Vandermonde Matrix

Let $t = (t_1, t_2, \ldots, t_n) \in \mathbb{R}^n$. Create the matrix $V \in \mathbb{R}^{n \times n}$ such that each row of V is a geometric series of t_i.

$$V = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & t_n^2 & \cdots & t_n^{n-1} \end{bmatrix}$$

The (i, j) element of the Vandermonde matrix is given by $V_{i,j} = t_i^{j-1}$ where $i, j \in \{1, 2, \ldots, n\}$. Your task is to **efficiently** compute the Vandermonde matrix V.

Input Format

- 1d numpy array t of size n containing integers $t_1, t_2, \ldots, t_n$

Output Format

Output the resulting $n \times n$ Vandermonde matrix V. Each element should be returned as an integer.

Constraints

- $t \in \mathbb{R}^n$
- $1 \leq n \leq 1000$

Sample Input

```
t = [2, 3, 1]
```

Sample Output

```
[[1  2  4]
 [1  3  9]
 [1  1  1]]
```

Implementation

Goal: Fill in the following function:

```
def vandermonde_matrix(t):
    ...
    return ... # Return the resulting matrix
exec("\n".join(iter(input, "#Exit"))) # Don't remove this line
```

Hint

- numpy broadcasting (`https://numpy.org/doc/stable/user/basics.broadcasting.html`)