```
# Python for Healthcare

## Hospitals and Cost Narrative

### Import Standard Libraries
import os # Inlcuded in every script DC!
import pandas as pd # Incldued in every code script for DC!
import numpy as np # Incldued in every code script for DC!

### Set working directory to project folder
os.chdir("C:/Users/drewc/GitHub/python-for-healthcare/lessons/pymodule1") # Set wd
to project repository

### Verify
print("Ready") # Print result

#################### Break ####################

# Section A: 2018 MSPB by State (EDA)
print("Section A: Start") # Print result

## Step 1: Import Libraries and Data

### Import Libraries for Section
import geopandas as gp # Simple mapping with pandas like syntax
import matplotlib.pyplot as plt # Comprehensive graphing package in python

### Import CMS Data
df_cms = pd.read_csv("_data/cms_mspb_stage.csv", encoding = "ISO-8859-1") # Import
dataset saved as csv in _data folder

### Import State Shape File
gdf_state = gp.read_file("_data/_maps/state.shp") # Import shapefile saved with
corresponding

### Verify CMS
df_cms.info() # Get class, memory, and column info: names, data types, obs.
df_cms.head() # Print first 5 observations

## Step 2: Prepare Data for Analysis

### Select only State and Measure
df_filter = df_cms.filter(["State", "Score"]) # Keep only selected columns

### Group by State
df_group = df_filter.groupby(["State"], as_index = False).mean() # Group data By
Columns and Sum

### Rename Score as MSPB
df_rename = df_group.rename(columns = {"Score": "MSPB"}) # Rename column
```

```
### Drop NA values
df_na = df_rename.dropna() # Drop all rows with NA values

### Rename Dataframe
df_mspb = df_na # Rename sorted dataframe as MSPB for clarity

### Verify MSPB
df_mspb.info() # Get class, memory, and column info: names, data types, obs.
df_mspb.head() # Print first 5 observations

## Step 3: Conduct Analysis and Tests

### Summary Statistics for States
summary = df_mspb.describe() # Get summary statistics for numerical columns in data
frame

### Create Results Text File
text_file = open("_fig/mspb_summary.txt", "w") # Open text file and name with
subproject, content, and result suffix. To write or overwrite a new file, type "w".
To append, type "a".
text_file.write(str(summary)) # Line of text with string version of a data object
text_file.close() # Close file

### Verify MSPB
print(summary) # Print summary statistics

## Step 4: Create Visuals and Outputs

### Geo Join State and Geometry
gdf_join = pd.merge(gdf_state, df_mspb, on = "State", how = "inner") # Join by
column while keeping only items that exist in both, select outer or left for other
options

### Create Figure
plt.figure() # Create blank figure before creating plot

### Create Map Plot
gdf_join.plot(column = "MSPB", cmap = "Blues", legend = False).set_axis_off() #
Create simple chloropleth map in geopandas

### Set Labels and Titles
plt.title("Medicare Spending Per Beneficiary by State in 2018") # Title above the
plot

### Save to figure file
plt.savefig("_fig/mspb_chloro.jpeg", bbox_inches = "tight") # Save figure file to
_fig in directory, use tight to make a good looking image

## Verify
```

```
plt.show() # Show created plots

# End Section
print("THE END") # Print result

#################### Break ####################

# Section B: MSPB by State and Money (Q-Q)
print("Section B: Start") # Print result

## Step 1: Import Libraries and Data

### Import Statistics Packages
import statsmodels.api as sm # Regression modeling in scipy

### Import Money Data
df_money = pd.read_csv("_data/money_state_stage.csv", encoding = "ISO-8859-1") #
Import dataset saved as csv in _data folder

### Verify Money
df_money.info() # Get class, memory, and column info: names, data types, obs.
df_money.head() # Print first 5 observations

## Step 2: Prepare Data for Analysis

### Inner MSPB
df_join = pd.merge(df_mspb, df_money, on = "State", how = "inner") # Join by column
while keeping only items that exist in both, select outer or left for other options
df_join.info() # Get class, memory, and column info: names, data types, obs.

### Drop Values with NA
df_na = df_join.dropna() # Drop all rows with NA values, 0 = rows, 1 = columns

### Rename to Regression
df_reg = df_na

### Verify MSPB
df_reg.info() # Get class, memory, and column info: names, data types, obs.
df_reg.head() # Print first 5 observations

## Step 3: Conduct Analysis and Tests

### Linear Regression Model
features = df_reg.columns.drop(["MSPB", "State"]) # Drop outcome variable and
Geographies to isolate all predictor variable names as features
x = df_reg[features] # features as x
y = df_reg["MSPB"] # Save outcome variable as y
model = sm.OLS(y, x).fit() # Run Linear Regression Model This may but most likely
wont take time
result = model.summary() # Create Summary of final model
```

```
### Create Results Text File
text_file = open("_fig/mspb_fp_model.txt", "w") # Open text file and name with
subproject, content, and result suffix. To write or overwrite a new file, type "w".
To append, type "a".
text_file.write(str(result)) # Line of text with string version of a data object
text_file.close() # Close file

### Verify Regression
print(result) # Print result to verify

## Step 4: Create Visuals and Outputs

### Create Figure
plt.figure() # Create blank figure before creating plot

### Create Scatter Plot
plt.scatter(df_reg["ForProfit"], df_reg["MSPB"], c = "b") # Create scatter plot with
(x axis, y axis, color)

### Set Labels and Titles
plt.ylabel("Average State MSPB in 2018 (Released by CMS)") # Label Y axis
plt.xlabel("Percent of Hospitals that are For-Profit in State") # Label for X Axis
plt.title("Medicare Spending Per Beneficiary and For-Profit Hospitals by State in
2018") # Title above the plot

### Save to figure file
plt.savefig("_fig/mspb_fp_scatter.jpeg", bbox_inches = "tight") # Save figure file
to _fig in directory, use tight to make a good looking image

## Verify
plt.show() # Show created plots

# The End
print("THE END") # Print result
```