GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Lecture 2: Functions and expressions

Introduction to programming (LT2111)

Markus Forsberg
Språkbanken
University of Gothenburg

2011-09-13

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Changes in the schedule (sorry about that)

- Supervised lab, Week 40, Tuesday, has been moved to Thursday 10.15-12.00
- Lecture 5, Week 40, has been moved to Thursday 13.15-15.00, T307
- Lecture 6, Week 41, has been moved to Thursday 13.15-15.00, T116

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Values and types

- *values* are things a program do work on, such as numbers, strings or lists.
- Every *value* has a *type*. When we know the type, we know what we can do with the value (numbers may be added, strings may be concatenated, and so on).

```
>>> s = "I'm a string."
>>> type(s)
<type 'str'>
>>> dir(s)
[ ... , 'capitalize', 'center', 'count', 'decode',
'encode','endswith', 'expandtabs', 'find', 'format',
'index', 'isalnum','isalpha', 'isdigit', 'islower',
'isspace', 'istitle', 'isupper', 'join', 'ljust',
'lower', 'lstrip', 'partition', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
'split', 'splitlines', 'startswith', 'strip',
'swapcase', 'title', 'translate', 'upper', 'zfill']
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Expressions and statements

- *expressions* denote values, possibly after some computation (5+5 denotes 10, ‘a’ + ‘b’ denotes ‘ab’).
- *statements* perform actions, such as printing a string or assigning a name to a value.
- *variables* give names to values.

```
>>> import math
>>> math.sqrt(16)
4.0
>>> result = math.sqrt(16)
>>> result
4.0
>>> print result
4.0
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Type: string

- Strings are used to represent text data.
- Example:

```
'Python programming'
"Python's interpreter"
"""strings spanning
more than one line"""
```

- Strings are *immutable*, i.e., they cannot be changed (=string operations create new strings).
- There are many convenient Python functions for strings, e.g.
  - split
  - join
  - replace
  - lower

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Lists

- Lists are used to represent sequences of data, e.g., a token list.
- Example: `['Python', 'programming']`
- Lists are *mutable*, i.e., they may be changed.
- List are sequences, supporting indexing and slicing.

```
>>> list = [12,43,564,1,23]
>>> list[0]
12
>>> list[4]
23
>>> list[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> list[1:3]
[43, 564]
```

Procedure and Functions

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

```
def NAME(PARAMETERS):
    BODY
    (return EXPRESSION)
```

Procedure:

```
def print_twice(s):
    print s
    print s
>>> print_twice('hello')
hello
hello
```

Function:

```
def area(length, width):
    return length*width
>> print area(5,10)
50
```

Procedures are also functions

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

- A procedure is actually a function, returning 'None' (instead of actually give no value, it return a value saying there is no value).
- When you (surprisingly) get 'None', then you are using a procedure as a function.

```
def five():
    print 5

>>> five()
5
>>> x = five()
5
>>> print x
None
>>> type(x)
<type 'NoneType'>
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Conditionals

```
if CONDITION:
   CODE
elif CONDITION:
   CODE
elif CONDITION:
   CODE
...
else:
   CODE
```

```
def greater_than_hundred(n):
    if n > 100:
        print "n is greater than 100"
    elif n < 100:
        print "n is lower than 100"
    else:
        print "n is hundred"
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Numerical condition

<	less than
<=	less than or equal to
==	equal to (note this is two "=" signs, not one)
!=	not equal to
>	greater than
>=	greater than or equal to

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Combining conditions

- **not** *CONDITION*
- *CONDITION1* **and** *CONDITION2*
- *CONDITION1* **or** *CONDITION2*

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Some string condition

s.startswith(t)	test if s starts with t
s.endswith(t)	test if s ends with t
t in s	test if t is contained inside s
s.islower()	test if all cased characters in s are lowercase
s.isupper()	test if all cased characters in s are uppercase
s.isalpha()	test if all characters in s are alphabetic
s.isalnum()	test if all characters in s are alphanumeric
s.isdigit()	test if all characters in s are digits
s.istitle()	test if s is titlecased (all words in s have have initial capitals)

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Looping: for loop

```
for NAME in ITERATOR:
    BODY
```

- Almost everything in Python that you want to traverse is an iterator: lists, sets, strings, and more.

```
>>> for x in 'loop':
...     print x
...
l
o
o
p

>>> for x in ['the','dog','barked']:
...     print x
...
the
dog
barked
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Modules

- A module is a collection of related functions.
- We use `import module` to import a module.
- Functions (and more) is available through dot notation: `module.function`.

```
>>> import math
>>> math
<module 'math' (built-in)>
>>> math.cos(math.pi)
-1.0
```

Modules cont.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

- `from module import *` allows us to skip the dot notation, and just write `function`.
- In general, this is not a good idea, since:
  - the origin of `function` is lost;
  - the risk that we get accidental name collisions increases.

```
a1.py: a = 1
a2.py: a = 2
>>> from a1 import *
>>> from a2 import *
>>> a
2
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Introduction to assignment 1

- Assignment 1: Princeton WordNet

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Case study: interface design

- The rest of the lecture: Chapter 4 in Downey.
- http://www.greenteapress.com/thinkpython/html/book005.html