**Name: Adithya M Section: K SRN: PES1UG20CS621**

**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**

**from operator import truediv**

**from turtle import shape**

**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 = None**

**array = np.ones(shape)**

**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**

**det = np.linalg.det(array)**

**if det != 0:**

**cofactor = None**

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

**else:**

**return -1**

**return cofactor**

**# 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**

**# TODO**

**ans = None**

**np.random.seed(seed1)**

**W1 = np.random.rand(shape1[0])**

**np.random.seed(seed2)**

**W2 = np.random.rand(shape2[0])**

**np.random.seed(seed3)**

**try:**

**ans = W1 \* (X1 \*\* coef1) + W2 \* (X2 \*\* coef2)**

**b = np.random.rand(np.shape(ans))**

**ans = ans + b**

**except:**

**return np.array(-1)**

**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].fillna(df[column].mode()[0], inplace=True)**

**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)**

**v=column**

**"""**

**df[column] = df.groupby(group)[column].apply(lambda x: x.fillna(x.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:**

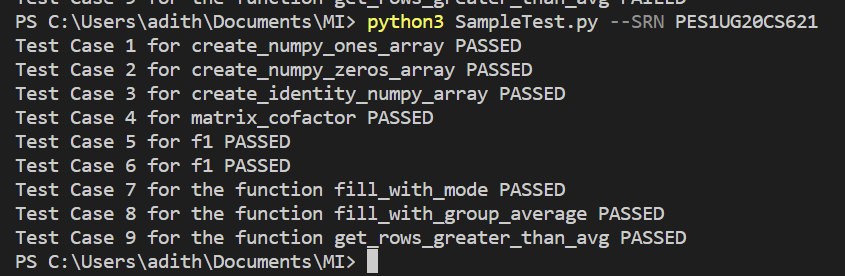
**df: Pandas DataFrame object.**

**"""**

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

**return df**

**Output:**

****