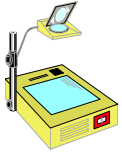




Python: Chapter 3 Basic Statements



Learning Python, Lutz & Ascher

1

statements

- Process objects like lists, tuples, etc
- Create objects
- Programs are composed of modules
- And modules contain statements

2

Kinds of statements

- Assignment create references
- Calls run functions
- Print
- If/elif/else
- For/else
- While/else
- pass
- Break/continue
- Try/except/finally/raise
- Import, from
- Class
- Global
- Del
- Exec
- assert

3

assignment

- Create object references
- Names are created when assigned
- Must assign to a name before you reference the name
- Implicit assignments: import, from, def, class, for, function arguments, etc.
- Multiple-target: a = b = 'cat'

4

Examples of assignment

```
>>>x = 1
>>>y = 2
>>> A, B = x, y      #tuples
>>>A, B
(1,2)
>>> [C, D] = [x, y]  #lists
>>>C, D
(1, 2)
<<<x, y = y, x #3-step swap
>>>x, y
(2, 1)
```

5

Rules for variable names

- Syntax: (letter)(letter|digit)*
- Case sensitive

6

Print statement

```
>>>print 'hello world'
hello world

>>>'hello world'
'hello world'

>>>import sys
>>>sys.stdout.write('hello world\n')
hello world
```

7

If statement

```
General form:
if <test1>:
    <statements>
elif <test2>:
    <statements>
else:
    <statements>
```

8

No switch statement in Python: use a dictionary!

```
>>>choice = 'ham'
>>>print {'spam':2.50,
...       'ham': 1.99,
...       'eggs':0.99,
...       'bacon':1.10}[choice]
1.99
```

*Dictionaries associate keys to values;
Can also contain functions for more
Complex actions!*

9

Syntax rules

- Statements execute one after another
- Statement and block boundaries are detected automatically: no semicolon or curly brace
- Statements can span lines if use \
- Compound statements =
header ':' indented statements

10

Some weird examples

```
If a==b and \
... x == y:
...     print 'nice'

X = 1; Y=2; print x

If 1: print 'hello'
```

11

and/or

```
x and y    -- returns an object
x or y     -- returns an object
not x      -- return 0, 1
```

and & or return an object, not 1 or 0;
Returns left operand if false, owise return
right operand whether true or false;
Use short circuit evaluation:

```
>>> [] or 3
3
>>> [] or {}
{}
>>> [] and {}
[]
```

12

While statement

- Most general iteration construct
- Repeatedly executes a block of indented statements
- Pre-test loop
- There are other statements that implicitly loop: *map, reduce, filter, in*, etc.

13

General form of while

```
while <test>:  
  <stmts1>  
else:  
  # optional else  
  <stmts2> # fun if didn't exit with break
```

```
while 1:  
  print 'type ctrl-C to stop'  
  
>>>x = 'cat'  
>>>while x:  
...   print x,  
...   x = x[1:]  
cat ca c
```

14

Break can obviate a flag

```
while x > 1:  
  if y % x == 0: #x evenly divides y  
    print y, ' has factor ', x  
    break  
  x = x - 1  
else:  
  print y, ' is prime'
```

15

General form of for loop

```
for <target> in <object>:  
  <stmts1>  
  if <test>: break # exit loop  
  if <test>: continue # go to top of loop  
else:  
  # optional else  
  <stmts2> # fun if didn't exit with break
```

```
>>>L = [1,2,3]  
>>>for x in L:  
...   print x,  
1 2 3
```

```
>>>T = [(1,2), (3,4), (5,6)]  
>>>for (a,b) in T:  
...   print a, b,  
1 2  
3 4  
5 6
```

16

Range statement

- Returns a list of successively higher integers
- Can be used to step through a for loop

```
>>>range(0, 10, 2)  
[0, 2, 4, 6, 8]
```

```
>>>for i in range(3)  
...   print i,  
0 1 2
```

Ranges are open-ended on the right!

17

Common coding problems

- Don't forget the colon
- Start in column 1
- Blank lines matter at interactive prompt: they end the statement
- Blank lines don't for import module
- Indent consistently (don't mix tabs/spaces)
- Don't code C in Python: **if (x == y):**
- Don't always expect a result: (list.sort()) returns None
- Use parens after a function name: **list.sort()**

18