

Python for Excel Users

```
31 def __init__(self):
32     self.file = None
33     self.fingerprints = set()
34     self.logdupes = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, 'requests.txt'),
39                         'a')
40         self.file.seek(0)
41         self.fingerprints.update([fingerprint(request) for request in self.requests])
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool('requests_log_debug')
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)
```

Why this presentation?

- Due to COVID-19 many roles have been put on hold, EXCEPT DATA!
- Most programs foundational elements, many students give up in this early stage

SO...

Let's start with tangible example and fold in foundational concepts as we go!

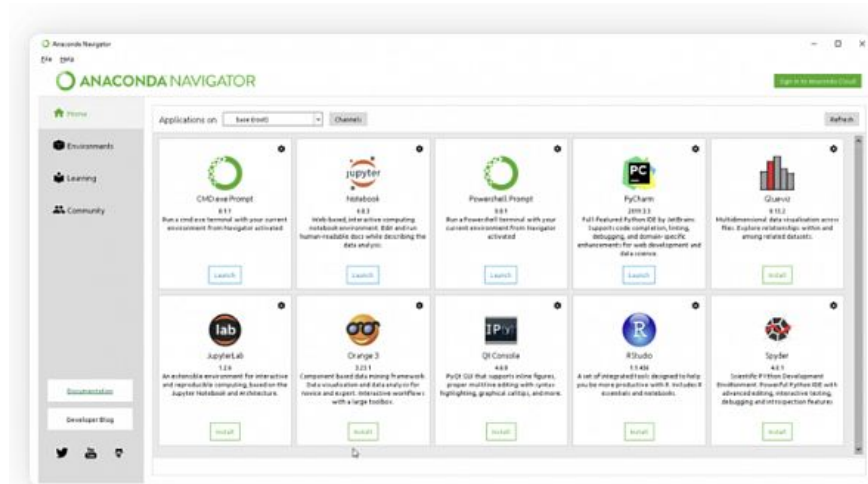
Where Python helps

- Gets user out of multiple sheet issue and embedded formulas with other files
- Difficult with VLOOKUP if many rows
- Ability to standardize and automate tasks
- Batteries included!



How to download Python

- Anaconda
 - <https://www.anaconda.com/products/individual>
- Miniconda
 - <https://docs.conda.io/en/latest/miniconda.html>
- Base Python
 - <https://www.python.org/downloads/>



Difference between Jupyter and scripts



```
In [17]: # Import the pandas package as pd
import pandas as pd
import seaborn as sns
```

```
In [18]: # Use the read_excel function to load ticket_sales.xlsx
sales= pd.read_excel('salesperson.xlsx')
```

```
In [19]: print(sales)
```

| | Salesperson | Company | Yearly_Sales |
|---|-------------|---------|--------------|
| 0 | John | A | 75000 |
| 1 | Suzie | A | 50000 |
| 2 | Bob | B | 95000 |

```
In [20]: sales.head(2)
```

Out[20]:

| | Salesperson | Company | Yearly_Sales |
|---|-------------|---------|--------------|
| 0 | John | A | 75000 |
| 1 | Suzie | A | 50000 |

```
In [21]: sales.info()
```

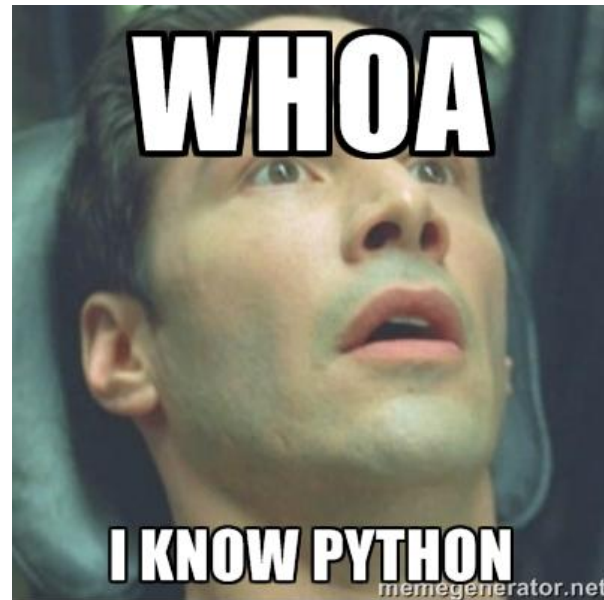
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Salesperson  3 non-null      object
1   Company      3 non-null      object
2   Yearly_Sales 3 non-null      int64
dtypes: int64(1), object(2)
memory usage: 200.0+ bytes
```

```
SparkDecisionTree.py
1 from pyspark.mllib.regression import LabeledPoint
2 from pyspark.mllib.tree import DecisionTree
3 from pyspark import SparkConf, SparkContext
4 from numpy import array
5
6 # Boilerplate Spark stuff:
7 conf = SparkConf().setMaster("local").setAppName("SparkDecisionTree")
8 sc = SparkContext(conf = conf)
9
10 # Some functions that convert our CSV input data into numerical
11 # features for each job candidate
12 def binary(YN):
13     if (YN == 'Y'):
14         return 1
15     else:
16         return 0
17
18 def mapEducation(degree):
19     if (degree == 'BS'):
20         return 1
21     elif (degree == 'MS'):
22         return 2
23     elif (degree == 'PhD'):
24         return 3
25     else:
26         return 0
27
```

Source: Oreilly

Other great free resources

- Data Analyst
 - <https://datatofish.com/>
 - <https://chrisalbon.com/>
 - <https://www.learnpython.org/>
 - <https://realpython.com/>
- Data Science
 - <https://jakevdp.github.io/PythonDataScienceHandbook/>
 - <https://www.datacamp.com/community/tutorials>
 - <https://towardsdatascience.com/>



Appendix A: DataFrame Methods

Like functions but accessed through package

`.describe()`

`.sort_values()`

`.info()`

`.head()`