SimpliPy: A notional machine for learning Python

International Insitute of Information Technology Hyderabad

Venkatesh Choppella, Moida Praneeth Jain, Gnaneswar Kulindala, Prabhav Shetty, Anushka Srikanth

Notional Machine for Python with Iteration

Syntax

- New statement: `while`
  - test expression
  - body
- `break` and `continue` instructions
- All while blocks must end in a `continue` instruction.

```
0  x = 5
1  while x < 10:
2      x = x + 1
3      continue
4
```

Control Transfer Functions

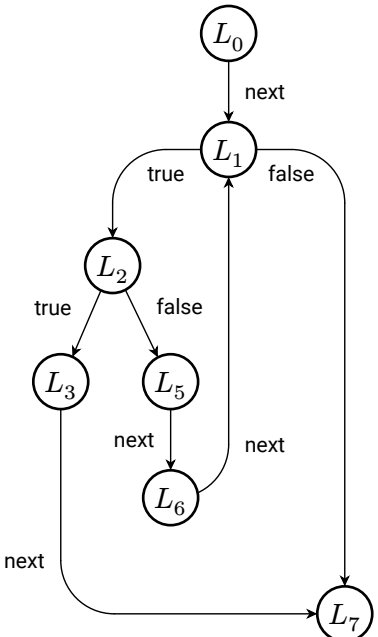
```
0  x = 5
1  while x < 10:
2      if x == 7:
3          break
4      else:
5          x = x + 1
6      continue
7
```

Loc	next	true	false	err
0	1	-	-	7
1	-	2	7	7
2	-	3	5	7
3	7	-	-	-
4	-	-	-	-
5	6	-	-	7
6	1	-	-	-
7	-	-	-	-

Notice that the `next` control transfer function for `continue` and `break` instructions are defined appropriately.

Control Flow Graph

```
0  x = 5
1  while x < 10:
2      if x == 7:
3          break
4      else:
5          x = x + 1
6      continue
7
```



State

No changes need to be made to the state to accommodate while loops.

$$\text{State} = \text{Loc} \times \text{Env}$$

while exp: transition

$$(i, e) \xrightarrow{\text{tick}} \begin{cases} (\text{true}(i), e) & \text{if res} = \text{true} \\ (\text{false}(i), e) & \text{if res} = \text{false} \\ (\text{err}(i), e) & \text{otherwise} \end{cases}$$

$$\text{if}$$

$$P_i ::= \text{while exp:}$$

$$\text{where}$$

$$\text{res} = \text{eval}(\text{exp}, e)$$

break and continue:

$$(i, e) \xrightarrow{\text{tick}} (\text{next}(i), e)$$

if

$$P_i := \text{break or continue}$$

Run of the Machine

```
0  x = 5
1  while x < 10:
2      if x == 7:
3          break
4      else:
5          x = x + 1
6      continue
7
```

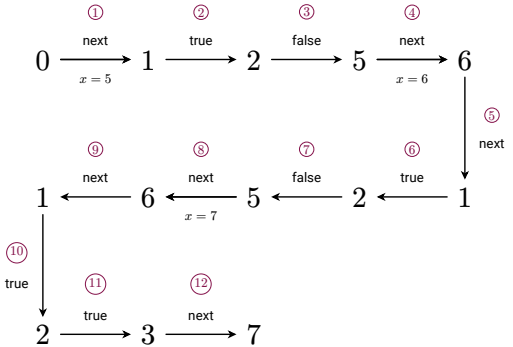
Execution Diagram

```
0  x = 5
1  while x < 10:
2      if x == 7:
3          break
4      else:
5          x = x + 1
6      continue
7
```

Execution Diagram

```
0  x = 5
1  while x < 10:
2      if x == 7:
3          break
4      else:
5          x = x + 1
6      continue
7
```

$$e = \begin{cases} \cancel{x \mapsto 5} & ① \\ \cancel{x \mapsto 6} & ④ \\ x \mapsto 7 & ⑧ \end{cases}$$



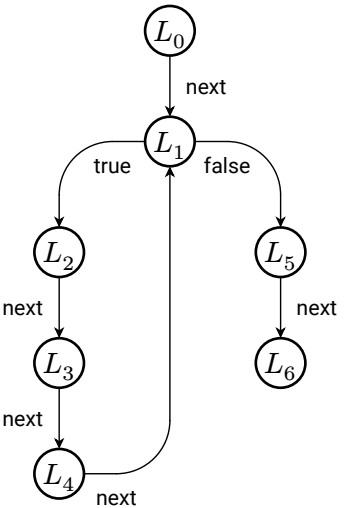
Example with error: Control Transfer Functions

```
0  x = 2
1  while x >= 0:
2      y = 10 // x
3      x = x - 1
4      continue
5  y = x * 2
6
```

Loc	next	true	false	err
0	1	-	-	6
1	-	2	5	6
2	3	-	-	6
3	4	-	-	6
4	1	-	-	-
5	6	-	-	6
6	-	-	-	-

Example with error: Control Flow Graph

```
0  x = 2
1  while x >= 0:
2      y = 10 // x
3      x = x - 1
4      continue
5  y = x * 2
6
```



Example with error: Run of the Machine

```
0  x = 2
1  while x >= 0:
2      y = 10 // x
3      x = x - 1
4      continue
5  y = x * 2
6
```

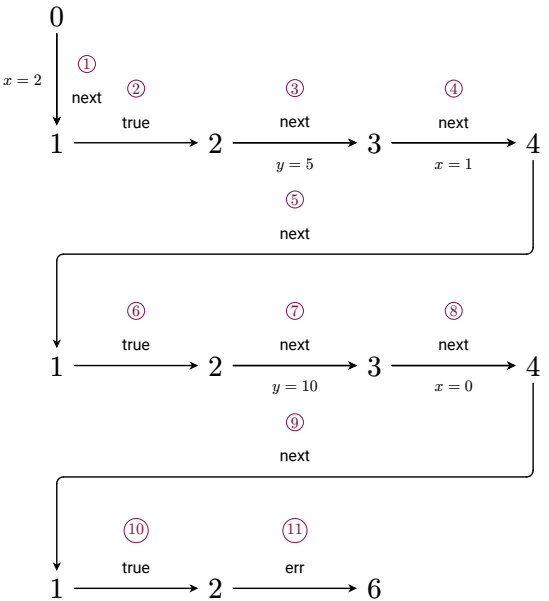
Example with error: Execution Diagram

```
0  x = 2
1  while x >= 0:
2      y = 10 // x
3      x = x - 1
4      continue
5  y = x * 2
6
```

Example with error: Execution Diagram

```
0  x = 2
1  while x >= 0:
2      y = 10 // x
3      x = x - 1
4      continue
5  y = x * 2
6
```

$$e = \begin{cases} \cancel{x \mapsto 2} & ① \\ \cancel{y \mapsto 5} & ③ \\ \cancel{x \mapsto 1} & ④ \\ y \mapsto 10 & ⑦ \\ x \mapsto 0 & ⑧ \end{cases}$$



Summary

- `while`, `break` and `continue` statements
- `true` and `false` control transfer functions for `while`
- `next` control transfer function for `break` and `continue`
- Transition for `while exp`
- Transitions for `break` and `continue`