FTSE Market Summary & Portfolio Analysis

KATE expects your code to define variables with specific names that correspond to certain things we are interested in.

KATE will run your notebook from top to bottom and check the latest value of those variables, so make sure you don't overwrite them.

- Remember to uncomment the line assigning the variable to your answer and don't change the variable or function names.
- Use copies of the original or previous DataFrames to make sure you do not overwrite them by mistake.

You will find instructions below about how to define each variable.

Once you're happy with your code, upload your notebook to KATE to check your feedback.

About the Dataset

In the following section, you will be analysing **The Financial Times Stock Exchange 100 (FTSE 100) Index** data, pronounced, the 'Footsie hundred', is a share index of the 100 companies listed on the London Stock Exchange with the highest market capitalisation.

The dataset includes information about:

- Company - Name of the publicly traded company
- Ticker - A ticker or stock symbol is an abbreviation used to uniquely identify a stock on a particular stock market
- Sector - Industry in which the company operates in
- Mid-price (p) - Average between the 'buying' and the 'selling' price of a particular stock
- Change - Difference between the current price and the previous day's market price of a particular stock
  - A positive change is what allows investors to make a profit.
- Our view - View of the analyst or the brokerage firm collected this data
- Brokers - Total number of brokerage firms tracking and analysing the stock
- 'Strong Buy', 'Buy', 'Neutral', 'Sell', 'Strong Sell' - columns indicates the weighted verdict of all Brokers
- Recommendation - Final verdict or recommendation for the stock
  - Overweight/Outperform means "buy", investors should assign a higher weighting to the stock in portfolios or funds
  - Underweight/Underperform means "sell" or "don't buy" recommendation for a specific stock

Importing Pandas Library

Make sure you run the following code cell import Pandas Library before you attempt any of the questions

```
In [1]:  import pandas as pd
```

Importing the dataset

First use the `.read_csv()` function to import the file `FTSE100.csv` from the `data` folder, assigning it to `df` DataFrame.

After using the `.head()` method to look at the first five rows of the `df` DataFrame. Also, consider using `.info()` method to further explore your data.

```
In [2]:  df = pd.read_csv('data/FTSE100.csv')
         df.head(10)
```

```
Out[2]:
```

	Company	Ticker	Sector	Mid-price (p)	Change	Our view	Brokers	Strong Buy	Buy	N
0	3i	III	Financial Services	1,099.50	0.0171	Hold	5	NaN	4	
1	Admiral Group	ADM	Nonlife Insurance	2,115.00	-0.42%	Hold	18	NaN	2	
2	Anglo American plc	AAL	Mining	1,744.00	0.0154	Hold	24	NaN	15	
3	Antofagasta	ANTO	Mining	848.2	0.0326	Hold	21	NaN	8	
4	Ashtead Group	AHT	Support Services	2,207.00	0.0347	Buy	17	NaN	12	
5	Associated British Foods	ABF	Food Producers	2,260.00	0.00	Hold	20	NaN	13	
6	AstraZeneca	AZN	Pharmaceuticals & Biotechnology	7,368.00	0.014	Buy	25	NaN	16	
7	Auto Trader Group	AUTO	Media	523.8	-0.42%	Hold	19	NaN	4	
8	Aveva	AVV	Software & Computer Services	3,698.00	-0.38%	Hold	15	NaN	5	
9	Aviva	AV.	Life Insurance	356.4	0.0082	Buy	17	NaN	12	

In [3]:

```
#df.info()

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Company         101 non-null    object 
 1   Ticker          101 non-null    object 
 2   Sector          101 non-null    object 
 3   Mid-price (p)   101 non-null    object 
 4   Change          101 non-null    object 
 5   Our view        101 non-null    object 
 6   Brokers         101 non-null    int64  
 7   Strong Buy      0 non-null      float64
 8   Buy             101 non-null    int64  
 9   Neutral         101 non-null    int64  
 10  Sell            101 non-null    int64  
 11  Strong Sell     101 non-null    int64  
 12  Recommendation  101 non-null    object 
dtypes: float64(1), int64(5), object(7)
memory usage: 10.4+ KB
```

1. Tidy Data

The dataset has a column with only `n/a` values, and also 101 rows (you may have been expecting 100!). This is because one of the companies (Royal Dutch Shell) has two entries. We can get rid of the first instance of these (RDSA).

Starting from a copy of `df`, create a new DataFrame called `clean_df` with the following changes:

- Drop the row with a `Ticker` value of `RDSA`
- Drop the `Strong Buy` column

In [4]:
```
#add your code below

#Make sure you call your new DataFrame clean_df
#clean_df = ...

clean_df = df.copy()
clean_df = clean_df[clean_df["Ticker"] != "RDSA"]
clean_df = clean_df.drop(columns=["Strong Buy"])
clean_df.head()
```

Out[4]:

	Company	Ticker	Sector	Mid-price (p)	Change	Our view	Brokers	Buy	Neutral	Sell	St
0	3i	III	Financial Services	1,099.50	0.0171	Hold	5	4	1	0	
1	Admiral Group	ADM	Nonlife Insurance	2,115.00	-0.42%	Hold	18	2	6	7	
2	Anglo American plc	AAL	Mining	1,744.00	0.0154	Hold	24	15	6	2	
3	Antofagasta	ANTO	Mining	848.2	0.0326	Hold	21	8	9	3	
4	Ashtead Group	AHT	Support Services	2,207.00	0.0347	Buy	17	12	5	0	

2. Change Column Data Type

Look at the values in the `Mid-price (p)` column. At first glance they may look like floats but in fact they have been interpreted as text. We need to change them to floats for them to be more useful.

Starting from a copy of `clean_df`, create a new DataFrame called `price_df` with the following change:

- Convert the values in the `Mid-price (p)` column to floats (keeping the column in the same place)

Hint: converting directly to a float may not work if there are commas in the strings; you may find the [str.replace (https://docs.python.org/3/library/stdtypes.html#str.replace)](https://docs.python.org/3/library/stdtypes.html#str.replace) method useful for fixing this before conversion.

```
In [5]:  #add your code below

         #Make sure you call your new DataFrame price_df
         #price_df = ...
         price_df = clean_df.copy()

         price_df["Mid-price (p)"] = price_df["Mid-price (p)"].str.replace(",","
         price_df["Mid-price (p)"] = price_df["Mid-price (p)"].str.replace(" ",""
         price_df["Mid-price (p)"].describe()
```

```
Out[5]:  count     100.000000
         mean     1866.543800
         std      1922.497883
         min        49.720000
         25%       519.350000
         50%      1088.750000
         75%      2366.000000
         max      7915.000000
         Name: Mid-price (p), dtype: float64
```

3. Format Change Values

Take a look at the values in the `Change` column. You'll see that they are in an inconsistent format, and stored as strings. The positive values need to be multiplied by 100. The negative values need to have the `%` sign removed.

Part 1: Create a function called `format_change` which takes a string such as those in the `Change` column and does the following:

1. If the last character is a % sign, remove it
2. Convert the string to a float
3. If that float is posiive, multiply it by 100
4. Return the resulting float

Hint: to convert string to a float you may find [float() (https://www.w3schools.com/python/ref_func_float.asp)](https://www.w3schools.com/python/ref_func_float.asp) function useful

```
In [6]:  #add your code below
         #def format_change(string):

         def format_change(string):
             if string.endswith("%"):
                 string = string[:-1]

             value = float(string)

             if value > 0:
                 value *= 100
             return value
```

Uncomment and run the following code cell to test that your function works as expected:

In [7]:
```
#format_change('0.45%')
format_change('0.45%')
```
Out[7]:
```
45.0
```

Part 2: Starting from a copy of `price_df`, create a new DataFrame called `change_df` with a new column called `Change (%)`:

- This should contain the result returned from the function created above when given the original `Change` column value as the argument

In [8]:
```
#add your code below

#Make sure you call your new DataFrame change_df
#change_df = ...
change_df = price_df.copy()
change_df["Change (%)"] = change_df["Change"].apply(format_change)
change_df.head()
```
Out[8]:

	Company	Ticker	Sector	Mid-price (p)	Change	Our view	Brokers	Buy	Neutral	Sell	Stro
0	3i	III	Financial Services	1099.5	0.0171	Hold	5	4	1	0	
1	Admiral Group	ADM	Nonlife Insurance	2115.0	-0.42%	Hold	18	2	6	7	
2	Anglo American plc	AAL	Mining	1744.0	0.0154	Hold	24	15	6	2	
3	Antofagasta	ANTO	Mining	848.2	0.0326	Hold	21	8	9	3	
4	Ashtead Group	AHT	Support Services	2207.0	0.0347	Buy	17	12	5	0	

4. Holding Summary

You are given the details of a share holding in a tuple, containing the company ticker code, number of shares, and price paid. Make sure you run the following code cell before attempting the question.

In [9]:
```
holding = ('BP.', 500, 1535.21)

print(holding[0])
print(holding[1])
print(holding[2])
```
```
BP.
500
1535.21
```

Starting from the `holding` above and `change_df`, build a new dictionary containing the following keys and the appropriate values in the given data formats.

```
{
    holding_cost: float,
    # The cost (in £, not pence) of the holding (number of shares *
price paid)
    holding_value: float,
    # The value (in £, not pence) of the holding (number of shares
* current mid-price)
    change_in_value: float,
    # The percentage change from the original cost of the holding t
o the current value
    (e.g. if the holding_value is 1.2 * the holding_cost, the chang
e_in_value should equal 20.0)

}
```

Call this dictionary `holding_dict`

Hint: If you want to get the first item in a series of values (such as a column of a filtered DataFrame), you can use `.values[0]` on the column

In [10]:

```
#add your code below

#Make sure you call your new dictionary holding_dict
#holding_dict = ...
ticker, num_shares, price_paid = holding
current_mid_price = change_df[change_df['Ticker'] == ticker]['Mid-price

holding_cost = num_shares * price_paid
holding_value = num_shares * current_mid_price
change_in_value = ((holding_value - holding_cost)/ holding_cost)*100



holding_dict = {"holding_cost": holding_cost/100 , "holding_value": hold

print(holding_dict)
```

```
{'holding_cost': 7676.05, 'holding_value': 8665.0, 'change_in_value': 1
2.88357944515734}
```

5. Market Comparison

Provided with the DataFrame you processed above, `change_df` , we would like to compare the % change in the mid-price for each company to the average % change for all companies in the market, along with a summary of the broker recommendations.

Create a DataFrame called `comparison_df` with the following columns added to a copy of `change_df` :

- **'Beat Market'** - This should be a Boolean column with `True` for stocks where `Change (%)` exceeds that of the average market change
- **'Buy Ratio'** - This should equal the `Buy` column divided by the `Brokers` column

Hint: Calculate the average market change % first then compare each value in the `Change`

In [11]:

```
#add your code below

#Make sure you call your new DataFrame comparison_df
#comparison_df = ...
comparison_df = change_df.copy()
average_market_change = comparison_df["Change (%)"].mean()
comparison_df["Beat Market"] = comparison_df["Change (%)"] > average_mar
comparison_df["Buy Ratio"] = (comparison_df["Buy"])/comparison_df["Broke
comparison_df.head()
```

Out[11]:

	Company	Ticker	Sector	Mid-price (p)	Change	Our view	Brokers	Buy	Neutral	Sell	Stro
0	3i	III	Financial Services	1099.5	0.0171	Hold	5	4	1	0	
1	Admiral Group	ADM	Nonlife Insurance	2115.0	-0.42%	Hold	18	2	6	7	
2	Anglo American plc	AAL	Mining	1744.0	0.0154	Hold	24	15	6	2	
3	Antofagasta	ANTO	Mining	848.2	0.0326	Hold	21	8	9	3	
4	Ashtead Group	AHT	Support Services	2207.0	0.0347	Buy	17	12	5	0	

6. Investigate

We want to identify any companies which match a given set of rules, so that we can look into them further.

We want to identify companies in `watchlist` that meet at least one of the following requirements:

i) Any company in `watchlist` whose prices is equal to or lower than the given target price.

ii) Any company in `watchlist` where `Buy Ratio` is 0.5 or greater.

Using the `watchlist` below and `comparison_df` you defined, create a list of companies meeting the requirements, call this list `companies_list`. The list should only have the company names, not the price.

Note: **A company meeting both requirements should only appear once in the list**.

Hint: create an empty list to add company names to, using a loop to work through the watchlist. If you want to get the first item in a series of values (such as a column of a filtered DataFrame), you can use `.values[0]` *on the column.*

In [12]:
```
watchlist = [('TUI', 820.0), ('Whitbread', 4300.0), ('AstraZeneca', 7500
             ('Standard Chartered', 920.0), ('Barclays', 135.5)]
watchlist
```

Out[12]:
```
[('TUI', 820.0),
 ('Whitbread', 4300.0),
 ('AstraZeneca', 7500.0),
 ('Standard Chartered', 920.0),
 ('Barclays', 135.5)]
```

In [14]:
```
companies_list = []

for i in watchlist:

  index = comparison_df[comparison_df['Company'].str.contains(i[0])].ind

  df2 = (comparison_df.loc[index, ['Mid-price (p)','Buy Ratio']])

  if (df2.iloc[0, 0]<=i[1]) or (df2.iloc[0,1]>0.5):

    companies_list.append(i[0])

print(companies_list)
```

```
['TUI', 'AstraZeneca', 'Standard Chartered']
```

In [ ]: