The Effect of Nonlocal Statements

- **Nonlocal Assignment**
  - Assignment binds name to object in the first frame of the current environment in which that name is bound.

- **Local Assignment**
  - Assignment binds name to object in the current frame.

From the Python 3 language reference:
- Names listed as a nonlocal statement must refer to pre-existing bindings in an enclosing scope.
- Names listed as a nonlocal statement must not collide with pre-existing bindings in the local frame.

Python Particulars
- Python pre-compiles each frame contains each name before executing the body of a function within a function. All instances of a name must refer to the same frame.

<table>
<thead>
<tr>
<th>Status</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonlocal statement</td>
<td>Create a new binding from name &quot;x&quot; to object 2 in the current frame.</td>
</tr>
<tr>
<td>Local assignment</td>
<td>Bind the name &quot;x&quot; to object 2 in the first frame of the current environment.</td>
</tr>
</tbody>
</table>

The Many Meanings of Assignment Statements

- **Evaluation order**
  - Evaluate all expressions right of =, from left to right.
  - Bind the names on the left to the resulting values in the current environment in which that name is bound.

- **Names listed in a nonlocal statement**
  - The name is bound in a non-local frame.
  - The name is bound in a non-local frame.

- **Names listed in a local assignment**
  - The name is bound in a non-local frame.
  - The name is bound in a non-local frame.

Mutable Functions

- **Non-local frame**
  - Create a new binding from name "x" to object 2 in the current frame.

- **Local frame**
  - Bind the name "x" to object 2 in the first frame of the current environment.

Persistent Local State Using Environments

- **Local assignment**
  - Re-bind name "x" to object 2 in the first frame of the current environment.

- **Non-local assignment**
  - Create a new binding from name "x" to object 2 in the current frame.

A Function with Behavior That Varies Over Time

- **Return value**
  - Remaining balance

- **Amount to withdraw**
  - Current balance

- **Literal value**
  - 0

- **Other value**
  - 0

Python Docs: an Introduction to the Names and Environments in Python

- **Local assignment**
  - Re-bind name "x" to object 2 in the first frame of the current environment.

- **Non-local assignment**
  - Create a new binding from name "x" to object 2 in the current frame.

The Many Meanings of Assignment Statements

- **Status**
  - My current statement
  - "x" is bound locally

- **Effect**
  - "x" is bound in a non-local frame
  - There is no binding for name "x" in the current frame

Python pre-compiles each frame contains each name before executing the body of a function within a function. All instances of a name must refer to the same frame.

```python
def make_withdraw(balance):
    """Return a withdraw function with a starting balance."""
    def withdraw(amount):
        """Declare the name 'balance' nonlocal at the top of the frame in which it is re-assigned."""
        if amount > balance:
            return 'Insufficient funds'
        balance -= amount
        return balance
    return withdraw
make_withdraw(50)
```
Mutable Values & Persistent Local State

Mutable values can be changed without a nonlocal statement.

Name-value binding cannot change because there is no nonlocal statement.

Name bound outside of withdraw

Element assignment changes a list.

Multiple Mutable Functions

Environment Diagrams

Go Bears!

Referential Transparency, Lost

-Mutation operations violate the condition of referential transparency because they do more than just return a value; they change the environment.