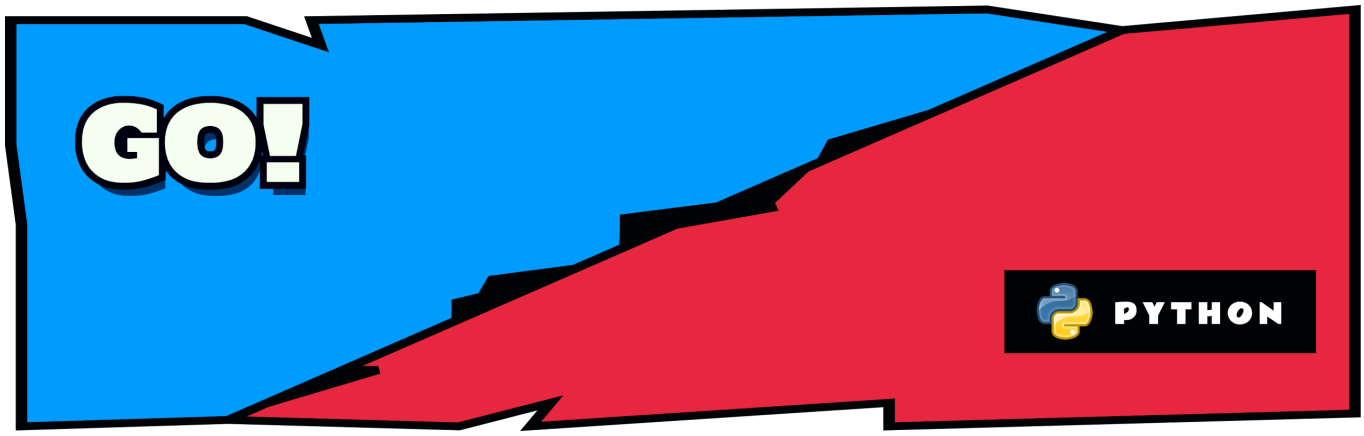

Spogliatoio | Python: Data on national teams around the world since 1993

Caderno de **Códigos**
Desenvolvedor Enzo Schitini

```
# SPOGLIATOIO - @ADIGE COMPANY

from functools import reduce
from time import sleep
import platform
import time
import matplotlib.pyplot as plt
import pandas as pd
import datetime
import pygame

from adige import ArquivoCSV
from adige import valore_abbreviato

avvio = time.time()
data = './base.csv'
df = pd.read_csv(data)
paese_da_cercare = 'Italy'

df = pd.read_csv(data)
print(df.head(n=10))

csv = ArquivoCSV(data)
print(f'\nCi sono {len(list(set(csv.extrair_coluna(1))))} nazionali.')

pygame.init()

pygame.mixer.music.load('1109.mp3')
pygame.mixer.music.play()
pygame.event.wait()

def graficos(liata_x:list, lista_y:list, colere_x:str, colore_y:str, nome:str):

  x = liata_x
  y = lista_y

  plt.plot(x,y,color='blue')
  plt.plot(x,y,color='black')
  plt.xticks(rotation = 90)
  #plt.title('Nome')
  #plt.xlabel('x')
  #plt.ylabel('y')
  #plt.grid()
  #plt.legend(['Estimativa', 'Dado'])
  plt.savefig(nome + '.jpg')
  plt.show()

for i in range(3):
    print('\n')
print('DATI PRELEVATI')

def valore_abbreviato(valore:str) -> str:
    alg_valore = list(valore)
    num_punt = sum(map(lambda x: x == '.', valore))
    for x in range(num_punt):
     alg_valore.remove('.')
```

```
    alg_valore = alg_valore.replace(".", "")
    numero = int("".join(alg_valore))
    numero_format = "{:,}".format(numero).replace(",", ".")
    return numero_format

line, colonne = df.shape

print(f'O arquivo {data} tem:')
print(f'{valore_abbreviato(str(line))} linhas.')
print(f'{valore_abbreviato(str(colonne))} colunas.')

#########################################################################################################################################################
# ------------------------- Classe delle prestazioni
#########################################################################################################################################################

class Prestazioni(object):

    def __init__(self, paese: str, data) -> None:
        self.paese = paese
        self.data = data

    def generale(self):
        df_paese_totale = self.data[(self.data['home_team'] == self.paese) | (self.data['away_team'] == self.paese)]
        totale = valore_abbreviato(str(len(df_paese_totale)))
        return totale

    def vittorie(self):

        def calcoli(posto, score1, score2):
            df_goal = self.data[['home_team', 'away_team', 'home_team_score', 'away_team_score', 'tournament']]
            totale_di_partite = df_goal[df_goal[posto] == self.paese]
            vittorie = df_goal[(df_goal[posto] == self.paese) & (self.data[score1] > self.data[score2])]

            vittorie = valore_abbreviato(str(len(vittorie)))

            # Proporzione
            proporzione = round((float(vittorie) / len(totale_di_partite)) * 100, 2)
            return len(totale_di_partite), vittorie, proporzione

        partite_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[0]
        vittorie_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[1]
        proporzione_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[2]

        partite_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[0]
        vittorie_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[1]
        proporzione_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[2]

        return partite_dentro_casa, vittorie_dentro_casa, proporzione_dentro_casa, partite_fuori_casa, vittorie_fuori_casa, proporzione_f

    def pareggi(self):

        def calcoli(posto, score1, score2):
            df_goal = self.data[['home_team', 'away_team', 'home_team_score', 'away_team_score', 'tournament']]
            totale_di_partite = df_goal[df_goal[posto] == self.paese]
            pareggi = df_goal[(df_goal[posto] == self.paese) & (self.data[score1] == self.data[score2])]

            pareggi = valore_abbreviato(str(len(pareggi)))

            # Proporzione
            proporzione = round((float(pareggi) / len(totale_di_partite)) * 100, 2)
            return len(totale_di_partite), pareggi, proporzione

        partite_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[0]
        pareggi_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[1]
        proporzione_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[2]

        partite_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[0]
        pareggi_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[1]
        proporzione_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[2]

        return partite_dentro_casa, pareggi_dentro_casa, proporzione_dentro_casa, partite_fuori_casa, pareggi_fuori_casa, proporzione_fuo

    def sconfitte(self):

        def calcoli(posto, score1, score2):
            df_goal = self.data[['home_team', 'away_team', 'home_team_score', 'away_team_score', 'tournament']]
            totale_di_partite = df_goal[df_goal[posto] == self.paese]
            sconfitte = df_goal[(df_goal[posto] == self.paese) & (self.data[score1] < self.data[score2])]

            sconfitte = valore_abbreviato(str(len(sconfitte)))

            # Proporzione
            proporzione = round((float(sconfitte) / len(totale_di_partite)) * 100, 2)
```

```
            return len(totale_di_partite), sconfitte, proporzione

        partite_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[0]
        sconfitte_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[1]
        proporzione_dentro_casa = calcoli('home_team', 'home_team_score', 'away_team_score')[2]

        partite_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[0]
        sconfitte_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[1]
        proporzione_fuori_casa = calcoli('away_team', 'away_team_score', 'home_team_score')[2]

        return partite_dentro_casa, sconfitte_dentro_casa, proporzione_dentro_casa, partite_fuori_casa, sconfitte_fuori_casa, proporzione

################################################################################################################################################
# ------------------------- Prestazioni generali
################################################################################################################################################

print('\n')
print('-------------------------------------------------------------')
italia = Prestazioni(paese_da_cercare, df)
print(f'{italia.paese} ha avuto {italia.generale()} partite.\n')

# Dentro casa
print(f'{italia.vittorie()[0]} partite, {italia.vittorie()[1]} vittorie e {italia.vittorie()[2]}% di prestazione -- Dentro casa --')
# Fuori casa
print(f'{italia.vittorie()[3]} partite, {italia.vittorie()[4]} vittorie e {italia.vittorie()[5]}% di prestazione -- Fuori casa --\n')

# Dentro casa
print(f'{italia.pareggi()[0]} partite, {italia.pareggi()[1]} pareggi e {italia.pareggi()[2]}% di prestazione -- Dentro casa --')
# Fuori casa
print(f'{italia.pareggi()[3]} partite, {italia.pareggi()[4]} pareggi e {italia.pareggi()[5]}% di prestazione -- Fuori casa --\n')

# Dentro casa
print(f'{italia.sconfitte()[0]} partite, {italia.sconfitte()[1]} sconfitte e {italia.sconfitte()[2]}% di prestazione -- Dentro casa --')
# Fuori casa
print(f'{italia.sconfitte()[3]} partite, {italia.sconfitte()[4]} sconfitte e {italia.sconfitte()[5]}% di prestazione -- Fuori casa --\n')

################################################################################################################################################
# ------------------------- Calcoli del torneo
################################################################################################################################################

def comportamento_in_un_torneo(paese, torneo:str, vedere: int):

    torneo_df = df[df['tournament'] == torneo]
    italia = Prestazioni(paese_da_cercare, torneo_df)

    lista = [italia.vittorie()[0], italia.vittorie()[1], italia.vittorie()[2], italia.vittorie()[3], italia.vittorie()[4], italia.vittori
             italia.pareggi()[0], italia.pareggi()[1], italia.pareggi()[2], italia.pareggi()[3], italia.pareggi()[4], italia.pareggi()[5]
             italia.sconfitte()[0], italia.sconfitte()[1], italia.sconfitte()[2], italia.sconfitte()[3], italia.sconfitte()[4], italia.sc

    if vedere == 1:
        #
        print('-------------------------------------------------------------------------')
        print(f'{paese} nel torneo ### {torneo.upper()} ### ha avuto {italia.generale()} partite:')
        print('-------------------------------------------------------------------------')

        print(f'{italia.vittorie()[0]} partite, {italia.vittorie()[1]} vittorie e {italia.vittorie()[2]}% di prestazione -- Dentro casa')
        print(f'{italia.vittorie()[3]} partite, {italia.vittorie()[4]} vittorie e {italia.vittorie()[5]}% di prestazione -- Fuori casa\n'

        print(f'{italia.pareggi()[0]} partite, {italia.pareggi()[1]} pareggi e {italia.pareggi()[2]}% di prestazione -- Dentro casa')
        print(f'{italia.pareggi()[3]} partite, {italia.pareggi()[4]} pareggi e {italia.pareggi()[5]}% di prestazione -- Fuori casa\n')

        print(f'{italia.sconfitte()[0]} partite, {italia.sconfitte()[1]} sconfitte e {italia.sconfitte()[2]}% di prestazione -- Dentro ca
        print(f'{italia.sconfitte()[3]} partite, {italia.sconfitte()[4]} sconfitte e {italia.sconfitte()[5]}% di prestazione -- Fuori cas

    return lista

# FIFAWorldCup Friendly CONCACAFNationsLeague UEFAEuroqualification FIFAWorldCupqualification CopaAmérica UEFAEuro
torneo_europa = ['FIFAWorldCup', 'Friendly', 'UEFAEuroqualification', 'FIFAWorldCupqualification', 'CopaAmérica', 'UEFAEuro']


for x in torneo_europa:
    try: # Dentro casa
        vittorie_casa = comportamento_in_un_torneo(paese_da_cercare, x, 0)[2]
        pareggi_casa = comportamento_in_un_torneo(paese_da_cercare, x, 0)[8]
        sconfitte_casa = comportamento_in_un_torneo(paese_da_cercare, x, 0)[14]

        vittorie_fuori = comportamento_in_un_torneo(paese_da_cercare, x, 0)[5]
        pareggi_fuori = comportamento_in_un_torneo(paese_da_cercare, x, 0)[11]
        sconfitte_fuori = comportamento_in_un_torneo(paese_da_cercare, x, 0)[17]

        vittorie = round(vittorie_casa + vittorie_fuori, 2)
        pareggi = round(pareggi_casa + pareggi_fuori, 2)
        sconfitte = round(sconfitte_casa + sconfitte_fuori, 2)
```

```
        comportamento_in_un_torneo(paese_da_cercare, x, 1)

        if (vittorie > pareggi) & (pareggi > sconfitte):
            ordine = [vittorie, pareggi, sconfitte]
            print(f'\nPrestazioni del torneo: {ordine}')

    except ZeroDivisionError:
        print('\n>>> ATTENZIONE!')
        print(f'    Mi sa che {paese_da_cercare} non ha mai partecipato al torneo: {x}.\n')

#########################################################################################################################################################
# ------------------------- Medie di gol ad ogni torneo
#########################################################################################################################################################

def media_goal(torneo:str, vedere: int):
    torneo_df = df[df['tournament'] == torneo]
    totale, _ = torneo_df.shape
    goal_1 = torneo_df[['home_team_score']].sum()
    goal_2 = torneo_df[['away_team_score']].sum()

    goal_1 = goal_1.values.tolist()
    goal_2 = goal_2.values.tolist()
    #print(goal_1[0], goal_2[0])
    #print(goal_1[0] + goal_2[0])
    #print(totale)

    media = (goal_1[0] + goal_2[0]) / totale

    return round(media, 2)

print('\n')
for i in torneo_europa:
    print(f'In {i}: media di gol {media_goal(i, 0)}')
print('\n')

#########################################################################################################################################################
# ------------------------- FIFA RANK: Come si è comportata la nazionale
#########################################################################################################################################################

def fifa_rank(paese):
    # Prendendo le partite giocate dalla squandra
    rank = df[['date', 'home_team', 'away_team', 'home_team_fifa_rank', 'away_team_fifa_rank']]
    rank = rank[(rank['home_team'] == paese) | (rank['away_team'] == paese)]

    # Giocate dentro casa
    home_team_fifa_rank = rank[['date', 'home_team', 'home_team_fifa_rank']] # home_team_fifa_rank # home_team_score
    home_team_fifa_rank = home_team_fifa_rank[home_team_fifa_rank['home_team'] == paese]
    home_team_lista = home_team_fifa_rank.values.tolist()

    # Giocate fuori casa
    away_team_fifa_rank = rank[['date', 'away_team', 'away_team_fifa_rank']] # away_team_fifa_rank # away_team_score
    away_team_fifa_rank = away_team_fifa_rank[away_team_fifa_rank['away_team'] == paese]
    away_team_lista = away_team_fifa_rank.values.tolist()

    # Lista principale (la somma delle partite dentro e fuori casa)
    lista_principale = home_team_lista + away_team_lista

    return lista_principale

lista = fifa_rank(paese_da_cercare)

anni = []

for x in lista:
    anno = x[0]
    anno = anno.split(sep='-')[0]
    anni.append(anno)

anni = list(set(anni))
anni.sort()

def media_punteggi_degli_anni(tutto: list, gli_anni: list):
    media_degli_anni = []
    for anno in gli_anni:
        punteggi = []
        for x in tutto:
            a = x[0]
            a = a.split(sep='-')[0]
            p = x[2]
            if a == anno:
                punteggi.append(p)
        media_punteggi = round((reduce(lambda x, y: x + y, punteggi)) / len(punteggi), 2)
```

```
        media_degli_anni.append(media_punteggi)
    return media_degli_anni

medie = media_punteggi_degli_anni(lista, anni)

media_del_periodo = round(reduce(lambda x, y: x + y, medie) / len(medie), 2)
print(f">>> {paese_da_cercare} ha preso {media_del_periodo} come punteggio medio nell'ultimo periodo.\n")

pygame.init()

pygame.mixer.music.load('1109.mp3')
pygame.mixer.music.play()
pygame.event.wait()

#########################################################################################################################################################
# ------------------------- FIFA RANK: La classifica
#########################################################################################################################################################

migliori1 = df[['home_team', 'home_team_fifa_rank']]
migliori1 =  migliori1[((migliori1['home_team_fifa_rank'] > 0) & (migliori1['home_team_fifa_rank'] < 16))]
migliori1 = migliori1.values.tolist()

migliori2 = df[['away_team', 'away_team_fifa_rank']]
migliori2 =  migliori2[((migliori2['away_team_fifa_rank'] > 0) & (migliori2['away_team_fifa_rank'] < 16))]
migliori2 = migliori2.values.tolist()

migliori = migliori1 + migliori2
nazionali = []

for x in migliori:
    nazionali.append(x[0])

nazionali = list(set(nazionali))

def calcola_classifica(anni, nazionali, periodo:None):
    classifica = []
    for nazionele in nazionali:
        lista_del_paese = fifa_rank(nazionele)
        if periodo != None:
            inizio_del_periodo = periodo.split(sep='-')[0]
            fine_del_periodo = periodo.split(sep='-')[1]
            periodo_lista = []
            for x in lista_del_paese:
                a = x[0]
                a = a.split(sep='-')[0]
                if  (a >= inizio_del_periodo) & (a <= fine_del_periodo):
                    periodo_lista.append(x)
            lista_del_paese = periodo_lista
            anni = list(range(int(inizio_del_periodo), int(fine_del_periodo)))
            lista_stringhe = []
            for elemento in anni:
                elemento_stringa = str(elemento)
                lista_stringhe.append(elemento_stringa)
            anni = lista_stringhe
        try:
            medie_del_paese = media_punteggi_degli_anni(lista_del_paese, anni)
        except TypeError:
            continue
        media_del_paese_nel_periodo = round(reduce(lambda x, y: x + y, medie_del_paese) / len(medie_del_paese), 2)
        string = [nazionele, media_del_paese_nel_periodo]
        classifica.append(string)
    return classifica

classifica = calcola_classifica(anni, nazionali, None) # 2010-2013

def lista_in_ordine(lista:list, indice:int, smistaggio:None) -> list:
    lista_ordinata = sorted(lista, key=lambda x: x[indice])
    if smistaggio != None:
        ordine = 0
        for x in lista_ordinata:
            ordine += 1
            print(f'{ordine}° {x[0]} con il {x[indice]}')
    return lista_ordinata

l = lista_in_ordine(classifica, 1, 1)

fine = time.time()

print('\nEnzo Schitini')

# Ottieni il sistema operativo corrente
os_name = platform.system()
```

```
# Ottieni la data corrente
now = datetime.datetime.now()

# Stampa la data nel formato "GGG GG GG"
print("Spogliatoio @ADIGE -", now.strftime("%d %B %Y"), "- Il sistema operativo in uso è:", os_name)

# Stampa la data e l'orario
print("La data e l'orario correnti sono:", now, "\n")
print(f'Durata: {round((fine - avvio), 2)}s\n')

# Fornendo il grafico
graficos(anni, medie, 'blue', 'black', 'Italia')

# Enzo Schitini
```