



3.2 第一个Python程序

□ 例:判断变量 num是否是正数

```
# 判断变量num是否是正数
num=0
if num > 0:
    print ("num是正数")
else:
    print ("num可能是0")
    print ("num也可能是负数")
```

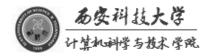
如果 num>0

输出: num是正数

否则

输出: num可能是0

num也可能是负数



□ 注释语句 & 赋值语句

```
# 判断变量num是否是正数
num=0
if num > 0:
    print ("num是正数")
else:
    print ("num可能是0")
    print ("num也可能是负数")
```

注释语句

是对程序进行**说明**的语句, 在程序运行过程中**不被执行**

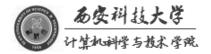
赋值语句

动态类型语言

- 不需要声明变量的语言
- 变量在使用前必须赋值
- 类型检查在运行阶段完成
- Python、JavaScript、Ruby......

静态类型语言

- 必须声明变量
- 类型检查在编译阶段完成
- C C++ Java.....





□ 条件语句 & 语句块

```
# 判断变量num是否是正数
num=②
if num > ②:
    print ("num是正数")
else:
    print ("num可能是0")
    print ("num也可能是负数")
```

```
if num > ∅:
    print ("num是正数")
else:
    print ("num可能是0")
print ("num也可能是负数")
```

条件语句

语句块

Python语句块直接通 过代码的缩进来表示

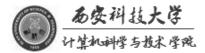
改变语句缩进,程序逻辑发生变化

如果 num>0

输出: num是正数

否则

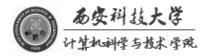
输出: num可能是0 输出num也可能是负数



■ C语言:悬挂else问题

C编译器是**忽略缩进**, 按照**就近原则**配对

```
( a
           if_{\bullet}(b > 0)
              printf("a和b都大于0");
      else
           _
printf("a小于0");
原意
结果
          (a > 0)
           if (b > 0)
               printf("a和b都大于0");
           else
               printf("a小于0");
```



Python 语言基础

■ Python语言

按照缩进来识别语句块,可以有效的避免其他语言中可能出现的错误配对问题

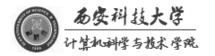
PEP8规范中,

规定语句块的缩进为4个空格

代码缩错误

IndentationError: unexpected indent

常见错误:混用键盘上的Tab键和空格键,造成缩进不一致。





□ 大小写敏感

```
#判断变量num是否是正数
num=0
Num=1
Num和num是不同的变量
if num > 0:
    print ("num是正数")
else:
    print ("num可能是0")
    print ("num也可能是负数")
```

```
可以将子句写在同一行上

num=0

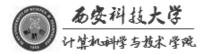
if num > 0: print ("num是正数")
else: print ("num可能是0")
print ("num也可能是负数")
```

运行结果:

num可能是⊘ num也可能是负数—

从新的一行开始打印

print()函数中**自动包含了换行**, 默认每次打印一行



TIPS

Python语句可以以分号结尾 不同的语句可以写在同一行上,以分号隔开

