Problem 15: Hidden Test Demo

Unlike the other problems in this suite, this problem is not a real test problem, but rather a demonstration of how "hidden test cells" will work on an assignment of exam.

This demo consists of a single exercise. It is followed by two test cells: one that you can see and use for debugging, and a "hidden" one, which you never get to see. (The autograder runs it and will give you some information back on whether the test case passed or not, but otherwise, you get no information.)

Exercise 0 (10 points: 5 points "exposed" and 5 points hidden). Let n be an integer value, and let $F(n)$ denote the n-th value of the Fibonacci sequence (or "n-th Fibonacci number"). The basic recursive definition is that $F(n) = F(n-1) + F(n-2)$. For example,

- `eval_fib(1) == 1`,
- `eval_fib(2) == 1`,
- `eval_fib(3) == eval_fib(2) + eval_fib(1) == 1 + 1 == 2`,
- `eval_fib(4) == eval_fib(3) + eval_fib(2) == 2 + 1 == 3`,
- `eval_fib(5) == eval_fib(4) + eval_fib(3) == 3 + 2 == 5`,
- `eval_fib(6) == eval_fib(5) + eval_fib(4) == 5 + 3 == 8`,
- and so on.

Here is a more precise and complete mathematical definition:

$$F(n) = \begin{cases} 0 & n \leq 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n \geq 2 \end{cases}.$$

Complete the function below, `eval_fib(n)`, so that it computes the `n`-th Fibonacci number. For instance, `eval_fib(1) == 1`, `eval_fib(2) == 1`

In [ ]:
```
def eval_fib(n):
    if n <= 1:
        return n
    else:
        return(eval_fib(n-1) + eval_fib(n-2))

# Demo:
for n in range(1, 7):
    print(f"eval_fib({n}) == {eval_fib(n)}")
```

```
eval_fib(1) == 1
eval_fib(2) == 1
eval_fib(3) == 2
eval_fib(4) == 3
eval_fib(5) == 5
eval_fib(6) == 8
```

In [ ]:
```python
# Test: `exposed_test`

# Per: https://en.wikipedia.org/wiki/Fibonacci_number#Identification
def is_fib(x):
    def is_perfsq(x):
        from math import sqrt
        r = int(sqrt(x))
        return x == r*r
    option_0 = 5*(x**2) + 4
    option_1 = option_0 - 8
    return is_perfsq(option_0) or is_perfsq(option_1)

assert eval_fib(1) == 1, f"eval_fib(1) == 1, not {eval_fib(1)}."
assert eval_fib(2) == 1, f"eval_fib(2) == 1, not {eval_fib(2)}."
fib_values = [1, 1]
for n in range(3, 10):
    print(f"Checking n={n}...")
    F_n = eval_fib(n)
    print(f"==> You computed {F_n}.")
    assert F_n, f"Error: {F_n} is not even a Fibonacci number!"
    if n >= 2:
        assert F_n == fib_values[-1] + fib_values[-2], f"Error: {F_n} is a Fibonacc
    fib_values.append(F_n)

print("\n(Passed!)")
```

```
Checking n=3...
==> You computed 2.
Checking n=4...
==> You computed 3.
Checking n=5...
==> You computed 5.
Checking n=6...
==> You computed 8.
Checking n=7...
==> You computed 13.
Checking n=8...
==> You computed 21.
Checking n=9...
==> You computed 34.

(Passed!)
```

In [ ]:
```python
# Test: `hidden_test`

print("""
This test cell will be replaced with one or more hidden tests.
You will only know the result after submitting to the autograder.
If the autograder times out, then either your solution is highly
```

```
inefficient or contains a bug (e.g., an infinite loop).
""")

###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

```
This test cell will be replaced with one or more hidden tests.
You will only know the result after submitting to the autograder.
If the autograder times out, then either your solution is highly
inefficient or contains a bug (e.g., an infinite loop).
```

Fin! This cell marks the end of the notebook. Remember, there is a test cell with a hidden test, so the only way to know if you get full credit is to submit your notebook to the autograder.