My Project Euler Solutions

Doug Beney

Contents

1 Problem #1

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

```
sum = 0

for i in range(1000):
    if i % 3 == 0 or i % 5 == 0:
        sum = sum + i

return sum
```

2 Problem #2

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

```
def fibify(index):
    if index > 2:
        return fibify(index-1) + fibify(index-2)
    return index

def fiblist(max):
    index = 2
    fib = 1
    nums = []
    while fib <= max:
        nums.append(fib)
        fib = fibify(index)
        index += 1
    return nums

return sum([n for n in fiblist(4000000)
            if n % 2 == 0])
```

3 Project #3

The prime factors of 13195 are 5, 7, 13 and 29.

What is the largest prime factor of the number 600851475143 ?

3.1 First I had to refresh myself on what a prime number is.

It is a number thats only factors are one and itself.

Example prime numbers: 2, 3, 5, 7, 11

So, let's create a prime number finder.

```
for i in range(1, 100):
    count = 0
```

```
    for n in range(1, i):
        if i % n == 0:
            count += 1
    if count == 1:
        print(i, "is prime")
```

3.2 Solving the problem

```
import math
import time

## Typical, slow method to test if number is prime
def is_prime_slow(number):
      if number <= 1:
          return False

      count = 0

      for n in range(2, i):
          if i % n == 0:
              count += 1

      if count == 0:
          return True

## Faster way to find if number is prime
def is_prime_fast(n):
    if n == 1:
        return False

    i = 2

    while i*i <= n:
        if n % i == 0:
            return False
        i += 1

    return True

def largest_prime_factors(number):
```

```
    largest = 0
    i = 2
    while i*i < number:
        if i > largest and is_prime_fast(i) and number % i == 0:
            largest = i
        i += 1
    return largest

print('The largest prime factor is',
      largest_prime_factors(600851475143))
```

4 Project #4

A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is $9009 = 91 \times 99$.

Find the largest palindrome made from the product of two 3-digit numbers.

```
import sys

def palindrome_p(num):
    num = str(num)
    num_r = ""
    for c in num:
        num_r = c + num_r
    return num == num_r

def find_largest_palindrone(max_num):
    num1 = max_num
    num2 = max_num
    largest_palindrone=0
    largest_palindrone_num1 = num1
    largest_palindrone_num2 = num2

    while True:
        product = num1 * num2
        if product > largest_palindrone and palindrome_p(product):
            largest_palindrone = product
            largest_palindrone_num1 = num1
            largest_palindrone_num2 = num2
```

```
        if num2 > 1:
            num2 -= 1
        else:
            num1 -= 1
            num2 = 999
            if num1 < 1:
                break

    print("The largest palindrone is:",
          largest_palindrone,
          "\nUsing", largest_palindrone_num1, "and", largest_palindrone_num2)

find_largest_palindrone(999)
```