Project Euler Problems

Syed Paymaan Raza

December 1, 2017

Contents

Trying out some project euler problems here. Mostly I do this to learn a language. It's been a while since I coded in Python so I thought to implement a few project euler problem solutions. I have done a few before so hopefully this should be relatively quick.

Used Python cheat sheets to revise concepts:

1. Matthes Python Cheat Sheets
2. Matthes Python Cheat Sheets - ALL
3. Limsi Python Cheat Sheet
4. Dave Child Python Cheat Sheet

Remember, in Python, whitespace matters!

1 Multiples of 3 and 5

```
def is_multiple_of(num, divisor):
    return num % divisor == 0

def p1():
    sum = 0
    for i in range(0, 1000):
        if is_multiple_of(i, 3) or is_multiple_of(i, 5):
            sum += i
    return sum

print(p1())
```

```
233168
```

Time complexity: O(n) where n = 1000; so really O(1) unless we want 1000 to be variable. Space complexity: O(1)

Another approach using list comprehensions:

```
def is_multiple_of(num, divisor):
    return num % divisor == 0

def p1_list_comprehensions():
    return sum(x for x in range(1000) if is_multiple_of(x, 3) or is_multiple_of(x, 5))

print(p1_list_comprehensions())
```

```
233168
```

2 Even Fibonacci Numbers

First, let's see simple fibonacci:

```
""" Exp time (solve recurrence), O(n) space due to stack frames """
def fib_recursive(n):
    if n == 0:
        return 1
    if n == 1:
        return 2
    return fib_recursive(n - 1) + fib_recursive(n - 2)

""" O(n) time, O(1) space """
def fib_iterative(n):
    fib_n_minus_two = 1
    fib_n_minus_one = 2
    if n == 0:
        return fib_n_minus_two
    if n == 1:
        return fib_n_minus_one
    fib_n = -1
    for i in range(2, n + 1):
        fib_n = fib_n_minus_one + fib_n_minus_two
        fib_n_minus_two = fib_n_minus_one
        fib_n_minus_one = fib_n
    return fib_n

print(fib_recursive(4))
print(fib_iterative(4))
```

```
8
8
```

Note we take $\text{fib} = 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$ Also, indexing is zero based i.e. $\text{fib}(0) = 1$, $\text{fib}(1) = 1$, $\text{fib}(2) = 3$ and so on.

Now, let's modify iterative solution to solve this problem:

```
def is_even(num):
    return num % 2 == 0

def p2(upper_bound):
    fib_n_minus_two = 1
    fib_n_minus_one = 2
    fib_n = -1
    sum = 2
    while (fib_n < upper_bound):
        fib_n = fib_n_minus_one + fib_n_minus_two
        fib_n_minus_two = fib_n_minus_one
        fib_n_minus_one = fib_n
        if (is_even(fib_n)):
            sum += fib_n
    return sum

print(p2(4 * (10 ** 6)))
```

```
4613732
```

Time is $\mathrm{O}(\text{upper}_{\text{bound}})$ but $\mathrm{O}(1)$ if $\text{upper}_{\text{bound}}$ is constant. Space is $\mathrm{O}(1)$.