

PYTHON BASICS

流程控制

2020-04-16

STEVEN WANG



上节总结

	字符串	列表	元组	字典	集合
创建	'string'	[1, 2]	(1, 2)	{'a':1, 'b':2}	{1, 2}
修改?	否	是	否	是	是
重复?	是	是	是	是	否
有序?	是	是	是	否	否
索引?	是	是	是	是	否
切片?	是	是	是	否	否
+, *?	是	是	是	否	否
in?	是	是	是	否	否

运行程序

流程控制

报错

正常

错误类型

异常处理

顺序

statement 1
statement 2
...
statement n

条件

if expr:
 statement 1
else:
 statement 2

重复

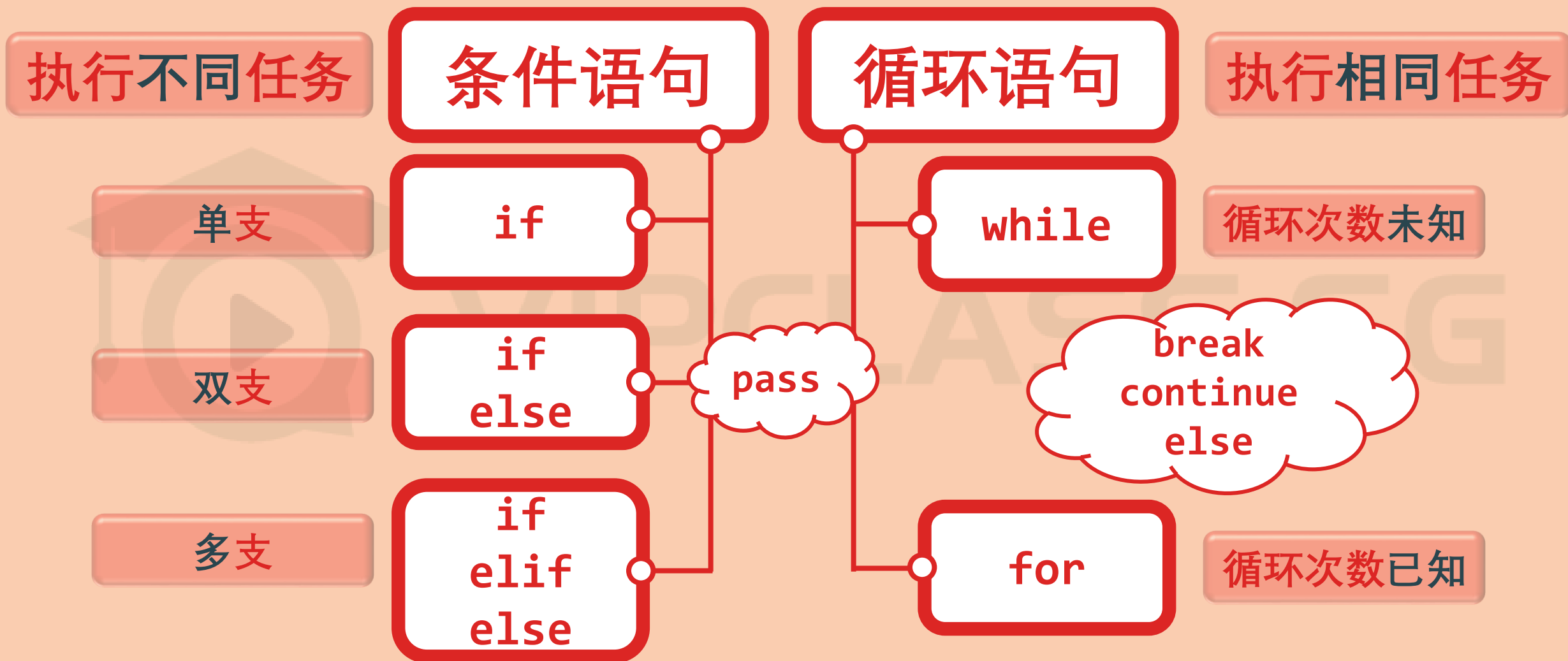
while expr:
 do the same thing
for a in A:
 do the same thing

不同任务

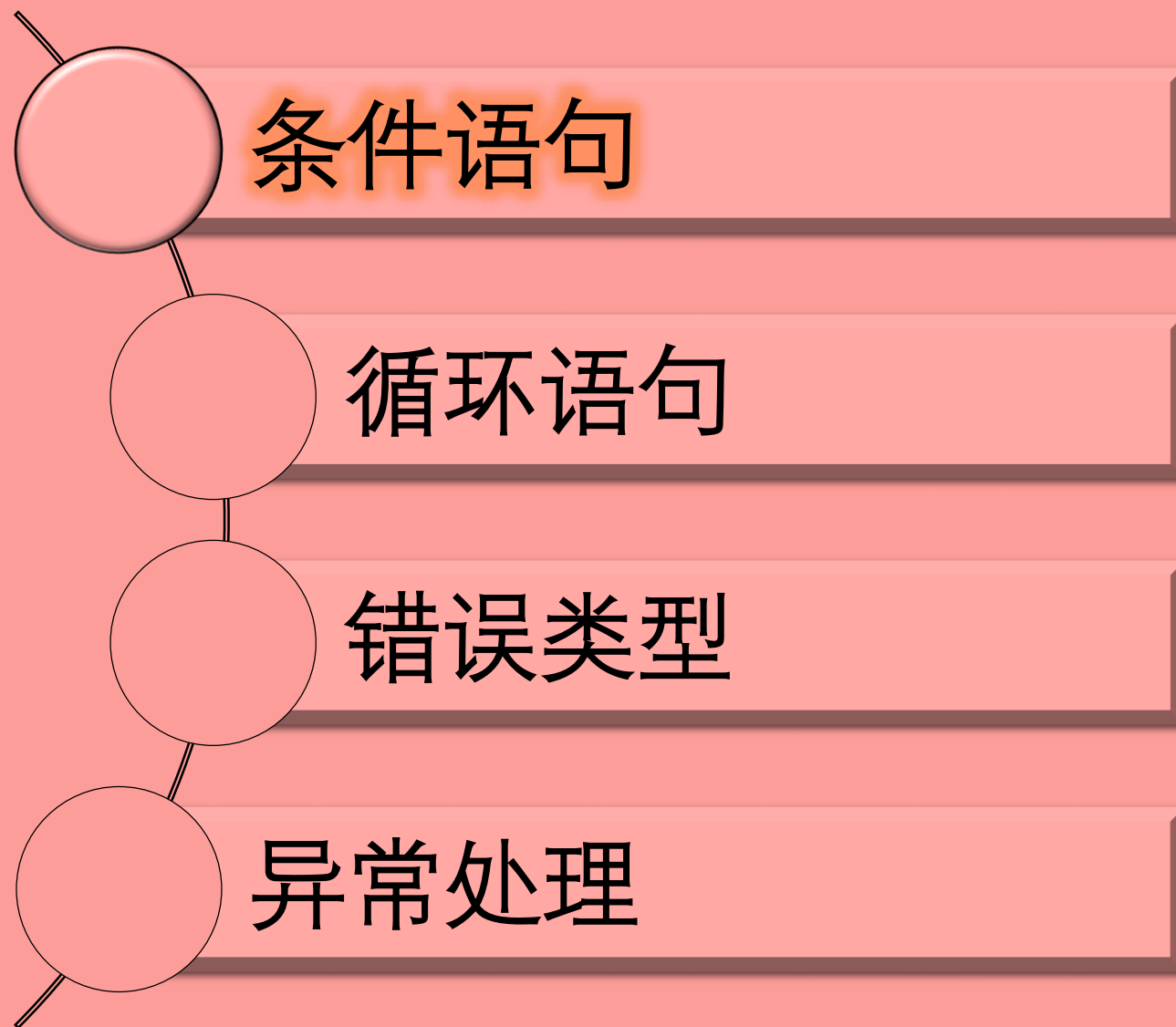
相同任务

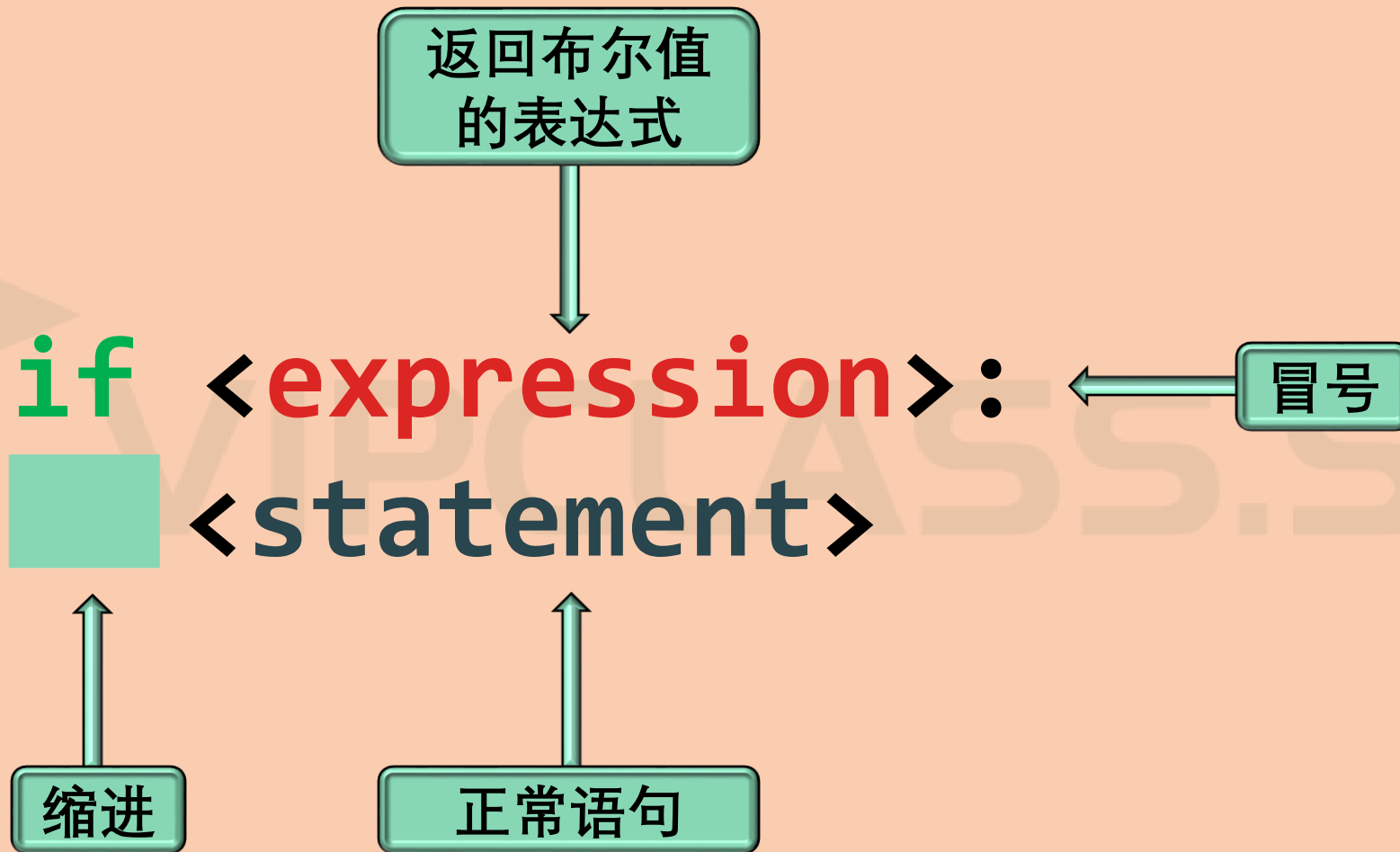
AssertionError
AttributeError
SyntaxError
TypeError
NameError
IndentationError
OSError
IndexError
KeyError
ValueError
ZeroDivisionError

```
try:  
# 事先预计可能会出错的代码  
except Exception:  
# 一旦异常被处理时运行的代码  
else:  
# 没有任何异常时运行的代码  
finally:  
# 任何情况下都会运行的代码
```



流程控制



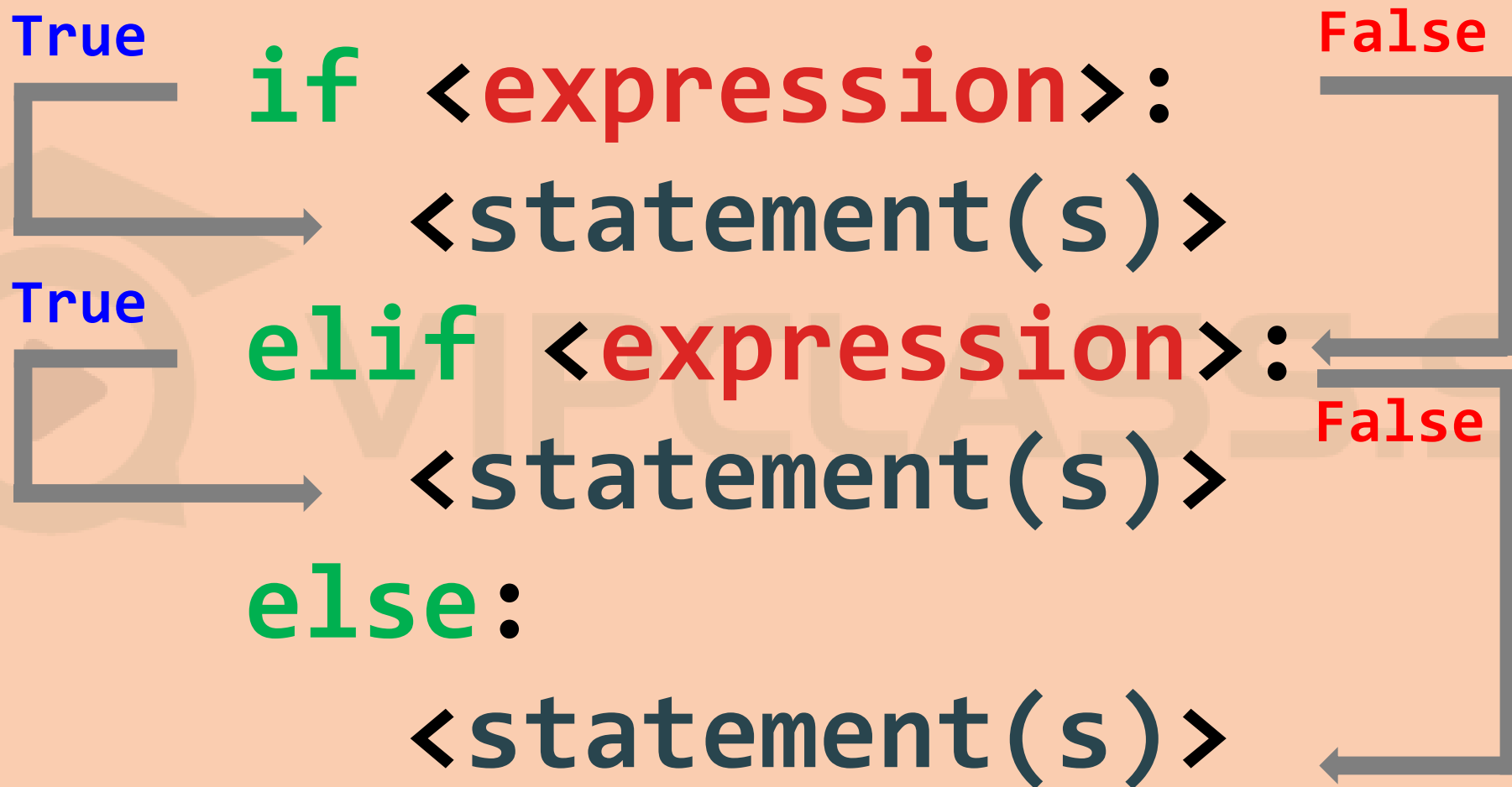




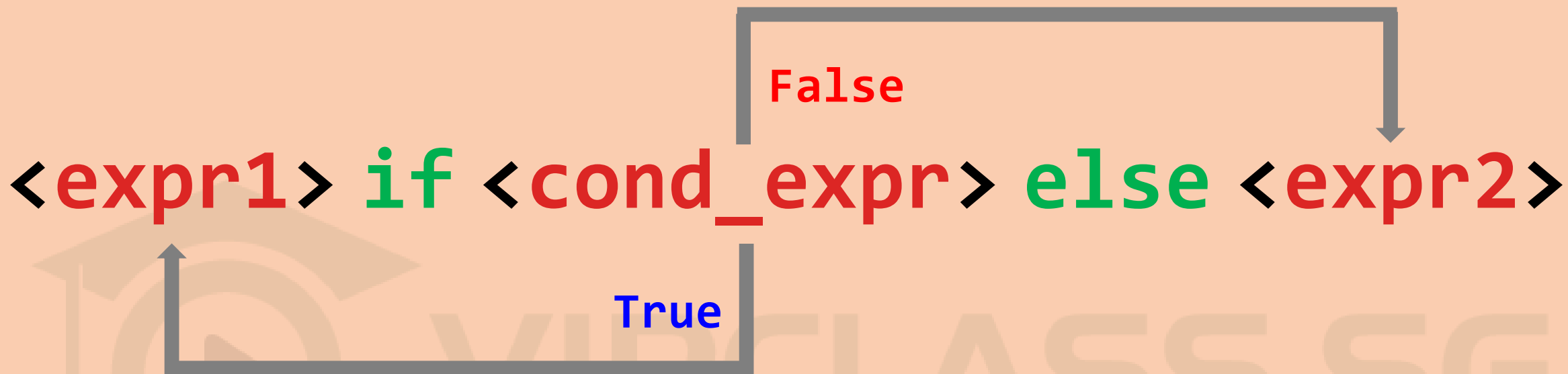
True False

```
if <expression>:  
    <statement(s)>  
else:  
    <statement(s)>
```









找出两个
数最大值，
并加上 1

```
x, y = 8, 9
```

```
1 + x if x > y else y
```

```
9
```



```
1 + (x if x > y else y)
```

```
10
```



有时我们想在某个地方编写一段代码，但是还未想好怎么具体来写，这时我们可以用关键词 **pass** 来先占位。

```
if True:  
    print('Done')
```

```
IndentationError: expected an indented block
```

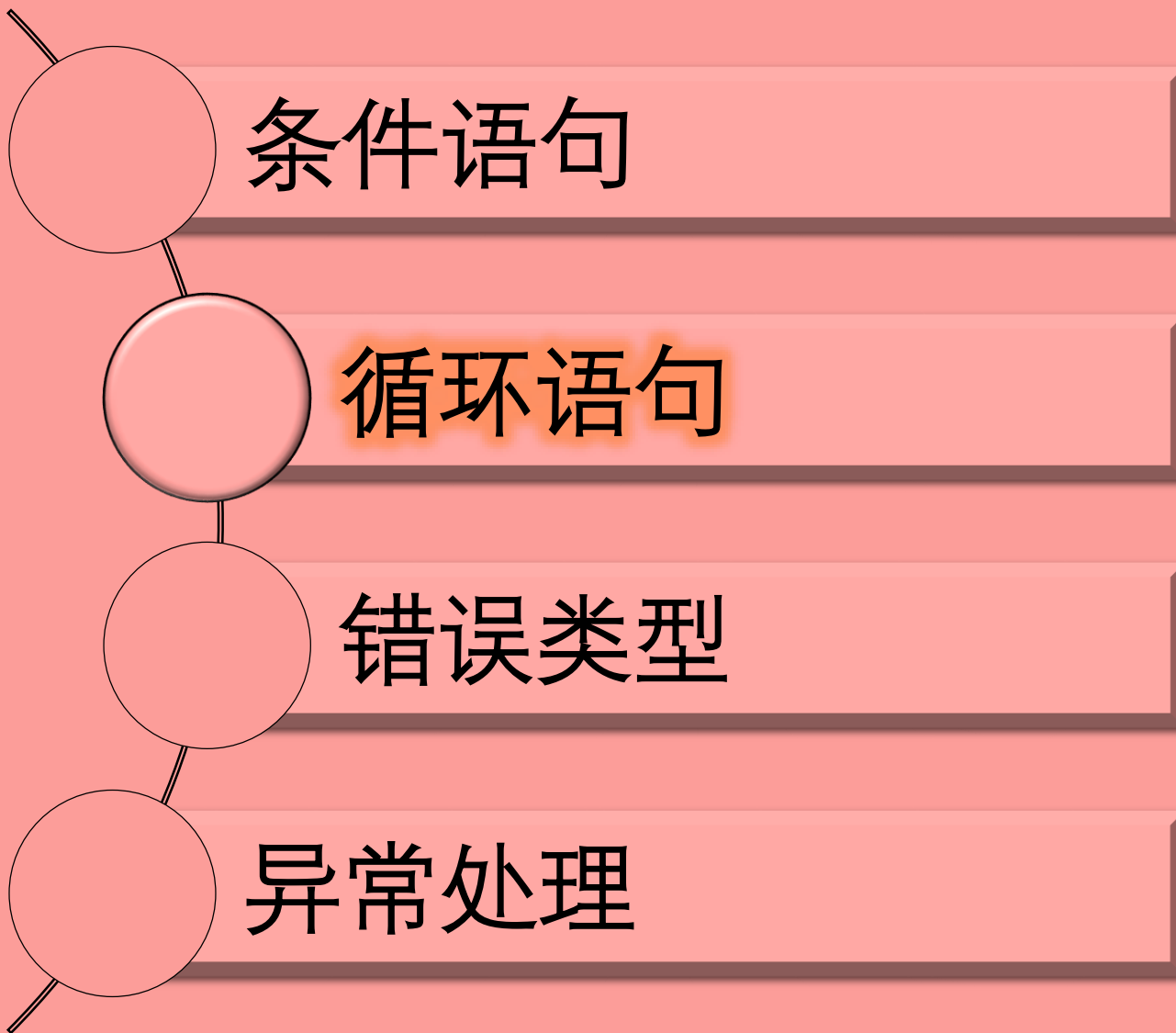


```
if True:  
    pass  
print('Done')
```

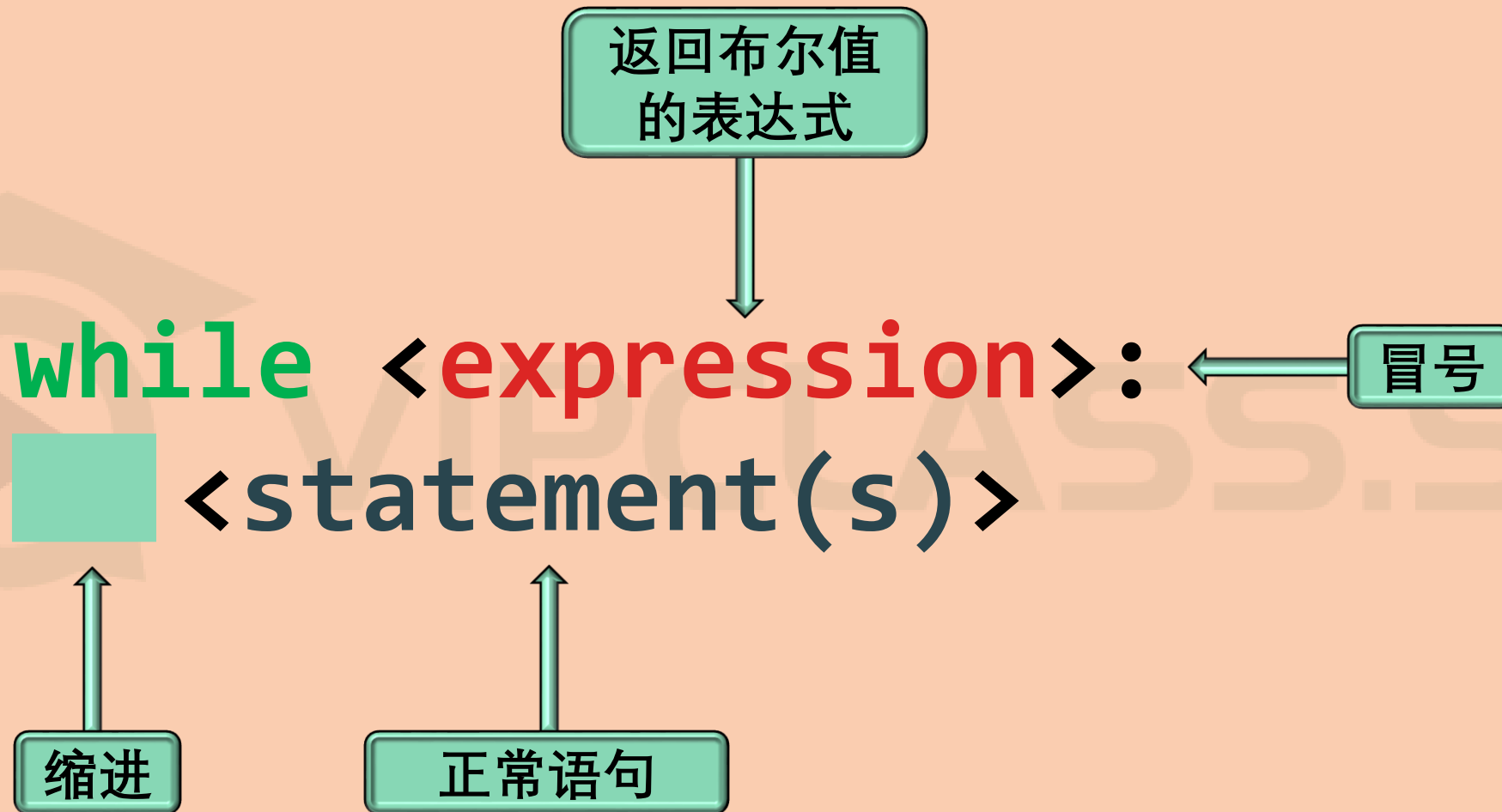
```
Done
```



流程控制



不确定次数的 while 循环



无限
while

```
while True:  
    <statement(s)>
```

嵌套
while

```
while <expression>:  
    while <expression>:  
        <statement(s)>  
    <statement(s)>
```

单行
while

```
while <expression>: <statement(s)>
```

确定次数的 for 循环

BASIC, Pascal

```
for i = 1 to 10  
  <ops on i>
```

C++, Jave, PHP, Perl

```
for (i = 1; i <= 10; i++)  
  <ops on i>
```

Python

```
for i in range(10)  
  <ops on i>
```

对一个长度为 10 容器型数据 A，如列表
A = [10, 3, 4, 5, 9, 100, 3, 67, 60, 1]

BASIC, Pascal

```
for i = 1 to 10  
  <ops on A[i]>
```

C++, Jave, PHP, Perl

```
for (i = 1; i <= 10; i++)  
  <ops on A[i]>
```

Python

```
for var in A  
  <ops on var>
```




元素

可迭代对象

```
for <var> in <iterable>:  
    <statement(s)>
```

Iterable

 $A = [1, 2, 3]$

iter()

Iterator

 $i = \text{iter}(A)$

next(i) → 1

next(i) → 2

next(i) → 3

next(i) → StopIteration

```
for (idx, var) in enumerate(iterable):  
    <statement(s)>
```

Iterable

```
A = ['Python', 'Matlab', 'VBA']
```

enumerate(iter())

Iterator

```
i = enumerate  
    (iter(A))
```

next(i) → (0, 'Python')

next(i) → (1, 'Matlab')

next(i) → (2, 'VBA')

next(i) → StopIteration

```
for (i1, i2) in zip(iter1, iter2):  
    <statement(s)>
```

Iterable

```
A = ['love', 'like', 'hate']  
B = ['Python', 'Matlab', 'VBA']
```

zip(iter(), iter())

Iterator

```
i = zip  
(iter(A), iter(B))
```

next(i) → ('love', 'Python')

next(i) → ('like', 'Matlab')

next(i) → ('hate', 'VBA')

next(i) → StopIteration

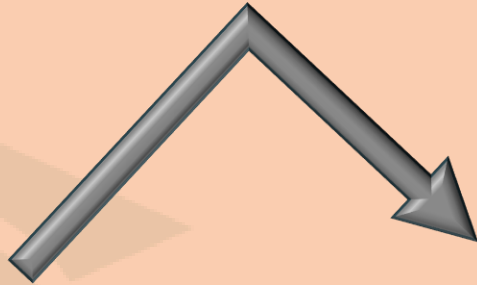
zip(

1	2	3
---	---	---

 ,

4	5	6
---	---	---

)



1	2	3
---	---	---

4	5	6
---	---	---

1	4
---	---

2	5
---	---

3	6
---	---

zip(

1	4
---	---

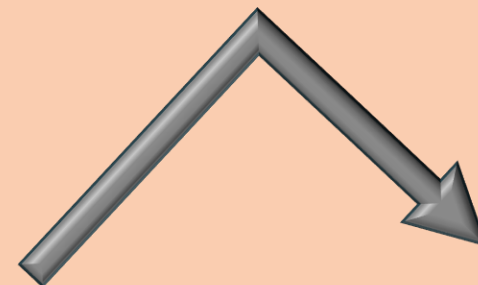
 ,

2	5
---	---

 ,

3	6
---	---

)



1	2	3
---	---	---

4	5	6
---	---	---

break continue else



<statement>

break

<statement>

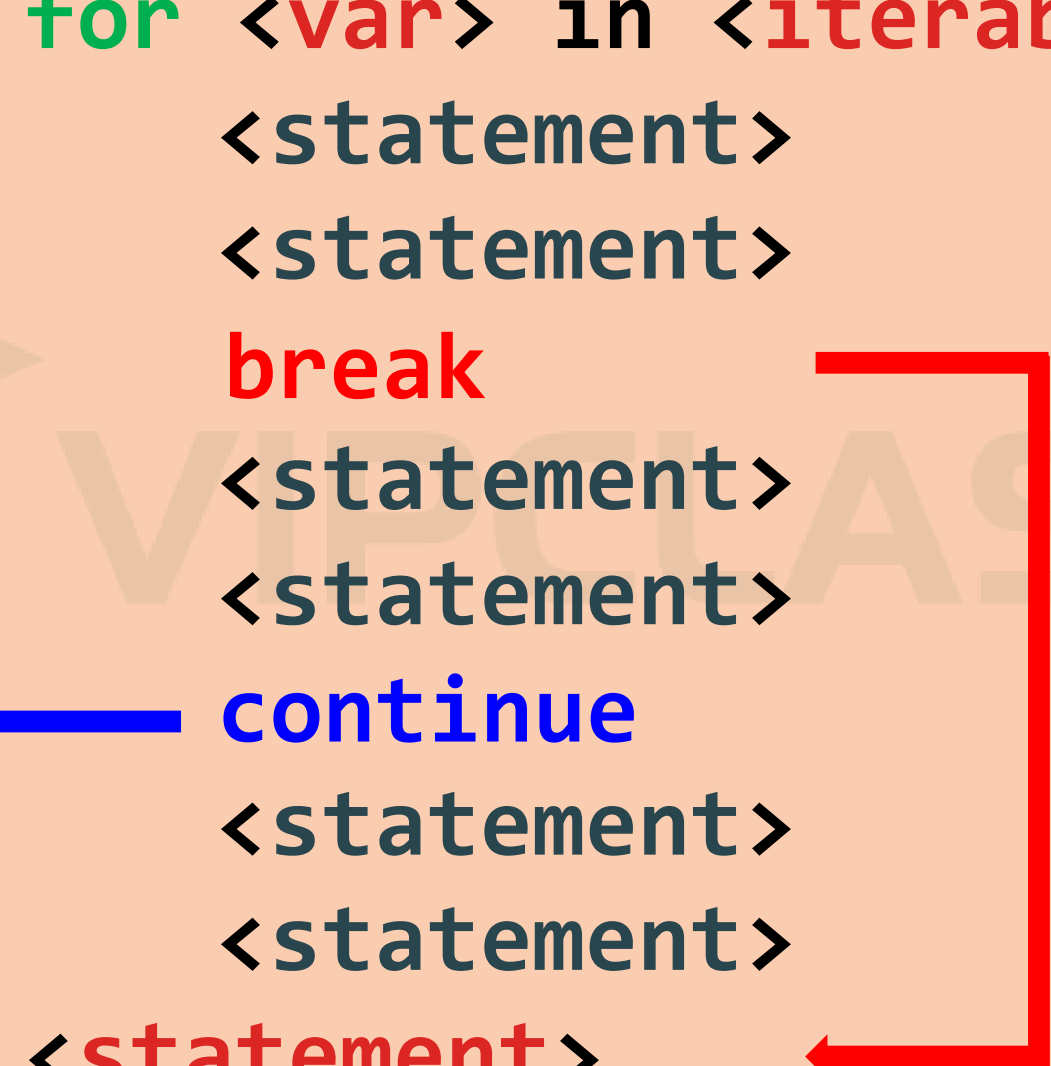
continue

<statement>

<statement>

<statement>

```
for <var> in <iterable>:  
    <statement>  
    <statement>  
    break  
    <statement>  
    <statement>  
    continue  
    <statement>  
    <statement>  
<statement>
```



while <expression>:	if <expression>:
<statement(s)>	<statement(s)>
else:	else:
<statement(s)>	<statement(s)>

```
while <expression>:  
    <statement(s)>
```

```
else:
```

```
    <statement(s)>
```

当 while 循环正
常结束时执行

```
while <expression>:  
    <statement(s)>  
<statement(s)>
```

什么时候
都执行


```
for <var> in <iterable>:  
    <statement(s)>  
else:  
    <statement(s)>
```

```
for <var> in <iterable>:  
    <statement(s)>
```

```
else:
```

```
    <statement(s)>
```

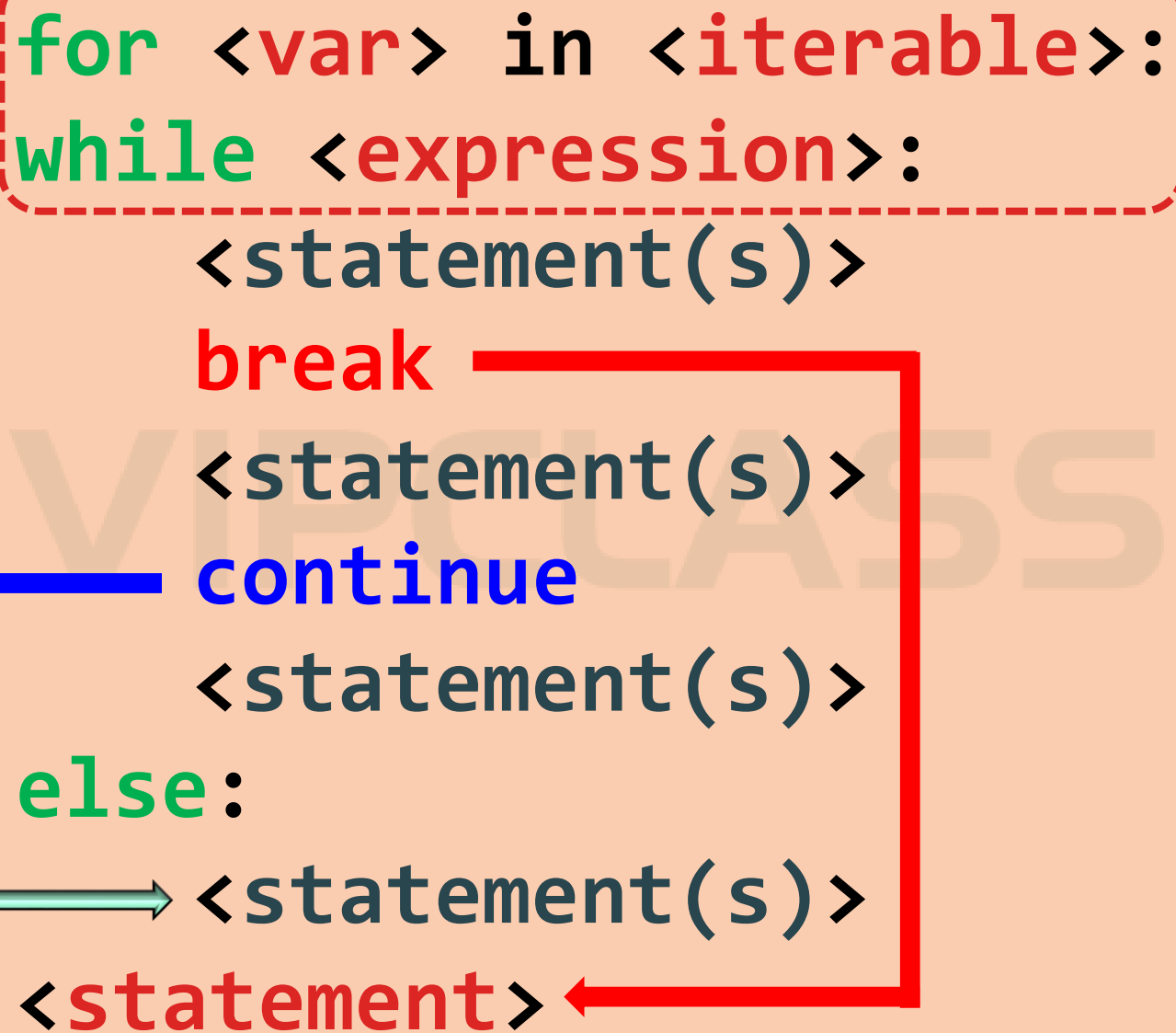
当 for 循环正常结束时执行

```
for <var> in <iterable>:  
    <statement(s)>
```

```
<statement(s)>
```

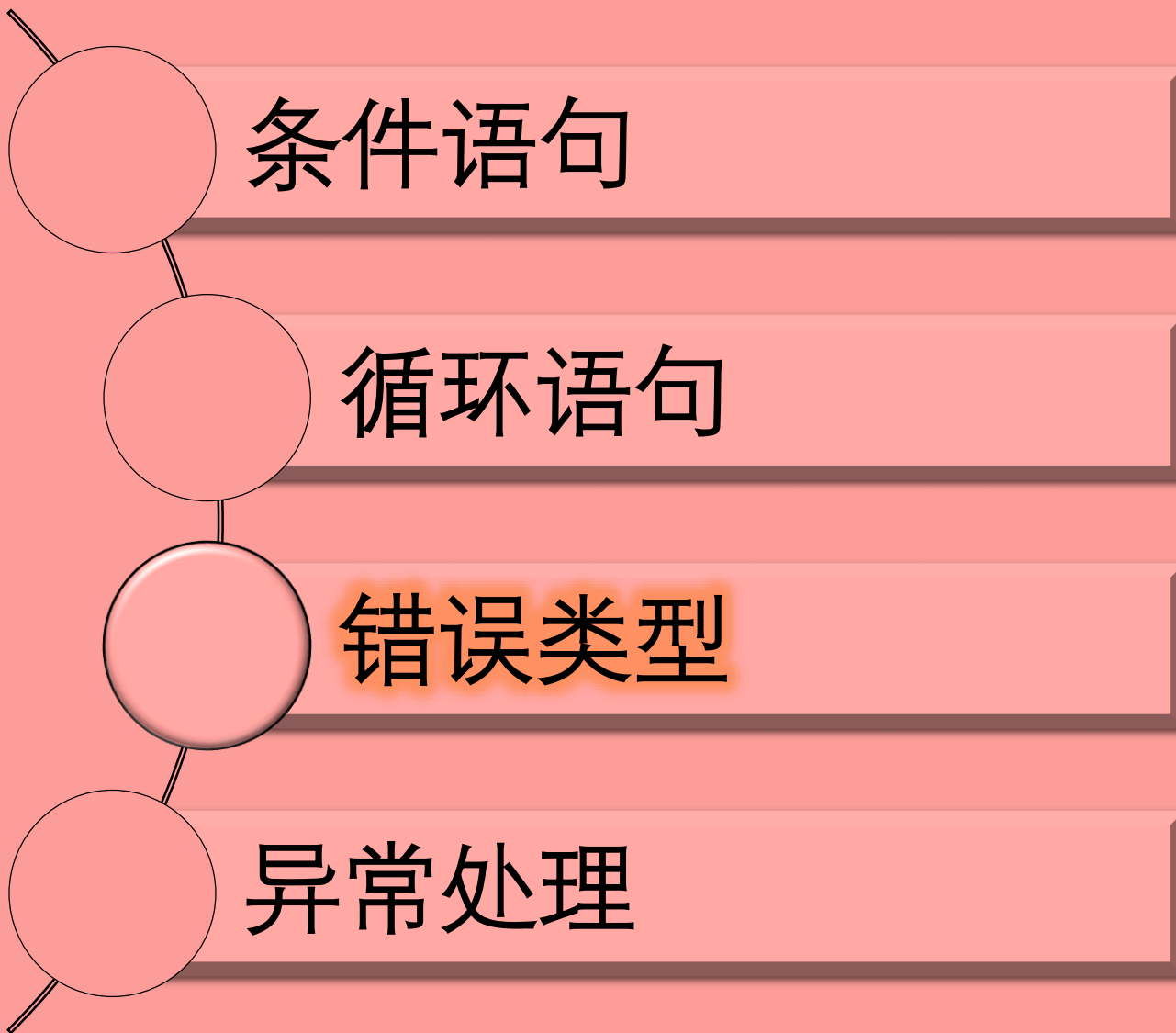
什么时候都执行

```
for <var> in <iterable>:  
while <expression>:  
    <statement(s)>  
    break  
    <statement(s)>  
    continue  
    <statement(s)>  
else:  
    <statement(s)>  
<statement>
```



循环正常
结束时执行

流程控制



编译时报错

语法错误

昨天，送给我几本书。

```
print('I love you)
```



运行时报错

运行错误

我飞到月球去看风景。

```
a = 10  
b = 0  
c = a/b
```



运行完不报错

逻辑错误

因为我满仓入市，
所以股市一定涨。

```
(a, b) = (10, 8)  
print('The mean of  
a and b is', a+b/2)
```



编译时报错

语法错误

- 关键词拼错
- 函数最后没带冒
- 字符串最后没引号
- 括号不匹配
- 缩进位置错误



运行时报错

运行错误

- 分母为零
- 整数加字符串
- 使用没定义的变量
- 调用没定义的函数
- 读取不存在的文件



运行完不报错

逻辑错误

- 用错变量名
- 缩进错了整块代码
- 用错操作符优先级
- 用错布尔表达式为

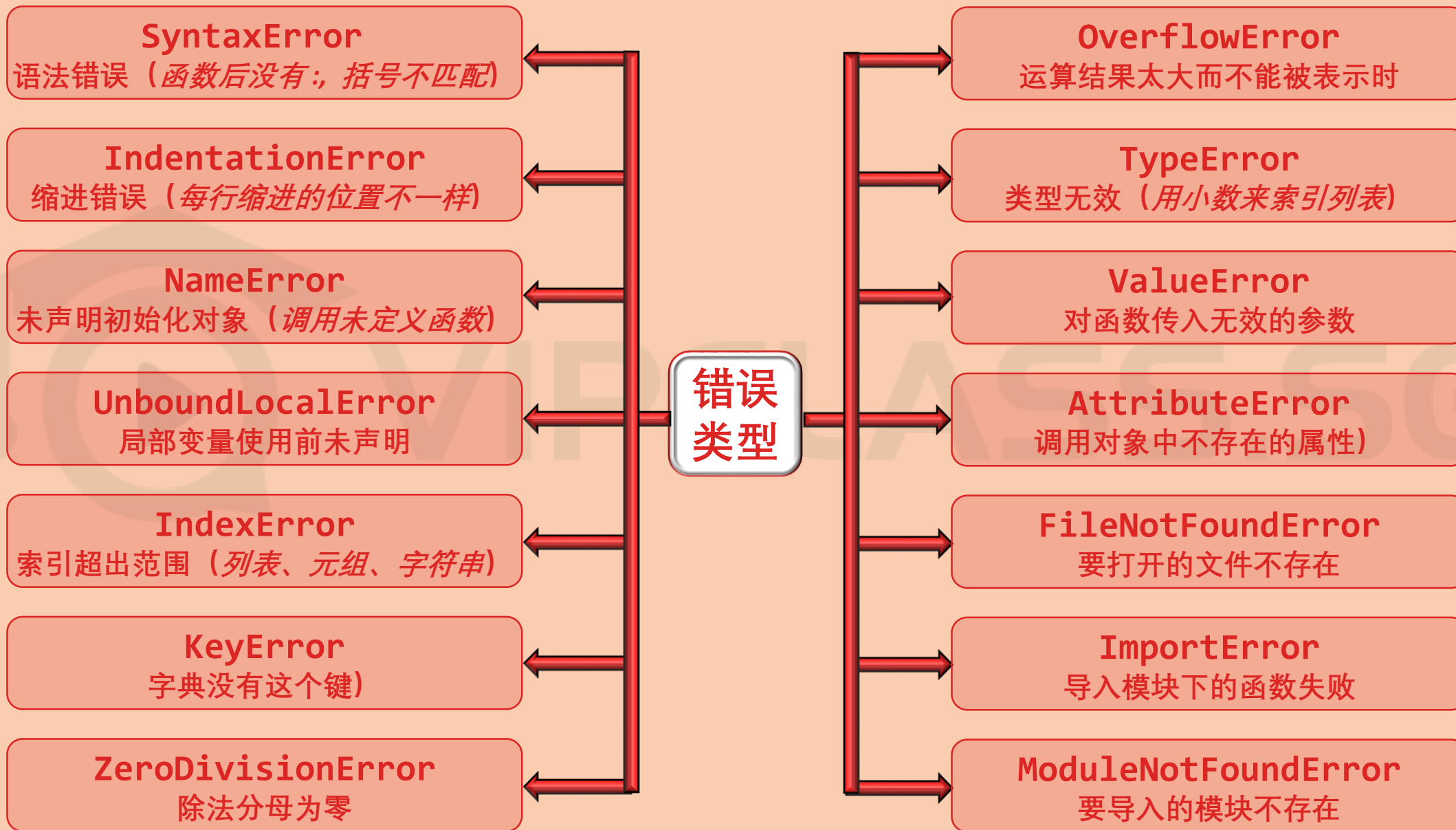


BaseException

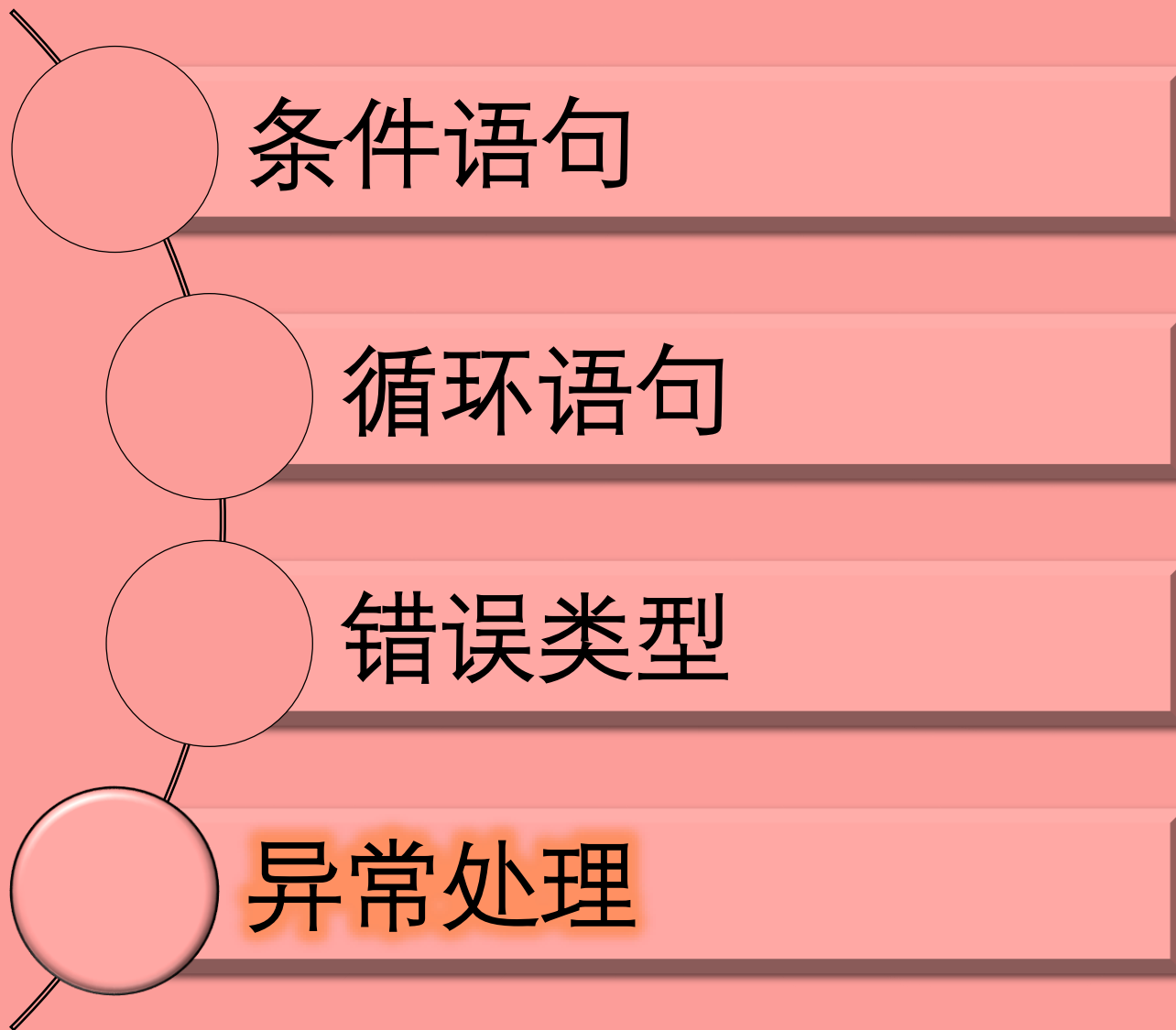
- +-- SystemExit
- +-- **KeyboardInterrupt**
- +-- GeneratorExit
- +-- Exception
 - +-- **StopIteration**
 - +-- ArithmeticError
 - +-- FloatingPointError
 - +-- **OverflowError**
 - +-- **ZeroDivisionError**
 - +-- **AssertionError**
 - +-- **AttributeError**
 - +-- BufferError
 - +-- EOFError
 - +-- **ImportError**
 - +-- **LookupError**
 - +-- **IndexError**
 - +-- **KeyError**
- +-- MemoryError
 - +-- **NameError**
 - +-- UnboundLocalError

+-- OSError

- +-- BlockingIOError
- +-- ChildProcessError
- +-- ConnectionError
- +-- FileExistsError
- +-- **FileNotFoundError**
- +-- InterruptedError
- +-- IsADirectoryError
- +-- NotADirectoryError
- +-- PermissionError
- +-- ProcessLookupError
- +-- TimeoutError
- +-- ReferenceError
- +-- RuntimeError
 - +-- NotImplementedError
- +-- **SyntaxError**
 - +-- **IndentationError**
 - +-- TabError
- +-- SystemError
- +-- **TypeError**
- +-- **ValueError**



流程控制



raise

assert

Exception

try

catch

防患未然，知错就改

语法错误

```
print( 1/0 )
```

```
File "<...>", line 1
```

```
print( 1/0 )
```

```
      ^  
SyntaxError: invalid syntax
```

箭头指向解析器
遇到语法错误的地方

运行错误

```
print( 1/0 )
```

```
Traceback (most recent call last)
```

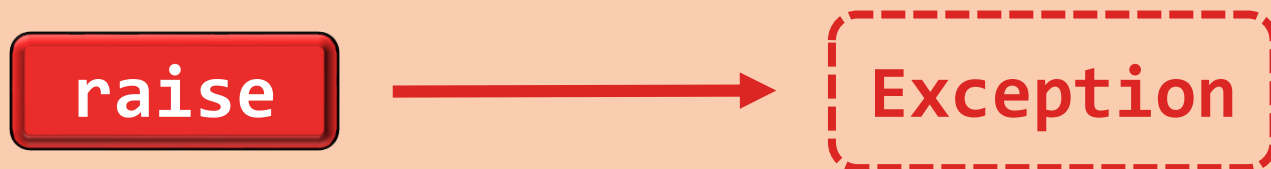
```
<...> in <module>
```

```
----> 1 print( 1/0 )
```

```
ZeroDivisionError: division by zero
```

异常处理
是处理运行错误

如果在某个条件下
你想抛出一个错误



声明某个条件成立
反之抛出一个错误



然后呢？

try:

事先预计可能会出错的代码

except Exception:

任何异常被处理时运行的代码



try:

事先预计可能会出错的代码

except some_exception:

特定异常被处理时运行的代码



try:

事先预计可能会出错的代码

except exception_1:

异常 1 被处理时运行的代码

...

except exception_n:

异常 n 被处理时运行的代码

try:

事先预计可能会出错的代码

except (exception_1, ..., exception_n):

异常 1 到 n 任意一个被处理时运行的代码



try:

事先预计可能会出错的代码

except Exception:

任何异常被处理时运行的代码

else:

没有任何异常时运行的代码

try:

事先预计可能会出错的代码

except Exception:

任何异常被处理时运行的代码

else:

没有任何异常时运行的代码

finally:

任何情况下都会运行的代码

try:

运行这里代码

except:

报错了，运行这里代码

else:

没报错，运行这里代码

finally:

无论如何，运行这里代码

try:

代码 1



× 代码 2



× 代码 3



× 代码 n+1

**except exception_1:**

处理异常 1 发生时运行的代码

except exception_2:

处理异常 2 发生时运行的代码

...

except exception_n:

处理异常 n 发生时运行的代码

else:

没发生异常时运行的代码

finally:

任何时候都要运行的代码

总结

流程
正常

流程
异常

内容	语法	用处
条件语句	if	按条件执行
循环语句	while, for	重复执行
错误类型	*Error	要处理先了解
异常处理	raise/assert try-except- else-finally	先预防 后处理

流程
控制

下节预告：函数

终身学习 快乐学习

王圣元 Steven Wang
微信公众号：王的机器