# Pandas and DataFrames

An alternative that is more accessible then just the csv library

# What will be covered today

- Pandas
- Series
- DataFrames
- Cleaning Data

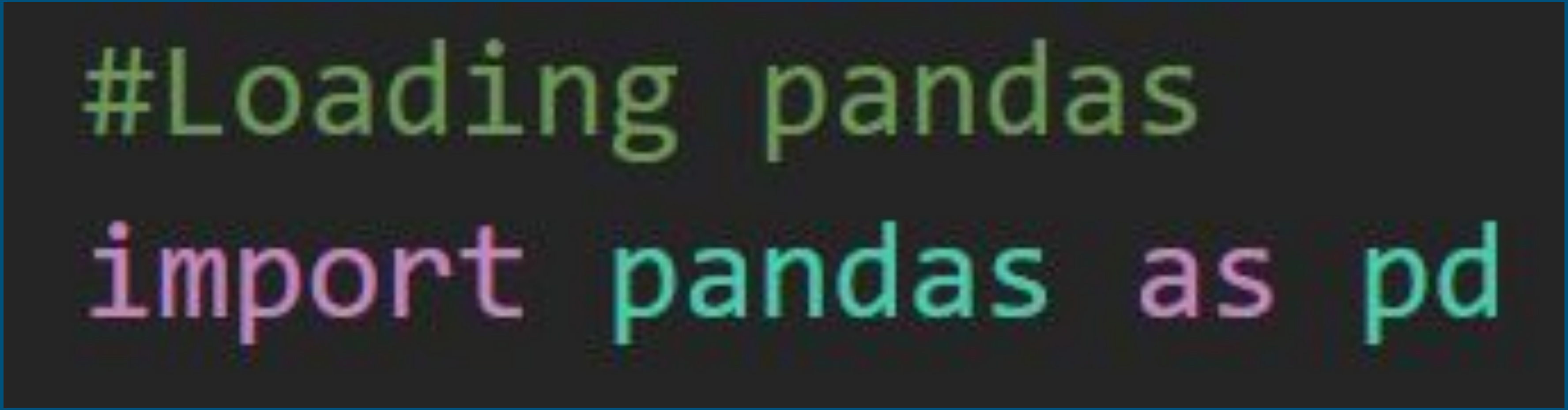
# Reminder

- Mini Project 1 is going on and is due next Tuesday before class!
- If you have not made your survey and sent it to people, do so ASAP
- Any questions about Mini Project 1?

# Pandas- What is it?

- Library for managing lots of data fast
- Wrapper for visualization and math libraries too.
- Literally the coolest

# Import pandas library



```python
#Loading pandas
import pandas as pd
```

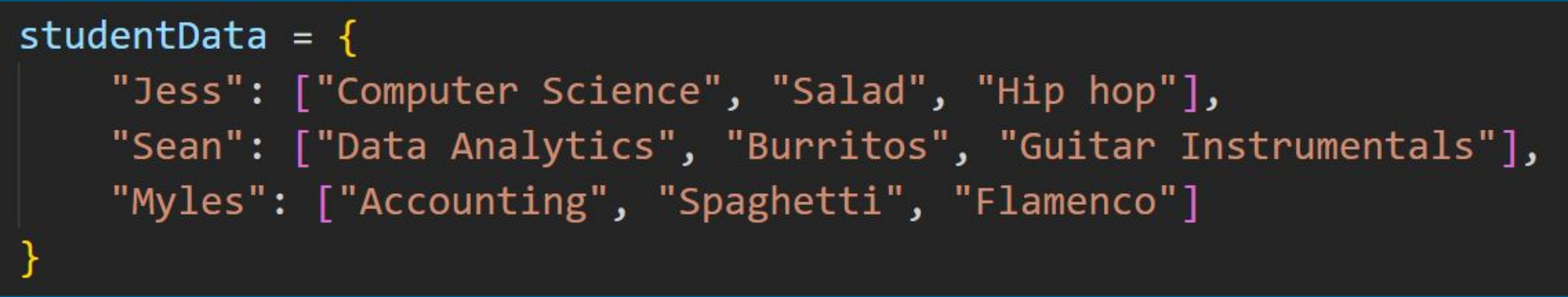
# Pandas DataFrames

- Pandas stores datas in table called “DataFrames”
    - Notice that the “D” and “F” are capitalized.
- But how do we make one?

# Making a DataFrame with a Dictionary

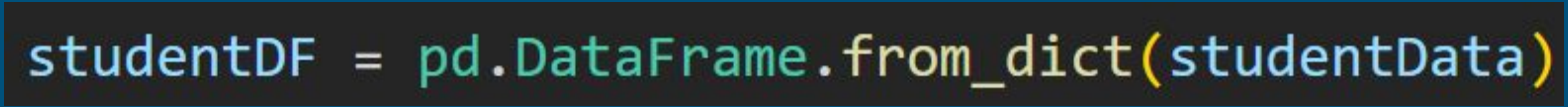
- Lets make a DataFrame where each row is a student, and the columns are “major”, “fave food”, and “fave music genre”
- We’ll start by making a dictionary

# Ex: Dictionary of data



```
studentData = {
    "Jess": ["Computer Science", "Salad", "Hip hop"],
    "Sean": ["Data Analytics", "Burritos", "Guitar Instrumentals"],
    "Myles": ["Accounting", "Spaghetti", "Flamenco"]
}
```

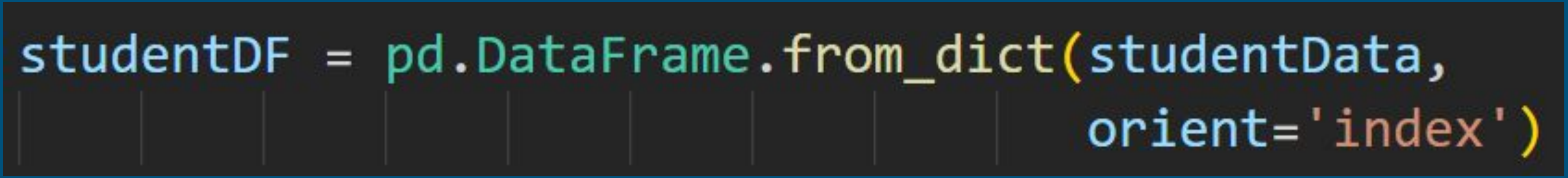
# How do we make a DataFrame from the Dictionary?



```
studentDF = pd.DataFrame.from_dict(studentData)
```

# Result is not bad...

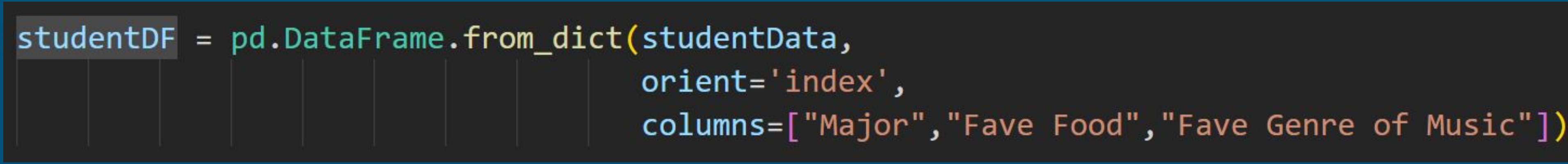
- ....but what if I want each row to a different student?
- Then I should do this instead:



```
studentDF = pd.DataFrame.from_dict(studentData,
                                   orient='index')
```

# Pretty good….

- …. But it would be nice if the columns had better names.
- We can do that too!:



```
studentDF = pd.DataFrame.from_dict(studentData,
                                   orient='index',
                                   columns=["Major","Fave Food","Fave Genre of Music"])
```

# How do we get data from this table?

- loc method!
- Give examples of getting entire row
- Give examples of getting one value
- Give examples of getting entire column

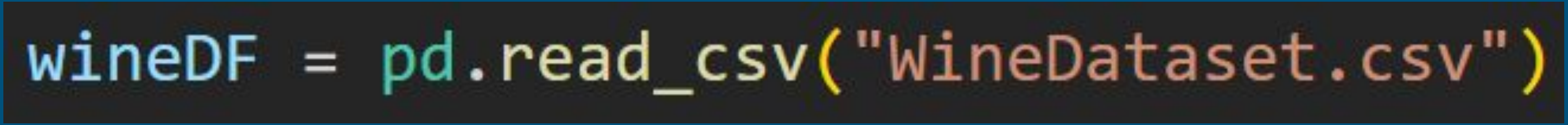
## Cool, but what about data from CSV or JSON files?

- No problem!

# Lets explore a dataset on different wines

- Provide data to class

# Load CSV data as a DataFrame



```
wineDF = pd.read_csv("WineDataset.csv")
```

- Print it out. What do you see?

# Printing out less

- head and tail methods

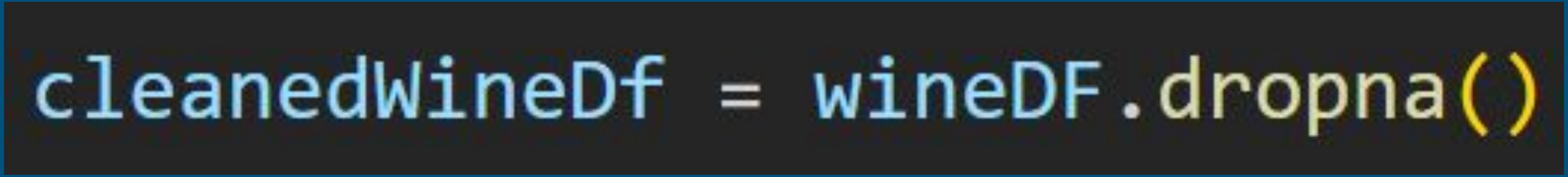
# Lets see what we can do with this dataset!

- Pandas is really good at cleaning and preprocessing data.
- It is also fast!
  - Built in C
- Works well with large datasets!

# How do we get number of rows and columns?

- Shape method

# Using pandas to drop NA values

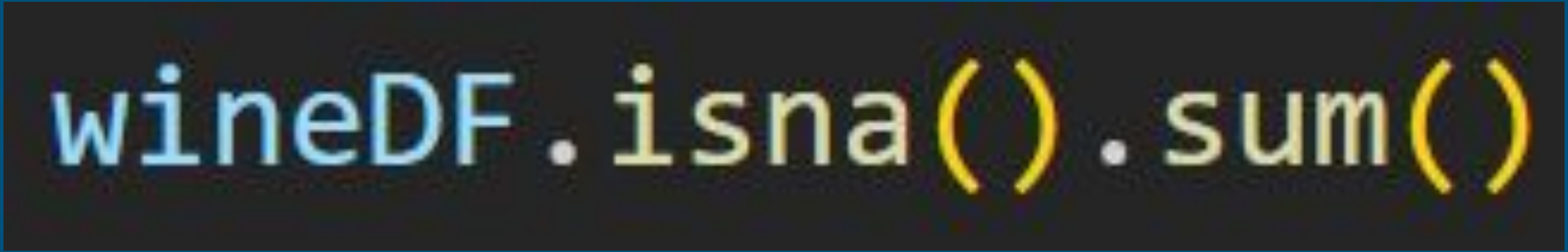


```
cleanedWineDf = wineDF.dropna()
```

# How many rows were there originally? How many rows after dropping all rows with NA values

- What caused this?
- What columns seem to be the main culprit?

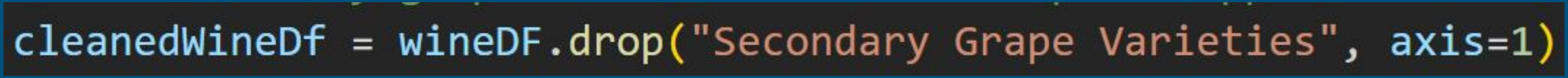
# We can check the amt of NAs with code!



```
wineDF.isna().sum()
```

# Lets maybe just remove the 2-3 cols with the most NAs.

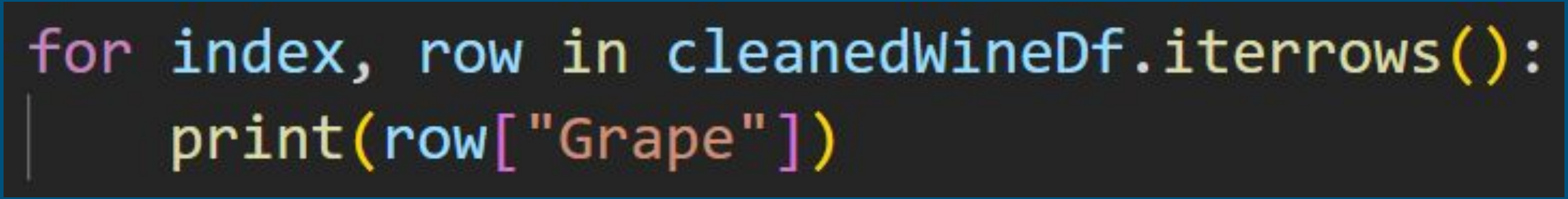
Ex:



```
cleanedWineDf = wineDF.drop("Secondary Grape Varieties", axis=1)
```

- axis = 0  means row
- axis = 1 means column

# How to loop through a Pandas DataFrame?



```
for index, row in cleanedWineDf.iterrows():
    print(row["Grape"])
```

# IMPORTANT!- You should NOT try to loop through a Pandas DataFrame (usually)

- Why?
    - Much slower and inefficient
- There are built in functions that are much quicker
    - Faster due to Vectorization
    - If there is a time, do example on whiteboard with row vector times
        - This technique can use up more memory tho

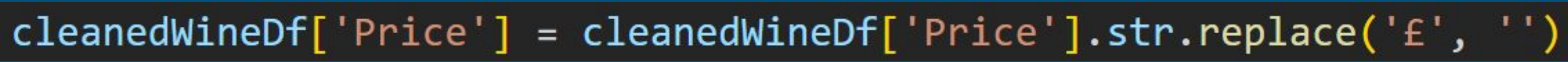
# Lets answer some questions about the dataset!

# Q1: What is the average price of the dataset!

- Pandas has function called mean().
- Very easy!
- Try it!
    - What is the problem?
    - How to fix it?

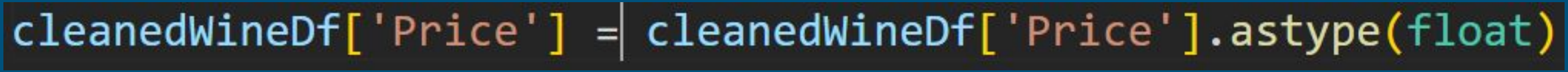
# Using Pandas to reformat data

- Can replace characters of strings
- Ex:



```
cleanedWineDf['Price'] = cleanedWineDf['Price'].str.replace('£', '')
```

# Can convert the data type of entire column



```
cleanedWineDf['Price'] = cleanedWineDf['Price'].astype(float)
```

# Did it work? Was everything resolved?

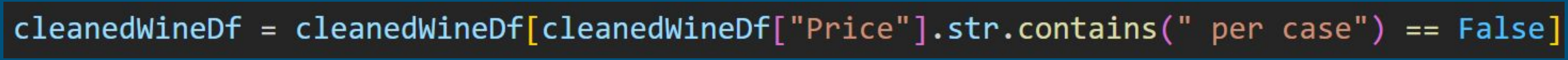
- If not, then explore until it is!

# Need to remove a row that meets a certain condition?

- Do this!

  myDF = myDF.loc[ myDF[“colName”] *boolean condition check* ]

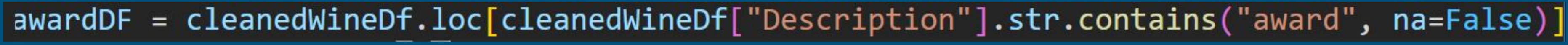
- Ex:



```
cleanedWineDf = cleanedWineDf[cleanedWineDf["Price"].str.contains(" per case") == False]
```

# So what is average price of a bottle of wine in this dataset?

# Q2: What wines have the word “award” written in their description? “Fruity”? “Dry”? etc



```
awardDF = cleanedWineDf.loc[cleanedWineDf["Description"].str.contains("award", na=False)]
```

- na = False means to ignore rows that have any na value

# Q3: How many wines have “Raspberry” as one of their characteristics? What is the average price of these?

# Q4: How many wines have a raspberry characteristic and have won an award?

- Must use “and”