cp_reference

1 cp基本语法

1.1 输出 print

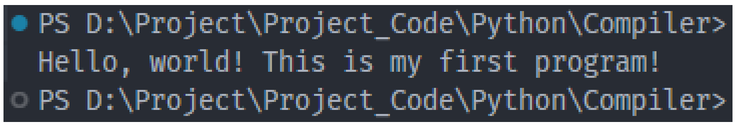
输出格式为

```
print(表达式 [, 表达式, ... , 表达式], [sep=表达式], [end=表达式])
```

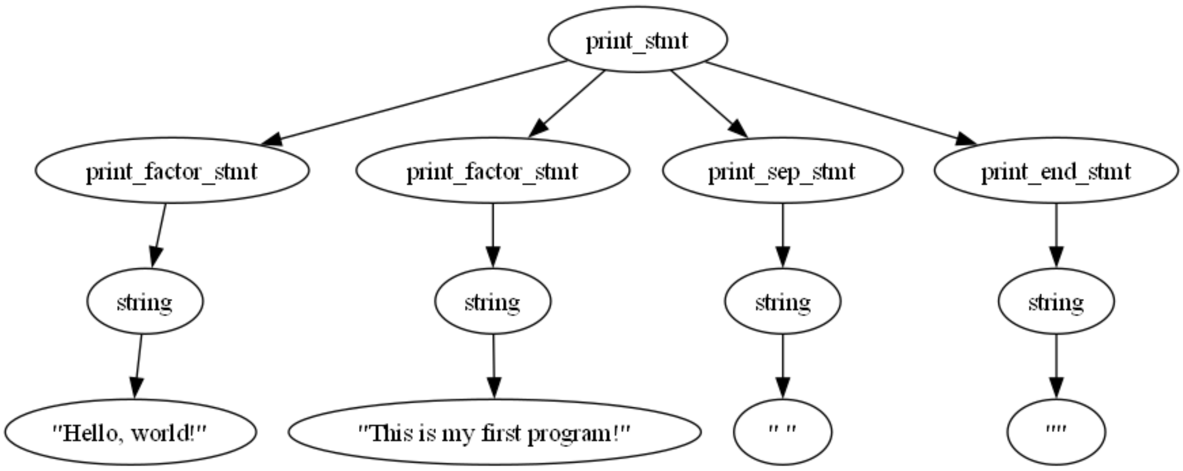
sep 和 **end** 分别是输出间隔符和输出结束符，默认为空格和换行符。

以下是一个简单的 cp 程序，可以输出单词，"Hello World"。

```
print("Hello, world!", "This is my first program!", sep=" ", end="")
```



```
PS D:\Project\Project_Code\Python\Compiler>
Hello, world! This is my first program!
PS D:\Project\Project_Code\Python\Compiler>
```



2 数据类型

2.1 基本数据类型

cp 提供了 4 种基本数据类型，见下表。

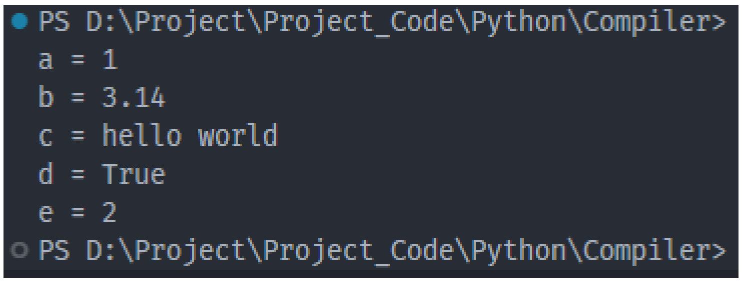
数据类型	关键字
int	整数
float	浮点数
bool	布尔值
string	字符串

2.2 变量声明

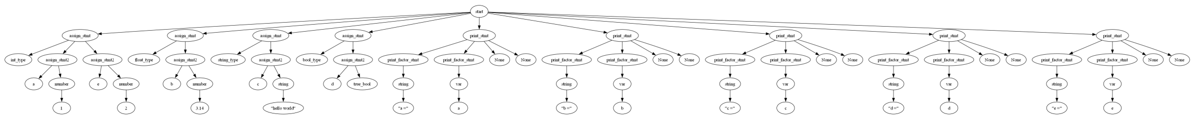
变量声明格式为

```
基本数据类型 变量名 = 初始值 [, 变量名2 = 初始值, ... , 变量名n = 初始值]
```

```
int a = 1, e = 2
float b = 3.14
string c = "hello world"
bool d = True
print("a =", a)
print("b =", b)
print("c =", c)
print("d =", d)
print("e =", e)
```



```
PS D:\Project\Project_Code\Python\Compiler>
  a = 1
  b = 3.14
  c = hello world
  d = True
  e = 2
PS D:\Project\Project_Code\Python\Compiler>
```



3 运算符

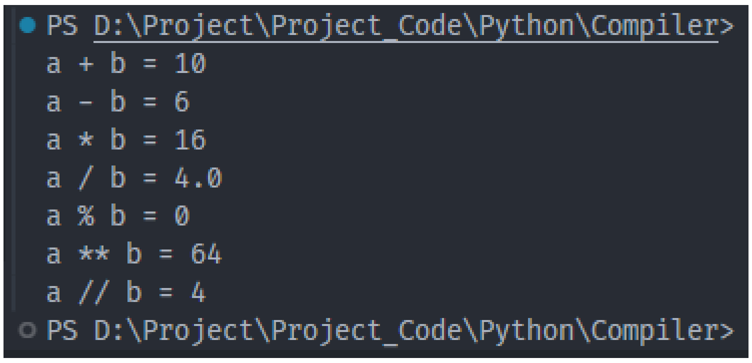
cp 提供以下 4 种类型的运算符：

- 算术运算符
- 比较运算符
- 赋值运算符
- 逻辑运算符
  接下来，我将一一展示。

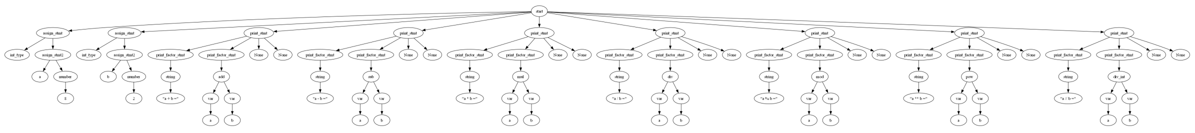
3.1 算术运算符

运算符	描述
+	加 - 两个对象相加
-	减 - 得到负数或是一个数减去另一个数
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串
/	除 - x除以y
%	取模 - 返回除法的余数
**	幂 - 返回x的y次幂
//	取整除 - 返回商的整数部分（向下取整）

```
int a = 8
int b = 2
print("a + b =", a+b)
print("a - b =", a-b)
print("a * b =", a*b)
print("a / b =", a/b)
print("a % b =", a%b)
print("a ** b =", a**b)
print("a // b =", a//b)
```



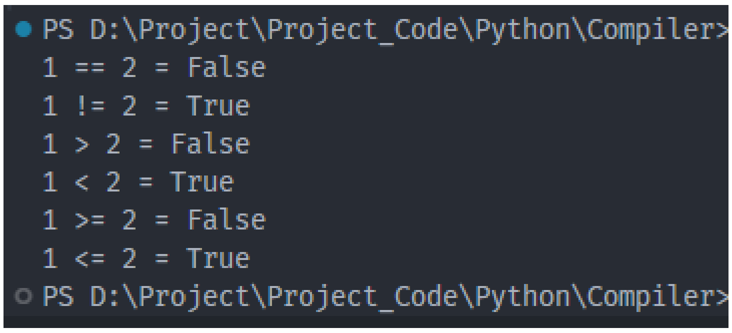
```
PS D:\Project\Project_Code\Python\Compiler>
a + b = 10
a - b = 6
a * b = 16
a / b = 4.0
a % b = 0
a ** b = 64
a // b = 4
PS D:\Project\Project_Code\Python\Compiler>
```



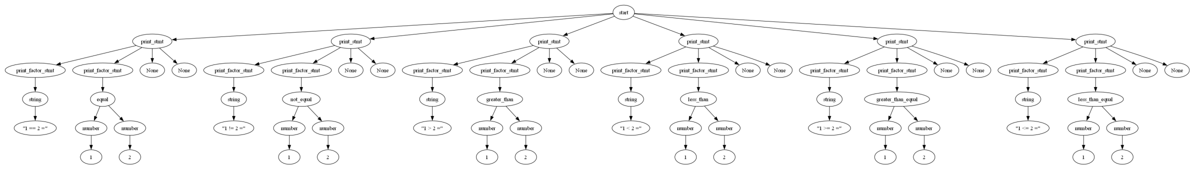
3.2 比较运算符

运算符	描述
==	等于 - 比较对象是否相等
!=	不等于 - 比较两个对象是否不相等
>	大于 - 返回x是否大于y
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量 True 和 False 等价。
>=	大于等于 - 返回x是否大于等于y。
<=	小于等于 - 返回x是否小于等于y。

```
print("1 == 2 =", 1==2)
print("1 != 2 =", 1!=2)
print("1 > 2 =", 1>2)
print("1 < 2 =", 1<2)
print("1 >= 2 =", 1>=2)
print("1 <= 2 =", 1<=2)
```



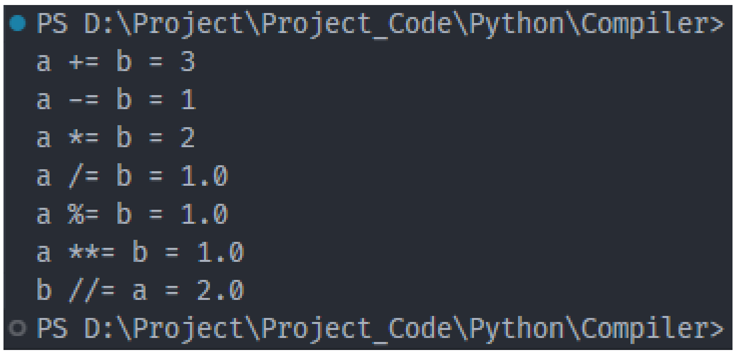
```
PS D:\Project\Project_Code\Python\Compiler>
1 == 2 = False
1 != 2 = True
1 > 2 = False
1 < 2 = True
1 >= 2 = False
1 <= 2 = True
PS D:\Project\Project_Code\Python\Compiler>
```



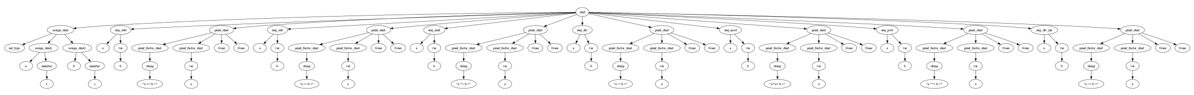
3.3 赋值运算符

运算符	描述
=	简单的赋值运算符
+=	加法赋值运算符
-=	减法赋值运算符
*=	乘法赋值运算符
/=	除法赋值运算符
%=	取模赋值运算符
**=	幂赋值运算符
//=	取整除赋值运算符

```
int a = 1, b = 2
a += b
print("a += b =", a)
a -= b
print("a -= b =", a)
a *= b
print("a *= b =", a)
a /= b
print("a /= b =", a)
a %= b
print("a %= b =", a)
a **= b
print("a **= b =", a)
a //= b
print("a //= b =", a)
```



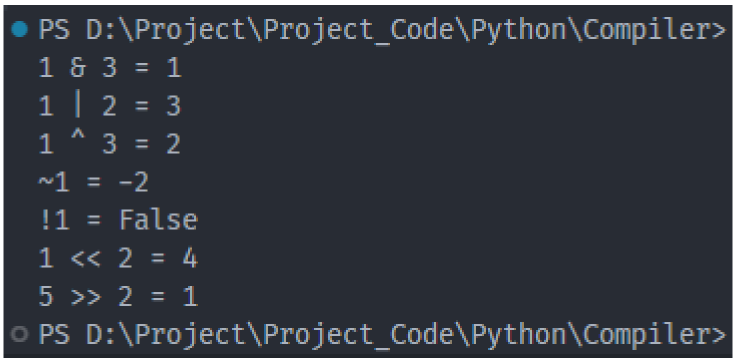
```
PS D:\Project\Project_Code\Python\Compiler>
  a += b = 3
  a -= b = 1
  a *= b = 2
  a /= b = 1.0
  a %= b = 1.0
  a **= b = 1.0
  b //= a = 2.0
PS D:\Project\Project_Code\Python\Compiler>
```



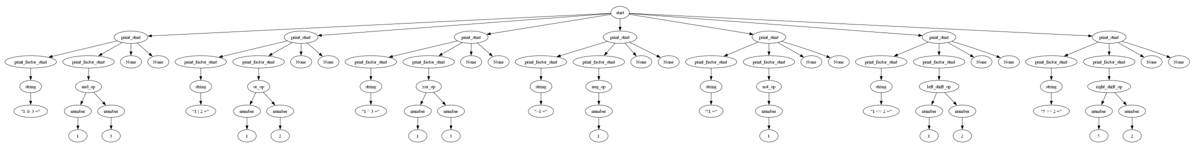
3.4 逻辑运算符

运算符	描述
&	与
|	或
^	异或
~	取反
!	非
>>	右移
<<	左移

```
print("1 & 3 =", 1 & 3)
print("1 | 2 =", 1 | 2)
print("1 ^ 3 =", 1 ^ 3)
print("~1 =", ~1)
print("!1 =", !1)
print("1 << 2 =", 1 << 2)
print("5 >> 2 =", 5 >> 2)
```



```
PS D:\Project\Project_Code\Python\Compiler>
1 & 3 = 1
1 | 2 = 3
1 ^ 3 = 2
~1 = -2
!1 = False
1 << 2 = 4
5 >> 2 = 1
PS D:\Project\Project_Code\Python\Compiler>
```



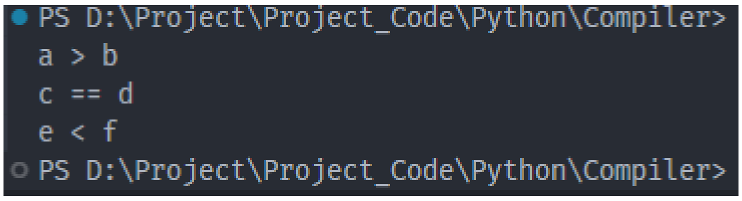
4 条件控制

条件控制语句格式为：

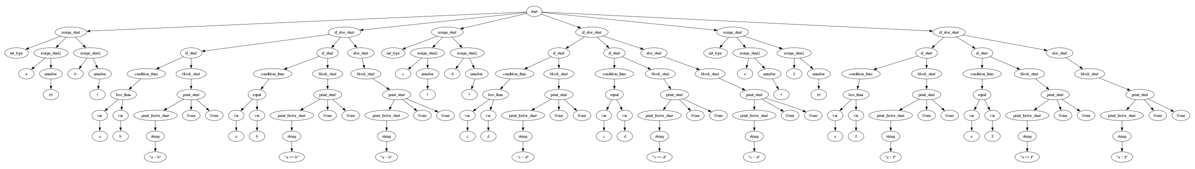
```
if (condition) {
  statement
}
[elif (condition) {
  statement
}]
[elif (condition) {
  statement
}]
[...]
[else {
  statement
}]
```

现有一例子如下：

```
int a = 10, b = 5
if(a < b) {
  print("a < b")
} elif(a == b) {
  print("a == b")
} else {
  print("a > b")
}
int c = 5, d = 5
if(c < d) {
  print("c < d")
} elif(c == d) {
  print("c == d")
} else {
  print("c > d")
}
int e = 5, f = 10
if(e < f) {
  print("e < f")
} elif(e == f) {
  print("e == f")
} else {
  print("e > f")
}
```



```
PS D:\Project\Project_Code\Python\Compiler>
a > b
c == d
e < f
PS D:\Project\Project_Code\Python\Compiler>
```



)

5 循环控制

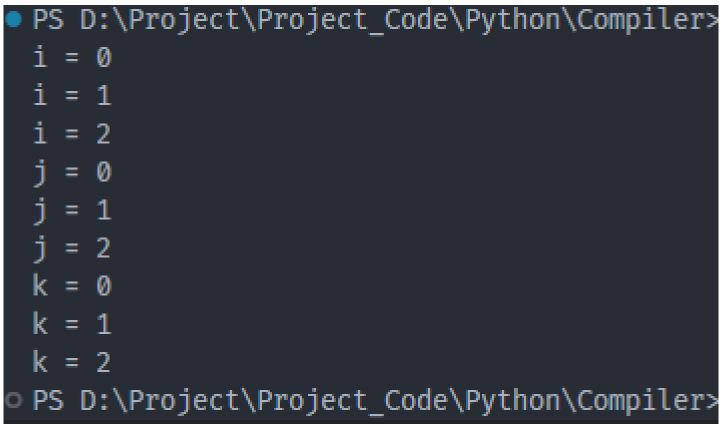
5.1 for 循环

for 循环格式为：

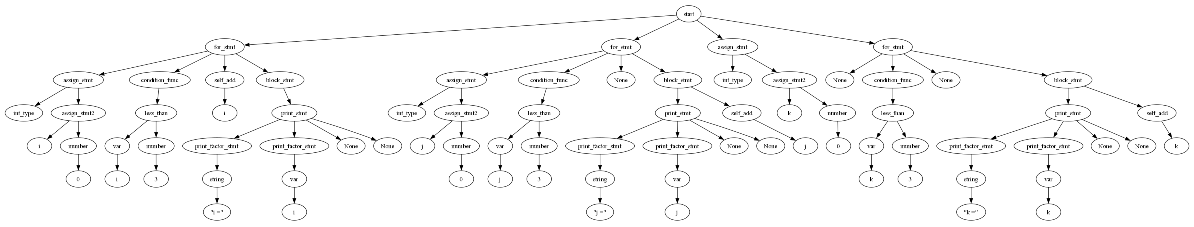
```
for ([statement]; [condition]; [statement]) {
  statement
}
```

现有一例子如下：

```
for (int i = 0; i < 3; i++) {
  print("i =", i)
}

for (int j = 0; j < 3;) {
  print("j =", j)
  j ++
}

int k = 0

for (; k < 3;) {
  print("k =", k)
  k ++
}
```



```
PS D:\Project\Project_Code\Python\Compiler>
i = 0
i = 1
i = 2
j = 0
j = 1
j = 2
k = 0
k = 1
k = 2
PS D:\Project\Project_Code\Python\Compiler>
```



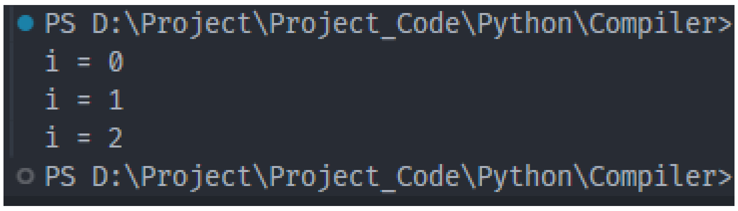
5.2 while 循环

while 循环格式为：

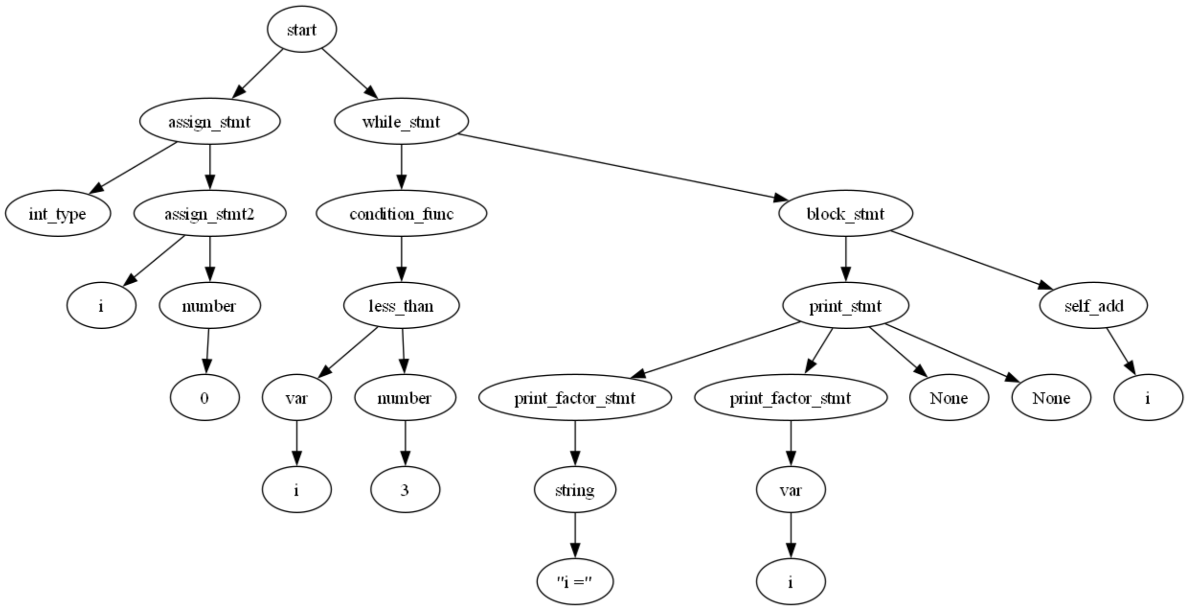
```
while (condition) {
  statement
}
```

现有一例子如下：

```
int i = 0
while (i < 3) {
  print("i =", i)
  i ++
}
```



```
PS D:\Project\Project_Code\Python\Compiler>
i = 0
i = 1
i = 2
PS D:\Project\Project_Code\Python\Compiler>
```



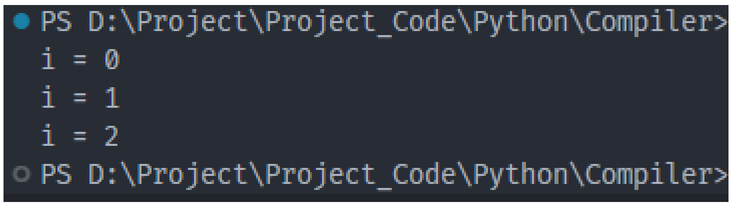
5.3 do-while 循环

do-while 循环格式为：

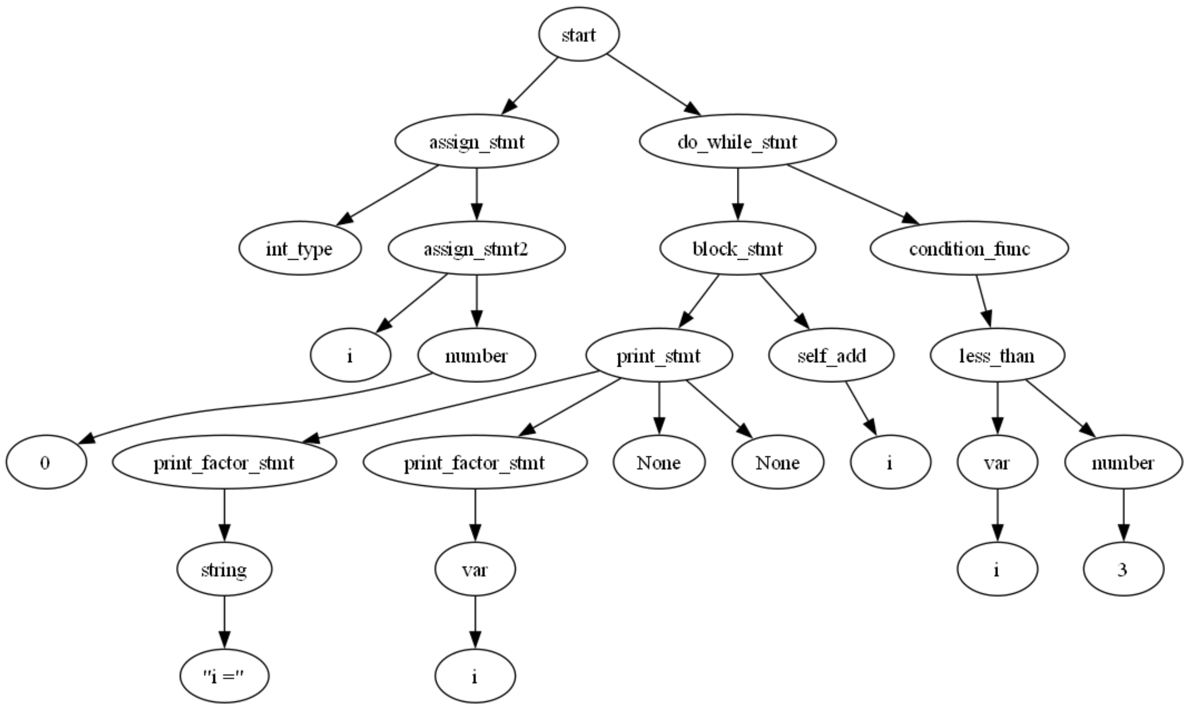
```
do {
  statement
} while (condition)
```

现有一例子如下：

```
int i = 0
do {
  print("i =", i)
  i ++
} while (i < 3)
```



```
PS D:\Project\Project_Code\Python\Compiler>
i = 0
i = 1
i = 2
PS D:\Project\Project_Code\Python\Compiler>
```



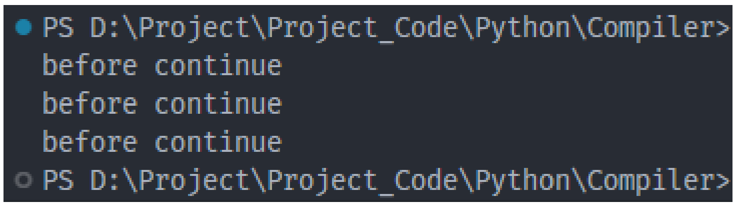
5.4 continue 语句

continue 语句格式为：

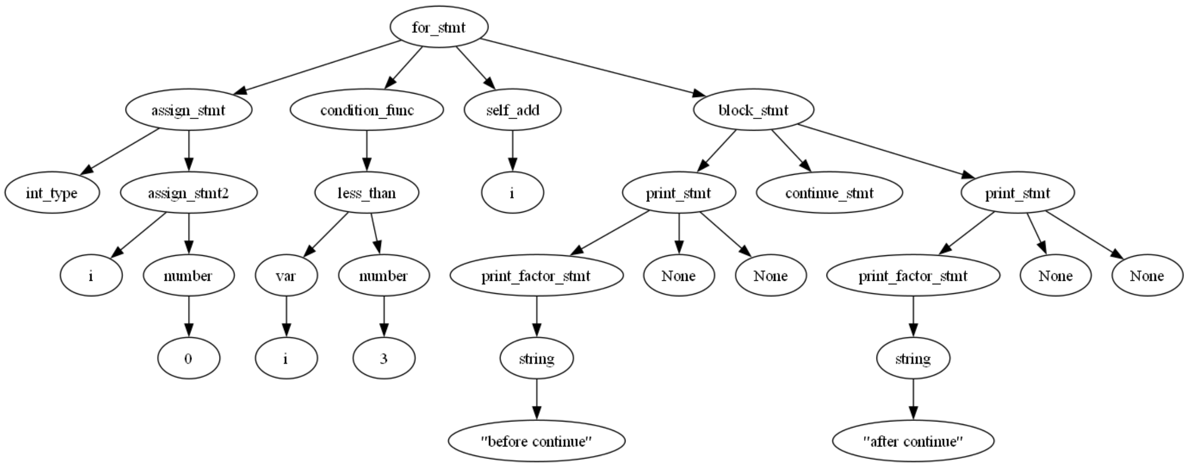
```
for(statement; condition; statement) {
  [statement]
  continue
  [statement]
}
```

现有一例子如下：

```
for(int i = 0; i < 3; i ++) {
  print("before continue")
  continue
  print("after continue")
}
```



```
PS D:\Project\Project_Code\Python\Compiler>
before continue
before continue
before continue
PS D:\Project\Project_Code\Python\Compiler>
```



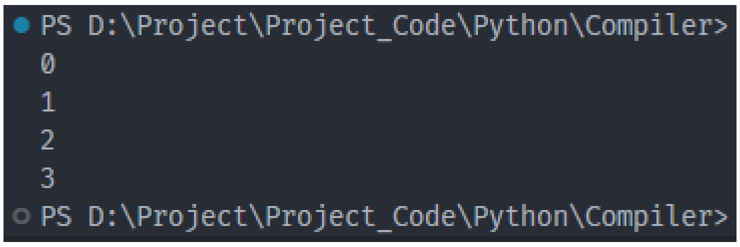
5.5 break 语句

break 语句格式为：

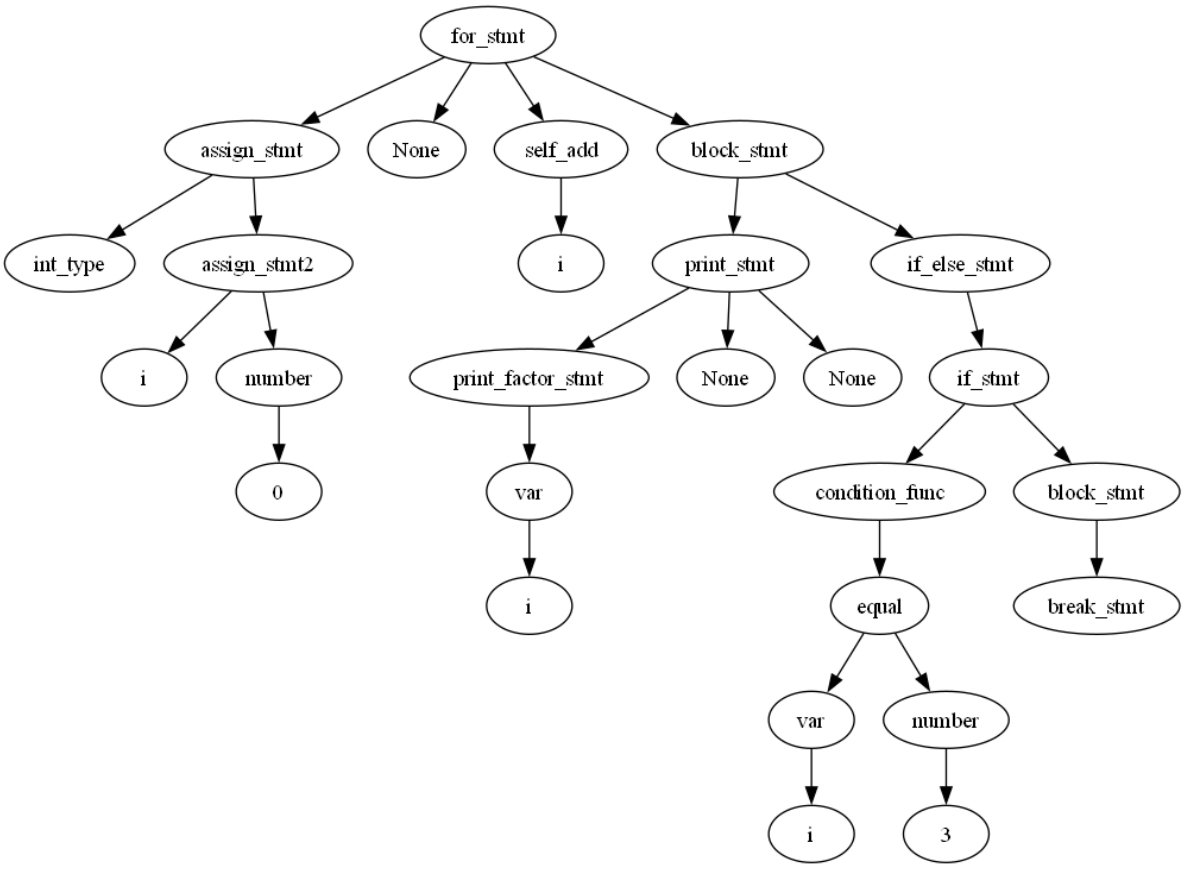
```
for(statement; condition; statement) {
  [statement]
  break
  [statement]
}
```

现有一例子如下：

```
for(int i = 0; ; i ++) {
  print(i)
  if(i == 3) {
    break
  }
}
```



```
PS D:\Project\Project_Code\Python\Compiler>
0
1
2
3
PS D:\Project\Project_Code\Python\Compiler>
```


for_stmt
assign_stmt
None
self_add
block_stmt
int_type
assign_stmt2
i
print_stmt
if_else_stmt
i
number
print_factor_stmt
None
None
if_stmt
0
var
condition_func
block_stmt
i
equal
break_stmt
var
number
i
3