Exercises II

Aksel Hiorth

University of Stavanger

Jan 18, 2024

Contents

1 Exercise 1: Create a function for extracting data

Question: Create a function from the following code. The function should take as argument the field name and return a DataFrame with field data. Include a docstring in the function.

```
import pandas as pd
df_prod=pd.read_excel('../data/field_production_gross_monthly.xlsx')
df=df_prod[df_prod['Field (Discovery)'] == 'DRAUGEN']
```

```
def get_data(field):
        """
        Extracts data for a specific field
        """
    df_prod=pd.read_excel('../data/field_production_gross_monthly.xlsx')
        df= df_prod[df_prod['Field (Discovery)'] == field]
        return df

# example of use
df=get_data('DRAUGEN')
```

2 Exercise 2: Improve the previous function

To check if a DataFrame, `df`, contains any data, we can use `df.empty`. If `df.empty` is `True`, there are no data in the DataFrame.

Question: Extend the previous function, by using `df.empty` to give a message if no data are available for a field.

```
def get_data(field):
        """
        Extracts data for a specific field
        """
    df_prod=pd.read_excel('../data/field_production_gross_monthly.xlsx')
        df= df_prod[df_prod['Field (Discovery)'] == field]
    if df.empty:
       print('No data for ', field)
        return df

# example of use
df=get_data('draugen')
```

3 Exercise 3: More improvements

Question: Extend the previous function such that it is case insensitive, i.e. that `get_data('draugen')` would actually return data.

```
def get_data(field):
        """
```

```
        Extracts data for a specific field
        """
    df_prod=pd.read_excel('../data/field_production_gross_monthly.xlsx')
    field = field.upper()
        df= df_prod[df_prod['Field (Discovery)'] == field]
    if df.empty:
       print('No data for ', field)
        return df

# example of use
df=get_data('draugen')
```

4 Exercise 4: Increase speed

The previous function is a bit slow, because for each field we read the big Excel file `field_production_gross_monthly.xlsx` each time. It is much better to read it only once.

Question: Read the data outside the function and pass the production DataFrame, using a default argument.

Solution: Variables defined outside a function in Python is considered as global, thus you do not send them as argument, but it is generally considered bad coding practice to use global variables, hence

```
df_prod=pd.read_excel('../data/field_production_gross_monthly.xlsx')

def get_data(field,df_prod=df_prod):
        """
        Extracts data for a specific field
        """
    field = field.upper()
        df= df_prod[df_prod['Field (Discovery)'] == field]
    if df.empty:
       print('No data for ', field)
        return df
```

```
# example of use
df=get_data('draugen')
df2=get_data('ekofisk')
```

5 Exercise 5: Encapsulate in a class

In the previous exercise we had to use a global variable `df_prod` and then pass it to our function. In many cases it might be much easier to use a class.

Question: Create a class:

1. where the `__init__` function reads in the data
2. and add a class function that returns data for a given field

```
class ProdData:
    """
    A class to extract production data from FactPages
    """
    def __init__(self):
        self.df_prod=pd.read_excel('../data/field_production_gross_monthly.xlsx')

       def get_data(self,field):
        """
        Extracts data for a specific field
        """
        field=field.upper()
        df= self.df_prod[(self.df_prod['Field (Discovery)'] == field)]
        if df.empty:
             print('No data for ', field)
        return df
ff=ProdData()
```

Note that the class contains all the data, we do not have to use a global variable

```
#example of use
df=ff.get_data('draugen')
df2=ff.get_data('ekofisk')
```

References