Solution 4

Module 4: Logical Operations

1.

Answer:

(F,F,F): notB=T notA=T A∧notB=F notA∧C=F result=False
(F,F,T): notB=T notA=T A∧notB=F notA∧C=T result=True
(F,T,F): notB=F notA=T A∧notB=F notA∧C=F result=False
(F,T,T): notB=F notA=T A∧notB=F notA∧C=T result=True
(T,F,F): notB=T notA=F A∧notB=T notA∧C=F result=True
(T,F,T): notB=T notA=F A∧notB=T notA∧C=F result=True
(T,T,F): notB=F notA=F A∧notB=F notA∧C=F result=False
(T,T,T): notB=F notA=F A∧notB=F notA∧C=F result=False

2.

Answer:

a) Bug: < 2 has no left-hand variable. Fix: if temp > -5 and temp < 2:
(or equivalently: if -5 < temp < 2: which Python allows)
b) Bug: De Morgan error — 'neither is true' requires not (A or B),
which equals not A AND not B. Fix: if not frost_risk and not heavy_rain:
c) Bug: = True is an assignment, not a comparison — SyntaxError.
Fix: if sensor_a or sensor_b or sensor_c:
(Truthy booleans don't need == True; just use the variable directly)

3.

Answer:

a) co2_ppm=1600, fan_on=False, temp=38.0, humidity=85.0, time_of_day='day', heating_on=False
b)
```
if co2_ppm > 1500 and not fan_on:
    fan_on = True
elif temp > 35.0 or humidity > 90.0:
    open_roof_vent()
elif time_of_day == 'night' and temp < 12.0:
    heating_on = True
else:
    print("All conditions normal")
```
c) co2_ppm=1600 > 1500 → True, fan_on=False → not False = True → True and True → first block runs.
fan_on = True. Remaining elif/else blocks are skipped.

4.

Answer:

a) Precedence: NOT first → AND next → OR last.
Evaluation: 1. not recently_fertilised 2. soil_n < 50 and (result of step 1) 3. ndvi < 0.4 or (result of step 2)
So the code reads as: ndvi < 0.4 OR (soil_n < 50 AND not recently_fertilised)

b) No — the code does NOT match the intention. The intended logic is:
(ndvi < 0.4 OR soil_n < 50) AND not recently_fertilised
Corrected code: apply = (ndvi < 0.4 or soil_n < 50) and not recently_fertilised

c) Parentheses make the programmer's intent explicit and unambiguous.
They protect code from subtle bugs caused by operator precedence assumptions.
Code is read by humans as well as machines — clarity reduces mistakes.
Even experienced programmers add parentheses in complex conditions as a safety habit.