PYTHON STATEMENT

Lecturer: Terry

Outline

1. Introduction to expression/statement
2. Expression
3. Simple statement
4. Compound statement

1 Introduction to expression/statement

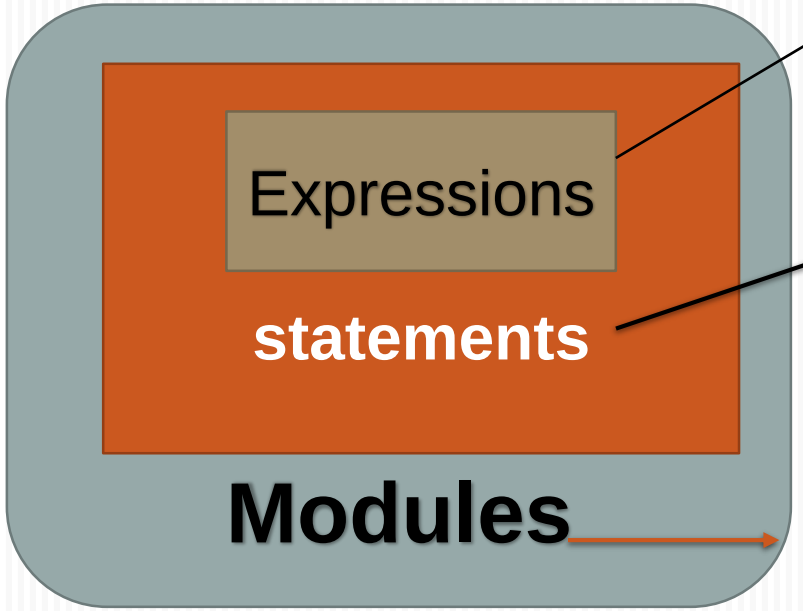
1.1 The Python Conceptual Hierarchy

Python programs can be decomposed into modules, statements, expressions, and objects.

1. Programs are composed of modules.
2. Modules contain statements.
3. Statements contain expressions.
4. Expressions create and process objects.

1
Introduction

1.1 The Python Conceptual Hierarchy



Expressions create and process objects.

Statements are the things you write to tell Python what your programs should do.

- Every file of Python source code whose name ends in a .py extension is a **module**.

- packages program code and data for reuse,
- provides self-contained namespaces that minimize variable name clashes across your programs.

1.2 Expression vs statement

- If you can print it, or assign it to a variable, it’s an expression. If you can’t, it’s a statement.
- If you type an expression on the command line, the interpreter evaluates it and displays the result.
- When you type a statement on the command line, Python executes it and displays the result, if there is one.

2 Expression

- An expression is a combination of values, variables, and operators.

Operators	Description
`yield x`	Generator function send protocol
`lambda args: expression`	Anonymous function generation
`x if y else z`	Ternary selection (x is evaluated only if y is true)
`x or y`	Logical OR (y is evaluated only if x is false)
`x and y`	Logical AND (y is evaluated only if x is true)
`not x`	Logical negation
`x in y, x not in y`	Membership (iterables, sets)
`x is y, x is not y`	Object identity tests
`x < y, x <= y, x > y, x >= y`	Magnitude comparison, set subset and superset;
`x == y, x != y`	Value equality operators
`x | y`	Bitwise OR, set union
`x ^ y`	Bitwise XOR, set symmetric difference
`x & y`	Bitwise AND, set intersection
`x << y, x >> y`	Shift x left or right by y bits

2
Expression

- An expression is a combination of values, variables, and operators.

Operators	Description
`yield x`	Generator function send protocol
`lambda args: expression`	Anonymous function generation
`x if y else z`	Ternary selection (x is evaluated only if y is true)
`x or y`	Logical OR (y is evaluated only if x is false)
`x and y`	Logical AND (y is evaluated only if x is true)
`not x`	Logical negation
`x in y, x not in y`	Membership (iterables, sets)
`x is y, x is not y`	Object identity tests
`x < y, x <= y, x > y, x >= y`	Magnitude comparison, set subset and superset;
`x == y, x != y`	Value equality operators
`x | y`	Bitwise OR, set union
`x ^ y`	Bitwise XOR, set symmetric difference
`x & y`	Bitwise AND, set intersection
`x << y, x >> y`	Shift x left or right by y bits
`x + y`	Addition, concatenation;
`x - y`	Subtraction, set difference
`x * y`	Multiplication, repetition;
`x % y`	Remainder, format;
`x / y, x // y`	Division: true and floor
`-x, +x`	Negation, identity
`~x`	Bitwise NOT (inversion)
`x ** y`	Power (exponentiation)
`x[i]`	Indexing (sequence, mapping, others)
`x[i:j:k]`	Slicing
`x(...)`	Call (function, method, class, other callable)
`x.attr`	Attribute reference
`(...)`	Tuple, expression, generator expression
`[...]`	List, list comprehension
`{...}`	Dictionary, set, set and dictionary comprehensions

3 Simple statement

Statements can be divided into two types.

- Simple statements comprised within a single logical line.

- Compound statements
  - have other statements nested inside them;
  - generally span multiple lines.

Header line:
    Nested statement block

Statement

Logical And Physical Lines

- A physical line is what you see when you write the program.
- A logical line is what Python sees as a single statement.
- Python implicitly assumes that each physical line corresponds to a logical line.

- more than one logical line on a single physical line
- more than one physical line for a single logical line
- **Explicit/implicit line joining**

Python Indentation

- Whitespace at the beginning of the line is important. This is called indentation.
- Statements which go together must have the same indentation. Each such set of statements is called a block.
- Use a single tab or four spaces for each indentation level. Choose either of these two indentation styles.
- Do not use a mixture of tabs and spaces for the indentation as it does not work across different platforms properly.

2.1 Simple statements

tement is comprised within a
l line.

```
5+6
11
3+5; 6+7; 5
5
```

Expression statements

```
a = 9 # bind
a='Python' # rebind
```

Assignment statements

3.1 Simple Statement

2.2 Expression **statements**

- used to compute and write a
- or (usually) to call a procedu

```
5+6
11
3+5; 6+7; 5
5
```

Simple Statement

2.3 Assignment statements

```
a = 9 # bind
```

```
a='Python' # rebind
```

Assignment statements are

- bind names to values
- rebind names to values

The evaluation of an expression produces a value, which is why expressions can appear on the right hand side of assignment statements.

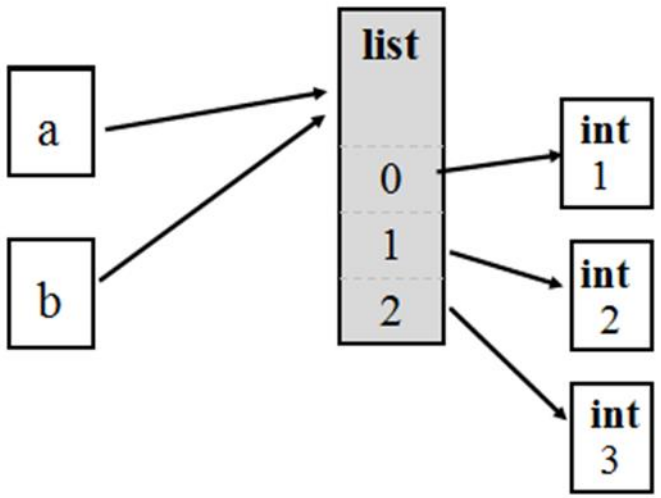
Simple Statement

Assignment statements

are used to

- modify attributes or objects.

```
>>> a=[1,2,3]
>>> b=a
```



2.4 Assignment statement forms

- **Basic form**

```
a = 'Python'
```

2.4 Assignment statement forms

- **Sequence assignment**

```
a,b,c,d = 'Pyth'
```

✓ —any sequence of names can be assigned to any sequence of values,

✓ and Python assigns the items one at a time by position.

```
a, b = 'Python', 'Finance'
```

```
[a, b] = [ 'Python', 'Finance']
```

```
a
'Python'
b
'Finance'
```

2.4 Assignment statement forms

- **Multiple-target assignments**

```
a =b= 'Python'
```

Python assigns a reference to the same object (the object farthest to the right) to all the targets on the left.

Simple Statement

2.4 Assignment statement forms

- **Augmented assignments**

  a shorthand that combines an expression and an assignment in a concise way.

```
a=7
```

```
a+=2
```

```
a
```

```
9
```

```
x = [1,2,4]
x+=[2,3]
x
```

```
[1, 2, 4, 2, 3]
```

3 Compound statement

Compound statements

- contain (groups of) other statements
- span multiple lines
- affect or control the execution of those other statements in some way.

Compound Statement

In order to control the flow of a program, we have two main weapons:

- **conditional programming** (also known as branching)
- **looping**.

Compound Statement

3.1 Conditional programming

The main tool is the if statement.

如果大盤上漲，買入股票A。
否則，賣出股票A。

```
if condition1:
    statement1
else:
    statement2
```

Compound Statement

3.1 Conditional programming

如果大盤上漲，買入統一股票。
同時，電子指數成份股也上漲，買台積電股票。
如果大盤和電子指數股都沒有上漲。則，不進場。

```
if condition1:
    statement1
elif condition2:
    statement2
else:
    statement3
```

Compound Statement

3.2 Looping programming

- statements that repeat an action over and over.

  ✓ the ***while*** statement,
  provides a way to code general loops.

  ✓ the ***for*** statement, is designed for
  - stepping through the items in a sequence or other iterable object
  - and running a block of code for each.

Compound Statement

3.2.1 for loops

- The for loop is a generic iterator in Python:
  - it can step through the items in any ordered sequence or other iterable object.

Compound Statement

3.2.1 for loops

General Format:

```
for target in object:   # Assign object items to target
    statements          # Repeated loop body: use target
```

Compound Statement

3.2.2 while loops

Python keeps evaluating the test at the top and executing the statements nested in the loop body until the test returns a false value:

```
while test:                # Loop test
    statements             # Loop body
else:                      # Optional else
    statements             # Run if didn't exit loop with break
```

Compound Statement

3.2.3 break and continue statements

Compound Statement