Making Choices

1.

a. `True`
a. `NameError`
a. `True`
a. `True`
a. `True`

2.

a. `x and y`
b. `not x`
c. `x or y`

3.

```
(full or empty) and not (full and empty)
```

4. Yes. If you try substituting all possible Boolean values for `(light < 0.01)`, which we'll abbreviate as L, and `(temperature > 0.0)` as T, you'll find the following Boolean expressions are logically equivalent:

L	T	L != T	L or T and not (L and T)
True	**True**	False	False
True	**False**	True	True
False	**True**	True	True
False	**False**	False	False

5.

```
result = abs(x) == x
```

6.

```
def different(a, b):
    """ (number, number) -> bool

    Return True if a and b refer to the different values, or False,
otherwise.

    >>> different(0, 1)
    True
```

```
    >>> different(1, 1)
    False
    """
    return a != b
```

7.

a.

```
if population < 10000000:
    print(population)
```

b.

```
if population >= 10000000 and population <= 35000000:
    print(population)
```

Note: the solution above is inclusive, because it includes 10000000 and 35000000. The solution below is exclusive:

```
if population > 10000000 and population < 35000000:
    print(population)
```

c.

```
if (population / land_area) > 100:
    print('Densely populated')
```

d.

```
if (population / land_area) > 100:
    print('Densely populated')
else:
    print('Sparsely populated')
```

8.

```
def convert_temperatures(t, source, target):
    """ (number, str, str) -> number

    Convert t from source temperature scale to target temperature scale
    and return the result.

    >>> convert_temperatures(0, 'Celsius', 'Kelvin')
    273.15
    >>> convert_temperatures(100, 'Celsius', 'Fahrenheit')
    212.0
    """
    if source == 'Kelvin':
        celsius = t - 273.15
    elif source == 'Celsius':
        celsius = t
    elif source == 'Fahrenheit':
        celsius = (t - 32) * 5 / 9
    elif source == 'Rankine':
        celsius = (t - 491.67) * 5 / 9
    elif source == 'Delisle':
        celsius = 100 - t * 2 / 3
    elif source == 'Newton':
        celsius = t * 100 / 33
    elif source == 'Reamur':
        celsius = t * 5 / 4
    elif source == 'Romer':
```

```
        celsius = (t - 7.5) * 40 / 21
    if target == 'Kelvin':
        return celsius + 273.15
    elif target == 'Celsius':
        return celsius
    elif target == 'Fahrenheit':
        return celsius * 9 / 5 + 32
    elif target == 'Rankine':
        return (celsius + 273.15) * 9 / 5
    elif target == 'Delisle':
        return (100 - celsius) * 3 / 2
    elif target == 'Newton':
        return celsius * 33 / 100
    elif target == 'Reamur':
        return celsius * 4 / 5
    elif target == 'Romer':
        return celsius * 21 / 40 + 7.5
```

9.

The problem is the order of the two conditions. For any values less than 3.0, the first condition will be true and the body of the elif block will never be executed. To fix this, we reorder the conditions:

```
if ph < 3.0:
    print(ph, "is VERY acidic! Be careful.")
elif ph < 7.0:
    print(ph, "is acidic")
```

10. a. `It's acidic!` b. `It's acidic!` c.

```
ph = float(input("Enter the ph level: "))
if ph < 7.0:
    print("It's acidic!")
if ph < 4.0:
    print("It's a strong acid!")
```

11.

The reason why the code checks whether someone is light is because that is a better mental match with the table: the first cell is for young, light people, and so it is natural to have a Boolean variable to match that.

Here is the new code with the alternative version:

```
young = age < 45
heavy = bmi >= 22.0

if young and not heavy:
   risk = 'low'
elif young and heavy:
   risk = 'medium'
```

```
elif not young and not heavy:
   risk = 'medium'
elif not young and heavy:
   risk = 'high'
```