

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.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

程序的循环结构



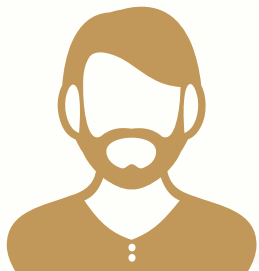
嵩 天
北京理工大学





单元开篇

程序的循环结构



- 遍历循环
- 无限循环
- 循环控制保留字
- 循环的高级用法





遍历循环

遍历循环

遍历某个结构形成的循环运行方式

for <循环变量> *in* <遍历结构> :
<语句块>

- 从遍历结构中逐一提取元素，放在循环变量中

遍历循环



for <循环变量> *in* <遍历结构> :
 <语句块>

- 由保留字for和in组成，完整遍历所有元素后结束
- 每次循环，所获得元素放入循环变量，并执行一次语句块

遍历循环的应用

计数循环(N次)

```
for i in range(N) :
```

<语句块>

- 遍历由range()函数产生的数字序列，产生循环

遍历循环的应用

计数循环(N次)

```
>>> for i in range(5):  
    print(i)
```

0

1

2

3

4

```
>>> for i in range(5):  
    print("Hello:",i)
```

Hello: 0

Hello: 1

Hello: 2

Hello: 3

Hello: 4

遍历循环的应用

计数循环(特定次)

```
for i in range(M,N,K) :
```

<语句块>

- 遍历由range()函数产生的数字序列，产生循环

遍历循环的应用

计数循环(特定次)

```
>>> for i in range(1,6):  
    print(i)
```

1
2
3
4
5

```
>>> for i in range(1,6,2):  
    print("Hello:",i)
```

Hello: 1
Hello: 3
Hello: 5

遍历循环的应用

字符串遍历循环

for c *in* s :

<语句块>

- s是字符串，遍历字符串每个字符，产生循环

遍历循环的应用

字符串遍历循环

```
>>> for c in "Python123":  
        print(c, end=",")
```

P,y,t,h,o,n,1,2,3,

遍历循环的应用

列表遍历循环

```
for item in ls :
```

 <语句块>

- ls是一个列表，遍历其每个元素，产生循环

遍历循环的应用

列表遍历循环

```
>>> for item in [123, "PY", 456] :  
        print(item, end=",")
```

123,PY,456,

遍历循环的应用

文件遍历循环

```
for line in fi :
```

<语句块>

- fi是一个文件标识符，遍历其每行，产生循环

遍历循环的应用

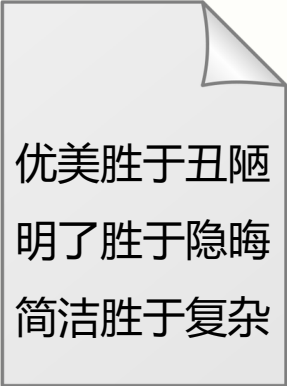
文件遍历循环

```
>>> for line in fi :  
        print(line)
```

优美胜于丑陋

明了胜于隐晦

简洁胜于复杂



优美胜于丑陋
明了胜于隐晦
简洁胜于复杂

遍历循环

for <循环变量> *in* <遍历结构> :
 <语句块>

- 计数循环(N次)
- 列表遍历循环
- 计数循环(特定次)
- 文件遍历循环
- 字符串遍历循环
-



无限循环

无限循环

由条件控制的循环运行方式

while <条件> :



<语句块>

- 反复执行语句块，直到条件不满足时结束

无限循环的应用

无限循环的条件

```
>>> a = 3
>>> while a > 0 :
    a = a - 1
    print(a)
```

2

1

0

```
>>> a = 3
>>> while a > 0 :
    a = a + 1
    print(a)
```

4

5

... (CTRL + C 退出执行)



循环控制保留字

循环控制保留字


break 和 continue

- **break**跳出并结束当前整个循环，执行循环后的语句
- **continue**结束当次循环，继续执行后续次数循环
- **break**和**continue**可以与**for**和**while**循环搭配使用

循环控制保留字


break 和 continue

```
>>> for c in "PYTHON" :  
    if c == "T" :  
        continue  
    print(c, end="")
```



PYHON

```
>>> for c in "PYTHON" :  
    if c == "T" :  
        break  
    print(c, end="")
```



PY

循环控制保留字

```
>>> s = "PYTHON"
>>> while s != "" :
    for c in s :
        print(c, end="")
    s = s[:-1]
```

PYTHONPYTHOPYTHPYTPYP

```
>>> s = "PYTHON"
>>> while s != "" :
    for c in s :
        if c == "T" :
            break
        print(c, end="")
    s = s[:-1]
```

PYPYPYPYPYP

- **break** 仅跳出当前最内层循环



循环的高级用法

循环的扩展

循环与else

<i>for</i> <变量> <i>in</i> <遍历结构> :	<i>while</i> <条件> :
<语句块1>	<语句块1>
<i>else</i> :	<i>else</i> :
<语句块2>	<语句块2>

循环的扩展

循环与else

- 当循环没有被break语句退出时，执行else语句块
- else语句块作为"正常"完成循环的奖励
- 这里else的用法与异常处理中else用法相似

循环的扩展

循环与else

```
>>> for c in "PYTHON" :  
    if c == "T" :  
        continue  
    print(c, end="")  
else:  
    print("正常退出")
```

PYTHON正常退出

```
>>> for c in "PYTHON" :  
    if c == "T" :  
        break  
    print(c, end="")  
else:  
    print("正常退出")
```

PY



单元小结

程序的循环结构

- *for...in* 遍历循环: 计数、字符串、列表、文件...
- *while* 无限循环
- *continue* 和 *break* 保留字: 退出当前循环层次
- 循环 *else* 的高级用法: 与 *break* 有关





小花絮

资深程序员如何看待编程工具？

PyCharm? No! Visual Studio? No! Anaconda? No!

- 编程工具琳琅满目，资深程序员都在用什么？如何选择？
- 资深程序员**更**理解逻辑、**更**期待效率、**更**重视简洁，因此：
 - 资深程序员都**不用**集成开发环境，**不用**复杂调试工具，**不用**复杂图形界面工具
 - 资深程序员都喜欢用**编辑器**类型的开发工具，小巧、灵活、可定制
 - 建议：Visual Studio Code (VSCode) 、Notepad++、Vim，足矣！

请初学者"老老实实"用IDLE，这个工具足够了

