Assignment 8

August 25, 2023

1 Python Project

1.0.1 Benjamin Michaels

1.0.2 8/25/23

1.0.3 The libraries you will need

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Load in DataFrames

[16]:
```
cost_of_living = pd.read_csv('Python Project/cost_of_living.csv')
ds_salaries = pd.read_csv('Python Project/ds_salaries.csv')
fyi_salary_data=  pd.read_csv('Python Project/Levels_Fyi_Salary_Data.csv')
country_codes = pd.read_excel('Python Project/country_codes.xlsx')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[16], line 1
----> 1 cost_of_living = pd.read_csv('Python Project/cost_of_living.csv')
      2 ds_salaries = pd.read_csv('Python Project/ds_salaries.csv')
      3 fyi_salary_data=  pd.read_csv('Python Project/Levels_Fyi_Salary_Data.csv')

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:211, in␣
 ↪deprecate_kwarg.<locals>._deprecate_kwarg.<locals>.wrapper(*args, **kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:331, in␣
 ↪deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args,␣
 ↪**kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:950, in␣
 ↪read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols ␣
 ↪squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values,␣
 ↪false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values,␣
 ↪keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,␣
 ↪infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates,␣
 ↪iterator, chunksize, compression, thousands, decimal, lineterminator,␣
 ↪quotechar, quoting, doublequote, escapechar, comment, encoding,␣
 ↪encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines,␣
 ↪delim_whitespace, low_memory, memory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:605, in␣
 ↪_read(filepath_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1442, in␣
 ↪TextFileReader.__init__(self, f, engine, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1735, in␣
 ↪TextFileReader._make_engine(self, f, engine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File ~\anaconda3\lib\site-packages\pandas\io\common.py:856, in␣
 ↪get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,␣
 ↪errors, storage_options)
    851 elif isinstance(handle, str):
    852     # Check whether the filename is to be opened in binary mode.
    853     # Binary mode does not support 'encoding' and 'newline'.
    854     if ioargs.encoding and "b" not in ioargs.mode:
    855         # Encoding
--> 856         handle = open(
    857             handle,
    858             ioargs.mode,
    859             encoding=ioargs.encoding,
    860             errors=errors,
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
    865         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'Python Project/
 ↪cost_of_living.csv'
```

Used to find the average salary of a Data Scientist, $108187.83

[ ]:
```
ds_salaries.groupby('job_title')['salary_in_usd'].agg([np.mean])
```

Rename columns to have them merged together

[3]:
```
country_codes.rename(columns = {'Alpha-2 code': 'country_codes'}, inplace = True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 country_codes.rename(columns = {'Alpha-2 code': 'country_codes'}, inplace␣
 ↪= True)

NameError: name 'country_codes' is not defined
```

Merging of dataframes

[4]:
```
salaries_and_codes = pd.merge(ds_salaries, country_codes, how= 'left', on=␣
↪'country_codes')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 salaries_and_codes = pd.merge(ds_salaries, country_codes, how= 'left', on=␣
 ↪'country_codes')

NameError: name 'ds_salaries' is not defined
```

Create a dataframe of average salary per country for a data scientist

[5]:
```
salary_averages = salaries_and_codes.groupby('Country')['salary_in_usd'].mean()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 salary_averages = salaries_and_codes.groupby('Country')['salary_in_usd'].
 ↪mean()

NameError: name 'salaries_and_codes' is not defined
```

Turn the Series into a Dataframe

[6]:
```
salary_averages_data = salary_averages.to_frame()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 salary_averages_data = salary_averages.to_frame()

NameError: name 'salary_averages' is not defined
```

Merging country_codes and salary_averages_data dataframes

[7]:
```
salary_averages_data = pd.merge(salary_averages_data, country_codes, how=␣
 ↪'left', on= 'Country')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 salary_averages_data = pd.merge(salary_averages_data, country_codes, how=␣
 ↪'left', on= 'Country')

NameError: name 'salary_averages_data' is not defined
```

Split City column into two comlumns, city and Country

```
[8]: cost_of_living[['city', 'Country']] = cost_of_living['City'].str.rsplit(',',␣
     ↪n=1, expand=True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[8], line 1
----> 1 cost_of_living[['city', 'Country']] = cost_of_living['City'].str.
 ↪rsplit(',', n=1, expand=True)

NameError: name 'cost_of_living' is not defined
```

Get two dataframes to merge even though they were being difficult, this felt like a round about way of doing so

```
[9]: cost_of_living_average = salary_averages_data.Country.str.split()
     salary_averages_data['Country_new'] = cost_of_living_average.str[:2].str.join('␣
     ↪')
     index = cost_of_living.Country.str.split()
     cost_of_living['Country_new'] = index.str[:2].str.join(' ')

     country_salary_data = pd.merge(salary_averages_data, cost_of_living, how=␣
     ↪'left', on= 'Country_new')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 cost_of_living_average = salary_averages_data.Country.str.split()
      2 salary_averages_data['Country_new'] = cost_of_living_average.str[:2].str.
 ↪join(' ')
      3 index = cost_of_living.Country.str.split()

NameError: name 'salary_averages_data' is not defined
```

Create a salary effectiveness column

```
[10]: country_salary_data["effective_salary"] = country_salary_data['salary_in_usd'] /␣
      ↪country_salary_data['Cost of Living Index']
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 country_salary_data["effective_salary"] =␣
 ↪country_salary_data['salary_in_usd'] / country_salary_data['Cost of Living␣
 ↪Index']
```

```
NameError: name 'country_salary_data' is not defined
```

Sort by salary effectiveness

[11]:
```
country_salary_data.sort_values('effective_salary', ascending=False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 country_salary_data.sort_values('effective_salary', ascending=False)

NameError: name 'country_salary_data' is not defined
```

Get rid of all rows that show nothing in every value

[12]:
```
country_salary_data =␣
 ↪country_salary_data[country_salary_data['effective_salary'].notna()]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 country_salary_data =␣
  ↪country_salary_data[country_salary_data['effective_salary'].notna()]

NameError: name 'country_salary_data' is not defined
```

Select the top 5 places

[13]:
```
top_5_places = country_salary_data[:5]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 top_5_places = country_salary_data[:5]

NameError: name 'country_salary_data' is not defined
```

Get rid of unnecessary columns

[14]:
```
top_5_places = top_5_places.
 ↪drop(['Country_x','salary_in_usd','country_codes','Alpha-3␣
 ↪code','Numeric','Country_new', 'Rank', 'Cost of Living Plus Rent Index'],␣
 ↪axis=1)
```

```
---------------------------------------------------------------------------
```

```
NameError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 top_5_places = top_5_places.
 ↪drop(['Country_x','salary_in_usd','country_codes','Alpha-3␣
 ↪code','Numeric','Country_new', 'Rank', 'Cost of Living Plus Rent Index'],␣
 ↪axis=1)

NameError: name 'top_5_places' is not defined
```

Create finished bar plot

```
[15]: top_5_places.plot(x='City', kind='bar',stacked=True,
                     title= 'Cost of Living')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[15], line 1
----> 1 top_5_places.plot(x='City', kind='bar',stacked=True,
      2                      title= 'Cost of Living')

NameError: name 'top_5_places' is not defined
```

```
[ ]:
```