Lecture 5

How to check if an integer is prime?

Naive primality testing: Try dividing n by all integers between 2 and $[\sqrt{n}]$. Complexity $O(\sqrt{n})$. Too slow.

Psuedoprimality testing:

Recall Fermat's Little Theorem

$a^{p-1} \equiv 1 \bmod p$ for p prime and all $a \in (\mathbb{Z}/p\mathbb{Z})^\times$

Thus, if $a^{n-1} \not\equiv 1 \bmod n$ for some $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ then n is not a prime. In particular, if $2^{n-1} \not\equiv 1 \bmod n$, then n is not a prime.

Suprise: If $2^{n-1} \equiv 1 \bmod n$, then probability for a prime $\rightarrow 0$ as $n \rightarrow \infty$.

For a randomly chosen 1024 bit number, $p < 10^{-41}$.

The test is sufficient for random large numbers in practice, but not good for large numbers that are not randomly chosen.

We may decrease the error further by testing with more values of a. Unfortunately, there exist infinitely many composite integers n s.t. $a^{n-1} \equiv 1 \bmod n$ for all $a \in (\mathbb{Z}/n\mathbb{Z})^\times$

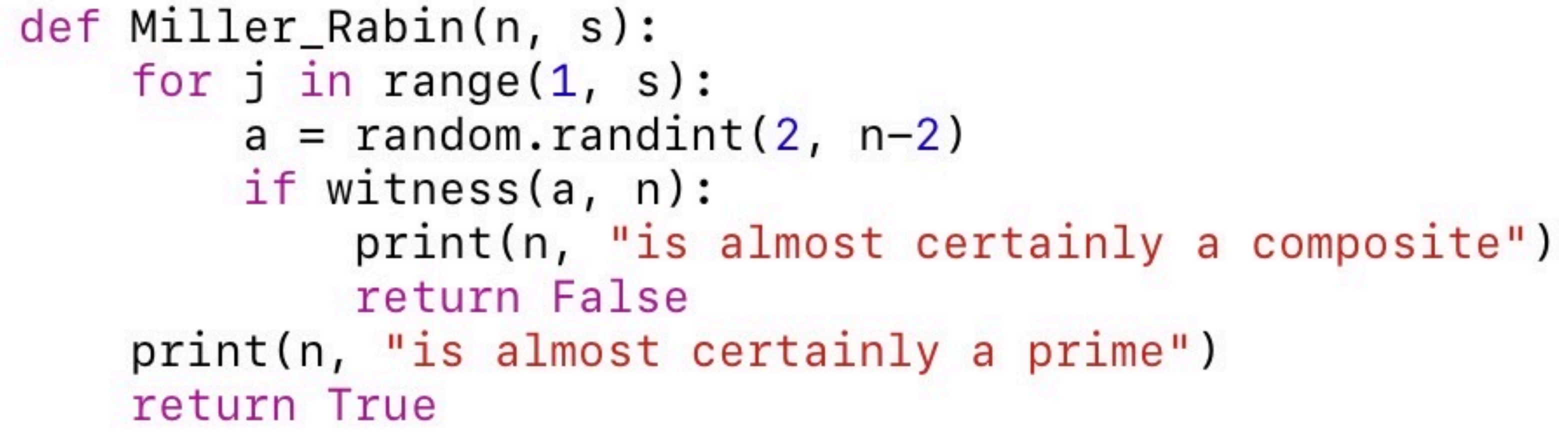
↑ Carmichael numbers

The Miller-Robin randomized primality test:

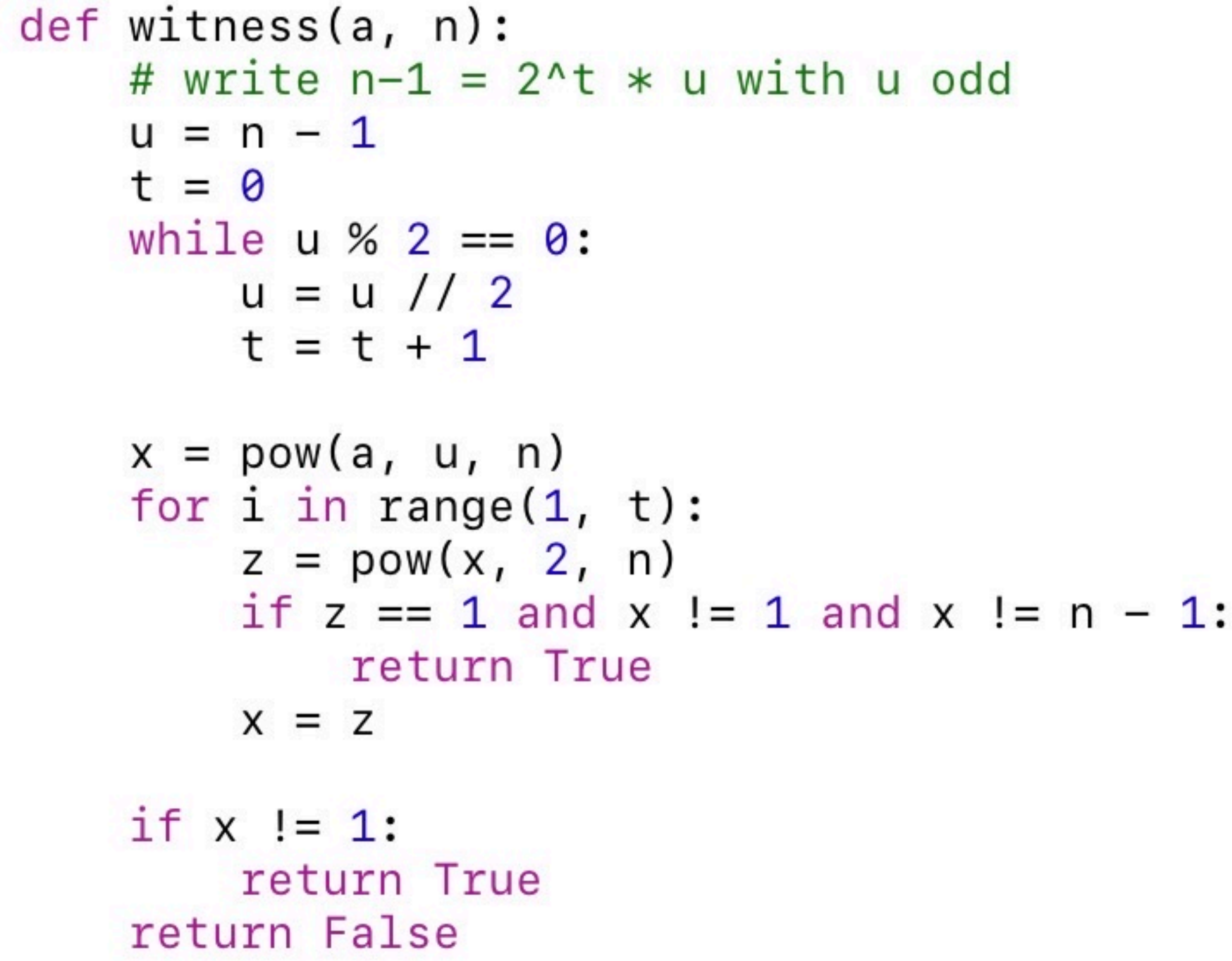
Two improvements:

1) It tries s randomly chosen base values a to test if $a^{n-1} \equiv 1 \mod n$

2) When computing a^{n-1}, it looks for nontrivial square root of 1 and a.



```python
def Miller_Rabin(n, s):
    for j in range(1, s):
        a = random.randint(2, n-2)
        if witness(a, n):
            print(n, "is almost certainly a composite")
            return False
    print(n, "is almost certainly a prime")
    return True
```



```python
def witness(a, n):
    # write n-1 = 2^t * u with u odd
    u = n - 1
    t = 0
    while u % 2 == 0:
        u = u // 2
        t = t + 1

    x = pow(a, u, n)
    for i in range(1, t):
        z = pow(x, 2, n)
        if z == 1 and x != 1 and x != n - 1:
            return True
        x = z

    if x != 1:
        return True
    return False
```

Error rate of the Miller-Rabin primality test.

Theorem. If n is an odd composite number, then the number of witnesses (i.e. witness(a, n) = true) is at least $(n-1)/2$.

Theorem. For any odd integer $n \geq 2$ and positive s, the probability that Miller-Rabin(n, s) errs is at most 2^{-s}.

Advantage: we can control the error rate by increasing s.