Import pandas as pd

import pandas as pd

pd.set\_option('max\_rows', 5)

from learntools.core import binder; binder.bind(globals())

from learntools.pandas.creating\_reading\_and\_writing import \*

print("Setup complete.")

Series vs DataFrame

-Fruits = pd.DataFrame([[30,21]],columns = [‘Banana’,’Apple])

print (Fruits)

-fruit\_sales = pd.DataFrame([[35, 21], [41, 34]], columns=['Apples', 'Bananas'],

index=['2017 Sales', '2018 Sales'])

-ingredients = pd.Series(['4 cups','1 cup', '2 large', '1 can'], index = ['Flour','Milk','Eggs','Spam',], name = 'Dinner')

-reviews = pd.read\_csv('../input/wine-reviews/winemag-data\_first150k.csv',index\_col = 0) sep = \’t’

Saves the panda exercise as a CSV file

-animals.to\_csv('cows\_and\_goats.csv')

Connecting a SQL script to Pandas

import sqlite3

conn = sqlite3.connect('../input/pitchfork-data/database.sqlite')

music\_reviews = pd.read\_sql\_query('SELECT \* FROM artists', conn)

q6.check()

music\_reviews

first\_description = reviews.description.iloc[0]

q2.check()

first\_description

#Check the first value

first\_row = reviews.iloc[0]

#Select the first row of the data set

first\_descriptions = reviews.description.iloc[:10]

Show data as Series

indices = [1,2,3,5,8]

sample\_reviews = reviews.loc[indices]

q5.check()

Sample\_reviews

# Display indices 1,2,3,4,5,8 from the Dataset

col = ['country','province','region\_1','region\_2']

indices = [0,1,10,100]

df = reviews.loc[indices,col]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| country | province | region\_1 | region\_2 |  |
| 0 | Italy | Sicily & Sardinia | Etna | NaN |
| 1 | Portugal | Douro | NaN | NaN |
| 10 | US | California | Napa Valley | Napa |
| 100 | US | New York | Finger Lakes | Finger Lakes |

Create a new view

q6.check()

df

italian\_wines = reviews[reviews.country == 'Italy']

Bring only the wines that are from Italy

top\_oceania\_wines = reviews[(reviews.country.isin(['Australia','New Zealand'])) & (reviews.points >= 95)]

q9.check()

top\_oceania\_wines

Create a DataFrame `top\_oceania\_wines` containing all reviews with at least 95 points (out of 100) for wines from Australia or New Zealand.

cols = ['country','variety']

df = reviews.head(100).loc[:,cols]

q7.check()

df

Correct

|  |  |  |
| --- | --- | --- |
|  | country | variety |
| 0 | Italy | White Blend |
| 1 | Portugal | Portuguese Red |
| ... | ... | ... |
| 98 | Italy | Sangiovese |
| 99 | US | Bordeaux-style Red Blend |

**Summary functions and maps workbook**

* median\_points = reviews.points.median()
* countries = reviews.country.unique()
* reviews\_per\_country = (reviews.country.value\_counts())
* centered\_price = (reviews.price - reviews.price.mean())
* bargain\_idxmax = (reviews.points / reviews.price).idxmax() # Best price - reviews

bargain\_wine = reviews.loc[bargain\_idxmax,'title']

* n\_trop = reviews.description.map(lambda desc: "tropical" in desc).sum() - Count how **many times a word shows in the description**

n\_fruity = reviews.description.map(lambda desc: "fruity" in desc).sum()

descriptor\_counts = pd.Series([n\_trop, n\_fruity], index=['tropical', 'fruity'])

* Pushing reviews on a new column

def stars(row):

if row.country == 'Canada':

return 3

elif row.points >= 95:

return 3

elif row.points >= 85:

return 2

else:

return 1

star\_ratings = reviews.apply(stars, axis = 'columns')

print (star\_ratings)

* check\_reviews = reviews.groupby('taster\_twitter\_handle').size()

# Group the highest reviewer

* best\_rating\_per\_price = reviews.groupby('price')['points'].max().sort\_index()

Sort based on price

* price\_extreme = reviews.groupby('variety').price.agg([min,max])

Group by variety and check min and max prices.

* sorted\_varieties = price\_extremes.sort\_values(by=['min', 'max'], ascending=False)

Sort varieties based on price

* reviewer\_mean\_ratings = round(reviews.groupby('taster\_name').points.mean())

Sort by points

* country\_variety\_counts = reviews.groupby(['country','variety']).size().sort\_values(ascending = False)
* Check type of data

dtype = reviews.points.dtype

* point\_strings = reviews.points.astype(str) Change data to string
* n\_missing\_prices = reviews.price.isnull().sum() check is null values
* reviews\_per\_region = reviews.region\_1.fillna('Unknown').value\_counts().sort\_values(ascending = False)

Filling up the Null values

* Rename columns in Pandas

renamed = reviews.rename(columns = dict(region\_1 = 'region', region\_2 = 'locale'))

* Change the index name of a dataset

reindexed = reviews.rename\_axis('wines')

Concat function - Pandas

gaming\_products = pd.read\_csv("../input/things-on-reddit/top-things/top-things/reddits/g/gaming.csv")

gaming\_products['subreddit'] = "r/gaming"

movie\_products = pd.read\_csv("../input/things-on-reddit/top-things/top-things/reddits/m/movies.csv")

movie\_products['subreddit'] = "r/movies"

combined\_products **=** pd.concat([gaming\_products, movie\_products])

**Join**

powerlifting\_meets = pd.read\_csv("../input/powerlifting-database/meets.csv")

powerlifting\_competitors = pd.read\_csv("../input/powerlifting-database/openpowerlifting.csv")

**powerlifting\_combined = powerlifting\_meets.set\_index('MeetID').join(powerlifting\_competitors.set\_index('MeetID'))**

**print (powerlifting\_combined)**

Generate a list with the (a,b,c,d,e,f) , randint 8 - size 30

**ser = pd.Series(np.take(list('abcdefgh'), np.random.randint(8, size=30)))**

**final\_count = ser.value\_counts()**

**print (ser)**

**print (final\_count)**

**Check the position of ser1 on ser2**

**[pd.Index(ser1).get\_loc(i) for i in ser2]**

**Slice (Make dataframe smaller)**

**new\_table = df.loc[:,'school':'guardian']**

**Change the first letter of a string**

**capitalize = lambda x: x.capitalize()**

**print(new\_table['Mjob'].apply(capitalize))**

**Create a new column conditional based on a condition - Pandas**

**df['mayority'] = np.where(df['age'] >= 18 , True , False)**

**Multiply every number by 10**

**def multiply(x):**

**if type (x) is int:**

**return x \* 10**

**print (df.applymap(multiply).head(5))**

**Check the type of the dataset**

**print(df.info())**

**Change the date of a Dataset**

**df['Year'] = pd.to\_datetime(df['Year'],format = '%Y')**

**Change the index**

**df.set\_index('Year', drop = True)**

**Delete a column**

**df.drop('Total',axis=1, inplace=True)**

**Sum Decades**

**#Step 8. Group the year by decades and sum the values**

**# Uses resample to sum each decade**

**crimes = df.resample('10AS').sum()**

**# Uses resample to get the max value only for the "Population" column**

**population = df['Population'].resample('10AS').max()**

**# Updating the "Population" column**

**df['Population'] = population**

**crime.idxmax(0)**

**Rename columns**

**data\_set.rename(columns = {0:'bedrs', 1: 'bathrs', 2: 'price\_sqr\_meter'}, inplace= True)**

**Print column teams that scored more than 6 goals.**

**print (euro12[euro12.Goals > 6])**

**Print a column that starts with the letter G**

**(euro12[euro12.Team.str.startswith('G')])**

**Filter by two conditions**

**army[(army['deaths']> 500) | (army['deaths'] < 50)]**

**Between 500 and 50**

**Filter a range that doesn’t include a word**

**army[(army['regiment'] != 'Dragoons')]**

**Check the mean of a column based on a criteria**

**print(regiment[regiment['regiment'] == 'Nighthawks'].groupby('regiment').mean())**

**Group two conditions based on one criteria**

**print(regiment.groupby(['regiment','company']).preTestScore.mean())**

**Group two conditions not following hierarchical sorting**

**print(regiment.groupby(['regiment','company']).preTestScore.mean().unstack())**

**Iterate over a group and print the name and the whole data from the regiment**

**for name, group in regiment.groupby('regiment'):**

**print(name)**

**print(group)**

**for name,group in regiment.groupby('company'):**

**print(group)**

dtype: int64

regiment company name preTestScore postTestScore

0 Nighthawks 1st Miller 4 25

1 Nighthawks 1st Jacobson 24 94

4 Dragoons 1st Cooze 3 70

5 Dragoons 1st Jacon 4 25

8 Scouts 1st Sloan 2 62

9 Scouts 1st Piger 3 70

regiment company name preTestScore postTestScore

2 Nighthawks 2nd Ali 31 57

3 Nighthawks 2nd Milner 2 62

6 Dragoons 2nd Ryaner 24 94

7 Dragoons 2nd Sone 31 57

10 Scouts 2nd Riani 2 62

11 Scouts 2nd Ali 3 70

**Change the format of a date from INT64 TO DATETIME64**

crime.Year = pd.to\_datetime(crime.Year, format = '%Y')

**Set a column as the index**

crime.set\_index('Year', drop = True)

**Delete a specific column**

crime.drop(crime.columns[2], axis = 1)

del crime['Total']

**Group by Decade**

print(crime.groupby(pd.Grouper(key = 'Year', freq= '10AS')).sum())

**Slice some columns**

df.loc[:,'school':'guardian']

**Step 6. Capitalize both Mjob**

capitalizer = lambda x: x.capitalize()

df['Mjob'].apply(capitalizer)

**Step 6. Create a Function that checks if a**

#Create a function called majority that return a boolean value to a new column called legal\_drinker (Consider majority as older than 17 years old)

def mayority(x):

if x >= 17:

return True

else:

return False

df['legal\_drinker'] = df['age'].apply(mayority)

**Step 7 multiply everything by 10**

#Multiply every number of the dataset by 10.

def multiply(x):

if type(x) == int:

return x \* 10

return x

df.applymap(multiply)

**Create a price list and group orders vs quantity and order\_id**

#Create a scatterplot with the number of items orderered per order price

chipo.item\_price = [float(value[1:-1])for value in chipo.item\_price]

orders = chipo.groupby('order\_id').sum()

print(orders)

**Filter All positive values and exclude negative values**

final\_check = online\_rt[online\_rt >= 0]

**Re-arrange**

pokemon = pokemon[['name', 'type', 'hp', 'evolution', 'pokedex']]

print(pokemon)

**Showcase only the dates that are at the end of the month.**

apple = apple.resample('BM')

**Substract the first and last date of the dataset**

result = (apple.index.max() - apple.index.min()).days

**Check how many months are in the dataset**

apple\_months = apple.resample('M').mean()

len(apple\_months)

= 404

### **Plot the 'Adj Close' value. Set the size of the figure to 13.5 x 9 inches**

check = apple['Adj Close'].plot(title = "Apple Stock")

### **Start and now date**

import datetime as dt

start = dt.datetime(2015,1,1)

end = dt.datetime.now()

### **Drop NA on a sample**

monthly = monthly.dropna

### **Assign names to columns**

iris.columns = []

### **Count the missing columns**

pd.isnull(iris).sum()

### **Changing the columns to NAN**

iris.iloc[10:30,2] = np.nan

### **Delete rows with NAN**

iris = iris.dropna(how = 'any')

### **Reset Index**

iris.reset\_index(drop = True)

### **Delete columns**

wine = wine.drop(wine.columns[[0,3,6,8,10,12,13]],axis = 1)