Fibonnaci Sequence

Problem Statement

Implement a Fibonnaci Sequence in three different ways:

- Recursively
- Dynamically (Using Memoization to store results)
- Iteratively

Remember that a fibonacci sequence: 0,1,1,2,3,5,8,13,21,... starts off with a base case checking to see if n = 0 or 1, then it returns 1.

Else it returns fib(n-1)+fib(n+2).

Recursively

The recursive solution is exponential time Big-O , with O(2^n). However, its a very simple and basic implementation to consider:

In [1]:
```python
def fib_rec(n):

    # Base Case
    if n == 0 or n == 1:
        return n

    # Recursion
    else:
        return fib_rec(n-1) + fib_rec(n-2)
```

In [2]:
```python
fib_rec(10)
```

Out[2]:
```
55
```

Dynamically

In the form it is implemented here, the cache is set beforehand and is based on the desired **n** number of the Fibonacci Sequence. Note how we check it the cache[n] != None, meaning we have a check to know wether or not to keep setting the cache (and more importantly keep cache of old results!)

In [3]:
```python
# Instantiate Cache information
n = 10
cache = [None] * (n + 1)


def fib_dyn(n):

    # Base Case
    if n == 0 or n == 1:
        return n
    # Check cache
    if cache[n] != None:
        return cache[n]

    # Keep setting cache
    cache[n] = fib_dyn(n-1) + fib_dyn(n-2)

    return cache[n]
```

In [4]:
```
fib_dyn(10)
```

Out[4]:
```
55
```

Iteratively

In this solution we can take advantage of Python's tuple unpacking!

In [5]:
```
def fib_iter(n):

    # Set starting point
    a = 0
    b = 1

    # Follow algorithm
    for i in range(n):

        a, b = b, a + b

    return a
```

In [6]:
```
fib_iter(23)
```

Out[6]:
```
28657
```

Test Your Solution

Run the cell below to test your solutions, simply uncomment the solution functions you wish to test!

In [7]:
```
"""
UNCOMMENT THE CODE AT THE BOTTOM OF THIS CELL TO SELECT WHICH SOLUTIONS TO TEST.
THEN RUN THE CELL.
"""

from nose.tools import assert_equal

class TestFib(object):

    def test(self,solution):
        assert_equal(solution(10),55)
        assert_equal(solution(1),1)
        assert_equal(solution(23),28657)
        print('Passed all tests.')
# UNCOMMENT FOR CORRESPONDING FUNCTION
t = TestFib()

t.test(fib_rec)
```

```
#t.test(fib_dyn) # Note, will need to reset cache size for each test!
#t.test(fib_iter)
```

```
Passed all tests.
```

Conclusion

Hopefully this interview question served as a good excercise in exploring recursion, dynamic programming, and iterative solutions for a single problem! Its good to work through all three because in an interview a common question may just begin with requesting a recursive solution and then checking to se if you can implement the other forms!