



马哥教育

IT 人的高薪职业学院

Python封装和解构

讲师：Wayne

从业十余载，漫漫求知路

封装和解构

□ 封装

- 将多个值使用逗号分割，组合在一起
- 本质上，返回一个元组，只是省掉了小括号
- python特有语法，被很多语言学习和借鉴

t1 = (1,2) # 定义为元组

t2 = 1,2 # 将1和2封装成元组

type(t1)

type(t2)

封装和解构

□ 举例

a = 4

b = 5

temp = a

a = b

b = temp

等价于

a, b = b, a

上句中，等号右边使用了封装，而左边就使用了解构



解构

- 把线性结构的元素解开，并顺序的赋给其它变量
- 左边接纳的变量数要和右边解开的元素个数一致

- 举例

```
lst = [3, 5]
```

```
first, second = lst
```

```
print(first, second)
```



解构

□ 举例

`a,b = 1,2`

`a,b = (1,2)`

`a,b = [1,2]`

`a,b = [10,20]`

`a,b = {10,20}`

`a,b = {'a':10,'b':20}` # 非线性结构也可以解构

`a,b = {10,20,30}`

`a,*b = {10,20,30}`

`[a,b] = (1,2)`

`[a,b] = 10,20`

`(a,b) = {30,40}`



Python3的解构

- 使用 *变量名 接收，但不能单独使用
- 被 *变量名 收集后组成一个列表
- 举例

```
lst = list(range(1, 101, 2))
```

```
head, *mid, tail = lst
```

```
*lst2 = lst
```

```
*body, tail = lst
```

```
head, *tail = lst
```

```
head, *m1, *m2, tail = lst
```

```
head, *mid, tail = "abcdefghijklmn"
```

```
type(mid)
```



丢弃变量

- 这是一个惯例，是一个不成文的约定，不是标准
- 如果不关心一个变量，就可以定义改变量的名字为_
- _是一个合法的标识符，也可以作为一个有效的变量使用，但是定义成下划线就是希望不要被使用，除非你明确的知道这个数据需要使用

□ 举例

```
lst = [9,8,7,20]
```

```
first, *second = lst
```

```
head, *_ , tail = lst
```

```
print(head)
```

```
print(tail)
```

```
# _是合法的标识符，看到下划线就知道这个变量就是不想被使用
```

```
print(_)
```



丢弃变量

□ 举例

```
lst = [9,8,7,20]
```

```
first, *second = lst
```

```
_, *_ , tail = lst
```

```
print(_)
```

```
print(tail)
```

```
print(_)
```



丢弃变量

□ 总结

- `_` 这个变量本身无任何语义，没有任何可读性，所以不是用来给人使用的
- Python中很多库，都使用这个变量，使用十分广泛。请不要在不明确变量作用域的情况下，使用 `_` 导致和库中 `_` 冲突

□ 练习

- `lst = list(range(10))` # 这样一个列表，取出第二个、第四个、倒数第二个

练习

□ 练习

- 从 `lst = [1,(2,3,4),5]` 中，提取4出来
- 环境变量 `JAVA_HOME=/usr/bin`，返回环境变量名和路径
- 对列表 `[1, 9, 8, 5, 6, 7, 4, 3, 2]` 使用冒泡法排序，要求使用封装和解构来交互数据

练习

□ 练习

□ 从lst = [1,(2,3,4),5]中，提取4出来

```
lst = [1,(2,3,4),5]
```

```
a,(b,c,d),e = lst
```

```
print(a,b,c,d,e)
```

```
_,(*_,val),*_ = lst
```

```
print(val)
```

```
_,[*_,val],*_ = lst
```

```
print(val)
```



练习

□ 练习

- 环境变量JAVA_HOME=/usr/bin，返回变量名和路径

```
key, _, val = "JAVA_HOME=/usr/bin".partition('=')
```

```
print(key)
```

```
print(val)
```

□ 总结：

- 解构，是Python提供的很好的功能，可以方便的提取复杂数据结构的值
- 配合 _ 的使用，会更加便利

马哥教育
IT人的高薪职业学院

谢谢

咨询热线 400-080-6560

