1 Problem Statement

By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13. What is the 10001st prime number?

2 Code

```
# Project Euler #7 | Sidharth Baskaran | 01/08/2022

import time
import math

def is_prime(n):
    for x in range(math.ceil(math.sqrt(n)),1,-1):
        if n % x == 0:
            return False
    return True

def solve(count=10001):
    curr = 0
    p = 0
    idx = 0
    while curr <= count:
        p = 2*idx + 1
        if is_prime(p):
            curr += 1
        idx += 1
    print(p)


if __name__ == "__main__":
    s = time.time()
    solve()
    e = time.time()

    print('%.3fms' % ((e-s)*1000))
```

3 Explanation

We include a method `is_prime` to determine whether or not a number n is prime. It simply checks whether all numbers below $\sqrt{n}$ excluding 1 divide n. We only have to check from $\sqrt{n}$ and below because at least one factor of a nonprime number is below its square root.

The method `solve` finds the nth prime, with a default of $n = 10001$. A prime variable, p, is set to 2 and a counter variable is set to 1 (since 2 is prime). We then check whether the next odd p is still prime, and move our counter up until it is equal to 10001. Only checking odd numbers after 2 saves time, since there are no even primes past 2.

```
104759
396.011ms
```

Figure 1: Output