PLF Session 5 EDA/ File Operations & Exception and Error Handling

UCLA, Econ 10P: Introduction to Python for Economists

1. Install the Python packages **wooldridge** which contains 111 data sets often used in econometrics and **NumPy** which is a library supporting mathematical functions for large, multi-dimensional arrays and matrices.

In [1]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in /Users/paulschumacher/anaconda3/l
ib/python3.10/site-packages (1.25.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install wooldridge
```

```
Defaulting to user installation because normal site-packages is not write
able
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: wooldridge in /home/ca338041-01f5-4955-add
5-d8416d8f23a5/.local/lib/python3.11/site-packages (0.4.4)
Requirement already satisfied: pandas in /opt/conda/envs/anaconda-panel-2
023.05-py310/lib/python3.11/site-packages (from wooldridge) (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/conda/envs/
anaconda-panel-2023.05-py310/lib/python3.11/site-packages (from pandas->w
ooldridge) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/envs/anaconda-p
anel-2023.05-py310/lib/python3.11/site-packages (from pandas->wooldridge)
(2022.7)
Requirement already satisfied: numpy>=1.21.0 in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from pandas->wooldridg
e) (1.24.3)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/anaconda-panel
-2023.05-py310/lib/python3.11/site-packages (from python-dateutil>=2.8.1-
>pandas->wooldridge) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

2. Open the data set 'wage1' from wooldridge!

In [4]:
```
import wooldridge as woo
wage1 = woo.dataWoo('wage1')
```

3. Display the...

- number of rows and columns
- first 5 elements/lines of the data set
- last 5 elements/lines of the data set

- a general overview of the some statistical measures
- all the columns in the data set

In [7]:
```
wage1.shape
```

Out[7]:
```
(526, 24)
```

In [5]:
```
wage1.head()
```

Out[5]:

	wage	educ	exper	tenure	nonwhite	female	married	numdep	smsa	northcen	...	trcommpu
0	3.10	11	2	0	0	1	0	2	1	0	...	0
1	3.24	12	22	2	0	1	1	3	1	0	...	0
2	3.00	11	2	0	0	0	0	2	0	0	...	0
3	6.00	8	44	28	0	0	1	0	1	0	...	0
4	5.30	12	7	2	0	0	1	1	0	0	...	0

5 rows × 24 columns

In [5]:
```
wage1.tail()
```

Out[5]:

	wage	educ	exper	tenure	nonwhite	female	married	numdep	smsa	northcen	...	trcommp
521	15.00	16	14	2	0	1	1	2	0	0	...	
522	2.27	10	2	0	0	1	0	3	0	0	...	
523	4.67	15	13	18	0	0	1	3	0	0	...	
524	11.56	16	5	1	0	0	1	0	0	0	...	
525	3.50	14	5	4	1	1	0	2	0	0	...	

5 rows × 24 columns

In [6]:
```
wage1.describe()
```

Out[6]:

	wage	educ	exper	tenure	nonwhite	female	married	numd
count	526.000000	526.000000	526.00000	526.000000	526.000000	526.000000	526.000000	526.0000
mean	5.896103	12.562738	17.01711	5.104563	0.102662	0.479087	0.608365	1.0437
std	3.693086	2.769022	13.57216	7.224462	0.303805	0.500038	0.488580	1.2618
min	0.530000	0.000000	1.00000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	3.330000	12.000000	5.00000	0.000000	0.000000	0.000000	0.000000	0.0000
50%	4.650000	12.000000	13.50000	2.000000	0.000000	0.000000	1.000000	1.0000
75%	6.880000	14.000000	26.00000	7.000000	0.000000	1.000000	1.000000	2.0000
max	24.980000	18.000000	51.00000	44.000000	1.000000	1.000000	1.000000	6.0000

8 rows × 24 columns

In [8]:
```
wage1.columns
```

Out[8]:
```
Index(['wage', 'educ', 'exper', 'tenure', 'nonwhite', 'female', 'marrie
d',
       'numdep', 'smsa', 'northcen', 'south', 'west', 'construc', 'ndurma
n',
       'trcommpu', 'trade', 'services', 'profserv', 'profocc', 'clerocc',
       'servocc', 'lwage', 'expersq', 'tenursq'],
      dtype='object')
```

File Operations

Reading a .txt File

4. Open the .txt file "'read.txt'". Read each sentence in a single line.

In [34]:
```
f = open('read.txt', "r")
f.read()
```

Out[34]:
```
'Lecture, three hours.\nPython is commonly used programming language for
data science.\nIt is powerful and easy to learn tool that can be applied
to make simple histograms or fit complicated machine learning models.\nIn
troduction to using Python for basic data exploration, analysis, and visu
alization.\nEmphasis on applications with economic data and econometric a
nalysis.\nP/NP grading.\n\n'
```

`readline()` method returns the current line of the file and then advances to the next line

In [33]:
```
f.readlines()
```

Out[33]:
```
['Lecture, three hours.\n',
 'Python is commonly used programming language for data science.\n',
 'It is powerful and easy to learn tool that can be applied to make simpl
e histograms or fit complicated machine learning models.\n',
 'Introduction to using Python for basic data exploration, analysis, and
visualization.\n',
 'Emphasis on applications with economic data and econometric analysi
s.\n',
 'P/NP grading.\n',
 '\n']
```

In [37]:
```
f = open('read.txt', "r")
for line in f:
        print(line, end = "")
```

```
Lecture, three hours.
Python is commonly used programming language for data science.
It is powerful and easy to learn tool that can be applied to make simple
histograms or fit complicated machine learning models.
Introduction to using Python for basic data exploration, analysis, and vi
sualization.
Emphasis on applications with economic data and econometric analysis.
P/NP grading.
```

Writing a new .txt File

Create and open the .txt file "'text.txt'". Write "Hello World!" and close the file. Check you work through reading the newly created file.

In [68]:
```
g = open("text.txt", "w")
```

In [69]:
```
g.write("Hello World!")
```

Out[69]:
```
12
```

Even after calling `.write()`, we won't be able to inspect the new text we've written until we close the connection:

In [70]:
```
g.close()
```

Once we've closed the connection, we can't write any more text to output_file:

In [71]:
```
g = open('text.txt', "r")
g.read()
```

Out[71]:
```
'Hello World!'
```

5. Create a basic function that first multiplies the input by 3 and then devides this result by 2. Once you are done input "1" as an integer and then"one" as a string. The goal is the get an output in the first case and an error in the second one.

In [1]:
```
#Not CLEAN error catch, you get red error (TypeError) text box when you try
def multiply_and_divide(x):
    """
    multiply input by 3 and then divide result by 2
    """

    output = x*3
    output = output/2

    return(output)
```

In [2]:
```
multiply_and_divide(1)
```

Out[2]:
```
1.5
```

In [3]:
```
multiply_and_divide("one")
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Cell In[3], line 1
----> 1 multiply_and_divide("one")

Cell In[1], line 8, in multiply_and_divide(x)
      3 """
      4 multiply input by 3 and then divide result by 2
      5 """
      7 output = x*3
----> 8 output = output/2
     10 return(output)

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

6. Include an if statement in the function to specify the Error and raise and exception for "TypeError". If a TypeError occors display the output: "This function is designed to work only with floats and ints.

In [4]:
```
def multiply_and_divide_advanced(x):
    """
    multiply input by 3 and then divide result by 2
    """

    if type(x) not in [float, int]:
        raise TypeError("This function is designed to work only with floats

    output = x*3
    output = output/2

    return(output)
```

In [5]:
```
multiply_and_divide_advanced("one")
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Cell In[5], line 1
----> 1 multiply_and_divide_advanced("one")

Cell In[4], line 7, in multiply_and_divide_advanced(x)
      2 """
      3 multiply input by 3 and then divide result by 2
      4 """
      6 if type(x) not in [float, int]:
----> 7     raise TypeError("This function is designed to work only with
floats and ints. ")
      9 output = x*3
     10 output = output/2

TypeError: This function is designed to work only with floats and ints.
```

7. Create a try and except statement for the same function but show the error message only. Use "six" as an input.

In [6]:
```
try:
    multiply_and_divide_advanced("six")
except TypeError as e:
    print(e)
```

```
This function is designed to work only with floats and ints.
```

8. Integrate the try and except statement in your function. Run the function with "1" and "one"

In [8]:
```
def multiply_and_divide_more_advanced(x):
    output = x*3
    try:
        output = output/2
    except TypeError:
        print("This function is designed to work with floats and ints, retu
        return x

    return output
```

In [9]:
```
multiply_and_divide_more_advanced(1)
```

Out[9]:
```
1.5
```

In [10]:
```
multiply_and_divide_more_advanced("one")
```

```
This function is designed to work with floats and ints, returning origina
l input instead.
```

Out[10]:
```
'one'
```

Class Examples

In [11]:
```
f = open("raed.txt")
```

```
---------------------------------------------------------------------------
--
FileNotFoundError                         Traceback (most recent call las
t)
Cell In[11], line 1
----> 1 f = open("raed.txt")

File ~/anaconda3/lib/python3.10/site-packages/IPython/core/interactiveshe
ll.py:284, in _modified_open(file, *args, **kwargs)
    277 if file in {0, 1, 2}:
    278     raise ValueError(
    279         f"IPython won't let you open fd={file} by default "
    280         "as it is likely to crash IPython. If you know what you a
re doing, "
    281         "you can use builtins' open."
    282     )
--> 284 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'raed.txt'
```

In [12]:
```
try:
    f = open("raed.txt")
except Exception: #for the remaining errors
    print("Sorry. This file does not exist!.")
#else:
 #   pass
#finally:
 #   pass
```

```
Sorry. This file does not exist!.
```

In [13]:
```
try:
    f = open("read.txt")
    a = b #another error
except FileNotFoundError: #catches a specific error
    print("Sorry. This file does not exist!")
except Exception: #for the remaining errors
    print("Sorry. Something went wrong.")
#else:
 #   pass
#finally:
 #   pass
```

```
Sorry. Something went wrong.
```

In [14]:
```
try:
    f = open("read.txt")
    a = b #another error
except FileNotFoundError as e: #catches a specific error
    print(e)
except Exception as e: #for the remaining errors
    print(e)
#else:
 #   pass
#finally:
 #   pass
```

```
name 'b' is not defined
```

In [15]:
```
try:
    f = open("read.txt")
except FileNotFoundError as e: #catches a specific error
    print(e)
except Exception as e: #for the remaining errors
    print(e)
else: #since we do NOT have an exception, we can run the code
    print(f.read())
    f.close()
#finally:
 #   pass
```

```
Lecture, three hours.
Python is commonly used programming language for data science.
It is powerful and easy to learn tool that can be applied to make simple
histograms or fit complicated machine learning models.
Introduction to using Python for basic data exploration, analysis, and vi
sualization.
Emphasis on applications with economic data and econometric analysis.
P/NP grading.
```

In [16]:
```
try:
    f = open("read.txt")
except FileNotFoundError as e: #catches a specific error
    print(e)
except Exception as e: #for the remaining errors
    print(e)
else: #since we do NOT have an exception, we can run the code
    print(f.read())
    f.close()
finally: #runs no matter what happens
    print('Executing Finally...')
```

```
Lecture, three hours.
Python is commonly used programming language for data science.
It is powerful and easy to learn tool that can be applied to make simple
histograms or fit complicated machine learning models.
Introduction to using Python for basic data exploration, analysis, and vi
sualization.
Emphasis on applications with economic data and econometric analysis.
P/NP grading.


Executing Finally...
```

In [17]:
```
try:
    f = open("read.txt")
    if f.name == "read.txt":
        raise Exception
except FileNotFoundError as e: #catches a specific error
    print(e)
except Exception as e: #for the remaining errors
    print("Error")
else: #since we do NOT have an exception, we can run the code
    print(f.read())
    f.close()
finally: #runs no matter what happens
    print('Executing Finally...')
```

```
Error
Executing Finally...
```