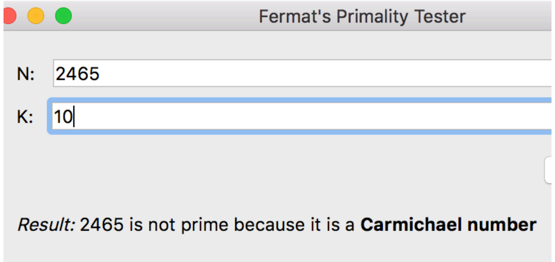
Emily Morrow
Section 2

Tests showing instances of Camichael, prime, and composite

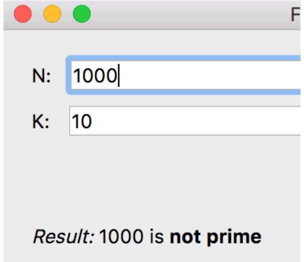


Fermat's Primality Tester

N: 2465

K: 10

Result: 2465 is not prime because it is a **Carmichael number**

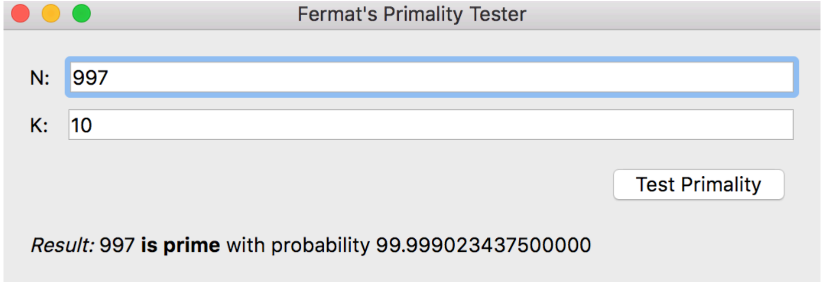


F

N: 1000

K: 10

Result: 1000 is **not prime**



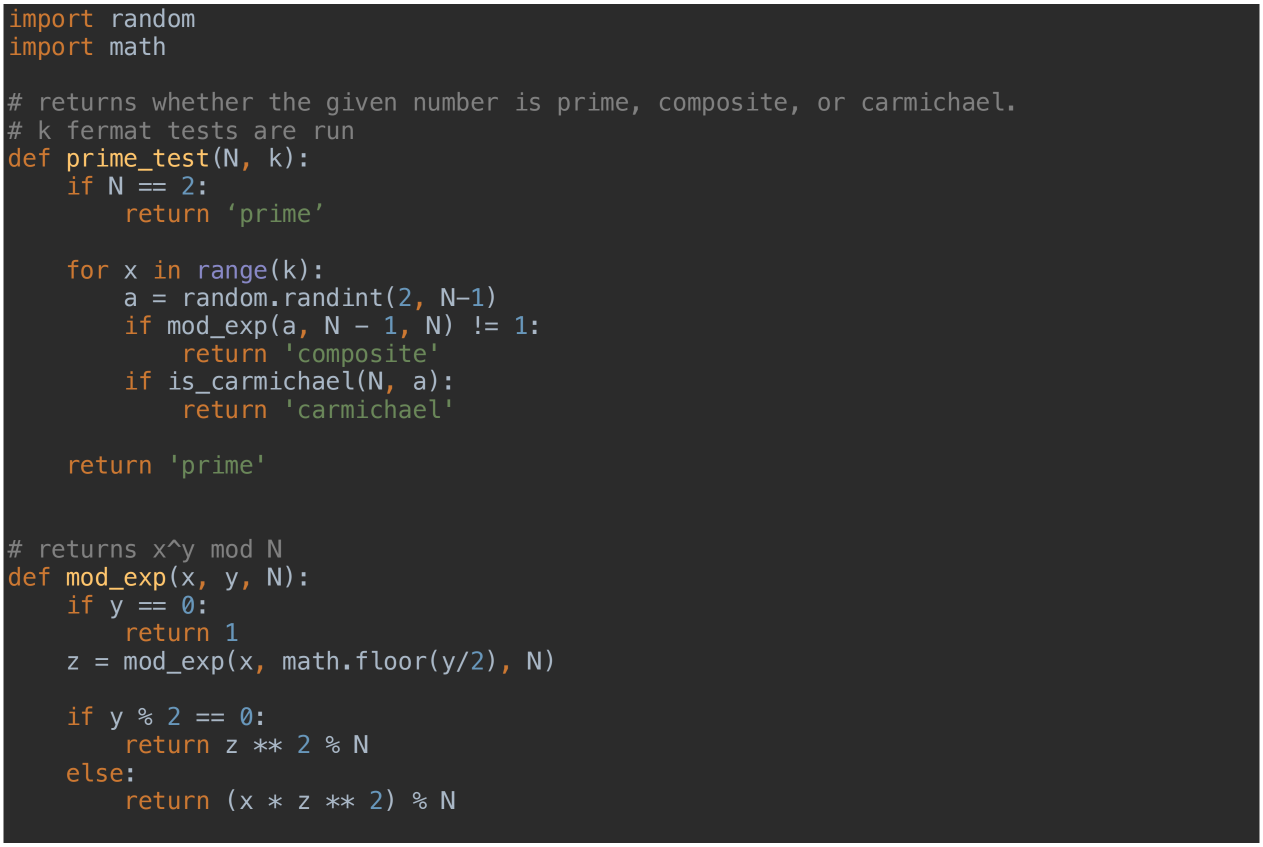
Fermat's Primality Tester

N: 997

K: 10

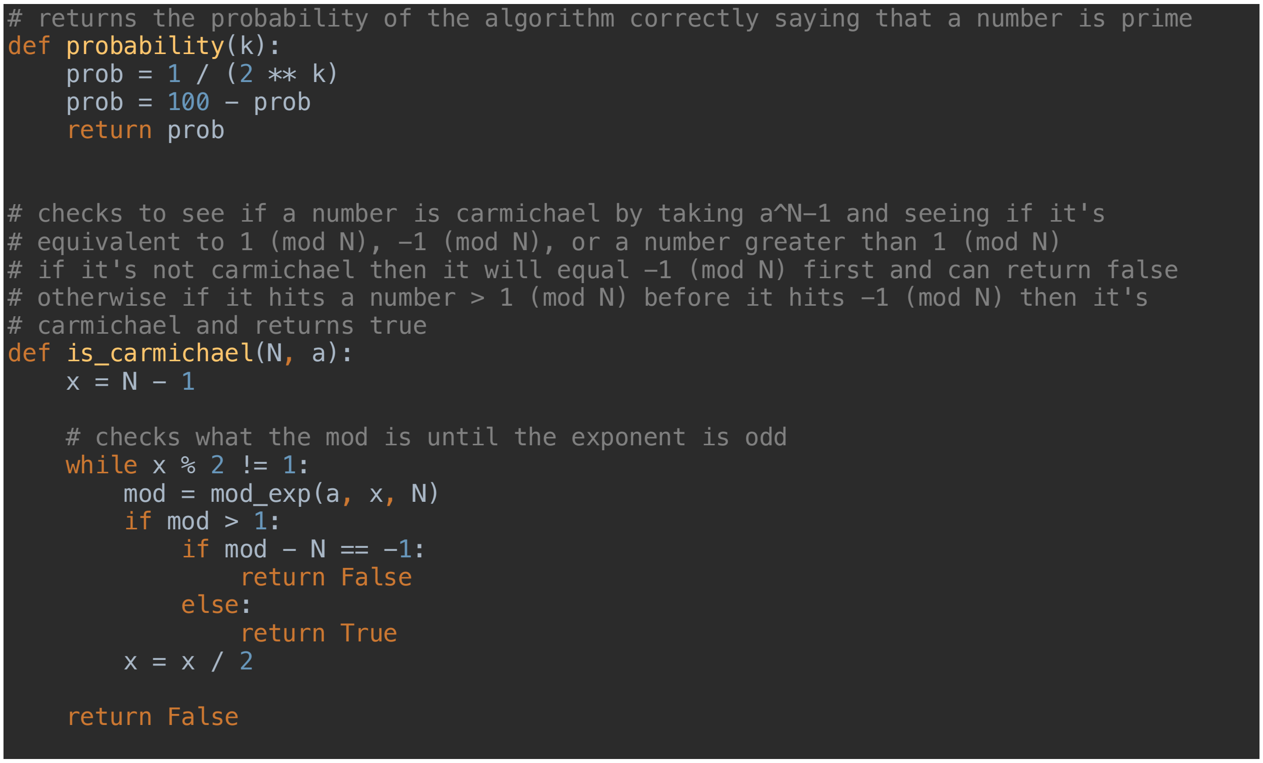
Result: 997 **is prime** with probability 99.999023437500000

Fermat.py code



```
import random
import math

# returns whether the given number is prime, composite, or carmichael.
# k fermat tests are run
def prime_test(N, k):
    if N == 2:
        return ‘prime’

    for x in range(k):
        a = random.randint(2, N-1)
        if mod_exp(a, N - 1, N) != 1:
            return 'composite'
        if is_carmichael(N, a):
            return 'carmichael'

    return 'prime'


# returns x^y mod N
def mod_exp(x, y, N):
    if y == 0:
        return 1
    z = mod_exp(x, math.floor(y/2), N)

    if y % 2 == 0:
        return z ** 2 % N
    else:
        return (x * z ** 2) % N
```

Emily Morrow
Section 2



```python
# returns the probability of the algorithm correctly saying that a number is prime
def probability(k):
    prob = 1 / (2 ** k)
    prob = 100 - prob
    return prob


# checks to see if a number is carmichael by taking a^N-1 and seeing if it's
# equivalent to 1 (mod N), -1 (mod N), or a number greater than 1 (mod N)
# if it's not carmichael then it will equal -1 (mod N) first and can return false
# otherwise if it hits a number > 1 (mod N) before it hits -1 (mod N) then it's
# carmichael and returns true
def is_carmichael(N, a):
    x = N - 1

    # checks what the mod is until the exponent is odd
    while x % 2 != 1:
        mod = mod_exp(a, x, N)
        if mod > 1:
            if mod - N == -1:
                return False
            else:
                return True
        x = x / 2

    return False
```

Space and Time Complexity

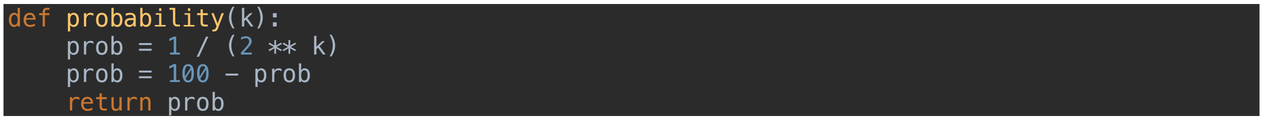
The mod_exp function is a recursive function that runs in $O(n^3)$ time assuming n is the largest of x, y and N.

The is_carmichael function has a while loop that runs in n/2 time but one of the steps inside is mod_exp so the runtime would be $\frac{n}{2} * O(n^3)$ or $O(\frac{n^4}{2})$ or $O(n^4)$ since 2 is a constant

The time complexity of the prime test function without accounting for the time complexity of mod_exp and is_carmichael is O(n) because it has a single for-loop. When we take into account the two functions being called inside the for-loop we would get about n*$\frac{n^4}{2} + n * n^3$ or O($\frac{n^5}{2}$) or $O(n^5)$ since 2 is a constant.

The space complexity is O(n)

Probability Equation



```python
def probability(k):
    prob = 1 / (2 ** k)
    prob = 100 - prob
    return prob
```

Emily Morrow
Section 2

To find the probablility that the output of the algorithm is correct I found the probability of error which is $\frac{1}{2^k}$ and subtracted that from 100 to find the probability of saying the number is prime and being right.