In Python, `and`, `or`, and `not` are logical operators used for boolean operations. They have specific precedence levels which determine the order in which they are evaluated.

The precedence of these operators is as follows:

1. `not` (highest precedence)
2. `and`
3. `or` (lowest precedence)

Let's go through an example to illustrate how operator precedence works:

```
# Example
x = True
y = False
z = True

result = x and y or not z

print(result)
```

Explanation:

1. `not z` is evaluated first because `not` has the highest precedence. Since `z` is `True`, `not z` evaluates to `False`.

2. Next, `x and y` is evaluated. `and` has higher precedence than `or`, so this operation is performed first. `x` is `True` and `y` is `False`, so `x and y` evaluates to `False`.

3. Finally, the result of `False or False` (which is the result of `x and y`) and `False` (the result of `not z`) is evaluated. Since `or` has lower precedence, it is evaluated last. `False or False` evaluates to `False`.

The final result is `False`.

Remember, using parentheses can help make the order of evaluation more explicit and can override the default precedence:

result = (x and y) or (not z)

This ensures that `x and y` is evaluated together before the `or` operation.