a = True
b = False
c = True
\((\neg a) \land b \) or c

What is the value of the expression at the end of the above code? Recall that NOT has the highest precedence, then AND, then OR.

A. True
B. False

Answer: A (True)

We could rewrite this expression as `\((\neg a) \land b \) \lor c\)`, however the value of the left side of the OR is irrelevant because c == True, thus making the whole expression True.
a = 3
b = (a != 3)
print(b)

What does the code above print?

A. True
B. False
C. 3
D. Syntax error

Answer: B

b is assigned the result of a relational operator, so it must be a boolean. Here `a == 3`, so b is False.
a = 3
b = (a == 3)
print(b)

What does the code above print?

A. True
B. False
C. 3
D. Syntax error

Answer: A
b is assigned the result of a relational operator, so it must be a boolean. Here a == 3, so b is True.
I would like an expression that evaluates to True when at least one of the following two conditions is true:

1. a and b are equal, \( a == b \)
2. when a has the value 5. \( a == 5 \)

Which of these expressions does that?

A. \( a == b == 5 \)
B. \( (a == b) \) or \( (a == 5) \)
C. \( (a == b) \) and \( (a == 5) \)
D. \( a == (b == 5) \)

Answer: B

Based on “at least one of” we will need an OR operator, that is \( a == b \) or \( a == 5 \).
Recall that chaining introduces an AND, so option A is \( (a == b) \) and \( (b == 5) \), while option D will compare a to boolean produced by \( b == 5 \). This is valid but not good style.
We should avoid comparing equality of different types. It is OK, however, to compare equality of numeric types like integers and floats.
Answer: B

if-else statements can be nested. Here we first evaluate \( x < 15 \). That is True so we next evaluate \( x > 8 \). That is False so we print 'two'. Regardless of the value of \( x \) only one of 'one', 'two', or 'three' will be printed.
```python
if temperature > 0:
    print("above freezing")
elif temperature == 0:
    print("at freezing")
else:
    print("below freezing")
else:
    print("above freezing")
```

These two code snippets will print the same thing for all values of `temperature`:

A. True
B. False

Answer: A (True)

Although the second version uses <=, because of the ordering of if-else branches, the `elif` will only be evaluated if `temperature != 0`. Thus it is equivalent to `temperature < 0`. 