```
import numpy as np


def w1(X):
    """
    Input:
    - X: A numpy array

    Returns:
    - A matrix Y such that Y[i, j] = X[i, j] * 10 + 100

    Hint: Trust that numpy will do the right thing
    """
    Y = X * 10 + 100
    return Y


def w2(X, Y):
    """
    Inputs:
    - X: A numpy array of shape (N, N)
    - Y: A numpy array of shape (N, N)

    Returns:
    A numpy array Z such that Z[i, j] = X[i, j] + 10 * Y[i, j]

    Hint: Trust that numpy will do the right thing
    """
    Z = X + 10 * Y
    return Z


def w3(X, Y):
    """
    Inputs:
    - X: A numpy array of shape (N, N)
    - Y: A numpy array of shape (N, N)

    Returns:
    A numpy array Z such that Z[i, j] = X[i, j] * Y[i, j] - 10

    Hint: By analogy to +, * will do the same thing
    """
    Z = X * Y - 10
    return Z


def w4(X, Y):
    """
    Inputs:
    - X: Numpy array of shape (N, N)
    - Y: Numpy array of shape (N, N)

    Returns:
    A numpy array giving the matrix product X times Y

    Hint:
    1. Be careful! There are different variants of *, @, dot
    2.  a = [[1,2],
             [1,2]]
        b = [[2,2],
             [3,3]]
        a * b = [[2,4],
                 [3,6]]
    Is this matrix multiplication?

    """
    Z = X @ Y
    return Z


def w5(X):
    """
    Inputs:
    - X: A numpy array of shape (N, N) of floating point numbers
    Returns:
    A numpy array with the same data as X, but cast to 32-bit integers

    Hint: Check .astype() !
    """
    Z = X.astype(dtype = np.int32)
    return Z


def w6(X, Y):
    """
    Inputs:
    - X: A numpy array of shape (N,) of integers
    - Y: A numpy array of shape (N,) of integers

    Returns:
    A numpy array Z such that Z[i] = float(X[i]) / float(Y[i])

    """
    Z = X.astype(dtype = np.float32) / Y.astype(dtype = np.float32)
    return Z


def w7(X):
    """
    Inputs:
    - X: A numpy array of shape (N, M)

    Returns:
    - A numpy array Y of shape (N * M, 1) containing the entries of X in row
      order. That is, X[i, j] = Y[i * M + j, 0]

    Hint:
    1) np.reshape
    2) You can specify an unknown dimension as -1
    """
    return np.reshape(X, (-1, 1))


def w8(N):
    """
    Inputs:
    - N: An integer

    Returns:
    A numpy array of shape (N, 2N)

    Hint: The error "data type not understood" means you probably called
    np.ones or np.zeros with two arguments, instead of a tuple for the shape
    """
    return np.zeros((N, 2 * N))


def w9(X):
    """
    Inputs:
    - X: A numpy array of shape (N, M) where each entry is between 0 and 1

    Returns:
    A numpy array Y where Y[i, j] = True if X[i, j] > 0.5

    Hint: Try boolean array indexing
    """
    return X > 0.5


def w10(N):
    """
    Inputs:
    - N: An integer

    Returns:
    A numpy array X of shape (N,) such that X[i] = i

    Hint: np.arange
    """
    return np.arange(N)
```

```
def w11(A, v):
    """
    Inputs:
    - A: A numpy array of shape (N, F)
    - v: A numpy array of shape (F, 1)

    Returns:
    Numpy array of shape (N, 1) giving the matrix-vector product Av
    """
    return A @ v


def w12(A, v):
    """
    Inputs:
    - A: A numpy array of shape (N, N), of full rank
    - v: A numpy array of shape (N, 1)

    Returns:
    Numpy array of shape (N, 1) giving the matrix-vector product of the inverse
    of A and v: A^-1 v
    """
    return np.linalg.inv(A) @ v


def w13(u, v):
    """
    Inputs:
    - u: A numpy array of shape (N, 1)
    - v: A numpy array of shape (N, 1)

    Returns:
    The inner product u^T v

    Hint: .T
    """
    # inner product = dot product
    return u.T @ v


def w14(v):
    """
    Inputs:
    - v: A numpy array of shape (N, 1)

    Returns:
    The L2 norm of v: norm = (sum_i^N v[i]^2)^(1/2)
    You MAY NOT use np.linalg.norm
    """
    norm = np.sqrt(np.sum(v ** 2))
    return norm


def w15(X, i):
    """
    Inputs:
    - X: A numpy array of shape (N, M)
    - i: An integer in the range 0 <= i < N

    Returns:
    Numpy array of shape (M,) giving the ith row of X
    """
    return X[i]


def w16(X):
    """
    Inputs:
    - X: A numpy array of shape (N, M)

    Returns:
    The sum of all entries in X

    Hint: np.sum
    """
    return np.sum(X)
```

```
def w17(X):
    """
    Inputs:
    - X: A numpy array of shape (N, M)

    Returns:
    A numpy array S of shape (N,) where S[i] is the sum of row i of X

    Hint: np.sum has an optional "axis" argument
    """
    return np.sum(X, axis = 1)


def w18(X):
    """
    Inputs:
    - X: A numpy array of shape (N, M)

    Returns:
    A numpy array S of shape (M,) where S[j] is the sum of column j of X

    Hint: Same as above
    """
    return np.sum(X, axis = 0)


def w19(X):
    """
    Inputs:
    - X: A numpy array of shape (N, M)

    Returns:
    A numpy array S of shape (N, 1) where S[i, 0] is the sum of row i of X

    Hint: np.sum has an optional "keepdims" argument
    """
    return np.sum(X, axis = 1, keepdims =  True)


def w20(X):
    """
    Inputs:
    - X: A numpy array of shape (N, M)

    Returns:
    A numpy array S of shape (N, 1) where S[i] is the L2 norm of row i of X
    """
    return np.sqrt(np.sum(X ** 2, axis = 1, keepdims =  True))
```