

**SUMMER 2024**

**APT3090 CRYPTOGRAPHY AND NETWORK SECURITY**

**Lab 1: Key Generation Algorithm Prime numbers**

**Name: Joy Kirui**

**AdmNo:664830**

**Programming language: Python**

import random

class PrimeKeyGenerator:

def \_\_init\_\_(self, start, end):

self.start = start

self.end = end

def is\_prime(self, n):

"""Check if a number is prime."""

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

i = 5

while (i \* i) <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

def generate\_prime(self):

"""Generate a random prime number within the given range."""

while True:

num = random.randint(self.start, self.end)

if self.is\_prime(num):

return num

def generate\_keys(self):

"""Generate two prime numbers, p and q."""

p = self.generate\_prime()

q = self.generate\_prime()

return p, q

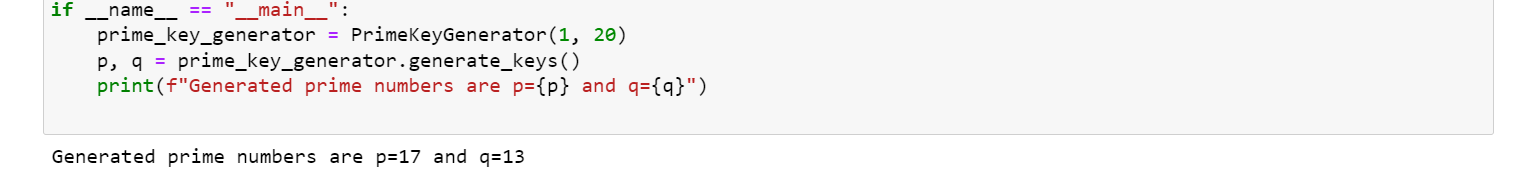
if \_\_name\_\_ == "\_\_main\_\_":

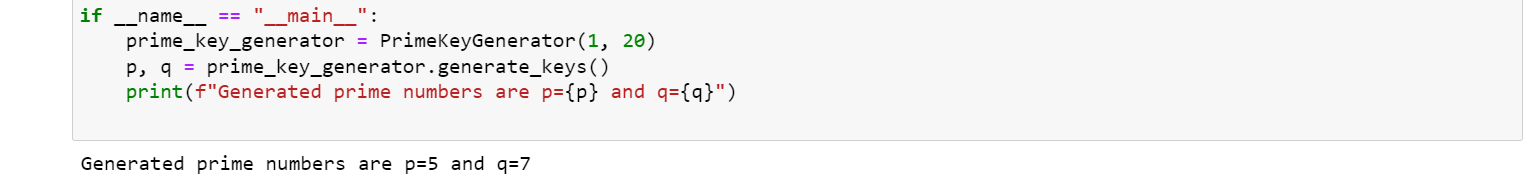
prime\_key\_generator = PrimeKeyGenerator(1, 20)

p, q = prime\_key\_generator.generate\_keys()

print(f"Generated prime numbers are p={p} and q={q}")

**Results**



Two prime numbers are randomly generated. 

**Explanation of the code**

**Instantiating the Object:**

prime\_key\_generator = PrimeKeyGenerator(1, 20) creates an instance of PrimeKeyGenerator with the range 1 to 20. This means the code will look for prime numbers between 1 and 20.

**Generation of the First Prime Number:**

The p = prime\_key\_generator.generate\_prime() is called.

Inside generate\_prime, a random number is generated within the range. Let's say it first generates 15.

is\_prime(15) returns False (since 15 is divisible by 3,5).

Another random number as in this case 17 is generated.

is\_prime(17) returns True (since 17 is actually a prime number).

p is set to 17.

**Generation of the Second Prime Number:**

The q = prime\_key\_generator.generate\_prime() is called.

Inside the generate\_prime, a random number is generated within the range. Let's say it first generates 8.

is\_prime(12) returns False (since 12 is divisible by 2).

Another random number as in this case 13 generated.

is\_prime(13) returns True (since 13 is a prime number).

q is set to 13.

**Lab 2: Euler’s Totient function**

**Euler's theorem** states that, “if p and q are relatively prime, then”, where φ is **Euler's** totient function for integers. That is, is the number of non-negative numbers that are less than q and relatively prime to q.Euler’s Totient function Φ(n) for an input n is count of numbers in {1, 2, 3, …, n} that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1.

import random

import math

class PrimeKeyGenerator:

def \_\_init\_\_(self, start, end):

self.start = start

self.end = end

def is\_prime(self, n):

"""Check if a number is prime."""

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

i = 5

while (i \* i) <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

def generate\_prime(self):

"""Generate a random prime number within the given range."""

while True:

num = random.randint(self.start, self.end)

if self.is\_prime(num):

return num

def euler\_totient(self, n):

"""Calculate Euler's Totient function φ(n)."""

result = n

p = 2

while p \* p <= n:

# Check if p is a divisor of n

if n % p == 0:

# If yes, then update n and result

while n % p == 0:

n //= p

result -= result // p

p += 1

# If n has a prime factor greater than sqrt(n)

if n > 1:

result -= result // n

return result

def generate\_key\_and\_totient(self):

"""Generate a prime number p and its Euler's Totient function φ(p)."""

p = self.generate\_prime()

phi\_p = self.euler\_totient(p)

return p, phi\_p

if \_\_name\_\_ == "\_\_main\_\_":

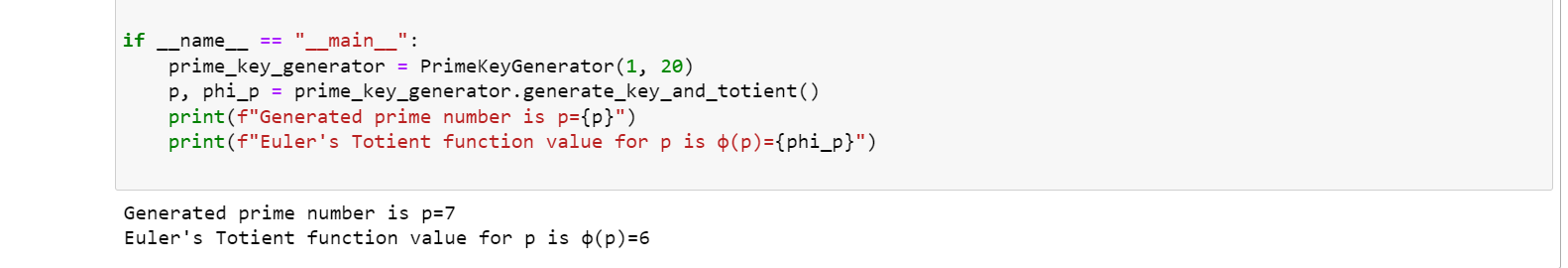
prime\_key\_generator = PrimeKeyGenerator(1, 20)

p, phi\_p = prime\_key\_generator.generate\_key\_and\_totient()

print(f"Generated prime number is p={p}")

print(f"Euler's Totient function value for p is φ(p)={phi\_p}")

**Result**



A number is randomly generated and its Euler’s Totient is calculated.

**Explanation of the code**

**Instantiating the Object:**

prime\_key\_generator = PrimeKeyGenerator(1, 20) initiates the object of the PrimeKeyGenerator class which has the range 1 to 20. This means that the code will be looking for prime numbers just between 1 and 20.

**Generation of the Prime Number p:**

p = prime\_key\_generator.generate\_prime() is called.

Inside generate\_prime, a random number is generated within the range. Let's say it first generates 10.

is\_prime(10) returns False (since 10 is divisible by 2).

Another random number is generated, say 7.

is\_prime(7) returns True (since 7 is a prime number).

p is set to 7.

**Calculating Euler's Totient Function φ(p):**

phi\_p = prime\_key\_generator.euler\_totient(p) is called with p being 7.

Inside euler\_totient, it starts with result = 7.

Iterates over potential prime factors starting from 2. Since 7 is prime, the only prime factor is 7 itself.

Updates result as result -= result // 7 which results in 6.