Python程序设计

输入输出

刘安
苏州大学，计算机科学与技术学院

http://web.suda.edu.cn/anliu/

本节涉及到的知识点

- 读取用户输入 - input函数
- 赋值语句
  - 多目标赋值
  - 增量赋值
  - 序列赋值
  - 序列解包
- 在标准输出上打印 - print函数

https://docs.python.org/3/library/functions.html#input

https://docs.python.org/3/library/functions.html#print

读取用户输入

- input函数读取用户的输入，并以字符串类型返回
- input函数接受一个字符串参数，并将其显示在标准输出上

```
>>> n = input('Enter an integer: ')
Enter an integer: 
```

```
>>> n = input('Enter an integer: ')
Enter an integer: 42
>>> n
'42'
>>> n + 1
Traceback (most recent call last):
  File "<pyshell#181>", line 1, in <module>
    n + 1
TypeError: must be str, not int
```

可以使用内置函数int、float、eval对input函数的返回值进行适当的处理

https://docs.python.org/3/library/functions.html

赋值语句

- 赋值语句将变量绑定到一个对象上
- 变量在使用前必须先赋值
- 进行赋值的地方
- 显式的赋值语句=
- for循环变量
- 函数定义、参数传递
- 模块导入以及类的定义

```
>>> a = 3.14
>>> for c in 'str':
        a = ord(c)
```

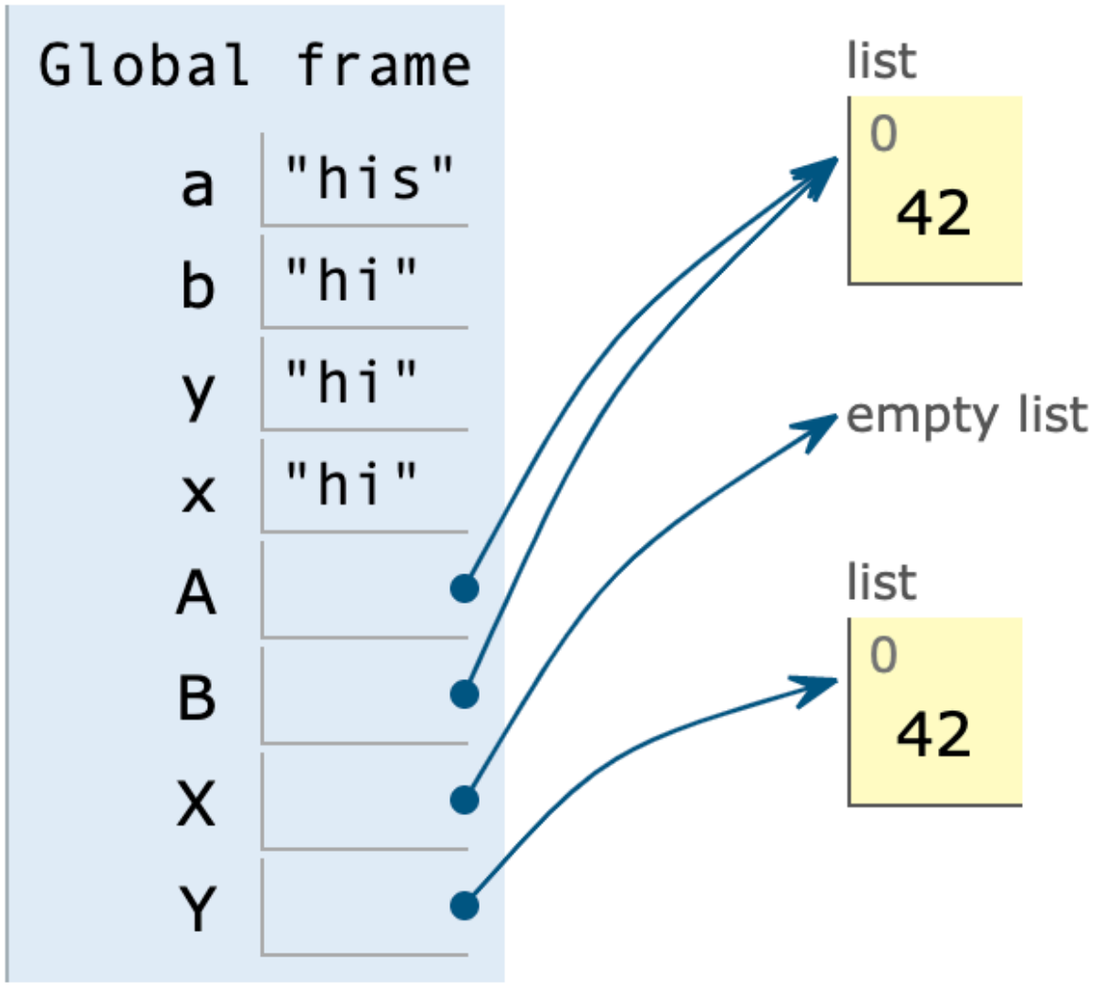
```
>>> def f(n):
        return n ** 2

>>> f(a)
12996
>>> import math
>>> math.pi
3.141592653589793
```

多目标赋值

- 将最右侧的对象依次赋值给左侧所有的变量
- 多目标赋值会形成共享引用，如果最右侧对象是可变的，要注意共享引用带来的副作用

```
>>> a = b = 'hi'
>>> y = 'hi'
>>> x = y
>>> a = 'his'
>>>
>>> A = B = []
>>> B.append(42)
>>> X = []
>>> Y = []
>>> Y.append(42)
```



增量赋值

- a = a + 1 ⟶ a += 1
- 将二元计算和赋值语句结合在一起
- 如果a是可变类型，增量赋值会原地修改对象

```
>>> a = 42
>>> a += 1
>>> a
43
>>> s = 'hi'
>>> s += '!'
>>> s
'hi!'
```

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

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

序列赋值

- 将右侧的对象序列（比如列表、字符串、元组等）赋值给左侧的变量序列（对象和变量一一匹配，数量需相等）

```
>>> [a, b] = [1, 2]
>>> a
1
>>> b
2
>>> a, b = [3, 5]
>>> a, b
(3, 5)
```

```
>>> a, b = 3, 5
>>> a, b # tuple here
(3, 5)
>>> a, b = b, a #exchange
>>> a, b
(5, 3)
>>> a, b, c = 'str'
>>> a, b, c
('s', 't', 'r')
```

```
>>> a, b, c = 'python'
Traceback (most recent call last):
  File "<pyshell#130>", line 1, in <module>
    a, b, c = 'python'
ValueError: too many values to unpack (expected 3)
```

序列解包

- 通过在变量前面加上星号实现更灵活的对象-变量匹配：带*的变量会绑定到一个列表，该列表收集了右侧序列中没被赋值给左边变量的所有对象

```
>>> a, *b = '1234'
>>> a
'1'
>>> b
['2', '3', '4']
>>> *a, b = '1234'
>>> a
['1', '2', '3']
>>> b
'4'
```

```
>>> a, *b, c = '1234'
>>> a
'1'
>>> b
['2', '3']
>>> c
'4'
>>> a, b, *c = '12'
>>> a
'1'
>>> b
'2'
>>> c
[]
```

使用print函数在标准输出上进行打印

- print(value1, value2, …, sep = ' ', end = '\n', …)
  - value1, value2, … ：待打印的对象
  - sep：对象文本之间插入的字符串，默认是一个空格
  - end：打印文本末尾的字符串，默认是换行符，如果设置为空字符串，则使得下一个print函数将继续在当前行的尾部打印

```
>>> a, *b, c = 'python'
>>> print(a, b, c)
p ['y', 't', 'h', 'o'] n
>>> print(a, b, c, sep = '...')
p...['y', 't', 'h', 'o']...n
```

```
>>> for i in range(1, 10, 2):
        print(i, end = '')
13579
```