Python条件控制语句

计算机之所以能做很多自动化的任务, 因为它可以自己做条件判断. if是Python中的主要选择工具, 代表Python程序所拥有的大多数逻辑! Python的if语句的格式为if测试, 后面跟着一个或多个可选的elif以及一个最终可选的else, 在if执行时Python会测试第一个计算结果为真的代码块, 如果测试都为假的时候 则执行else.

伪代码示例:

结构1:

```
if expression:
    argument1
    argument2
```

结构2:

```
if expression:
    argument1
else:
    argument2
```

结构3:

```
if expression:
    argument1
elif expression:
    argument2
elif expression:
    argument3
else:
    argument4
```

In [1]:
```
#实例代码1:

age = 20
if age >= 18:
    print 'your age is', age
    print 'adult'
```

```
your age is 20
adult
```

根据Python的缩进规则, 如果if语句判断是true, 就把缩进的两行print语句执行了. 否则什么也不做.

也可以给if添加一个else语句, 意思是如果if判断是false, 不要执行if的内容， 去把else执行了.

In [2]:
```
age = 3
if age >= 18:
    print('your age is', age)
    print('adult')
else:
    print('your age is', age)
    print('teenager')

# 当然上面的判断是很粗略的, 完全可以用elif做更细致的判断, 我们将其称之为多路分支.
```

```
('your age is', 3)
teenager
```

In [3]:
```
age = 3
if age >= 18:
    print('adult')
elif age >= 6:
    print('teenager')
else:
    print('kid')
```

```
kid
```

if语句执行有个特点, 它是从上往下判断, 如果在某个判断上是True, 把该判断对应的语句执行后, 就忽略掉剩下的elif和else， 所以， 请测试并解释为什么下面的程序打印的是teenager.

In [4]:
```
age = 20
if age >= 6:
    print('teenager')
elif age >= 18:
    print('adult')
else:
    print('kid')
```

```
teenager
```

Python的语法规则

1. 语句是逐个运行的Python会按照次序从头到尾执行文件中嵌套块当中的语句, 像if以及循环for这样的语句会让解释器在程序内跳跃, Python在程序中经过的路径叫做控制流程, 像if这种会对控制流程产生影响的语句叫做流程控制语句.
2. 块和语句的边界会自动检测 Python中没有begin或end这样的分隔符, 也没用大括号对包住一个区块的语句, Python使用的是首行缩进的形式区分代码的逻辑, 语句结束也不用以分号结尾.

In [5]:
```
#三元表达式

x = 10
if x > 8:
    print 'hello'
else:
    print 'world'

#以上为条件控制语句的传统写法， 似乎代码不够简洁优雅， 我们用三元表达式来实现相同的功能
```

```
hello
```

In [6]:
```
x = 10
'hello!' if x > 8 else 'world'
```

Out[6]:
```
'hello!'
```

In [7]:
```
'good' if 1 else 'bad'
```

Out[7]: 'good'

In [8]:
```
'good' if 0 else 'bad'
```

Out[8]: 'bad'