Prime Numbers with Sieve of Erastosthenes

Problem Statement:

You are given a positive integer `n`. Your task is to return all prime numbers less than or equal to `n`. To solve this efficiently, you can use **Sieve of Eratosthenes**, an ancient and efficient algorithm for finding all prime numbers up to a given limit.

A **prime number** is a natural number greater than 1 that has no divisors other than 1 and itself. The **Sieve of Erastosthenes** works by iteratively marking the multiples of each prime number starting from 2. Specifically:

- Start with a list of number from `2` to `n`.
- Mark the first unmarked number (starting with `2`) as prime.
- Eliminate all multiples of this number (as they cannot be prime).
- Repeat the process for the next unmarked number until you've processed all numbers up to `n`.

Constraints:

- $1 \leq n \leq 10^6$

Example 1:

Input:

```
n = 10
```

Output:

```
[2, 3, 5, 7]
```

Explanation:
The prime numbers less than or equal to 10 are 2, 3, 5, and 7.

Example 2:

Input:

```
n = 1
```

Output:

```
[]
```

Explanation:
There are no prime numbers less than or equal to 1.

Example 3:

Input:

```
n = 20
```

Output:

```
[2, 3, 5, 7, 11, 13, 17, 19]
```

Explanation:
The prime numbers less than or equal to 20 are listed in the output.

Hint:

1. Use a boolean array `is_prime` where all entries are initially set to `True`. Mark `is_prime[0]` and `is_prime[1]` as `False` since 0 and 1 are not prime.
2. For each number starting from 2, if it is still marked as prime, eliminate all its multiples by marking them as `False`.
3. Collect all indices of the `True` values in the `is_prime` array, as these represent the prime numbers.