UNIT TESTING, AGAIN

TOGETHER, QUICKLY...

```
def first_warm_day(temps: list[int]) -> int:
    """Return the index of the first warm day (>60 F),
    or return -1 if no such day exists.
    Inputs: temps, a list of temperatures
    Output: index of the first warm day
    """
    ...
```

If `list` is `[45, 61, 90, 34]`, we return...

If `list` is `[45, 54, 10, 34]`, we return...

If `list` is `[]`, we return...

If `list` is `[62, 63, 64, 65]`, we return...

TOGETHER, QUICKLY...

```
def first_warm_day(temps: list[int]) -> int:
    """Return the index of the first warm day (>60 F),
    or return -1 if no such day exists.
    Inputs: temps, a list of temperatures
    Output: index of the first warm day
    """
    ...
```

If `list` is `[45, 61, 90, 34]`, we return **1**

If `list` is `[45, 54, 10, 34]`, we return **-1**

If `list` is `[]`, we return **-1**

If `list` is `[62, 63, 64, 65]`, we return **0**

These give us test cases!

BUILDING THE FIRST TEST

If `list` is `[45, 61, 90, 34]`, we return **1**

- **Input:** `[45, 61, 90, 34]`
- **Expected:** 1
- **Actual:** Call `first_warm_day()`
- **Assertion:** Compare expected to actual

```
import unittest      # <--- unittest library
import warm_days     # <--- file where our implementation is
# put all test case functions in here ⬇️
class TestWarmDays(unittest.TestCase):
```

```
import unittest
import warm_days
class TestWarmDays(unittest.TestCase):
    # write a function signature def name(self):
    def test_first_warm_day_second_w_multiple(self):
```

Use a descriptive name for your test case so that you can see **what** failed.

```
import unittest
import warm_days
class TestWarmDays(unittest.TestCase):
    input_temps = [45, 61, 90, 34]
```

Pick your input(s)

```
import unittest
import warm_days
class TestWarmDays(unittest.TestCase):
    input_temps = [45, 61, 90, 34]
    expected_day = 1
```

Figure out (by hand!) what the expected return value *should be* for `warm_days.first_warm_day(input_temps)`.

```
import unittest
import warm_days
class TestWarmDays(unittest.TestCase):
    input_temps = [45, 61, 90, 34]
    expected_day = 1
    actual_day = warm_days.first_warm_day(input_temps)
```

Figure out (by hand!) what *is actually returned by* the call to `warm_days.first_warm_day(input_temps)`.

```
import unittest
import warm_days
class TestWarmDays(unittest.TestCase):
    input_temps = [45, 61, 90, 34]
    expected_day = 1
    actual_day = warm_days.first_warm_day(input_temps)
    self.assertEqual(expected_day, actual_day)
```

Write an assertion statement that compares what you expect to see with what you actually see.

BUGS ARE HARD TO SPOT!

```python
def first_warm_day(temps: list[int]) -> int:
    """Return the index of the first warm day (>60 F),
    or return -1 if no such day exists.
    Inputs: temps, a list of temperatures
    Output: index of the first warm day
    """
    for temp in temps:
        if temp > 60:
            return temp
    return -1
```

With a partner: come up with a test case (input/expected output pair) that would **fail** based on this implementation.

BUGS ARE HARD TO SPOT!

```
def first_warm_day(temps: list[int]) -> int:
    """Return the index of the first warm day (>60 F),
    or return -1 if no such day exists.
    Inputs: temps, a list of temperatures
    Output: index of the first warm day
    """
    for temp in temps:
        if temp > 60:
            return temp
    return -1
```

With a partner: come up with a test case (input/expected output pair) that would **pass** based on this implementation.

Sometimes tests pass, sometimes they fail. You're done when they **all** pass, and your confidence about correctness is higher when you have more tests.

`warm_days.py` and `test_warm_days.py` demo

FLATTEN TIME ON ED!