
print "Hej", namn
print "Hej " + namn

# Vad heter du? Terry G
# Hej Terry G
# Hej Terry G
```

2. Läsa tal, *if-else*-sats, villkor, teckenkodningskommentar

```
# -*- coding: utf-8 -*-   or iso-8859-1
# Maxberäkning 1

num1 = input("Ge första talet: ")
num2 = input("Ge andra talet: ")

if num1 > num2:
    print num1, "är störst!"
else:
    print num2, "är störst!"

# Note: In other languages:   (num1 > num2) ? num1 : num2
print num1 if (num1 > num2) else num2, "är störst!"

print (lambda:num2, lambda:num1)[num1 > num2](), "är störst!"

# Ge första talet: 8
# Ge andra talet: 5
# 8 är störst!
# 8 är störst!
# 8 är störst!
```

3. *While*-slinga, typkonvertering sträng $\longrightarrow$ heltal

```
# Maxberäkning 2

max = input("Ge första talet: ")   ## NB: bad name (builtin function)
ans = raw_input("Ge ett tal till (avsluta med Enter): ")
while ans != "":
    tal = int(ans)
    if tal > max:
        max = tal
    ans = raw_input("Ge ett tal till (avsluta med Enter):  ")

print "Störst var", max

# Ge första talet: 8
# Ge ett tal till (avsluta med Enter): 5
# Ge ett tal till (avsluta med Enter): 13
# Ge ett tal till (avsluta med Enter): 7
# Ge ett tal till (avsluta med Enter): 18
# Ge ett tal till (avsluta med Enter):
# Störst var 18
```

```
$ python
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> max
<built-in function max>
>>> type(max)
<type 'builtin_function_or_method'>
>>> help(max)
Help on built-in function max in module __builtin__:

max(...)
    max(iterable[, key=func]) -> value
    max(a, b, c, ...[, key=func]) -> value

    With a single iterable argument, return its largest item.
    With two or more arguments, return the largest argument.
```

Bland övriga funktioner finns: `dir` (visa variabler), `locals` (visa variabler och värden).

4. Aritmetik, typkonvertering heltal $\longrightarrow$ flyttal

```
# Average of three numbers

num1 = input("First number: ")
num2 = input("Second number: ")
num3 = input("Third number: ")
sum = num1 + num2 + num3
n = 3
avg1 = float(sum)/n
print "Average:", avg1
avg2 = sum/n
print "Average:", avg2

# First number: 5
# Second number: 3
# Third number: 2
# Average: 3.33333333333
# Average: 3
```

Pythons numeriska datatyper: *int*, *float*, *long*, *complex*.

5. Importera modul, slumptal

```
# Guessing

import random

ans = random.randint(1,10)
guess = input("Guess a number between one and ten: ")
while guess != ans:
    guess = input("Wrong, try again: ")
print "Correct!"
```

6. Lista, slumpa ur lista, sammansatt ord

```
# Word guess: Choose random word, count guesses.

import random

instruments = ["violin", "flute", "bassoon"]
answ = random.choice(instruments)
guess = raw_input("Five guesses: What instrument am I thinking of? ")
n = 1

while guess != answ and n < 5:
    guess = raw_input("Wrong, try again: ")
    n = n+1

if n <= 5:      # NB: spot error?
    print "Brilliant, you managed in only", n, "attempts!"
else:
    print "Sorry, you couldn't guess in five attempts."
```

7. Strängar, indexering, *slicing*, funktionen `len`, operatorn `in`, operatorn + för strängkonkatenering

```
# Girl's name: Convert a name to girl form.
#
# Assume the given name is a boy's name, remove the last letter (if a vowel),
# or the last two letters (otherwise). Finally, add "elle" at the end.
#
# Note: using Swedish vowels: AEIOUYÅÄÖ
#       not English: AEIOU(YW)

name = raw_input("Vad heter du? ")
n = len(name)
print "Ditt name har", n, "bokstäver."
last = name[n-1]
if last in "aeiouyåäö":
    name = name[:-1]
else:
    name = name[:-2]
girlname = name + "elle"
print "Ditt flickname är", girlname

# Vad heter du? Per
# Ditt name har 3 bokstäver.
# Ditt flickname är Pelle
#
# (Hmmmm....)
```

Ang. 'slicing'

Notera att ordningen mellan start/slut/steg är annorlunda än MATLAB:

```
s = 'hello world'
s[1:-1:2]  ## start : slut [: steg]
# 'el ol'
```

8. Strängmetoder, listor, for-slingor

```
# Translate to Yoda Speak.
# "You have much to learn." -> "Much to learn, you have."
#
# Note: proper Yoda Speak changes all sentences to OSV (object-subject-verb).
#       http://en.wikipedia.org/wiki/Object%E2%80%93Subject%E2%80%93Verb

sent = raw_input("Sentence: ")
n = len(sent)
end = sent[n-1] # remove punctuation
sent = sent[:-1]
lst = sent.split()

print lst[2].capitalize(),
for ordet in lst[3:]:
    print ordet.lower(),
print ",",
for ordet in lst[0:2]:
    print ordet.lower(),
print end

# Sentence: This is my home!
# My home , this is !

# TODO: rewrite so there are no spaces around the comma.
```

9. Funktioner, parametrar, returvärden, softspace

```
# Translate to Yoda Speak -- now with functions

import sys

def words(sentence):
    n = len(sentence)
    end = sentence[n-1]
    sentence = sentence[:-1]
    lst = sentence.split()  ## See: help("".split)
    return lst, end

def new_order(lst, end):
    print lst[2].capitalize(),
    for wrd in lst[3:]:
        print wrd.lower(),
    sys.stdout.softspace = 0  ## only affects next print call
    print ",",                ## See: type(sys.stdout); help(file)
    for wrd in lst[0:2]:
        print wrd.lower(),
    sys.stdout.softspace = 0
    print end

sentence = raw_input("Sentence: ")
lst, punctuation = words(sentence)
new_order(lst, punctuation)

# Sentence: This is my home!
# My home, this is!
```

Ang. *"softspace"*

```
import sys

print "first",
sys.stdout.softspace=False
print "second",
print "third"
# firstsecond third

sys.stdout.write("first")
sys.stdout.write("second")
sys.stdout.write("third")
# firstsecondthird
```

10. Listmetoden append, kopiering av lista

```
# Copy a list by value or by reference

def copy(lst):
    cp = []
    for el in lst:
        cp.append(el)
    return cp

# Note: could use variable name 'lst', it wouldn't be the same as above
l = [1, 7, 10, 13, 19, 23, 28] # "happy numbers"
cp1 = l
cp2 = copy(l)
cp3 = l[:]

print "-" * 40
print "l   = ", l
print "cp1 = ", cp1
print "cp2 = ", cp2
print "cp3 = ", cp3

cp1[3] = -1
cp2[4] = -2
cp3[5] = -3

print "-" * 40
print "l   = ", l
print "cp1 = ", cp1
print "cp2 = ", cp2
print "cp3 = ", cp3

# ----------------------------------------
# l   =  [1, 7, 10, 13, 19, 23, 28]
# cp1 =  [1, 7, 10, 13, 19, 23, 28]
# cp2 =  [1, 7, 10, 13, 19, 23, 28]
# cp3 =  [1, 7, 10, 13, 19, 23, 28]
# ----------------------------------------
# l   =  [1, 7, 10, -1, 19, 23, 28]
# cp1 =  [1, 7, 10, -1, 19, 23, 28]
# cp2 =  [1, 7, 10, 13, -2, 23, 28]
# cp3 =  [1, 7, 10, 13, 19, -3, 28]
```