def is_prime(n):  # pre: n >= 2
    for d in range(2, n):
        if n % d == 0:
            return False  # loop can stop early!
    return True

Current strategy: For the loop,
1. Between 1 and n-2 iterations
2. Cost per iteration is 1 step

If the loop completes every iteration, "return True" counts as 1 step.
So total # of steps is:
1. at least 1 step (loop takes >= 1 step)
2. at most \((n-2) \times 1 + 1\) steps
So the runtime is \# steps

1. \( \Omega(1) \geq 1 \)
2. \( O(n) \leq n \)

Ex

```python
def g(n):
x = n
while x > 1:
    if x % 2 == 0:
        x = x // 2
    else:
        x = 2 * x - 2
```

* Loop pattern is hard to predict

Need to put more effort into analysing the loop body (i.e., how \( x \) changes)
\\[ \forall x \in \mathbb{N}, x \geq 2 \implies \]
\[
\text{after 2 iterations, } x \text{ decreases by at least 1.}
\]

(Think... (and ask))

This means that the loop runs at most 2n times.

So... the runtime is \( O(n) \).

[In fact, the runtime is not \( \Theta(n) \)!]

[Exercise: can you prove the runtime is \( \Omega(\log n) \)?
\( O(\log n) \)?]
`def all_evens(L):`

L is a list of ints

```python
for x in L:
    if x % 2 == 1:
        return False
return True
```

For lists, "input size" means length of list.

E.g., length = 100

```
|   |   |   |   |   |   |
```

0 1 2 3 4 5 6

Runtime doesn't depend on just length, but where the first odd number is.

Worst-case runtime

```python
# steps
```

```python
n (list length)
```

```
50 100
```

```
50 100
```
Define let $P$ be a program. The worst-case running time of $P$ is the function $W_C P: \mathbb{N} \rightarrow \mathbb{R}^{\geq 0}$ defined as

$$W_C P(n) = \max \left\{ \text{runtime of } P(x) \mid x \text{ has size } n \right\}$$

(best-case runtime

$$B_C P(n) = \min \{ -3 \})$$

We want to communicate something like "$P$ has worst-case runtime $\Theta(n)$".

Part 1. What does $W_C P(n) \in O(f(n))$ mean?

$$\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow W_C P(n) \leq c f(n)$$

$$\exists c, n_0 \cdots \quad n \geq n_0 \Rightarrow \max \{ \text{runtime of } f(x) \mid x \text{ has size } n \} \leq c f(n)$$
Proving this is a universal argument based on properties of the code.

Let \( n \in \mathbb{N} \), and let \( x \) be a list of length \( n \).

Then \( \text{all} \_\text{even}(x) \) takes \( \leq n \) steps.

Then \( \text{WCall} \_\text{even} \in O(n) \).

Part 2 What does \( \text{WCall} \_p(n) \in \Omega(f(n)) \) mean?

\( \exists c, n_0 \in \mathbb{R}^+ \), \( \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow \max \{ \ldots \} \geq c f(n) \).

\( \exists c, n_0 \ldots n \geq n_0 \Rightarrow \exists x \in I_n, \text{runtime } P(x) \geq c f(n) \).
Ex (all-evens)

Let $n \in \mathbb{N}$. Let $x = [0, 0, 0, \ldots, 0]$ (with $n$ 0's).

[Trace the code on this input to count steps]

Then all-evens($x$) takes $n$ steps.

^ "at least"

Then $\text{WC}_{\text{all-evens}}(x) \in \Omega(n)$. ("is all you need for a lower bound")

Combining (1) + (2) we get $\text{WC}_{\text{all-evens}} \leq \Theta(n)$.

Ex (harder!!)

def palindrome_prefix(s):
    for k = len(s), len(s) - 1, ..., 2, 1:
        is_pal = True
        # check if $s[0:k]$ is a palindrome
# Check if \( s[0:k] \) is a palindrome

```python
for j=0...k-1:
    if \( s[j] \neq s[k-1-j] \):
        is_pal = False
        break (out of inner loop)
if is_pal:
    return True
```