Day - 25

**Working with data:**

**CSV and Pandas**

Working with CSV files and Analyzing Data with PANDAS

**25.1 CSV: Comma Separated Values**

* CSVs are a very common way of representing tabular data, so that, data fits into tables like a spreadsheet. And CSV, stands for Comma Separated Values.
* In this file, each row is a single set of data and each piece of data is separated by a comma WITHOUT space.
* No library: Accessing CSV file data, without using any library

#*No library: Accessing CSV file data, without using any library*

**with** **open**(file= "weather\_data.csv", mode="r") **as** weather\_data:

    day\_list = weather\_data**.readlines**()

**print**(day\_list)

* Returns a list of string. This list needs lots of cleaning.
* CSV python-inbuilt-library: Accessing CSV file data, with default CSV python-library
* The ***csv.reader()*** method:

reader(**csvfile**: Iterable[Text], dialect: Any=..., \*\*fmtparams: Any) -> \_reader

* param csvfile: ***Iterable[Text]***

csv\_reader = **reader**(iterable [, dialect='excel'] [optional keyword args])

* The "***iterable***" argument can be any object that returns a line of input for each iteration, such as a file object or a list.
* The function also accepts optional keyword arguments which override settings provided by the ***dialect***.
* The ***returned*** ***object*** is an ***iterator***. Each ***iteration*** returns a ***row of the CSV file*** (which can span multiple input lines).
* In below, ***data*** is a \_csv.reader object. "dt" are lists. Following retrieve the temperature except the title "temp".

|  |  |
| --- | --- |
| #*CSV python-library: Accessing CSV file data, with default CSV python-library*  **import** csv  temparatures = []  **with** **open**(file= "weather\_data.csv", mode="r") **as** data\_file:      data = csv**.reader**(data\_file)  **for** dt **in** data:  **print**(dt)  **if** dt[1] **!=** "temp":              temparatures**.append**(**int**(dt[1]))    **print**(temparatures) | Weather\_data.csv |
| day,temp,condition  Monday,12,Sunny  Tuesday,14,Rain  Wednesday,15,Rain  Thursday,14,Cloudy  Friday,21,Sunny  Saturday,22,Sunny  Sunday,24,Sunny |

* PANDAS: Accessing CSV file data, with PANDAS library. Work with more complex data.

**pip install pandas**

* Tutorials: You can learn more about pandas in the tutorials, and more about JupyterLab in the JupyterLab documentation.
* Books: The book we recommend to learn pandas is Python for Data Analysis, by Wes McKinney, creator of pandas.

L:\1\_Development\z-codes-2.1-PYTHON\MLp1\_py\_ang\_100dy\_day25\_CSV\_pandas>pip install pandas

Collecting pandas

Downloading pandas-1.1.5-cp36-cp36m-win32.whl (7.8 MB)

|████████████████████████████████| 7.8 MB 726 kB/s

Collecting pytz>=2017.2

Downloading pytz-2021.3-py2.py3-none-any.whl (503 kB)

|████████████████████████████████| 503 kB 656 kB/s

Collecting python-dateutil>=2.7.3

Downloading python\_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)

|████████████████████████████████| 247 kB 1.1 MB/s

Collecting numpy>=1.15.4

Downloading numpy-1.19.5-cp36-cp36m-win32.whl (11.0 MB)

|████████████████████████████████| 11.0 MB 656 kB/s

Collecting six>=1.5

Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)

Installing collected packages: six, pytz, python-dateutil, numpy, pandas

#*PANDAS: Accessing CSV file data, with PANDAS library.*

**import** pandas

data\_2 = pandas**.read\_csv**("weather\_data.csv")

**print**(data\_2)

#*printing temperature: Easyly with pandas*

**print**(data\_2["temp"])

All-code at once

#*No library: Accessing CSV file data, without using any library*

**with** **open**(file= "weather\_data.csv", mode="r") **as** weather\_data:

    day\_list = weather\_data**.readlines**()

**print**(day\_list)

#*CSV python-library: Accessing CSV file data, with default CSV python-library*

**import** csv

temparatures = []

**with** **open**(file= "weather\_data.csv", mode="r") **as** data\_file:

    data = csv**.reader**(data\_file)

**for** dt **in** data:

**print**(dt)

**if** dt[1] **!=** "temp":

            temparatures**.append**(**int**(dt[1]))

**print**(temparatures)

#*PANDAS: Accessing CSV file data, with PANDAS library.*

**import** pandas

data\_2 = pandas**.read\_csv**("weather\_data.csv")

**print**(data\_2)

#*printing temperature: Easyly with pandas*

**print**(data\_2["temp"])

#*python intro\_csv\_panda.py*

Output

L:\1\_Development\z-codes-2.1-PYTHON\MLp1\_py\_ang\_100dy\_day25\_CSV\_pandas>python intro\_csv\_panda.py

['day,temp,condition\n', 'Monday,12,Sunny\n', 'Tuesday,14,Rain\n', 'Wednesday,15,Rain\n', 'Thursday,14,Cloudy\

n', 'Friday,21,Sunny\n', 'Saturday,22,Sunny\n', 'Sunday,24,Sunny']

['day', 'temp', 'condition']

['Monday', '12', 'Sunny']

['Tuesday', '14', 'Rain']

['Wednesday', '15', 'Rain']

['Thursday', '14', 'Cloudy']

['Friday', '21', 'Sunny']

['Saturday', '22', 'Sunny']

['Sunday', '24', 'Sunny']

[12, 14, 15, 14, 21, 22, 24]

day temp condition

0 Monday 12 Sunny

1 Tuesday 14 Rain

2 Wednesday 15 Rain

3 Thursday 14 Cloudy

4 Friday 21 Sunny

5 Saturday 22 Sunny

6 Sunday 24 Sunny

**25.2 Pandas: Intro**

* Pandas data structures: There are two primary data structures of pandas: ***series*** and a ***dataframe***. ***Series (1-dimensional)*** and ***DataFrame (2-dimensional)***, handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.!

|  |  |
| --- | --- |
| **import** pandas  data = pandas**.read\_csv**("weather\_data.csv")  **print**(**type**(data))  **print**(**type**(data["temp"])) | **<class 'pandas.core.frame.DataFrame'>**  **<class 'pandas.core.series.Series'>** |

* ***DataFrame*** and ***series:*** A ***DataFrame*** is kind of the equivalent of your *whole* *table* here. And the ***series*** is basically equivalent to a ***list***. It's kind of like a *single* *column* in your table.
* So the temperature column would be a ***Series***, and the CSV itself is a ***Dataframe***.
* The idea is that the whole table is basically a ***data*** ***frame*** in pandas and every single column is a ***series*** kind of like a ***list*** in pandas, then you're pretty much half of the way there to understanding how this library works.

**import** pandas

data = pandas**.read\_csv**("weather\_data.csv")

# Converting DataFrame to dictionary

data\_dict = data**.to\_dict**()

**print**(data\_dict)

#Converting Series to list

temp\_list = data["temp"]**.to\_list**()

**print**(temp\_list)

**25. 3 Accessing Data using Pandas**

* Pandas Built-in function: Dig through the ***Panadas*** Documentation.

**import** pandas

data = pandas**.read\_csv**("weather\_data.csv")

temp\_list = data["temp"]**.to\_list**()

#*-------- Average Temperatue : Using built-in Python methods "sum" ---------*

average = **sum**(temp\_list)/**len**(temp\_list)

**print**(f"Average Temperatue : Using built-in Python method is   : {average}")

#*-------- Average Temperatue : Using built-in PANDAS method: mean(), median(), mode(), max() etc--------*

average\_pandas = data["temp"]**.mean**()

**print**(f"Average Temp : Using built-in PANDAS method   : {average\_pandas}")

max\_value = data["temp"]**.max**()

**print**(f"Max Temp : Using built-in PANDAS method   : {max\_value}")

* Accessing columns and rows:
* Get data in COLUMNS: the string must match to the column-title

**import** pandas

data = pandas**.read\_csv**("weather\_data.csv")

* Treat as a "Dictionary": **print**(data["condition"])
* Treat as an "Object": Alternative way **print**(data**.**condition)
* Get data in ROWS: Notice all **[]** are used. First with ***DataFrame*** then with ***Series***. We also could use object form,

*# Notice the conditional statement*

**print**(data[data["day"] **==** "Monday"])

Essentially when we get our DataFrame and then we use some ***square*** ***brackets***, and inside those square brackets if we only put the name of our column, day, temp, or condition, then we would get the *entire* *column*. But if we fill to that *column* by a *condition* say when a particular column is equal to a particular value, then we actually get hold of the row instead.

More Examples

#*Print day with maximum teperature*

**print**(data[data["temp"] **==** data["temp"]**.max**()])

#*or Equivalently*

**print**(data[data**.**temp **==** data**.**temp**.max**()])

#*Accessing particular/specific data from a ROW*

mnday = data[data**.**day **==** "Monday"]

**print**(mnday**.**condition)

#*temperature Cel to Ferh*

mnday\_temp = **int**(data[data**.**day **==** "Monday"]**.**temp)

**print**(mnday\_temp\*100)

#*or Equivalently*

temp\_of\_mon = **int**(mnday**.**temp)

ferenheit = (temp\_of\_mon \* (9/5)) + 32

**print**(ferenheit)

* Multiple condition on multiple rows (more at the end):

rOw = data[data**.**day **==** 21][data**.**month **==** 1]

**print**(rOw**.**name)

* The ideal way is: rOw = data[(data**.**day **==** 21) & (data**.**month **==** 1)]
* Transforming data to CSV using Pandas:

#*Create a Dataframe/CSV from scratch*

data\_dict = {

    "students": ["Amy", "James", "Angela"],

    "scores": [76, 56, 65]

}

* Creaings the Dataframe

created\_dt\_frm = pandas**.DataFrame**(data\_dict) #*this creates the Dataframe*

**print**(created\_dt\_frm)

* Creating CSV file

created\_dt\_frm**.to\_csv**("created\_dt\_frm.csv") #*This converts to CSV*

All code at once

**import** pandas

data = pandas**.read\_csv**("weather\_data.csv")

**print**(**type**(data))

**print**(**type**(data["temp"]))

#*Converting DataFrame to dictionary*

data\_dict = data**.to\_dict**()

**print**(data\_dict)

#*Converting Series to list*

temp\_list = data["temp"]**.to\_list**()

**print**(temp\_list)

#*-------- Average Temperatue : Using built-in Python methods "sum" ---------*

average = **sum**(temp\_list)/**len**(temp\_list)

**print**(f"Average Temperatue : Using built-in Python method is   : {average}")

#*-------- Average Temperatue : Using built-in PANDAS method: mean(), median(), mode(), max() etc---------*

average\_pandas = data["temp"]**.mean**()

**print**(f"Average Temp : Using built-in PANDAS method   : {average\_pandas}")

max\_value = data["temp"]**.max**()

**print**(f"Max Temp : Using built-in PANDAS method   : {max\_value}")

#*Get data in COLUMNS: the string must match to the column-title*

**print**(data["condition"]) #*treat as a "Dictionary"*

**print**(data**.**condition)  #*treat as an "Object": Alternative way*

#*Get data in ROWS: Notice all [] arew used. First with DataFrame then with Series*

#*Notice the conditional statement*

**print**(data[data["day"] **==** "Monday"])

#*Print day with maximum teperature*

**print**(data[data["temp"] **==** data["temp"]**.max**()])

#*or Equivalently*

**print**(data[data**.**temp **==** data**.**temp**.max**()])

#*Accessing particular/specific data from a ROW*

mnday = data[data**.**day **==** "Monday"]

**print**(mnday**.**condition)

#*temperature Cel to Ferh*

mnday\_temp = **int**(data[data**.**day **==** "Monday"]**.**temp)

**print**(mnday\_temp\*100)

#*or Equivalently*

temp\_of\_mon = **int**(mnday**.**temp)

ferenheit = (temp\_of\_mon \* (9/5)) + 32

**print**(ferenheit)

#*Create a Dataframe/CSV from scratch*

data\_dict = {

    "students": ["Amy", "James", "Angela"],

    "scores": [76, 56, 65]

}

created\_dt\_frm = pandas**.DataFrame**(data\_dict) #*this creates the Dataframe*

**print**(created\_dt\_frm)

created\_dt\_frm**.to\_csv**("created\_dt\_frm.csv") #*This converts to CSV*

#*More on numPy , Matplotlib and others*

#*python more\_panda\_opr.py*

* Exercise 25.1: The Great Squirrel Census Data Analysis (with Pandas!)

Practice version

**import** pandas

data = pandas**.read\_csv**("2018\_Central\_Park\_Squirrel\_Census\_Squirrel\_Data.csv")

fur\_color = data["Primary Fur Color"]

gray = 0

cinamon = 0

black = 0

**for** squrl\_colr **in** fur\_color:

**if** squrl\_colr **==** "Gray":

        gray += 1

**elif** squrl\_colr **==** "Cinnamon":

        cinamon += 1

**elif** squrl\_colr **==** "Black":

        black += 1

**print**(f"Gray : {gray}, Black {black}, Cinamon : {cinamon}")

result = {

    "Fur Color": ["Gray", "Cinnamon", "Black"],

    "Count": [gray, cinamon, black]

}

result\_csv = pandas**.DataFrame**(result)

**print**(result\_csv)

result\_csv**.to\_csv**("Squirrel\_fur\_color\_count.csv")

#*python squirrel\_data\_pandas.py*

Instructors solution

**import** pandas

data = pandas**.read\_csv**("2018\_Central\_Park\_Squirrel\_Census\_Squirrel\_Data.csv")

#*Access the rows and count them*

gray = **len**(data[data["Primary Fur Color"] **==** "Gray"])

cinamon = **len**(data[data["Primary Fur Color"] **==** "Cinnamon"])

black = **len**(data[data["Primary Fur Color"] **==** "Black"])

result = {

    "Fur Color": ["Gray", "Cinnamon", "Black"],

    "Count": [gray, cinamon, black]

}

result\_csv = pandas**.DataFrame**(result)

**print**(result\_csv)

result\_csv**.to\_csv**("Squirrel\_color\_count.csv")

#*python squirrel\_data\_pandas\_soln.py*

* Exercise 25.2: U.S. States Game

Practice version

**import** turtle

**import** pandas

scrn = turtle**.Screen**()

scrn**.title**("Us State Game")

#*setting the map image*

scrn**.register\_shape**("./blank\_states\_img.gif")

tur = turtle**.Turtle**(shape = "./blank\_states\_img.gif")

scrn**.tracer**(0)

#*Read CSV data*

data = pandas**.read\_csv**("50\_states.csv")

game\_on =**True**

**while** game\_on:

    #*screen.textinput("NIM", "Name of first player:")*

    name = scrn**.textinput**("Input Name", "Enter the name of a state")**.capitalize**()

    coord = data[data**.**state **==** name]

    x = **int**(coord**.**x)

    y = **int**(coord**.**y)

    tur**.speed**("fastest")

    tur**.color**("red")

    tur**.penup**()

    tur**.goto**(x, y)

    tur**.write**(name, align="center", font=('Courier', 10, 'normal'))

    tur**.goto**(0, 0)

    scrn**.update**()

#*python us\_game\_main.py*

Instructor solution

**import** turtle

**import** pandas

scrn = turtle**.Screen**()

scrn**.title**("Us State Game")

image = "./blank\_states\_img.gif"

scrn**.addshape**(image)

turtle**.shape**(image)

scrn**.tracer**(0)

#*Read CSV data*

data = pandas**.read\_csv**("50\_states.csv")

all\_states = data**.**state**.to\_list**()

guessed\_state = []

**while** **len**(guessed\_state) **<** 50 :

    #*screen.textinput("NIM", "Name of first player:")*

    name = (scrn**.textinput**(title= f"{len(guessed\_state)}/50 States Correct",prompt= "Enter the name of a state"))**.title**()

        #*"list" is needed to check "name" inside the "data" so we converted "data.state" to "list"*

**if** name **==** "Exit":

        missed\_states = []

        #*Create a missed states CSV*

**for** staTe **in** all\_states:

**if** staTe **not** **in** guessed\_state:

                missed\_states**.append**(staTe)

**print**(missed\_states)

        new\_data = pandas**.DataFrame**(missed\_states)

        new\_data**.to\_csv**("missed\_states.csv")

**break**

**elif** name **in** all\_states:

        guessed\_state**.append**(name)

        tur = turtle**.Turtle**()

        tur**.speed**("fastest")

        tur**.color**("red")

        tur**.penup**()

        tur**.hideturtle**()

        found\_state = data[data**.**state **==** name]

        tur**.goto**(**int**(found\_state**.**x), **int**(found\_state**.**y))

        tur**.write**(found\_state**.**state**.item**(), align="center", font=('Courier', 10, 'normal'))

        scrn**.update**()

#*python us\_game\_soln.py*

* Coordinate finding(With map)

**import** turtle

scrn = turtle**.Screen**()

scrn**.title**("Us State Game")

image = "./blank\_states\_img.gif"

scrn**.addshape**(image)

turtle**.shape**(image)

tur = turtle**.Turtle**()

#*turtle.onscreenclick(fun, btn=1, add=None)*

    #*Parameters*

        #*fun = a function with two arguments which will be called with the coordinates of the clicked point on the canvas*

        #*btn = number of the mouse-button, defaults to 1 (left mouse button)*

        #*add = True or False . if True, a new binding will be added, otherwise it will replace a former binding*

#*Eg: screen.onclick(turtle.goto)*

**def** **mouse\_click\_coord**(x, y):

**print**(x, y)

**for** i **in** **range**(0, 10):

    scrn**.onscreenclick**(mouse\_click\_coord)

#*follown runs an infinite loop*

turtle**.mainloop**()

#*python mouse\_click.py*

* Stack-overflow code:

**import** turtle

**def** **mouse\_click\_coord**(x, y):

**print**(x, y)

turtle**.onscreenclick**(mouse\_click\_coord)

#*following runs an infinite loop*

turtle**.mainloop**()

#*python mouse\_click\_stack\_overflow.py*

* ***turtle.mainloop()*** is alternative way to not-to disappear screen, because ***scrn.exitonclick()*** make screen disapper when clicked

**pandas.Series.item**

Series.item()

Return the first element of the underlying data as a Python scalar.

**How to filter a pandas DataFrame by multiple columns in Python**

tom\_and\_42 = df[(df["Name"]=="Tom") & (df["Age"]==42)]

tom\_or\_34 = df[(df["Name"]=="Tom") | (df["Age"]==34)]

tom42\_or\_34 = df[((df["Name"]=="Tom") & (df["Age"]<=42)) | (df["Age"]<=34)]

* Check if the row is empty

Use " empty "

rOw = data[(data**.**day **==** 221) & (data**.**month **==** 1)]

**if** rOw**.**empty:

**print**("Empty")

**print**(rOw**.**name)