Functions.py code

```
"""Functions for analysis of agricultural yield"""
# from pandas.core.frame import DataFrame
# import matplotlib.pyplot as plt
import pandas as pd
# import seaborn as sns
# import statsmodels as sms
# from sklearn.ensemble import RandomForestRegressor

test_df = pd.read_csv("yield_df.csv")


# i wrote a return statement for each function
# because pylint gave me an error for
# "uneccessary pass statement"
def wrangle_frame(dataframe):
    """function intended to merge the required
     csv files "yield", "rainfall", "pesticides" and "temp",
      and filter out unneeded columns
     and rows with missing data. I plan to make
     dummy variables for each crop type, and lagged
     variables for rain if possible to see if multiyear droughts
     or high rain has an effect"""
    return dataframe


def graph_agri():
    """returns informative graphs exploring the
    relationship between different variables that
    could effect agricultural yeild, including tempurature,
    rainfall, pesticide use, and crop type"""


def summary_stats(column_name):
    """"returns summary statistics for
    each relevant column in the dataframe,
    in order to better understand the distribution
    of data, called for each column and returns a list,
    of mean, st. dev, variance, and median"""
    return column_name


def ols_model(dataframe):
    """runs an ols regression predicting yield,
       based on relevent variables such as tempurature,
```

```
       pesticide use, rainfall , and other relevent variables.
       uses training and test sets to k-fold cross validate.
       returns model and prediction accuracy"""
    return dataframe


# might test out a boosting and bagging model as well

def random_forest_model(dataframe):
    """runs a more flexible model, random forest
       in order to predict crop yield
       and the relevent variables.
       uses training and test sets to k-fold cross validate.
       returns random forest model and prediction accuracy"""
    return dataframe
```

test.py code

```
"""tests for functions in module"""
import pandas as pd
import matplotlib.pyplot as plt
import functions as fn


test_df = pd.read_csv("yield_df.csv")


def test_data_wrangle():
    """will test the data_wrangle function
    in order to check whether the crop yield data
    was correctly organized, cleaned and merged.
    and has the 6 extra columns, 5 for each crop type,
    and 1 for lagged rainfall """
    assert len(fn.wrangle_frame(test_df).columns) == 12
    original_len = len(test_df["rain"])
    assert len(fn.wrangle_frame(test_df)["lagged_rain"]) <= original_len

    print("data wrangle tests passed")


def test_graph_agri():
    """will test the graph_agri function
    in order to check that graphs are returned without
    error. Preliminary plans to make 6 plots"""
```

```
    def subtest_1():
        fn.graph_agri()
        assert plt.gcf().number == 6

    def subtest_2():
        num_figures_before = plt.gcf().number
        fn.graph_agri()
        num_figures_after = plt.gcf().number
        assert num_figures_before < num_figures_after

    subtest_1()

    subtest_2()

    print("graphing tests passed")


def test_summary_stats():
    """Will test the summary_stats function,
    to make sure the summary stats are internally
    consitent and no data is being dropped or incorrectly
    summarized. """
    wrang_data = fn.wrangle_frame(test_df)
    mean_value_rain = sum(wrang_data["rain"]) / len(wrang_data["rain"])
    assert mean_value_rain == fn.summary_stats("rain")[1]
    print("summary stat tests passed")


def test_ols():
    """test whether ols and mse are calculated correctly by the
    ols_model function. Return statement will return fit model, but
    summary and mse will still be printed"""
    x_model = fn.wrangle_frame(test_df).drop(["hg/ha_yield"])
    y_model = fn.wrangle_frame(test_df)["hg/ha_yield"]
    assert len(fn.ols_model(y_model).predict(x_model)) == len(
                y_model)

    print("ols tests passed")


def test_random_forest():
    """test whether random forest and
    mse are correctly and consistently calculated
    by ht erandom_forest_model function"""
    x_model = fn.wrangle_frame(test_df).drop(["hg/ha_yield"])
```

```
    y_model = fn.wrangle_frame(test_df)["hg/ha_yield"]
    assert len(fn.random_forest_model(y_model).predict(x_model)) == len(
                          y_model)
    print("random forest tests passed")
```