Exercise 5

With all you've learned, you can start writing much more interesting programs. See if you can solve the problems below.

As always, run the setup code below before working on the questions.

In [ ]:
```
# Run this cell to setup the exercise
import sys
from pathlib import Path
learntools_dir = Path().absolute().parents[1]
sys.path.append(str(learntools_dir))
from learntools.core import binder; binder.bind(globals())
from learntools.python.ex5 import *
print('Setup complete.')
```

1.

Have you ever felt debugging involved a bit of luck? The following program has a bug. Try to identify the bug and fix it.

In [ ]:
```
def has_lucky_number(nums):
    """Return whether the given list of numbers is lucky. A lucky list conta
    at least one number divisible by 7.
    """
    for num in nums:
        if num % 7 == 0:
            return True
        else:
            return False
```

Try to identify the bug and fix it in the cell below:

In [ ]:
```
def has_lucky_number(nums):
    """Return whether the given list of numbers is lucky. A lucky list conta
    at least one number divisible by 7.
    """
    for num in nums:
        if num % 7 == 0:
            return True
        else:
            return False

# Check your answer
q1.check()
```

In [ ]:
```
# q1.hint()
# q1.solution()
```

2.

Look at the Python expression below. What do you think we'll get when we run it? When you've made your prediction, uncomment the code and run the cell to see if you were right.

In [ ]:
```
# [1, 2, 3, 4] > 2
```

R and Python have some libraries (like numpy and pandas) compare each element of the list to 2 (i.e. do an 'element-wise' comparison) and give us a list of booleans like `[False, False, True, True]`.

Implement a function that reproduces this behaviour, returning a list of booleans corresponding to whether the corresponding element is greater than n.

In [ ]:
```
def elementwise_greater_than(L, thresh):
    """Return a list with the same length as L, where the value at index i i
    True if L[i] is greater than thresh, and False otherwise.

    >>> elementwise_greater_than([1, 2, 3, 4], 2)
    [False, False, True, True]
    """
    pass

# Check your answer
q2.check()
```

In [ ]:
```
# q2.solution()
```

3.

Complete the body of the function below according to its docstring.

In [ ]:
```
def menu_is_boring(meals):
    """Given a list of meals served over some period of time, return True if
    same meal has ever been served two days in a row, and False otherwise.
    """
    pass

# Check your answer
q3.check()
```

In [ ]:
```
# q3.hint()
# q3.solution()
```

4. 🌶️

Next to the Blackjack table, the Python Challenge Casino has a slot machine. You can get a result from the slot machine by calling `play_slot_machine()`. The number it returns is your winnings in dollars. Usually it returns 0. But sometimes you'll get lucky and get a big payday. Try running it below:

In [ ]:
```
play_slot_machine()
```

By the way, did we mention that each play costs $1? Don't worry, we'll send you the bill later.

On average, how much money can you expect to gain (or lose) every time you play the machine? The casino keeps it a secret, but you can estimate the average value of each pull using a technique called the **Monte Carlo method**. To estimate the average outcome, we simulate the scenario many times, and return the average result.

Complete the following function to calculate the average value per play of the slot machine.

In [ ]:
```
def estimate_average_slot_payout(n_runs):
    """Run the slot machine n_runs times and return the average net profit p
    Example calls (note that return value is nondeterministic!):
    >>> estimate_average_slot_payout(1)
    -1
    >>> estimate_average_slot_payout(1)
    0.5
    """
    pass
```

When you think you know the expected value per spin, run the code cell below to view the solution and get credit for answering the question.

In [ ]:
```
# Check your answer (Run this code cell to receive credit!)
q4.solution()
```

Keep Going

Many programmers report that dictionaries are their favorite data structure. You'll get to **learn about them** (as well as strings) in the next lesson.
