Branch: master ▾ Introduction-To-Data-Science-In-Python / Assignment+2.py

Find file Copy path

Satya Assignment 2 88888f4 on Aug 1, 2018

0 contributors

204 lines (148 sloc) | 7.48 KB

Raw Blame History

```
# coding: utf-8

# ---
#
# _You are currently looking at **version 1.2** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter noteb
#
# ---

# # Assignment 2 - Pandas Introduction
# All questions are weighted the same in this assignment.
# ## Part 1
# The following code loads the olympics dataset (olympics.csv), which was derrived from the Wikipedia entry on [All Time Olympic Games Meda
#
# The columns are organized as # of Summer games, Summer medals, # of Winter games, Winter medals, total # number of games, total # of meda

# In[5]:

import pandas as pd

df = pd.read_csv('olympics.csv', index_col=0, skiprows=1)

for col in df.columns:
    if col[:2]=='01':
        df.rename(columns={col:'Gold'+col[4:]}, inplace=True)
    if col[:2]=='02':
        df.rename(columns={col:'Silver'+col[4:]}, inplace=True)
    if col[:2]=='03':
        df.rename(columns={col:'Bronze'+col[4:]}, inplace=True)
    if col[:1]=='№':
        df.rename(columns={col:'#'+col[1:]}, inplace=True)

names_ids = df.index.str.split('\s\(') # split the index by '('
#names_ids
df.index = names_ids.str[0] # the [0] element is the country name (new index)
df['ID'] = names_ids.str[1].str[:3] # the [1] element is the abbreviation or ID (take first 3 characters from that)

df = df.drop('Totals')
df.head(10)


# ### Question 0 (Example)
#
# What is the first country in df?
#
# *This function should return a Series.*

# In[6]:

# You should write your whole answer within the function provided. The autograder will call
# this function and compare the return value against the correct solution value
def answer_zero():
    # This function returns the row for Afghanistan, which is a Series object. The assignment
    # question description will tell you the general format the autograder is expecting
    return df.iloc[0]

# You can examine what your function returns by calling it in the cell. If you have questions
# about the assignment formats, check out the discussion forums for any FAQs
answer_zero()


```

```
# ### Question 1
# Which country has won the most gold medals in summer games?
#
# *This function should return a single string value.*

# In[7]:

def answer_one():
    max_gold = df['Gold'].max()
    df1 = df[df['Gold']==max_gold]
    return str(df1.iloc[0].name)


# ### Question 2
# Which country had the biggest difference between their summer and winter gold medal counts?
#
# *This function should return a single string value.*

# In[8]:

def answer_two():
    #import numpy as np
    diff = df['Gold'] - df['Gold.1']
    diff=diff.abs()
    res=diff.max()
    diff=diff[diff==res]
    return str(diff.index.values[0])


# ### Question 3
# Which country has the biggest difference between their summer gold medal counts and winter gold medal counts relative to their total gold
#
# $$\frac{Summer~Gold - Winter~Gold}{Total~Gold}$$
#
# Only include countries that have won at least 1 gold in both summer and winter.
#
# *This function should return a single string value.*

# In[96]:

def answer_three():
    res=df[(df['Gold']>0) & (df['Gold.1']>0)]
    res = (res['Gold']-res['Gold.1'])/(res['Gold'] + res['Gold.1'] + res['Gold.2'])
    res=res[res==res.max()]
    return res.index.values[0]

answer_three()


# ### Question 4
# Write a function that creates a Series called "Points" which is a weighted value where each gold medal (`Gold.2`) counts for 3 points, si
#
# *This function should return a Series named `Points` of length 146*

# In[10]:

def answer_four():
    res = df['Gold.2']*3 + df['Silver.2']*2 + df['Bronze.2']
    res.name = 'Points'
    return res


# ## Part 2
# For the next set of questions, we will be using census data from the [United States Census Bureau](http://www.census.gov). Counties are p
#
# The census dataset (census.csv) should be loaded as census_df. Answer questions using this as appropriate.
#
# ### Question 5
# Which state has the most counties in it? (hint: consider the sumlevel key carefully! You'll need this for future questions too...)
#
# *This function should return a single string value.*

# In[11]:

census_df = pd.read_csv('census.csv')
census_df.head()


# In[12]:

def answer_five():
    res=census_df[census_df['SUMLEV']==50]
    res=res.groupby('STNAME').count()['SUMLEV']
    return res[res==res.max()].index.values[0]


# ### Question 6
# **Only looking at the three most populous counties for each state**, what are the three most populous states (in order of highest populat
#
# *This function should return a list of string values.*
```

```
# In[39]:

def answer_six():
    res=census_df[['SUMLEV', 'STNAME', 'CENSUS2010POP']]
    res=res[res['SUMLEV']==50]
    res=res.sort(columns=['STNAME', 'CENSUS2010POP'], ascending=[True,False])
    res=res.groupby('STNAME').head(3)
    res=res.groupby('STNAME').sum()
    res=res.nlargest(3,'CENSUS2010POP')
    return list(res.index.values)


# ### Question 7
# Which county has had the largest absolute change in population within the period 2010-2015? (Hint: population values are stored in column
#
# e.g. If County Population in the 5 year period is 100, 120, 80, 105, 100, 130, then its largest change in the period would be |130-80| =
#
# *This function should return a single string value.*

# In[98]:

def answer_seven():
    import numpy as np
    res=census_df[census_df['SUMLEV']==50]
    res=res[['CTYNAME','POPESTIMATE2010', 'POPESTIMATE2011','POPESTIMATE2012','POPESTIMATE2013','POPESTIMATE2014','POPESTIMATE2015']]
    max_p=map(max,res['POPESTIMATE2010'],res['POPESTIMATE2011'],res['POPESTIMATE2012'],res['POPESTIMATE2013'],res['POPESTIMATE2014'],res['P
    max_p=np.array(list(max_p))
    min_p=map(min,res['POPESTIMATE2010'],res['POPESTIMATE2011'],res['POPESTIMATE2012'],res['POPESTIMATE2013'],res['POPESTIMATE2014'],res['P
    min_p=np.array(list(min_p))
    chg=max_p-min_p
    return res.iloc[chg.argmax()].values[0]


# ### Question 8
# In this datafile, the United States is broken up into four regions using the "REGION" column.
#
# Create a query that finds the counties that belong to regions 1 or 2, whose name starts with 'Washington', and whose POPESTIMATE2015 was
#
# *This function should return a 5x2 DataFrame with the columns = ['STNAME', 'CTYNAME'] and the same index ID as the census_df (sorted asce

# In[94]:

def answer_eight():
    res=census_df[(census_df['SUMLEV']==50) & (census_df['REGION'] <= 2) & (census_df['CTYNAME'].str.match('^Washington')) & (census_df['PO
    return res[['STNAME',  'CTYNAME']].sort_index()


# In[ ]:
```