






rurigi-waweru /

dsfpt10-p1-phase-1-code-challenge-codegrade



<> Code

Issues

Pull requests

Actions

Projects


Wiki


Security


Insights

Settings


dsfpt10-p1-phase-1-code-challenge-codegrade / p1_code_challenge.ipynb





 rurigi-waweru completed p1-code-challenge-codegrade!

ca70273 · 26 minutes ago




1156 lines (1156 loc) · 107 KB


Preview


Code


Blame

Raw









Rav

Do

Co

Co

Vie

✓

W

C

✓

De

Phase 1 Code Challenge

This code challenge is designed to test your understanding of the Phase 1 material. It covers:

- Pandas
- Data Visualization
- Exploring Statistical Data
- Python Data Structures

Read the instructions carefully. Your code will need to meet detailed specifications to pass automated tests.

Code Tests

We have provided some code tests for you to run to check that your work meets the item specifications. Passing these tests does not necessarily mean that you have gotten the item correct - there are additional hidden tests. However, if any of the tests do not pass, this tells you that your code is incorrect and needs changes to meet the specification. To determine what the issue is, read the comments in the code test cells, the error message you receive, and the item instructions.

Part 1: Pandas [Suggested Time: 15 minutes]

In this part, you will preprocess a dataset from the video game [FIFA19](#), which contains data from the players' real-life careers.

In [177...

```
# Run this cell without changes

import pandas as pd
import numpy as np
from numbers import Number
import warnings
warnings.filterwarnings('ignore')
```

1.1) Read `fifa.csv` into a pandas DataFrame named `df`

Use pandas to create a new DataFrame, called `df`, containing the data from the dataset in the file `fifa.csv` in the folder containing this notebook.

Hint: Use the string `'./fifa.csv'` as the file reference.

In [178...

```
# CodeGrade step1.1
# Replace None with appropriate code

df = pd.read_csv('./fifa.csv')
df.head()
```

Out[178...

	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	
0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	Bæ
1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	J
2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Pari (
3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Mar
4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Mar

5 rows × 88 columns

In [179...

```
# This test confirms that you have created a DataFrame named df

assert type(df) == pd.DataFrame
```

1.2) Convert the 'Release Clause' values from Euros to dollars

The 'Release Clause' variable contains prices denominated in Euros. Use the exchange rate 1 Euro = 1.2 Dollars to convert the prices to dollars.

```
In [180...
# CodeGrade step1.2
# Replace None with appropriate code

df['Release Clause'] = df['Release Clause'] * 1.2
```

```
In [181...
df['Release Clause'].head(5) # brian-added
```

```
Out[181...
0    271800.0
1    152520.0
2    273720.0
3    166320.0
4    235680.0
Name: Release Clause, dtype: float64
```

1.3) Drop rows from df with missing values for the 'Release Clause' feature.

Make sure that df remains the name of the dataset with the dropped rows.

```
In [182...
# CodeGrade step1.3
# Replace None with appropriate code

df = df.dropna(subset = ['Release Clause'])
```

```
In [183...
# This test confirms that your dataset has the correct number of observations after dropping

assert df['Release Clause'].shape[0] == 16643
```

1.4) Create a list top_10_countries containing the names of the 10 countries with the most players (using the 'Nationality' column).

Hint: Your answer should include England, Germany, Spain, France, and Argentina

```
In [184...
# CodeGrade step1.4
# Replace None with appropriate code

top_10_countries = df['Nationality'].value_counts().head(10).index.tolist()
# top_10_countries
```

```
In [185...
# This test confirms that you have created a list named top_10_countries

assert type(top_10_countries) == list

# This test confirms that top_10_countries contains England, Germany, Spain, France, and Argentina

assert set(['England', 'Germany', 'Spain', 'France', 'Argentina']).issubset(set(top_10_countries))
```

Part 2: Data Visualization [Suggested Time: 20 minutes]

This part uses the same FIFA dataset, and asks you to plot data using matplotlib .

```
In [186...
# Run this cell without changes

import matplotlib
import matplotlib.pyplot as plt
```

2.1) Create a matplotlib figure player_count_figure containing a labeled bar chart with the number of players from England, Germany, Spain, France, and Argentina

Use the strings provided below (bar_chart_title , bar_chart_count_label , and bar_chart_series_label) to title and label your bar chart.

Hint: These are the countries with the top 5 numbers of players, so you may be able to adapt some of the code you used for question 1.4. If you were unable to complete 1.4, use the following values:

Country Name	Num Players
England	16643
Germany	15252
Spain	27372
France	16632
Argentina	23568

England	1000
Germany	900
Spain	800
France	700
Argentina	600

In [187...

```
# CodeGrade step2.1
# Replace None with appropriate code

bar_chart_countries = ['England', 'Germany', 'Spain', 'France', 'Argentina']

bar_chart_title = '5 Countries with the Most Players'
bar_chart_count_label = 'Number of Players'
bar_chart_series_label = 'Nationality'

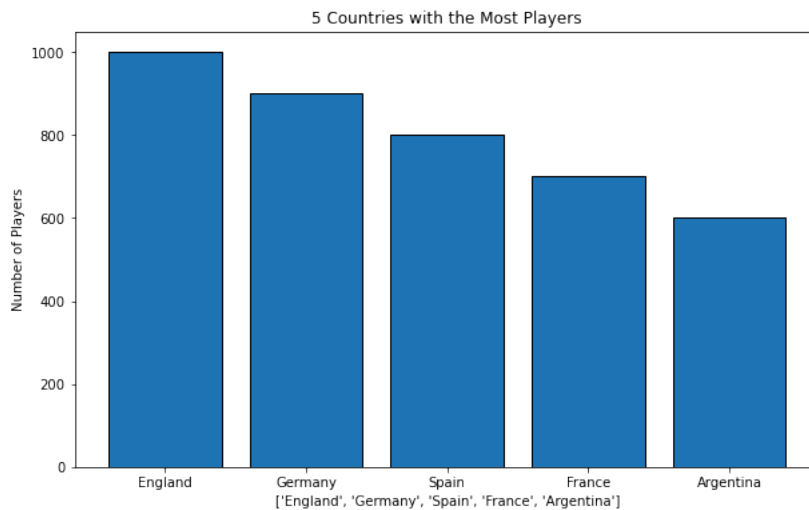
# -----
top_5_countries = bar_chart_countries
# -----

# analysis
player_count = [1000, 900, 800, 700, 600] # from above table

player_count_figure, ax = plt.subplots(figsize=(10, 6))

# -----
x = top_5_countries
heights = player_count

ax.bar(x, heights, edgecolor = 'black', linewidth = 1)
# title
ax.set_title(bar_chart_title)
# x-axis label
ax.set_xlabel(bar_chart_countries)
# y-axis label
ax.set_ylabel(bar_chart_count_label);
```



In [188...

```
# This test confirms that you have created a figure named player_count_figure

assert type(player_count_figure) == plt.Figure

# This test confirms that the figure contains exactly one axis

assert len(player_count_figure.axes) == 1
```

In [189...

```
# These tests confirm that the figure has a title and axis labels

assert player_count_figure.axes[0].get_title() != ''
assert player_count_figure.axes[0].get_ylabel() != ''
assert player_count_figure.axes[0].get_xlabel() != ''
```

2.2) Create a matplotlib figure tackle_figure containing a labeled scatter plot visualizing the relationship between StandingTackle (on X axis) and SlidingTackle (on Y axis)

Use the strings provided below (scatter_plot_title , standing_tackle_label , and sliding_tackle_label) to title and label your scatter plot.

In [190...

```
# CodeGrade step2.2
```

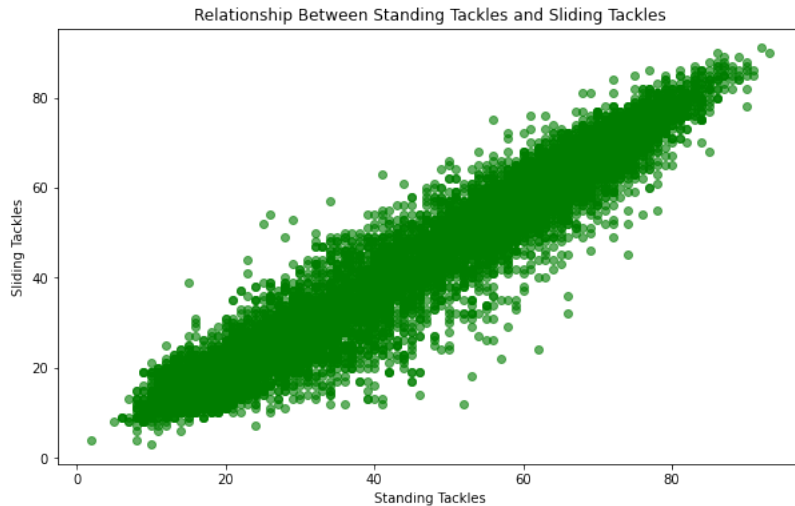
```

scatter_plot_title = 'Relationship Between Standing Tackles and Sliding Tackles'
standing_tackle_label = 'Standing Tackles'
sliding_tackle_label = 'Sliding Tackles'

tackle_figure, ax = plt.subplots(figsize=(10, 6))

# Your code here
ax.scatter(df.StandingTackle, df.SlidingTackle, alpha = 0.6, color = 'green')
# title
ax.set_title(scatter_plot_title)
# x-axis label
ax.set_xlabel(standing_tackle_label)
# y-axis label
ax.set_ylabel(sliding_tackle_label);

```



```

In [191... # This test confirms that you have created a figure named tackle_figure

assert type(tackle_figure) == plt.Figure

# This test confirms that the figure contains exactly one axis

assert len(tackle_figure.axes) == 1

```

Part 3: Exploring Statistical Data [Suggested Time: 20 minutes]

This part does some exploratory analysis using the same FIFA dataset.

3.1) Create numeric variables `mean_age` and `median_age` containing the mean and median player ages (respectively).

```

In [192... # CodeGrade step3.1
# Replace None with appropriate code

mean_age = df.Age.mean()
median_age = df.Age.median()

```

```

In [193... # These tests confirm that you have created numeric variables named mean_age and median_age

assert isinstance(mean_age, Number)
assert isinstance(median_age, Number)

```

3.2) Create a string variable `oldest_argentine_name` and a numeric variable `oldest_argentine_age` containing the name and age (respectively) of the oldest player with Argentina nationality.

```

In [194... # CodeGrade step3.2
# Replace None with appropriate code

argentines_dataframe = df[df.Nationality == 'Argentina']
oldest_argentine = argentines_dataframe.loc[argentines_dataframe.Age.idxmax()]

oldest_argentine_name = oldest_argentine.Name
oldest_argentine_age = oldest_argentine.Age

print('oldest_argentine_name:', oldest_argentine_name)

```

```
print('oldest_argentine_age:', oldest_argentine_age)
```

```
oldest_argentine_name: C. Muñoz
oldest_argentine_age: 41
```

```
In [195...
# This test confirms that you have created a string variable named oldest_argentine_name

assert type(oldest_argentine_name) == str

# This test confirms that you have created a numeric variable named oldest_argentine_age

assert isinstance(oldest_argentine_age, Number)
```

Part 4: Python Data Structures [Suggested Time: 20 min]

Below is a dictionary `players` with information about soccer players. The keys are player names and the values are dictionaries containing each player's age, nationality, and a list of teams they have played for.

```
In [196...
# Run this cell without changes

players = {
    'L. Messi': {
        'age': 31,
        'nationality': 'Argentina',
        'teams': ['Barcelona']
    },
    'Cristiano Ronaldo': {
        'age': 33,
        'nationality': 'Portugal',
        'teams': ['Juventus', 'Real Madrid', 'Manchester United']
    },
    'Neymar Jr': {
        'age': 26,
        'nationality': 'Brazil',
        'teams': ['Santos', 'Barcelona', 'Paris Saint-Germain']
    },
    'De Gea': {
        'age': 27,
        'nationality': 'Spain',
        'teams': ['Atletico Madrid', 'Manchester United']
    },
    'K. De Bruyne': {
        'age': 27,
        'nationality': 'Belgium',
        'teams': ['Chelsea', 'Manchester City']
    }
}
```

4.1) Create a list `player_names` of all the player names in dictionary `players`.

```
In [197...
# CodeGrade step4.1
# Replace None with appropriate code

player_names = list(players.keys())
player_names # ['L. Messi', 'Cristiano Ronaldo', 'Neymar Jr', 'De Gea', 'K. De Bruyne']
```

```
Out[197... ['L. Messi', 'Cristiano Ronaldo', 'Neymar Jr', 'De Gea', 'K. De Bruyne']
```

```
In [198...
# This test confirms that you have created a List named player_names

assert type(player_names) == list
```

4.2) Create a list of tuples `player_nationalities` containing each player's name along with their nationality.

```
In [199...
# CodeGrade step4.2
# Replace None with appropriate code

player_nationalities = [(Name, Value["nationality"]) for Name, Value in players.items()]

# printing result
player_nationalities
```

```
Out[199... [('L. Messi', 'Argentina'),
 ('Cristiano Ronaldo', 'Portugal'),
 ('Neymar Jr', 'Brazil'),
 ('De Gea', 'Spain'),
 ('K. De Bruyne', 'Belgium')]
```

In [200...

```
# This test confirms that you have created a list named player_nationalities

assert type(player_nationalities) == list
```

4.3) Define a function `get_players_on_team()` that returns a list of names of all the players who have played on a given team.

Your function should take two arguments:

- A dictionary of player information
- A string containing a team name (for which you are trying to find the player names)

In [201...

```
# CodeGrade step4.3

def get_players_on_team(player_dict, team_name):
    player_list = []

    # solution
    for name, details in player_dict.items():
        if team_name in details['teams']:
            player_list.append(name)

    return player_list

# testing the function
get_players_on_team(players, 'Real Madrid')
```

Out[201...

```
['Cristiano Ronaldo']
```

In [202...

```
# This test confirms that get_players_on_team() returns the right names for Manchester United

manchester_united_players = ['Cristiano Ronaldo', 'De Gea']
players_on_manchester_united = get_players_on_team(players, 'Manchester United')

assert players_on_manchester_united == manchester_united_players
```

THE END!