封装和解构

基本概念

```
t1 = 1, 2
print(type(t1)) # 什么类型

t2 = (1, 2)
print(type(t2))
```

Python等式右侧出现逗号分隔的多值的时候，就会将这几个值封装到元组中。这种操作称为**封装**packing。

```
x, y = (1, 2)
print(x) # 1
print(y) # 2
```

Python中等式右侧是一个容器类型，左侧是逗号分隔的多个标识符，将右侧容器中数据的一个个和左侧标识符一一对应。这种操作称为**解构**unpacking。

从Python3开始，对解构做了很大的改进，现在用起来已经非常的方便快捷。

封装和解构是非常方便的提取数据的方法，在Python、JavaScript等语言中应用极广。

```
# 交换数据
x = 4
y = 5
t = x
x = y
y = t


# 封装和解构，交换
x = 10
y = 11
x, y = y, x
```

简单解构

```python
# 左右个数相同
a,b = 1,2
a,b = (1,2)
a,b = [1,2]
a,b = [10,20]
a,b = {10,20} # 非线性结构
a,b = {'a':10,'b':20} # 非线性结构也可以解构

[a,b] = (1,2)
[a,b] = 10,20
(a,b) = {30,40}
```

那么，左右个数不一致可以吗?

```python
a, b = (10, 20, 30)
```

剩余变量解构

在Python3.0中增加了剩余变量解构（rest）。

```python
a, *rest, b = [1, 2, 3, 4, 5]
print(a, b)
print(type(rest), rest) #  <class 'list'> [2, 3, 4]
```

标识符rest将尽可能收集剩余的数据组成一个列表。

```python
a, *rest = [1, 2, 3, 4, 5]
print(a, rest)

*rest, b = [1, 2, 3, 4, 5]
print(rest, b)

*rest = [1, 2, 3, 4, 5]
print(rest) # 内容是什么?

a, *r1, *r2, b = [1, 2, 3, 4, 5] # ?
```

```python
a, *_, b = [1, 2, 3, 4, 5]
print(_) # 在IPython中实验，_是最后一个输出值，这里将把它覆盖

_, *b, _ = [1, 2, 3]
print(_) # 第一个_是什么
print(b) # 是什么
print(_) # 第二个_是什么
```

_是合法的标识符，这里它没有什么可读性，它在这里的作用就是表示不关心这个变量的值，我不想要。有人把它称作 丢弃(Throwaway)变量。

其它结构

```
x = [range(5)]
y = [*range(5)]
z = list(range(5))
print(x, y, z)
```

练习

- 从nums = [1, (2, 3, 4), 5]中，提取其中4出来
- 从list(range(10))中，提取第二个、第四个、倒数第二个元素