

# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

# 程序的分支结构

---



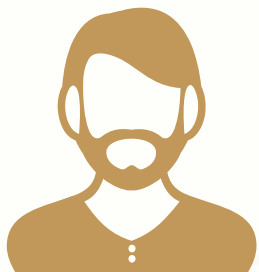
嵩 天  
北京理工大学





# 单元开篇

# 程序的分支结构



- 单分支结构
- 二分支结构
- 多分支结构
- 条件判断及组合
- 程序的异常处理



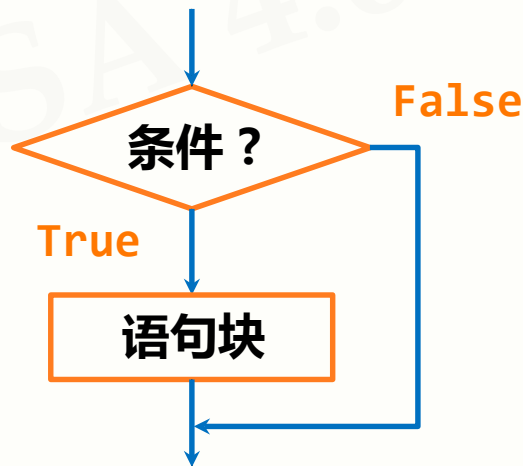


# 单分支结构

# 单分支结构

根据判断条件结果而选择不同向前路径的运行方式

*if*    <条件> :  
         <语句块>



# 单分支结构

## 单分支示例

```
guess = eval(input())
```

```
if guess == 99:  
    print("猜对了")
```

```
if True:  
    print("条件正确")
```



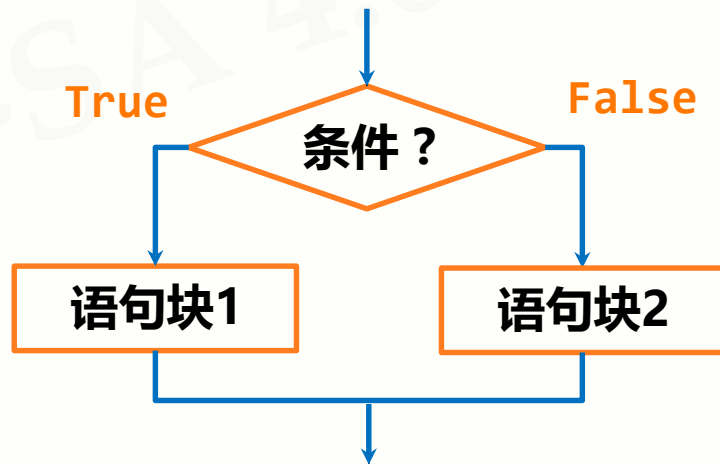
# 二分支结构



# 二分支结构

根据判断条件结果而选择不同向前路径的运行方式

*if* <条件> :  
    <语句块1>  
*else* :  
    <语句块2>



# 二分支结构

## 二分支示例

```
guess = eval(input())
```

```
if guess == 99:
```

```
    print("猜对了")
```

```
else :
```

```
    print("猜错了")
```

```
if True:
```

```
    print("语句块1")
```

```
else :
```

```
    print("语句块2")
```

# 二分支结构

紧凑形式：适用于简单表达式的二分支结构

＜表达式1＞ *if* ＜条件＞ *else* ＜表达式2＞

```
guess = eval(input())
```

```
print("猜{}了".format("对" if guess==99 else "错"))
```



# 多分支结构

# 多分支结构

*if* <条件1> :

<语句块1>

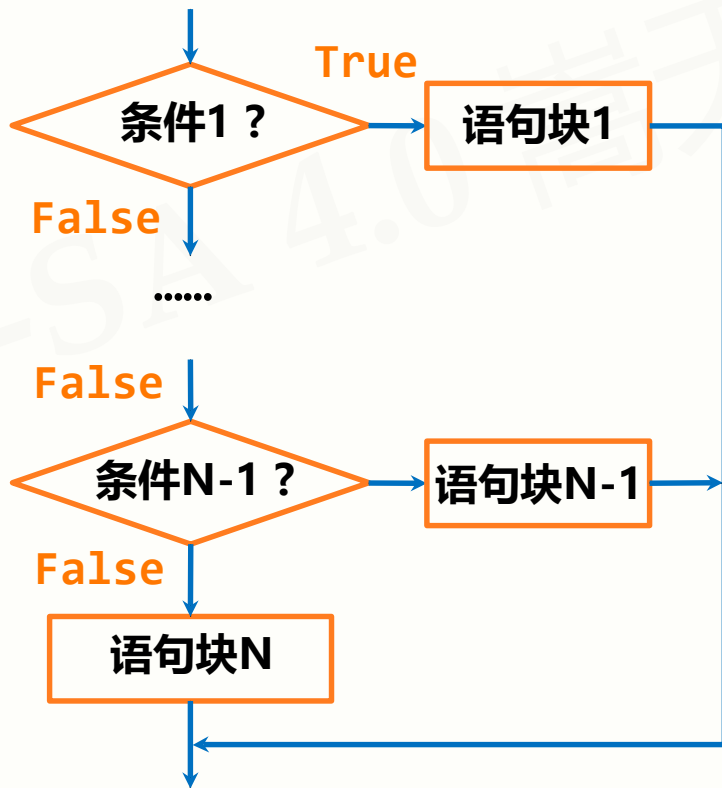
*elif* <条件2> :

<语句块2>

.....

*else* :

<语句块N>



# 多分支结构

## 对不同分数分级的问题

```
score = eval(input())
```

```
if score >= 60:
```

```
    grade = "D"
```

```
elif score >= 70:
```

```
    grade = "C"
```

```
elif score >= 80:
```

```
    grade = "B"
```

```
elif score >= 90:
```

```
    grade = "A"
```

```
print("输入成绩属于级别{}".format(grade))
```

- 注意多条件之间的包含关系

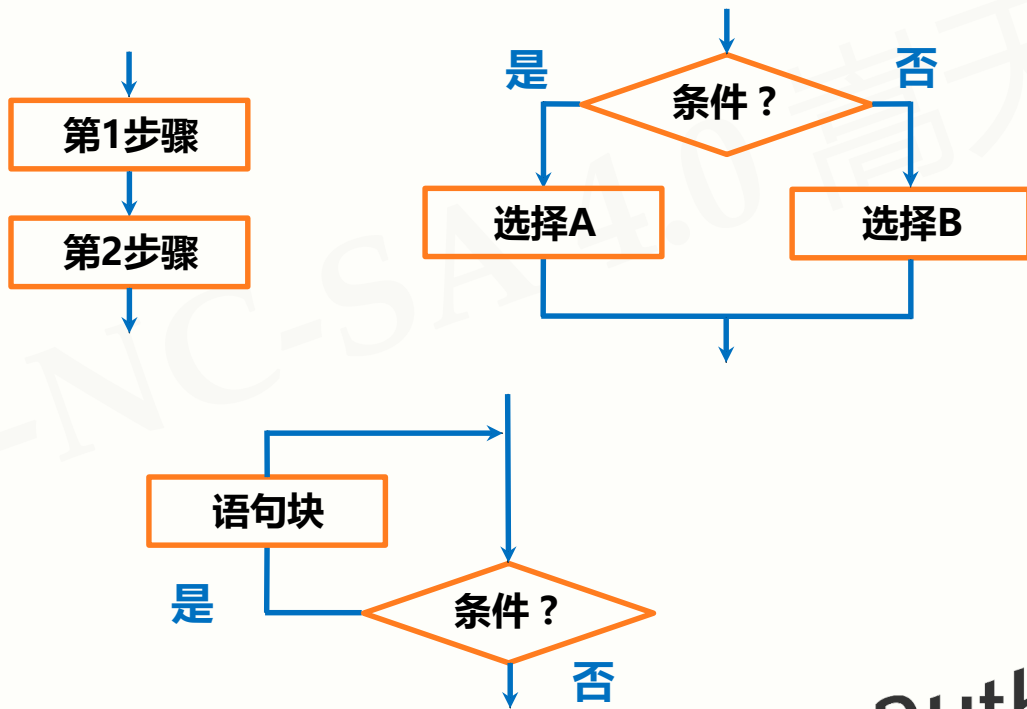
- 注意变量取值范围的覆盖

# “程序的控制结构”

- 顺序结构

- 分支结构

- 循环结构





# 条件判断及组合



# 条件判断

## 操作符

操作符	数学符号	描述
<	<	小于
<=	≤	小于等于
>=	≥	大于等于
>	>	大于
==	=	等于
!=	≠	不等于

# 条件组合

## 用于条件组合的三个保留字

操作符及使用	描述
x <b>and</b> y	两个条件x和y的逻辑 <b>与</b>
x <b>or</b> y	两个条件x和y的逻辑 <b>或</b>
<b>not</b> x	条件x的逻辑 <b>非</b>

# 条件判断及组合

## 示例

```
guess = eval(input())
```

```
if guess > 99 or guess < 99:
```

```
    print("猜错了")
```

```
else :
```

```
    print("猜对了")
```

```
if not True:
```

```
    print("语句块2")
```

```
else :
```

```
    print("语句块1")
```



# 程序的异常处理

# 异常处理

```
num = eval(input("请输入一个整数: "))  
print(num**2)
```

**当用户没有输入整数时，会产生异常，怎么处理？**

# 异常处理

## 异常发生的代码行数

Traceback (most recent call last):

File "t.py", line 1, in <module>

num = eval(input("请输入一个整数: "))

File "<string>", line 1, in <module>

NameError: name 'abc' is not defined

异常类型

异常内容提示

# 异常处理

## 异常处理的基本使用

*try* :

＜语句块1＞

*except* :

＜语句块2＞

*try* :

＜语句块1＞

*except* ＜异常类型＞ :

＜语句块2＞

# 异常处理

## 示例1

*try :*

```
num = eval(input("请输入一个整数: "))
```

```
print(num**2)
```

*except :*

```
print("输入不是整数")
```



# 异常处理

## 示例2

*try :*

```
num = eval(input("请输入一个整数: "))
```

```
print(num**2)
```

*except* NameError:

```
print("输入不是整数")
```

标注异常类型后，仅响应此类异常

异常类型名字等同于变量名

# 异常处理

*try* :

<语句块1>

异常处理的高级使用

*except* :

<语句块2>

*else* :

<语句块3>

*finally* :

<语句块4>

- *finally* 对应语句块4一定执行

- *else* 对应语句块3在不发生异常时执行



# 单元小结

# 程序的分支结构

- 单分支 *if* 二分支 *if-else* 及紧凑形式
- 多分支 *if-elif-else* 及条件之间关系
- *not and or > >= == <= < !=*
- 异常处理 *try-except-else-finally*





# 小花絮

## 关于小花絮的小花絮

- 在每周正式教学单元课件最后，都有一个**小花絮**部分
- 小花絮提供了一批**有趣资讯**，作为每个单元学习的**奖励**
- 小花絮鼓励大家在观看视频后**浏览课件**进行简要复习



