CS 3430: SciComp with Py
Assignment 1
Prime Factorization of Euclid Numbers

Vladimir Kulyukin
Department of Computer Science
Utah State University

January 14, 2017

1 Learning Objectives

1. Py Functions
2. Lists, Tuples, Strings, Loops
3. Euclid Numbers
4. Prime Factorization
5. Getting to know IDLE2 and IDLE3 on Raspberry Pi 3 Model B.

2 Introduction

In this assignment, you will write Py2 and Py3 programs to factorize the first 10 Euclid numbers. Euclid numbers feature prominently in cryptography and computability theory. Euclid of Alexandria, a Greek mathematician (323 - 283 B.C.E), constructed them to show that there are infinitely many prime numbers. Over 2,000 years after Euclid, Kurt Gödel, another brilliant mathematician, used Euclid numbers to construct Gödel numbers which laid the theoretical foundation of all modern compilers, interpreters, and operating systems.

3 Prime Numbers

We start our investigation of Euclid numbers by implementing the boolean function $is\_prime(n)$ that takes a number and returns $True$ or $False$, depending on whether n is a prime or not. Recall that a number is a prime if it is greater than or equal to 2 and does not have divisors other than 1 and itself. For exmaple, 2, 3, 5, 7, 11, 13, 17 are primes. A few test runs in Py 2.7 IDLE:

```
>>> is_prime(2)
True
>>> is_prime(3)
True
>>> is_prime(4)
False
>>> for n in xrange(20):
        if is_prime(n):
           print n, 'is prime'

2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
```

Now on to implementing the function $next\_prime\_after(p)$, where p is a prime. This function returns the smallest prime number $p' > p$ or $None$ if p is not a prime. A few test runs in Py 2.7 IDLE.

```
>>> next_prime_after(2)
3
>>> next_prime_after(19)
23
>>> next_prime_after(next_prime_after(19))
29
```

4 Euclid Numbers

A formal definition of the i^{th} Euclid number is as follows:

$$E_i = \prod_{0}^{i} p_i + 1. \tag{1}$$

In equation (1), p_i is the i^{th} prime number, i.e., $p_0 = 2, p_1 = 3, p_2 = 5, p_3 = 7, etc.$. Let us compute the first four Euclid numbers by hand.

- $E_0 = 2 + 1 = 3;$
- $E_1 = 2 \times 3 + 1 = 7;$
- $E_2 = 2 \times 3 \times 5 + 1 = 31;$
- $E_3 = 2 \times 3 \times 5 \times 7 + 1 = 211.$

Now implement the function $euclid\_number(i)$ that computes the i^{th} Euclid number. Below are a few test runs in Py 2.7 IDLE.

```
>>> euclid_number(0)
3
>>> euclid_number(1)
7
>>> euclid_number(2)
31
>>> euclid_number(3)
211
```

After you have implemented $euclid\_number(i)$, implement the function $compute\_first\_n\_eucs(n)$ that returns a list of the first n Euclid numbers. A few test runs in Py 2.7 IDLE are below.

```
>>> for euc in compute_first_n_eucs(10):
        print euc

3
7
31
211
2311
```

```
30031
510511
9699691
223092871
6469693231
>>> for euc in compute_first_n_eucs(20):
        print euc

3
7
31
211
2311
30031
510511
9699691
223092871
6469693231
200560490131
7420738134811
304250263527211
13082761331670031
614889782588491411
32589158477190044731
1922760350154212639071
117288381359406970983271
7858321551080267055879091
557940830126698960967415391
```

5 Factorizing Euclid Numbers

In number theory, the fundamental theory of arithmetic states that any natural number $n > 1$ either is a prime or is a product of primes. Put a different way, the theorem states that any natural number $n > 1$ is a product of prime factors. In his famous book "Elements," Euclid showed that this factorization is unique, i.e., no matter how one finds the prime factors, they will always be the same, regardless of how they are arranged in the final product. For example, no matter how one finds the prime factors of 15, one will always end up with 3 and 5 and no other prime factor. Because of this, the fundamental theorem of arithmetic is also known as the unique factorization theorem.

Implement the function $prime\_factors\_of(n)$ that returns a list of prime factors of a number $n > 1$ and [], i.e., an empty list if $n \leq 1$. A few test runs in Py 2.7 IDLE:

```
>>> prime_factors_of(8)
[2, 2, 2]
>>> prime_factors_of(15)
[3, 5]
>>> prime_factors_of(1200)
[2, 2, 2, 2, 3, 5, 5]
>>> 2*2*2*2*3*5*5
1200
```

Implement the function $tabulate\_euc\_factors(n)$ that computes the prime factorizations of the first n Euclid numbers. This function should return a list of 3-element tuples $(i, euc, factorization)$, where i is the euclid number's number, euc is the Euclid number itself returned by $euclid\_number(i)$, and $factorization$ is the list of prime factors returned by $prime\_factors\_of(euc)$. Here is a test run in Py 2.7 IDLE.

```
>>> for fac in tabulate_euc_factors(6):
        print fac

(0, 3, [3])
(1, 7, [7])
(2, 31, [31])
(3, 211, [211])
(4, 2311, [2311])
(5, 30031, [59, 509])
(6, 510511, [19, 97, 277])
```

6 What To Submit

Implement the above functions in Py2 and save them in the file *euclid_numbers.py*. Implement the above functions in Py3 and save them in the file *euclid_numbers_py3.py*. To save you some typing, I have attached the files *euclid_numbers.py* and *euclid_numbers_py3.py* with all the function stubs already defined. You can use those files to implement your solutions.

Please do not change the signatures and names of the functions in the Py files. It will make the graders' job much easier.

I recommend that you code up your functions in Py2 and then work on Py3 port. Remember the differences between *xrange*() (Py2) and *range*() (Py3) and between the *print* statement (Py2) and the *print* function (Py3) discussed in class.

You can use your favorite Py IDE on your desktop or laptop to develop your code. However, run your code on your Raspberry Pi before submitting it. We will test your code on our Rasberry Pi's.

Submit your *euclid_numbers.py* and *euclid_numbers_py*3*.py* via Canvas.

7 Bonus 1/2 Point Question

What is $prime\_factors\_of(euclid\_number(10))$?

Happy Hacking!