****

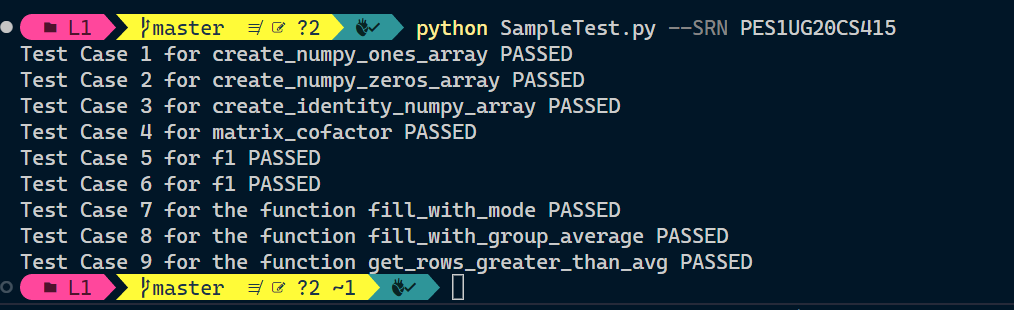
**SRN: PES1UG20CS415**

**Name: SHRUJAN**

**Section: G**

**Date: 20/8/2022**

**Screenshot**



**Code**

# *This weeks code focuses on understanding basic functions of pandas and numpy*

# *This will help you complete other lab experiments*

# *Do not change the function definations or the parameters*

*import* collections

*from* fileinput *import* filename

*from* re *import* X

*import* numpy as np

*import* pandas as pd

# *input: tuple (x,y)    x,y:int*

def *create\_numpy\_ones\_array*(shape):

    # *return a numpy array with one at all index*

    array = np.ones(shape)

    # *TODO*

*return* array

# *input: tuple (x,y)    x,y:int*

def *create\_numpy\_zeros\_array*(shape):

    # *return a numpy array with zeros at all index*

    array = None

    array = np.zeros(shape)

*return* array

#*input: int*

def *create\_identity\_numpy\_array*(order):

    # *return a identity numpy array of the defined order*

    array = None

    array = np.identity(order)

*return* array

# *input: numpy array*

def *matrix\_cofactor*(array):

    # *return cofactor matrix of the given array*

    array = np.linalg.inv(array).T\*np.linalg.det(array)

*return* array

# *Input: (numpy array, int ,numpy array, int , int , int , int , tuple,tuple)*

# *tuple (x,y)    x,y:int*

def *f1*(X1, coef1, X2, coef2, seed1, seed2, seed3, shape1, shape2):

    # *note: shape is of the forst (x1,x2)*

    # *return W1 x (X1 \*\* coef1) + W2 x (X2 \*\* coef2) +b*

    # *where W1 is random matrix of shape shape1 with seed1*

    # *where W2 is random matrix of shape shape2 with seed2*

    # *where B is a random matrix of comaptible shape with seed3*

    # *if dimension mismatch occur return -1*

    np.random.seed(seed1)

    W1 = np.random.rand(\*shape1)

    np.random.seed(seed2)

    W2 = np.random.rand(\*shape2)

    np.random.seed(seed3)

    B = np.random.rand(shape1[0], shape2[1])

*if* X1.shape[0] != shape1[1] or X2.shape[0] != shape2[1]:

*return* -1

    a1 = np.matmul(W1, X1\*\*coef1)

    a2 = np.matmul(W2, X2\*\*coef2)

*if* a1.shape[0] != a2.shape[0] or a1.shape[1] != a2.shape[1]:

*return* -1

    ans = a1+a2+B

*return* ans

def *fill\_with\_mode*(filename, column):

    """

    Fill the missing values(NaN) in a column with the mode of that column

    Args:

        filename: Name of the CSV file.

        column: Name of the column to fill

    Returns:

        df: Pandas DataFrame object.

        (Representing entire data and where 'column' does not contain NaN values)

        (Filled with above mentioned rules)

    """

    df = pd.read\_csv(filename)

    df[column] = df[column].fillna(df[column].mode()[0])

*return* df

def *fill\_with\_group\_average*(df, group, column):

    """

    Fill the missing values(NaN) in column with the mean value of the

    group the row belongs to.

    The rows are grouped based on the values of another column

    Args:

        df: A pandas DataFrame object representing the data.

        group: The column to group the rows with

        column: Name of the column to fill

    Returns:

        df: Pandas DataFrame object.

        (Representing entire data and where 'column' does not contain NaN values)

        (Filled with above mentioned rules)

    """

    df[column] = df[column].fillna(df.groupby(group)[column].transform('mean'))

*return* df

def *get\_rows\_greater\_than\_avg*(df, column):

    """

    Return all the rows(with all columns) where the value in a certain 'column'

    is greater than the average value of that column.

    row where row.column > mean(data.column)

    Args:

        df: A pandas DataFrame object representing the data.

        column: Name of the column to fill

    Returns:

        df: Pandas DataFrame object.

        """

    df = df[df[column] > df[column].mean()]

*return* df