A `break` statement with an argument appeared in a non-`loop` loop.

Example of erroneous code:

```
# let mut i = 1;
# fn satisfied(n: usize) -> bool { n % 23 == 0 }
let result = while true {
    if satisfied(i) {
        break 2 * i; // error: `break` with value from a `while` loop
    }
    i += 1;
};
```

The `break` statement can take an argument (which will be the value of the loop expression if the `break` statement is executed) in `loop` loops, but not `for`, `while`, or `while let` loops.

Make sure `break value;` statements only occur in `loop` loops:

```
# let mut i = 1;
# fn satisfied(n: usize) -> bool { n % 23 == 0 }
let result = loop { // This is now a "loop" loop.
    if satisfied(i) {
        break 2 * i; // ok!
    }
    i += 1;
};
```