Python 101

Lec04

Intermission

thoum

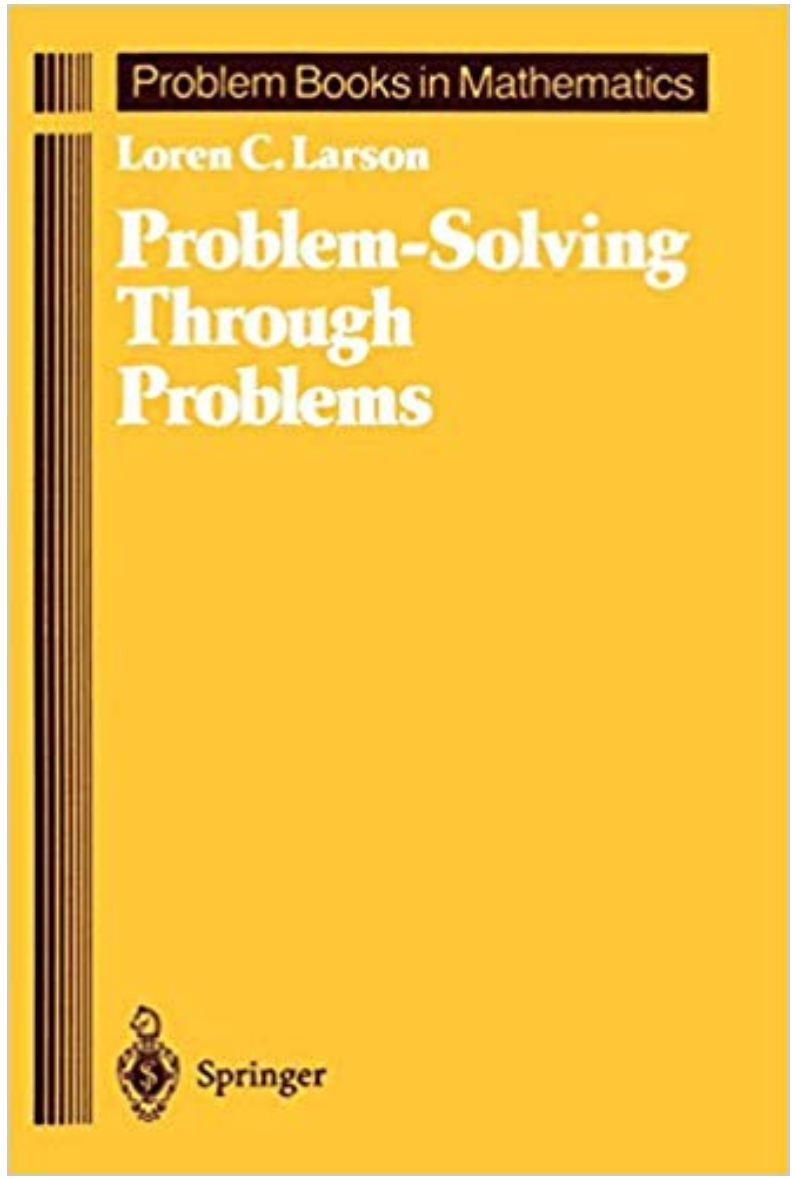
May 8, 2019

What have we learned so far?

Turing Complete

We *can* do everything$_{\text{a computer can}}$ with what we have learned so far.
But can *you*?

PSTP



https://www.amazon.com

Mini-Midterm

Open Book. Open Internet. Open Google. Open Stackoverflow.

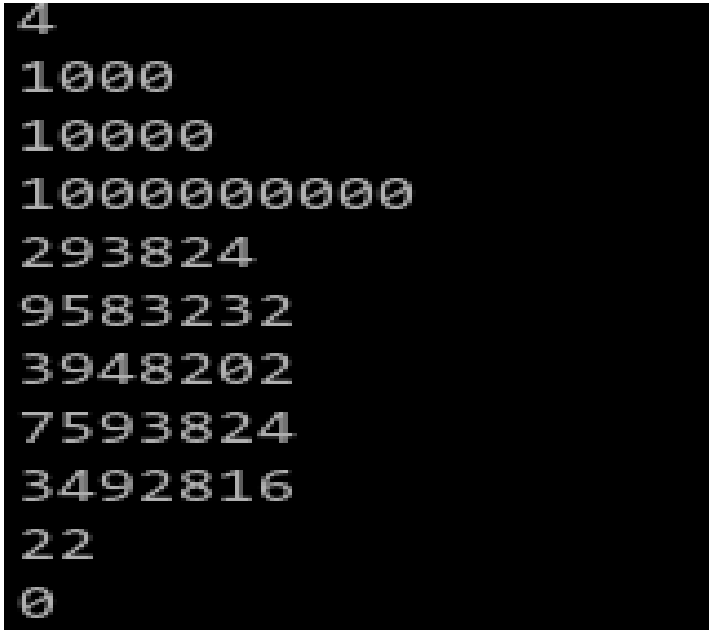
With an incentive

Those who finish upto Q4 until the end of the tutoring will get a **FREE** cup of coffee at Starbucks®.

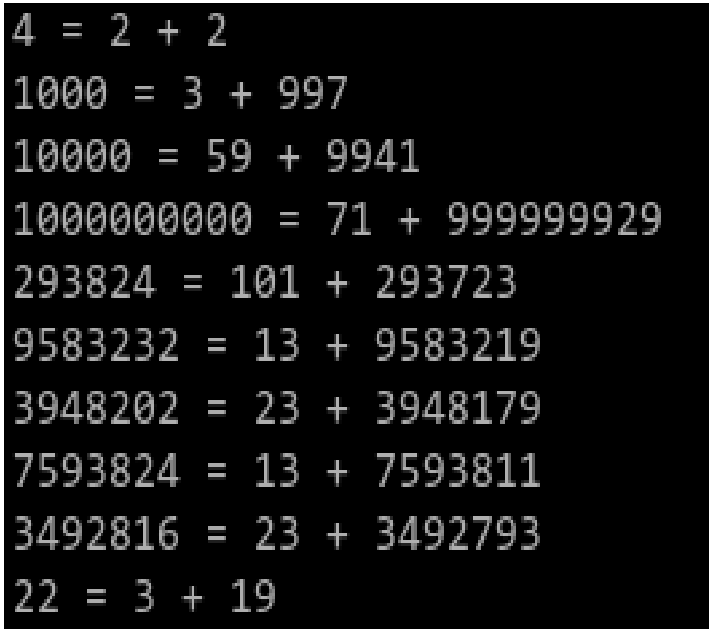
Q1. Goldbach Conjecture

Every even integer greater than 2 is the sum of two prime numbers.
Let's prove it upto 10,000,000,000.

Goldbach Conjecture



input



output

Step 0

Save as gbach.py

Step 1

1. The numbers come in seperate lines
2. Do we have to store each numbers?
3. We should terminate at 0.

Step 1

Can we get the correct input?

1. The numbers come in seperate lines
2. We should terminate at 0
3. Do we have to store each numbers?

Step 2

Now what?

1. Split the given number into two numbers
2. See if the two numbers are *primes*

Step 2.1

Checking if a number is a prime.
Ideas?

Step 2.5

Checking if a number is a prime.

1. Generate the sieve of eratosthenes and check membership?
2. Check N's divisors.

Step 2.5

Checking if a number is a prime.

1. The sieve: (X). We have to store 455,052,511 numbers.
2. Divisors: This seems better memory-wise.

Step 2.6

We need to make it into a function as is_prime is used repeatedly.

```
def is_prime(n):
    #Fill me in
```

Step 2.7

Anatomy of is_prime(n).
For k: 2 ... n-1 [1]
if n is divided by k:
not a prime number
if no divisor:
is a prime number

[1] There is a better value, which is?

Step 2.9

Check that is_prime works well. It is often a good idea to test your program with random numbers, big numbers, and small numbers.

1. is_prime(0)
2. is_prime(1)
3. is_prime(2)
4. is_prime(23334)
5. is_prime(1217)
6. is_prime(343313)
7. is_prime(10000000000)

Step 3.0

Splitting the numbers into two. (1, k-1) / (2, k-2)/ (3, k-3)...

Step 3.3

Now check if the split numbers are both primes
is_prime(k) and is_prime(n-k)

Step 4.0

Time to pretty print it. How? Look it up

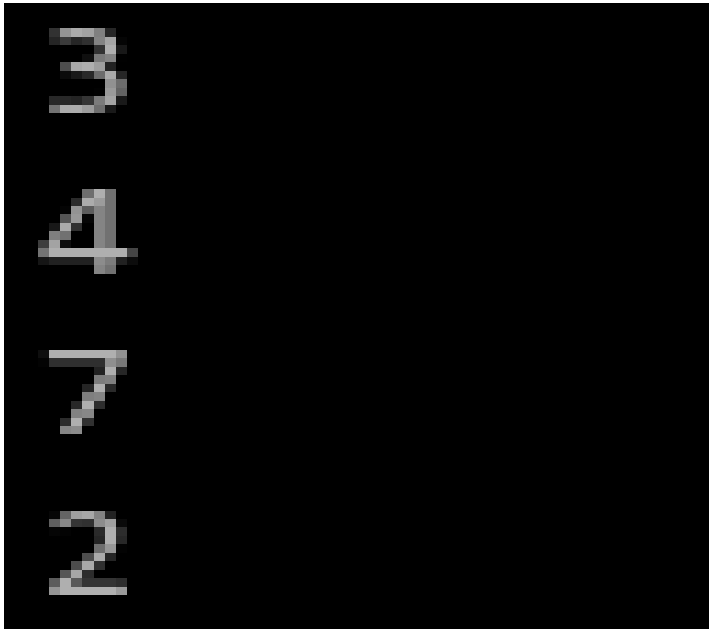
Q2. Partitioning

Calculate the number of cases of partitioning number N with 1,2,3.
4 =

1. 1+1+1+1
2. 1+1+2
3. 1+2+1
4. 2+1+1
5. 2+2
6. 1+3
7. 3+1

Partitioning

First Number: N repeat N times.



input

output

Step 0

Save as partition.py

Step 1

Any ideas?
Try writing some cases out.

Step 1.5

Can you find any relationships?

Step 1.9

Let's assume f(n) somehow magically calculates the answer for you.
Then What?

Step 2.0

Write out the relationship in your code.

Step 3.0

Test it for large numbers.... what should we do?

Challenge

We can change the recursive version with lists.
Assume lst[n] gives the answer for you.
Then what?

Q3. Dumb Multiplication

123*999 = 91827
999*999 = 818181

Step 0

I will be providing less guides.

Q4. Look and say sequence

Print the N^{th} look and say sequence, a.k.a ant sequence.
1, 11, 12, 1121, 122111....

Do it!