Generating Code for Loops
while loops

while (<cond_expr>) {
    <stmt_list>
}

WhileNode
  CmpOp
  StmtListNode
while loops

while (<cond_expr>) {
    <stmt_list>
}

l_loop:
<cond_expr>
b<!op> l_out
<stmt_list>
j l_loop
l_out:
for loops

for (<init_stmt>; <cond_expr>; <update_stmt>) {
   <stmt_list>
}
for loops

for (<init_stmt>;
     <cond_expr>;
     <update_stmt>) {
     <stmt_list>
}

<init_stmt>
l_loop:
<cond_expr>
b<!op> l_out
<stmt_list>
l_incr:
<update_stmt>
j l_loop
l_out:
continue and break statements

- Continue statements: skip past rest of block, perform incr_stmt and restart loop
- Break statements: jump out of loop (do not execute incr_stmt)
- Caveats:
  - Code for stmt_list is generated earlier—where do we jump?
  - Keep track of “loop depth” as you descend through AST

```
<init_stmt>  
l_loop:      
<cond_expr>  
b<!op> l_out  
<stmt_list>  
l_incr:      
<update_stmt>  
j l_loop  
l_out:
```