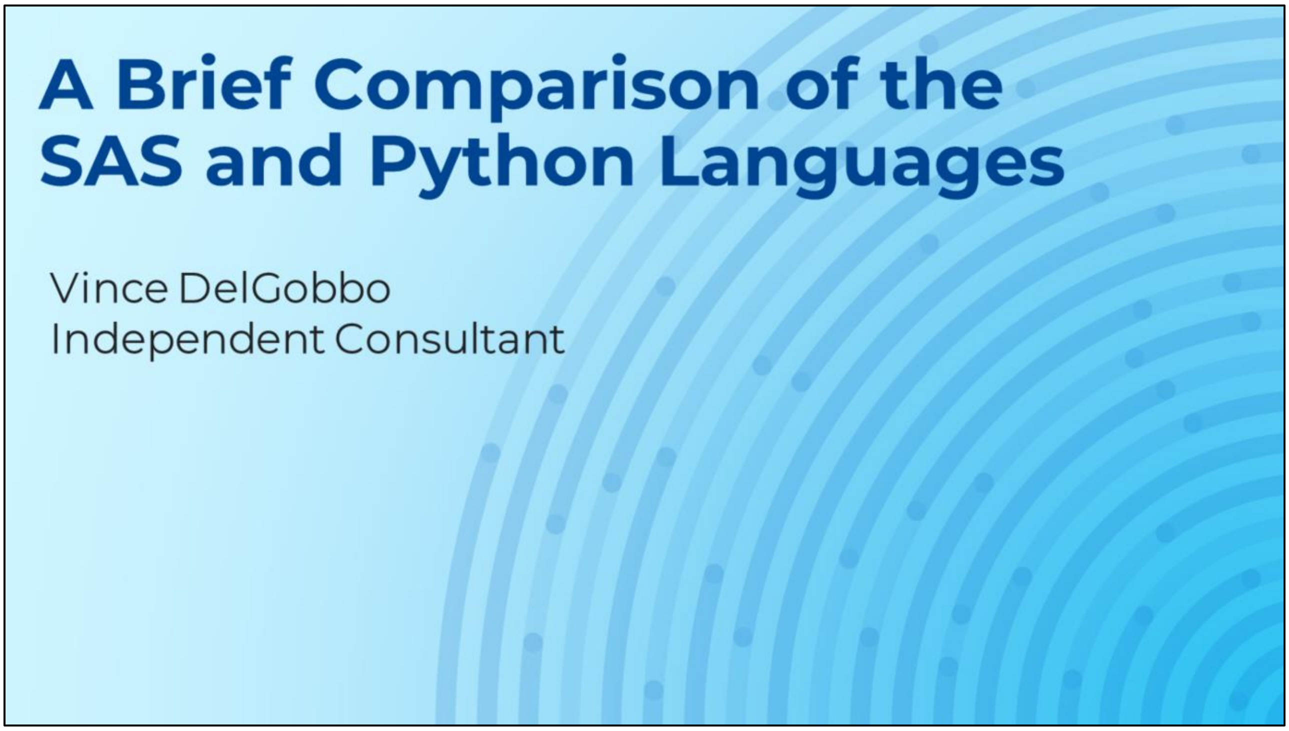


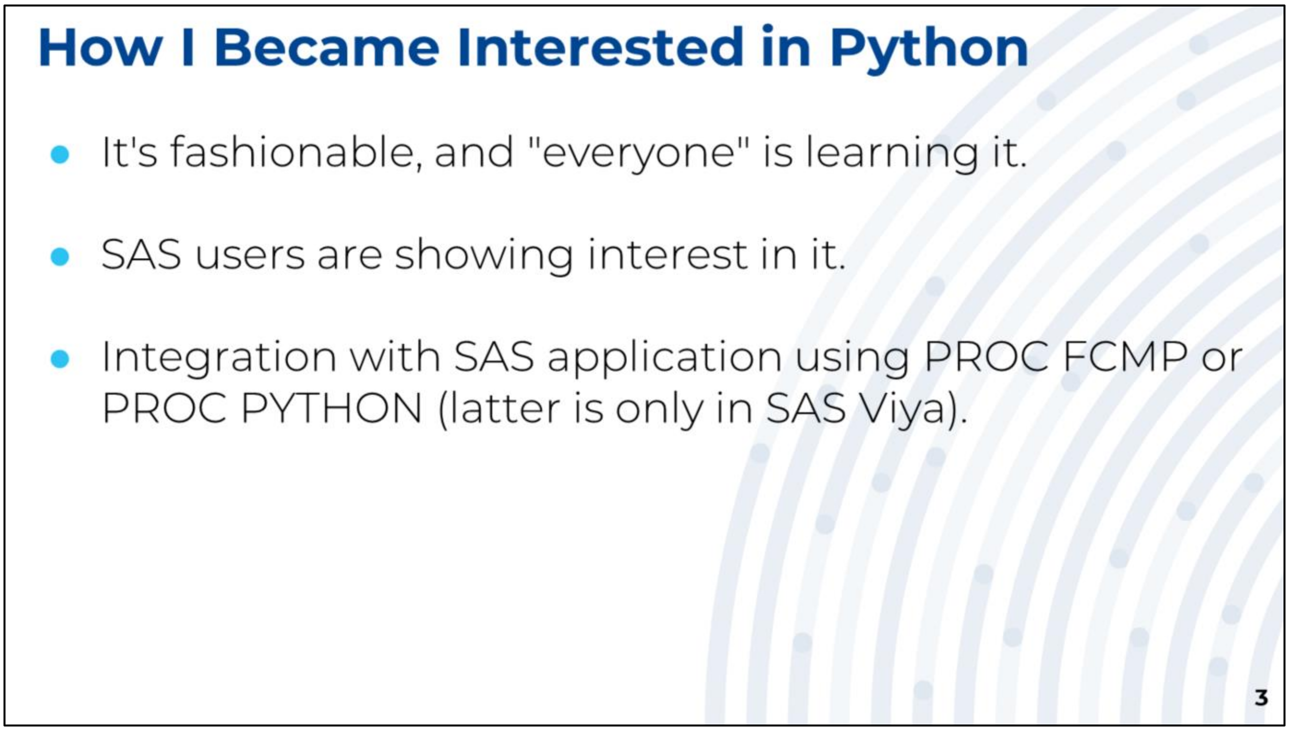
A Brief Comparison of the SAS and Python Languages

Vince DelGobbo
Independent Consultant

Boston Area SAS Users Group Meeting (BASUG)
April 20, 2022

A special thank you to Elizabeth Axelrod, Quentin McMullen, and the rest of the BASUG Steering Committee for inviting me to present this topic.

Introduction and Background



How I Became Interested in Python

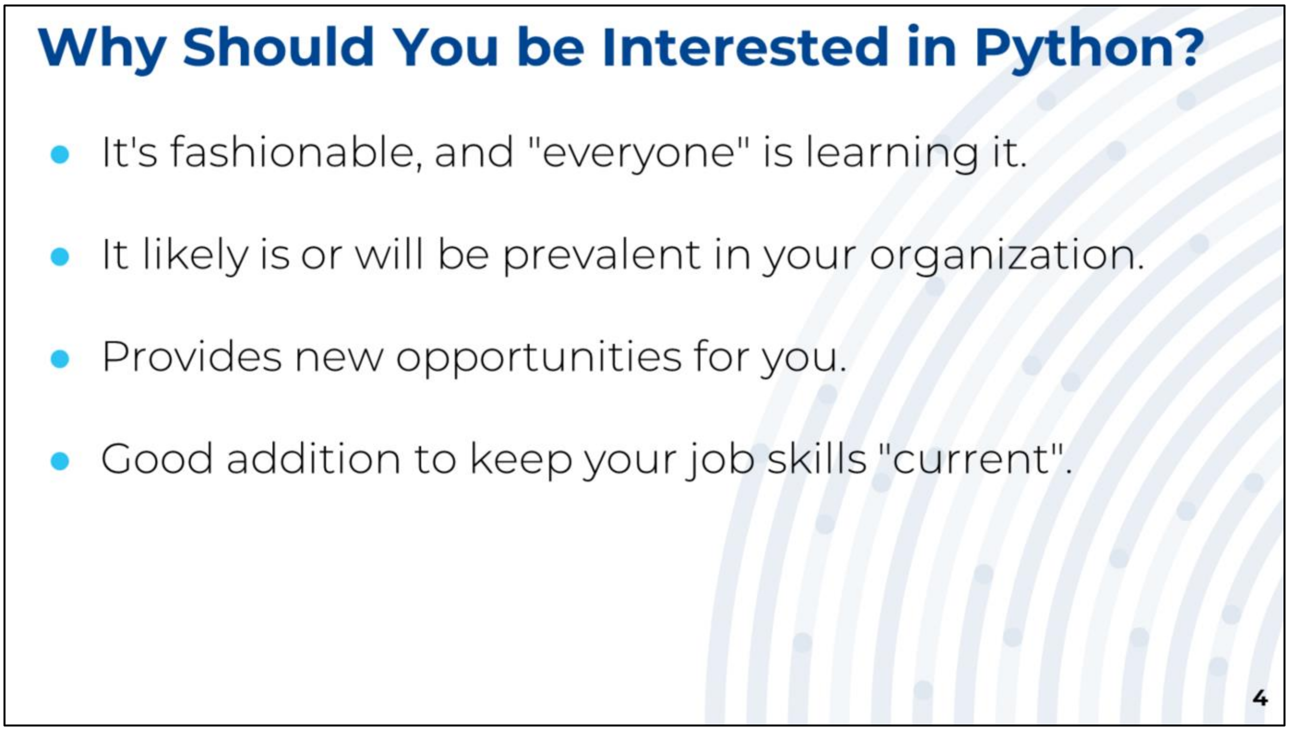
- It's fashionable, and "everyone" is learning it.
- SAS users are showing interest in it.
- Integration with SAS application using PROC FCMP or PROC PYTHON (latter is only in SAS Viya).

Interest in the Python language among programmers, including SAS programmers, has been growing steadily in recent years.

The interest shown by SAS programmers prompted me to look for integration points between Python and SAS. Enhancements to PROC FCMP in SAS 9, and the addition of PROC PYTHON in SAS Viya, provide those integration points.

The next step was to learn Python language. So I installed the Anaconda Python distribution, took an excellent O'Reilly online course, and then started working with the language.

During that learning period I noticed similarities between SAS and Python, and some of those findings became the basis for this presentation.



Why Should You be Interested in Python?

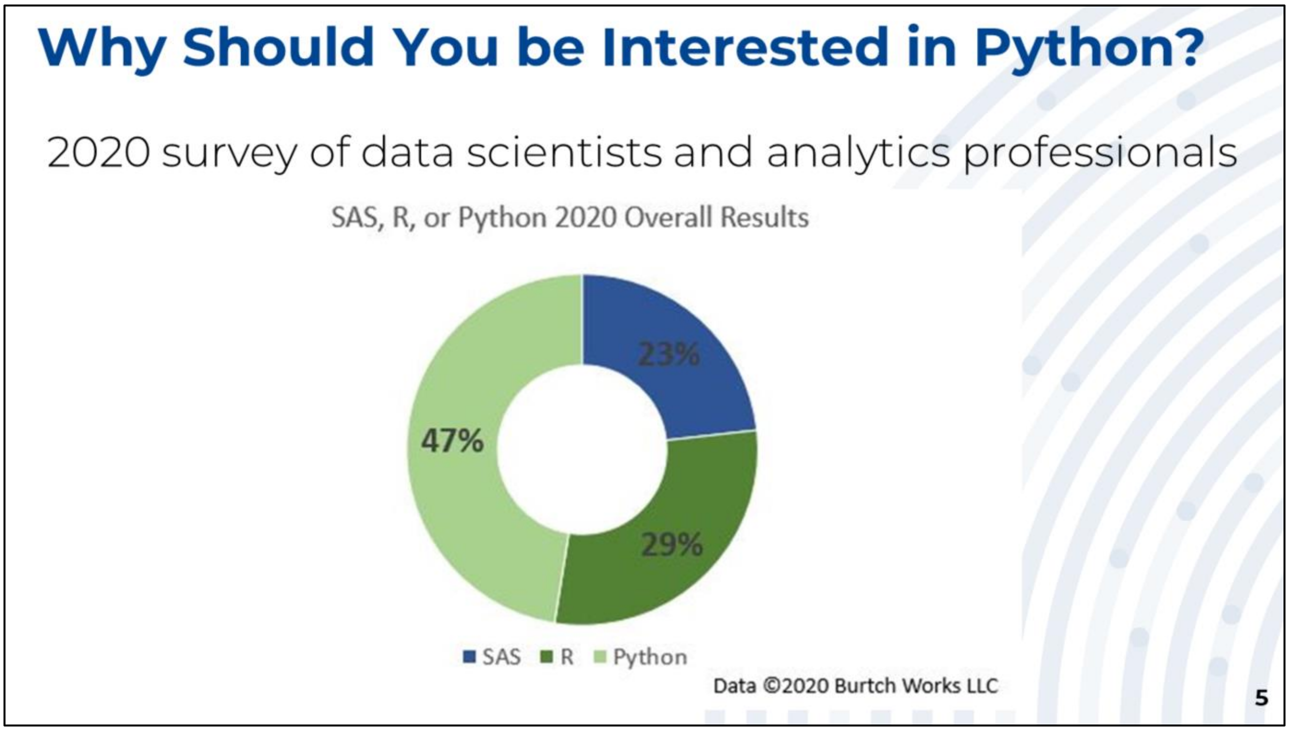
- It's fashionable, and "everyone" is learning it.
- It likely is or will be prevalent in your organization.
- Provides new opportunities for you.
- Good addition to keep your job skills "current".

The next few slides present information that show the popularity of Python.

Python is being used in in many organizations, and if it's not yet used in your organization, it will likely be used before too long. By learning Python, you will better positioned if corporate direction dictates that a project be completed using Python, or if you are asked to collaborate with Python developers.

Data Science and Machine Learning are two areas where Python is used a lot. It's a good idea to learn Python if you want to pursue work in these areas.

And having Python skills is advantageous should you decide to change jobs.



Why Should You be Interested in Python?

2020 survey of data scientists and analytics professionals

Each year for the past seven years, Burtch Works asked data scientists and analytics professionals whether they prefer to use SAS, R, or Python, and then examined their responses to see how tool preferences varied by several factors.

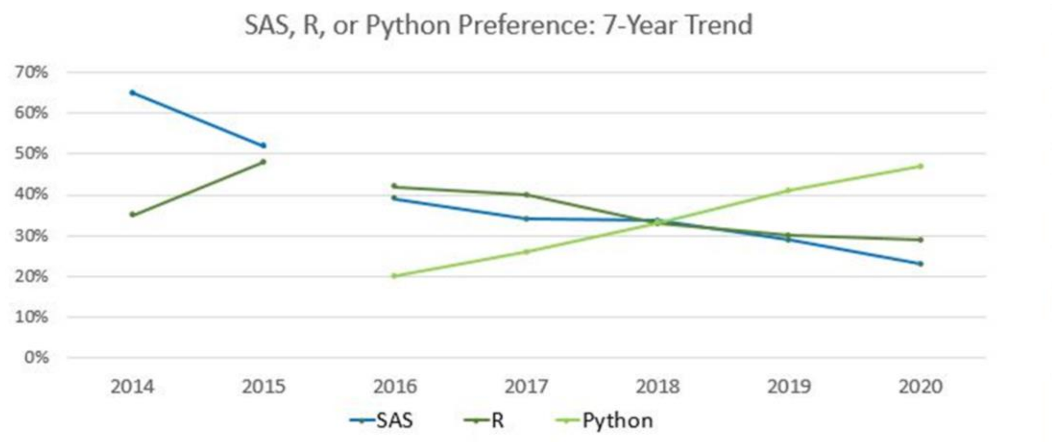
In the 2020 survey, 47% of the respondents prefer to use Python, with the remainder being split between R and SAS, with R having a 6% edge over SAS.

Source:

2020 SAS, R, or Python Survey Results: Which Tool do Data Scientists & Analytics Pros Prefer?
https://www.burtchworks.com/2020/12/16/2020-sas-r-or-python-survey-results-which-tool-do-data-scientists-analytics-pros-prefer/

Why Should You be Interested in Python?

2020 survey of data scientists and analytics professionals



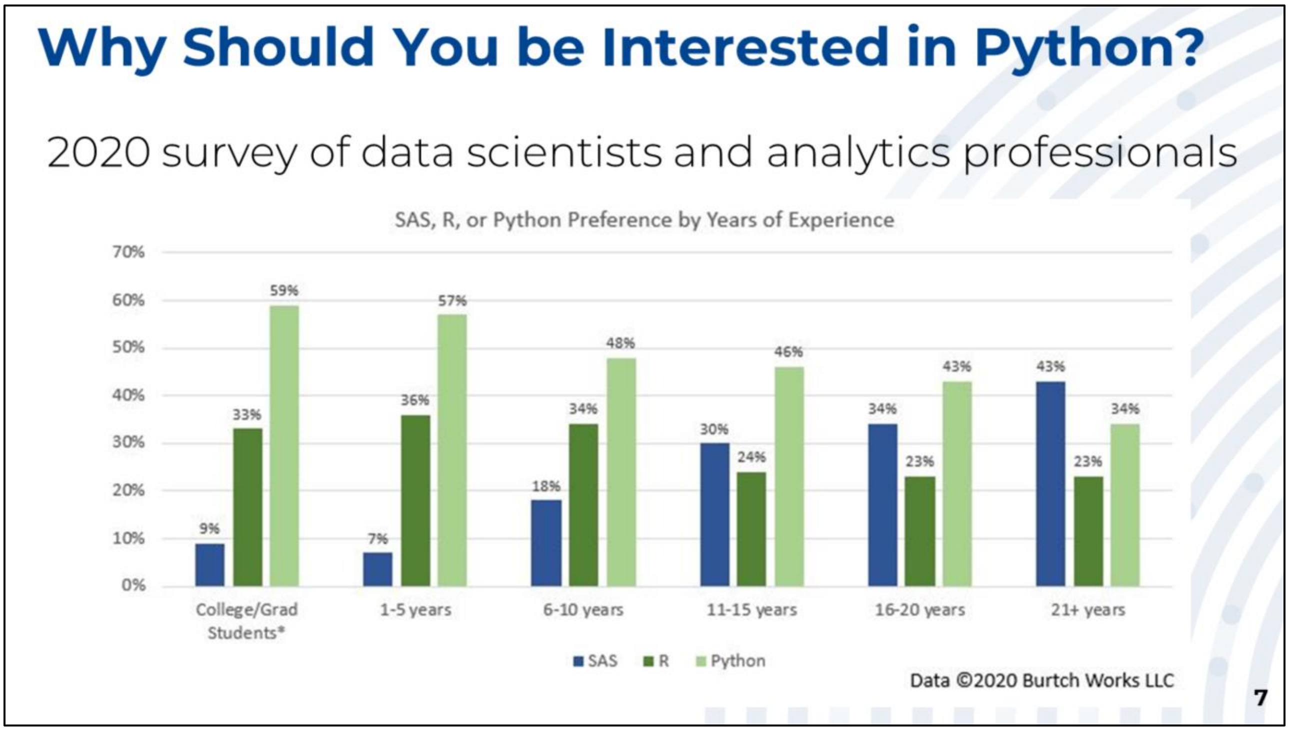
*Python added as an option in 2016.

Data ©2020 Burtch Works LLC

Burtch Works started tracking Python in 2016, when usage was far below R and SAS. Usage of SAS and R has been declining since then, while Python usage has been growing, with a near perfect tie happening in 2018. Python has been the dominant tool of preference since 2018.

Source:

2020 SAS, R, or Python Survey Results: Which Tool do Data Scientists & Analytics Pros Prefer?
https://www.burtchworks.com/2020/12/16/2020-sas-r-or-python-survey-results-which-tool-do-data-scientists-analytics-pros-prefer/



Why Should You be Interested in Python?

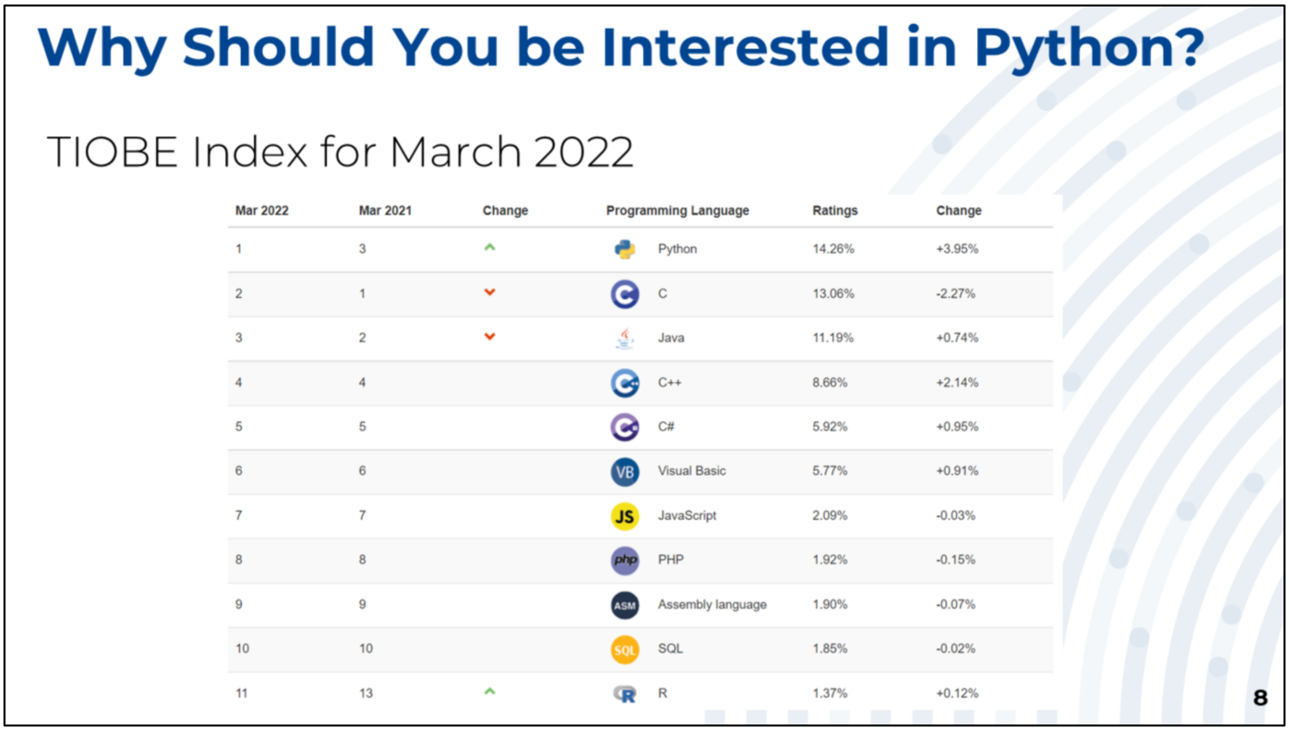
2020 survey of data scientists and analytics professionals

Data ©2020 Burtch Works LLC

Python usage greatly exceeds SAS and R usage in College/Grad students and early career professionals. This strong usage of Python is likely to manifest in your organization, so having Python skills seems to be worthwhile.

Source:

2020 SAS, R, or Python Survey Results: Which Tool do Data Scientists & Analytics Pros Prefer?
https://www.burtchworks.com/2020/12/16/2020-sas-r-or-python-survey-results-which-tool-do-data-scientists-analytics-pros-prefer/



Why Should You be Interested in Python?

TIOBE Index for March 2022

Mar 2022	Mar 2021	Change	Programming Language	Ratings	Change
1	3	^	Python	14.26%	+3.95%
2	1	˅	C	13.06%	-2.27%
3	2	˅	Java	11.19%	+0.74%
4	4		C++	8.66%	+2.14%
5	5		C#	5.92%	+0.95%
6	6		Visual Basic	5.77%	+0.91%
7	7		JavaScript	2.09%	-0.03%
8	8		PHP	1.92%	-0.15%
9	9		Assembly language	1.90%	-0.07%
10	10		SQL	1.85%	-0.02%
11	13	^	R	1.37%	+0.12%

The TIOBE Index, which is updated monthly, is an indicator of the popularity of programming languages. Major search engines are used to determine the popularity.

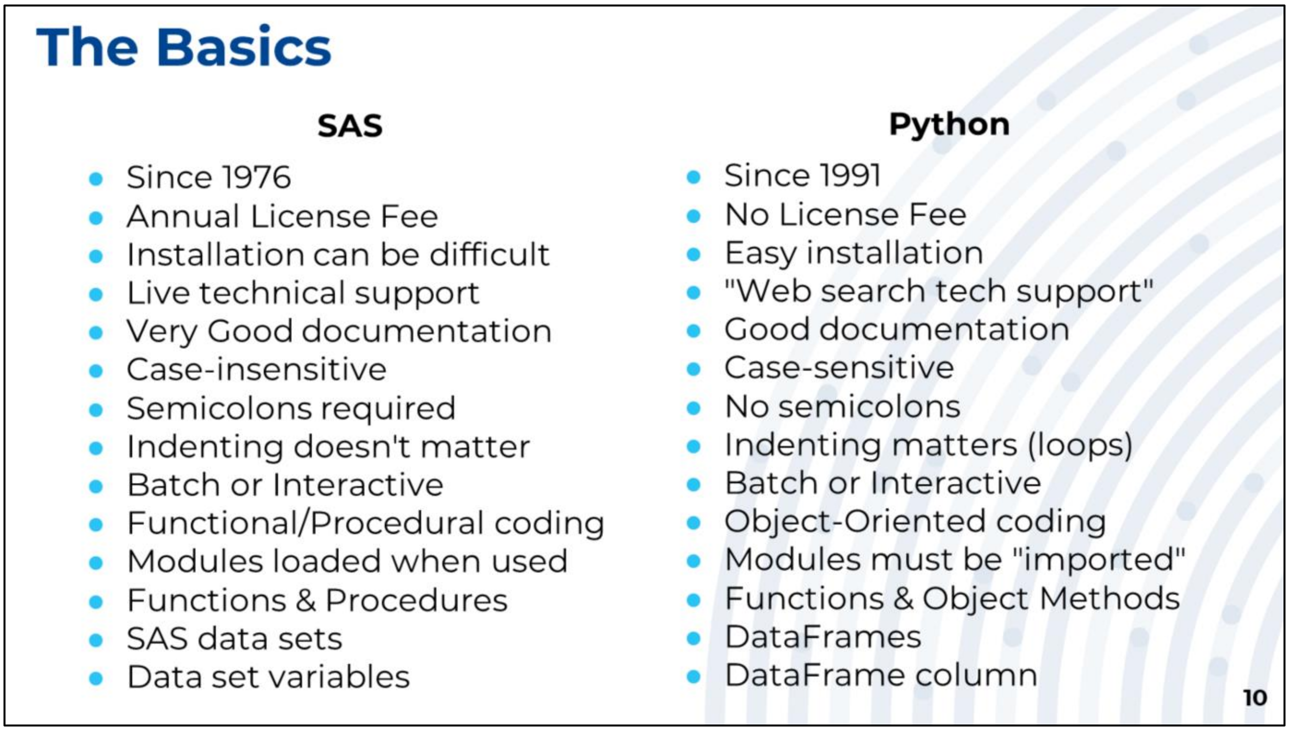
C, Java, and Python held the number 1, 2, and 3 spots respectively in March 2021. Python went to first place one year later. R is at number 11 and SAS is at number 24.

All the charts that we've looked at indicate that Python is a popular language, but make no statement that it is the "best" language.

Source:

TIOBE Index (updated monthly)
https://www.tiobe.com/tiobe-index

Comparison of the Languages



The Basics

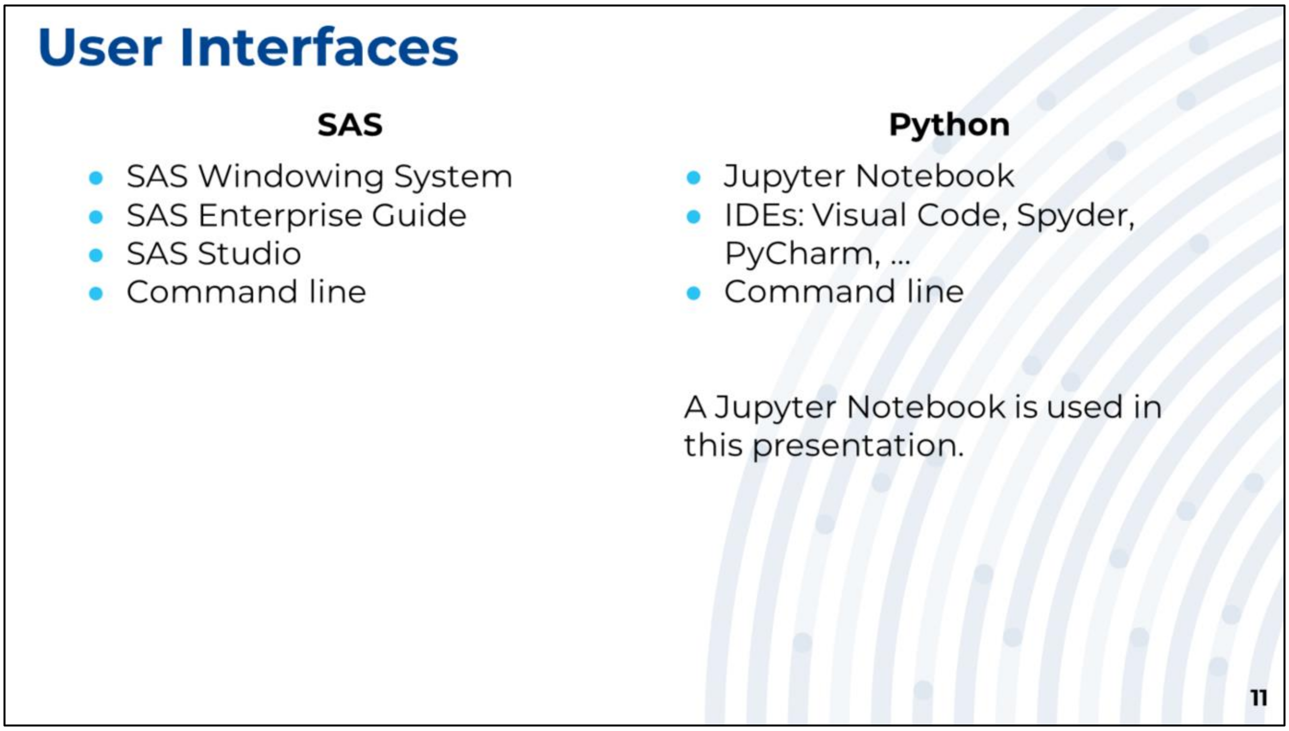
SAS	Python
Since 1976	Since 1991
Annual License Fee	No License Fee
Installation can be difficult	Easy installation
Live technical support	"Web search tech support"
Very Good documentation	Good documentation
Case-insensitive	Case-sensitive
Semicolons required	No semicolons
Indenting doesn't matter	Indenting matters (loops)
Batch or Interactive	Batch or Interactive
Functional/Procedural coding	Object-Oriented coding
Modules loaded when used	Modules must be "imported"
Functions & Procedures	Functions & Object Methods
SAS data sets	DataFrames
Data set variables	DataFrame column

The installation of SAS can be difficult, especially for server installations.

Unlike SAS, there is no formal technical support for Python. Much help is available through articles, tutorials, and discussion forums.

It's not exactly correct to say that Python documentation is not as good as SAS. This is somewhat of a biased opinion due to many years of reading SAS documentation. The Python documentation is different from SAS, and it takes a little time to acclimate to it.

A detailed discussion of Object Oriented Programming is beyond the scope of this discussion. Briefly, "everything" in Python is an object, and each object has a set of properties and functions (aka, "methods") that are specific to that function. Properties and methods are accessed using "dot notation", which is illustrated in the slides that follow.



User Interfaces

SAS

- SAS Windowing System
- SAS Enterprise Guide
- SAS Studio
- Command line

Python

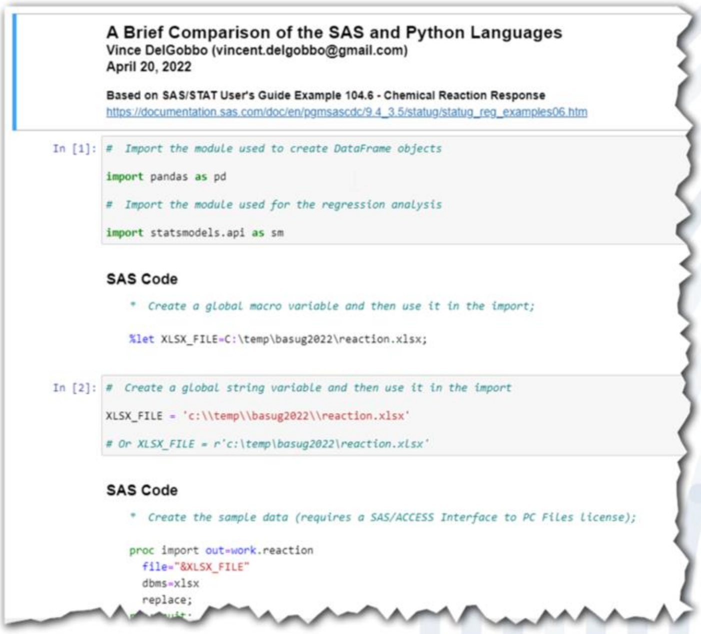
- Jupyter Notebook
- IDEs: Visual Code, Spyder, PyCharm, ...
- Command line

A Jupyter Notebook is used in this presentation.

You must use one of the SAS-provided clients to run SAS code.

You have many choices for Python clients.

Jupyter Notebook



A Brief Comparison of the SAS and Python Languages
Vince DelGobbo (vincent.delgobbo@gmail.com)
April 20, 2022

Based on SAS/STAT User's Guide Example 104.6 - Chemical Reaction Response
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/statug/statug_reg_examples06.htm

```
In [1]: #  Import the module used to create DataFrame objects

        import pandas as pd

        #  Import the module used for the regression analysis

        import statsmodels.api as sm
```

SAS Code

```
    *  Create a global macro variable and then use it in the import;

    %let XLSX_FILE=C:\temp\basug2022\reaction.xlsx;
```

```
In [2]: #  Create a global string variable and then use it in the import

        XLSX_FILE = 'c:\\temp\\basug2022\\reaction.xlsx'

        # Or XLSX_FILE = r'c:\temp\basug2022\reaction.xlsx'
```

SAS Code

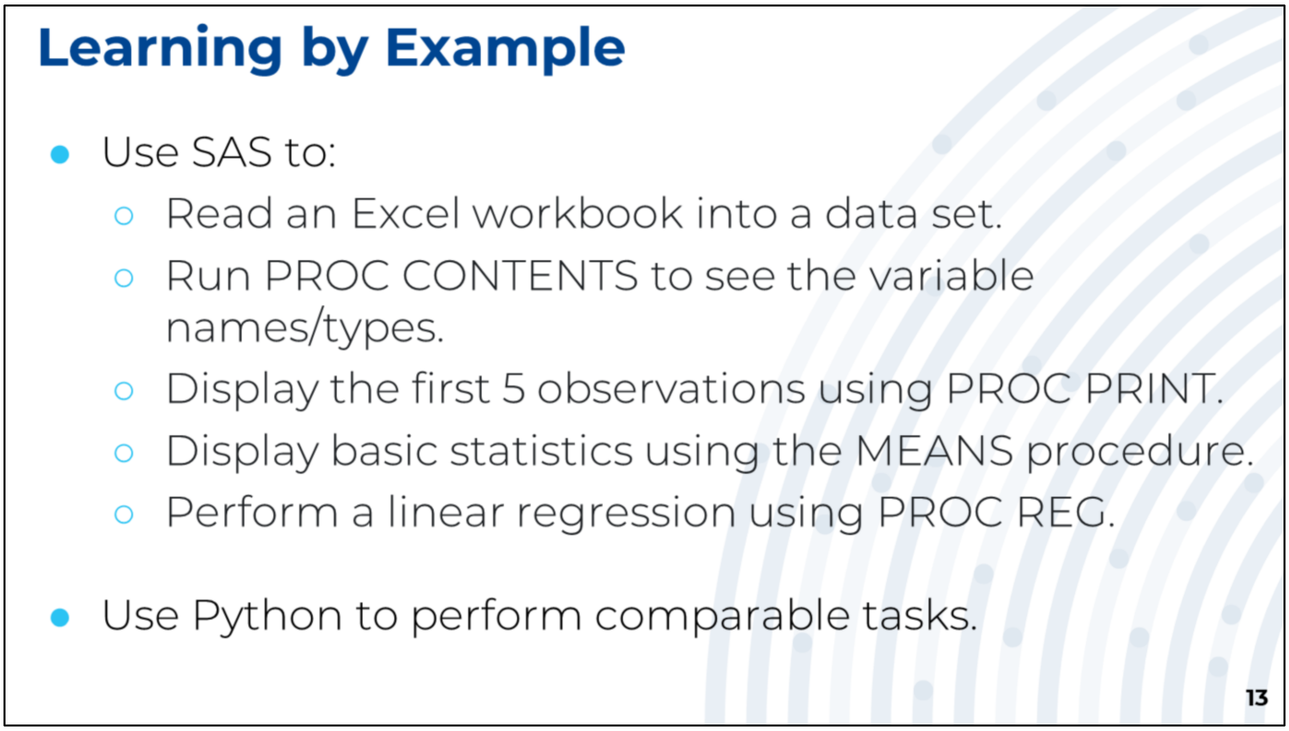
```
    *  Create the sample data (requires a SAS/ACCESS Interface to PC Files license);

    proc import out=work.reaction
      file="&XLSX_FILE"
      dbms=xlsx
      replace;
```

A Jupyter Notebook is a Web-based programming interface to the Julia, Python, and R languages. Code is specified within a cell and then executed interactively. The code and results are stored in the notebook in the order that they are created. You can also include cells containing HTML or Markdown text.

In the screenshot above you see explanatory text included with the Python code. No results are shown in this small portion of the notebook, but results are displayed in future slides.

You can send a Jupyter Notebook to a colleague and they will be able to execute the code, providing an easy way to collaborate on projects.

Jupyter Notebooks can be exported to several "static" formats, including HTML and PDF.



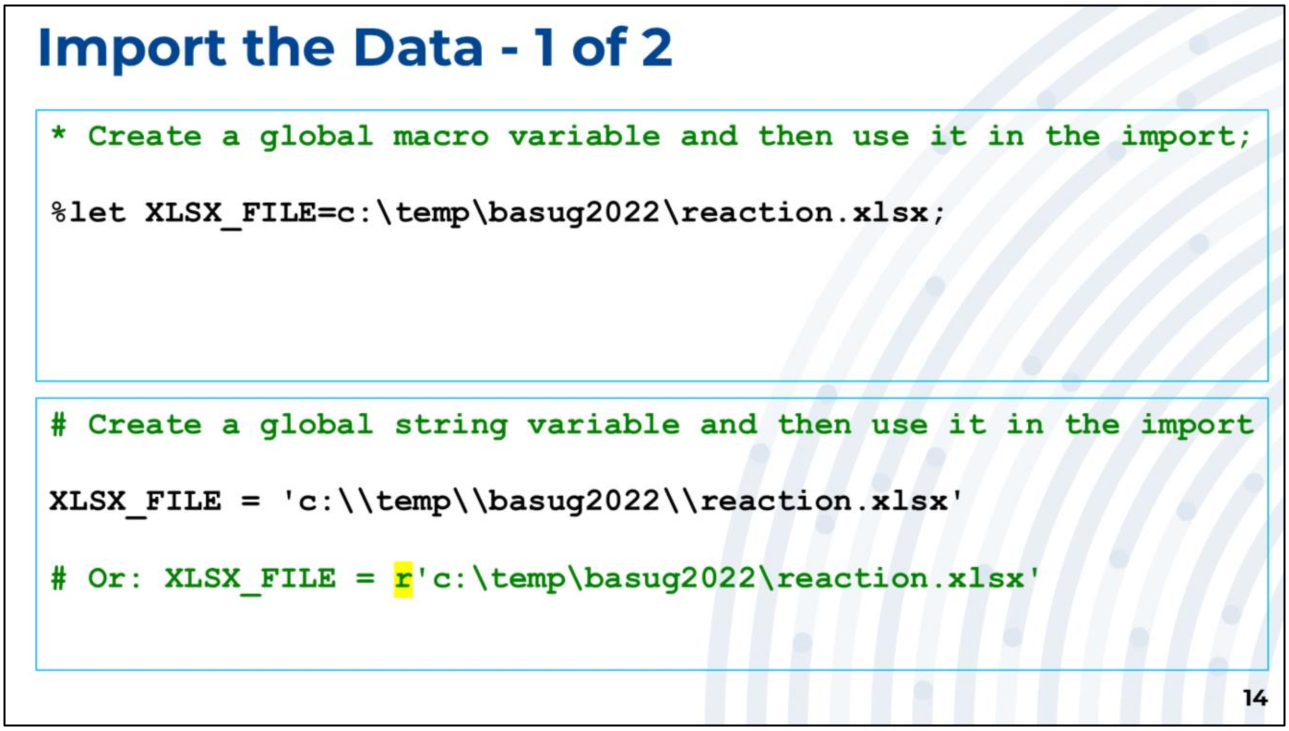
Learning by Example

- Use SAS to:
  - Read an Excel workbook into a data set.
  - Run PROC CONTENTS to see the variable names/types.
  - Display the first 5 observations using PROC PRINT.
  - Display basic statistics using the MEANS procedure.
  - Perform a linear regression using PROC REG.
- Use Python to perform comparable tasks.

A typical workflow consisting of importing and exploring data, and then performing a regression analysis, is used to compare he two languages.

The SAS regression analysis code based on this example:

SAS/STAT User's Guide, Example 104.6 Chemical Reaction Response
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/statug/statug_reg_examples06.htm



Import the Data - 1 of 2

```
* Create a global macro variable and then use it in the import;

%let XLSX_FILE=c:\temp\basug2022\reaction.xlsx;
```

```
# Create a global string variable and then use it in the import

XLSX_FILE = 'c:\\temp\\basug2022\\reaction.xlsx'

# Or: XLSX_FILE = r'c:\temp\basug2022\reaction.xlsx'
```

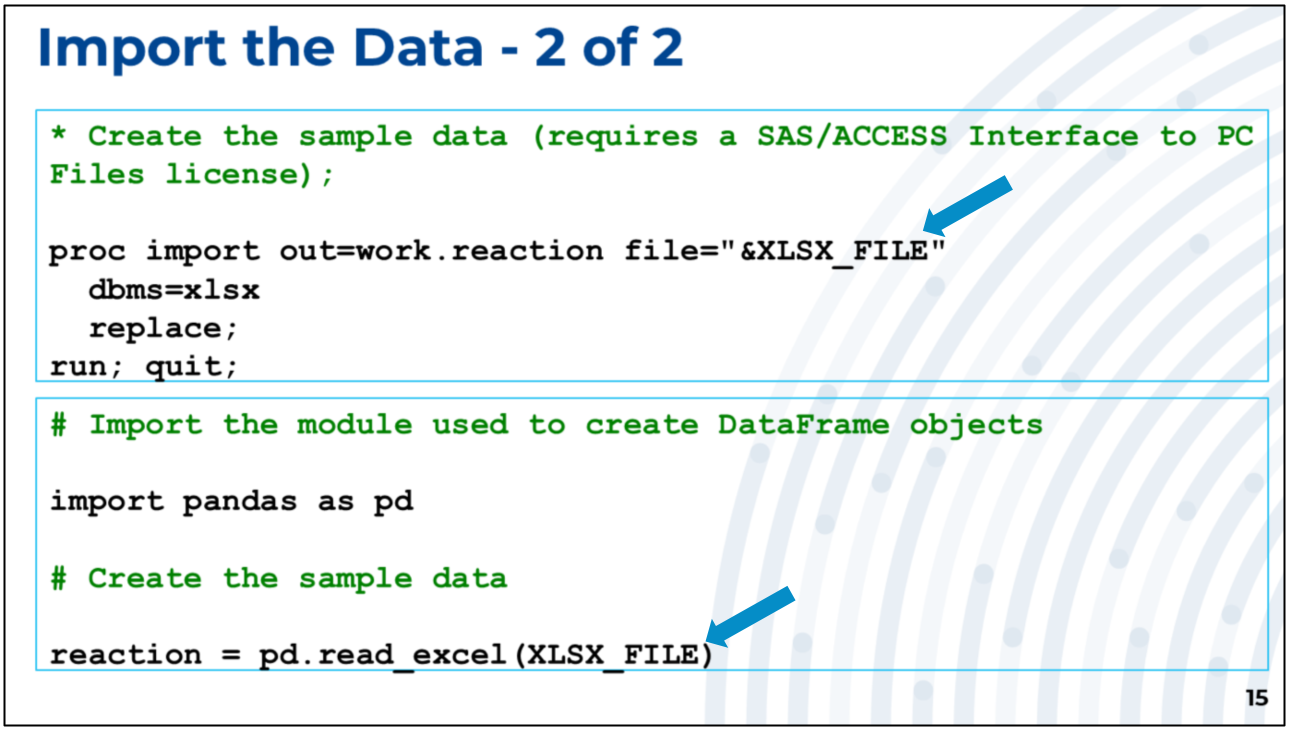
SAS uses the asterisk (*) character to denote a comment line and Python uses a pound (#) character.

SAS statements must end with a semicolon.

Variable names are case-insensitive in SAS but case-sensitive in Python.

Backslash characters (\) must be escaped in Python strings by specifying an additional backslash. Alternatively, you can use a "raw string":

```
XLSX_FILE=r'C:\Temp\BASUG2022\reaction.xlsx'
```



Import the Data - 2 of 2

```
* Create the sample data (requires a SAS/ACCESS Interface to PC
Files license);

proc import out=work.reaction file="&XLSX_FILE"
  dbms=xlsx
  replace;
run; quit;
```

```
# Import the module used to create DataFrame objects

import pandas as pd

# Create the sample data

reaction = pd.read_excel(XLSX_FILE)
```

15

The pandas module, used to create DataFrame objects, must be imported before it's first use. We specify an alias of "pd" to simplify code statements.

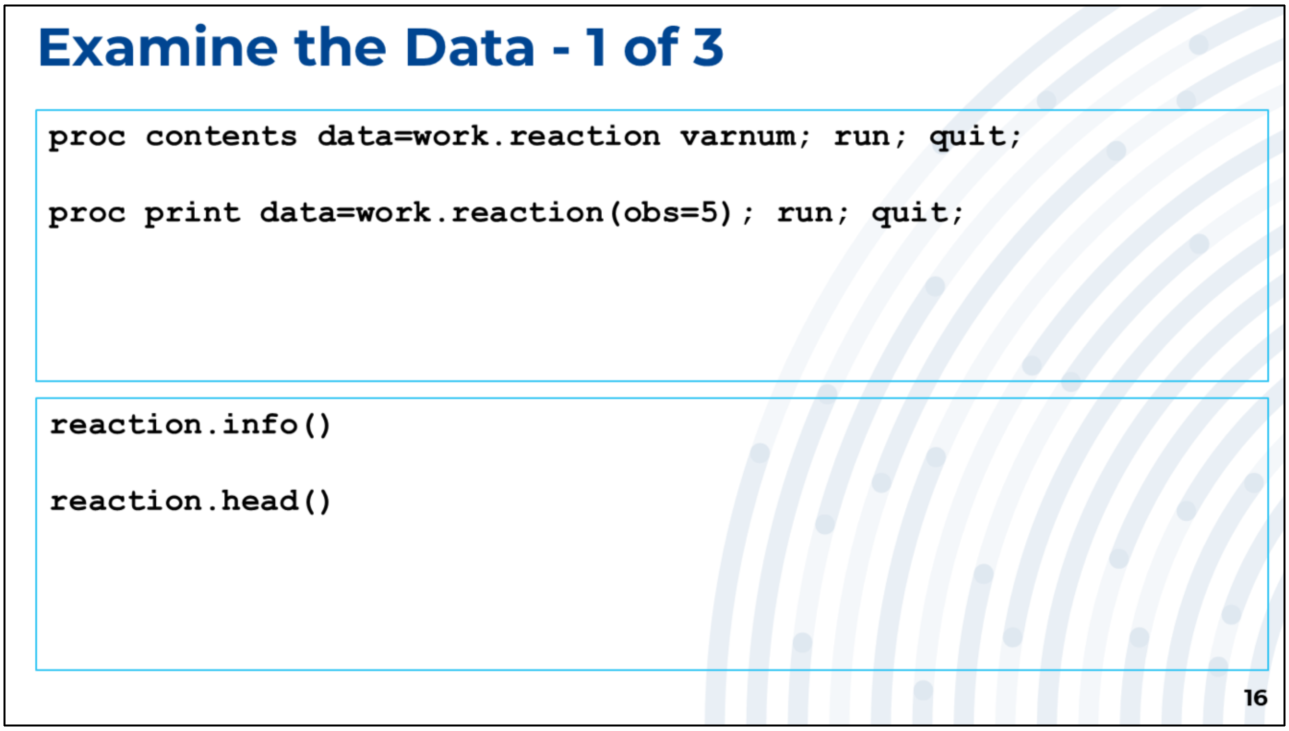
Here is the first example of object-oriented programming.

"Everything" in Python is an "object", and objects contain "methods" that are specific to that object. You can think of a method as being similar to a SAS function or procedure.

The pandas "read_excel" method reads the data from a sheet in the Excel workbook specified in the `XLSX_FILE` global variable, and then stores it in the "reaction" DataFrame object.

The "reaction" DataFrame object is similar to a SAS data set, and contains the data from the Excel workbook.

You can use pandas to read SAS data sets, CSV files, HTML files, and more.



Examine the Data - 1 of 3

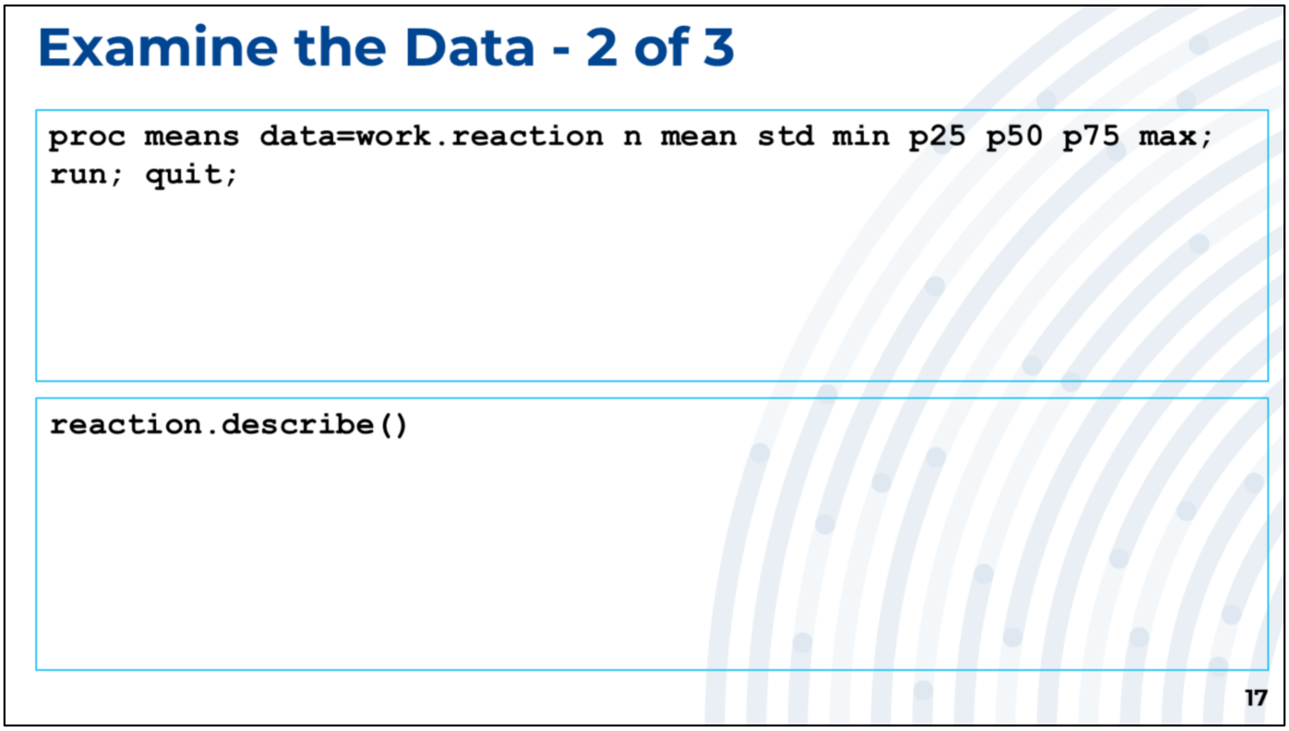
```
proc contents data=work.reaction varnum; run; quit;

proc print data=work.reaction(obs=5); run; quit;
```

```
reaction.info()

reaction.head()
```

The "info" and "head" methods are invoked on the data in the "reaction" DataFrame. The "info" method is similar to PROC CONTENTS, and the "head" method is similar to the PRINT procedure with the OBS option.

The "head" method displays the first 5 rows of the DataFrame by default. Specify an argument to the method to display a different number of rows, for example:

```
reaction.head(10)
```

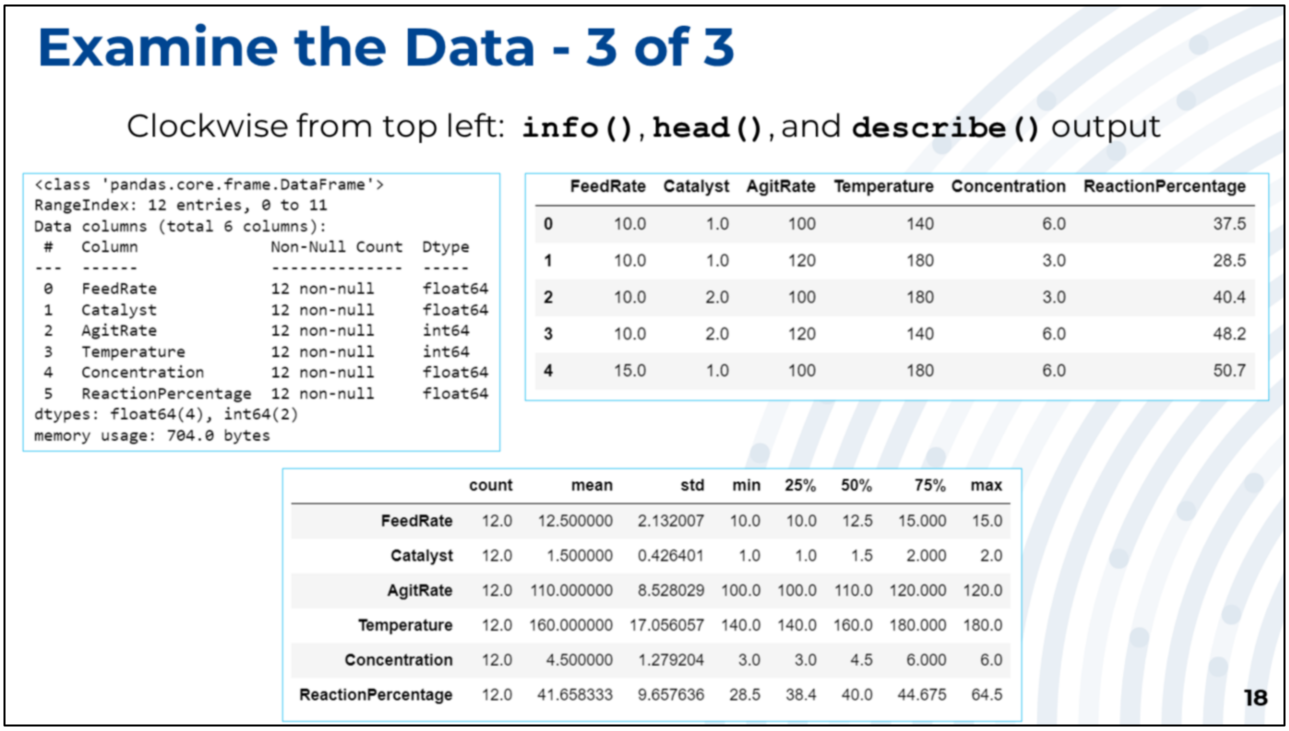


Examine the Data - 2 of 3

```
proc means data=work.reaction n mean std min p25 p50 p75 max;
run; quit;
```

```
reaction.describe()
```

The "describe" method displays basic statistics for the data in the "reaction" DataFrame object.



Examine the Data - 3 of 3

Clockwise from top left: **`info()`**, **`head()`**, and **`describe()`** output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   FeedRate            12 non-null     float64
 1   Catalyst            12 non-null     float64
 2   AgitRate            12 non-null     int64  
 3   Temperature         12 non-null     int64  
 4   Concentration       12 non-null     float64
 5   ReactionPercentage  12 non-null     float64
dtypes: float64(4), int64(2)
memory usage: 704.0 bytes
```

	FeedRate	Catalyst	AgitRate	Temperature	Concentration	ReactionPercentage
0	10.0	1.0	100	140	6.0	37.5
1	10.0	1.0	120	180	3.0	28.5
2	10.0	2.0	100	180	3.0	40.4
3	10.0	2.0	120	140	6.0	48.2
4	15.0	1.0	100	180	6.0	50.7

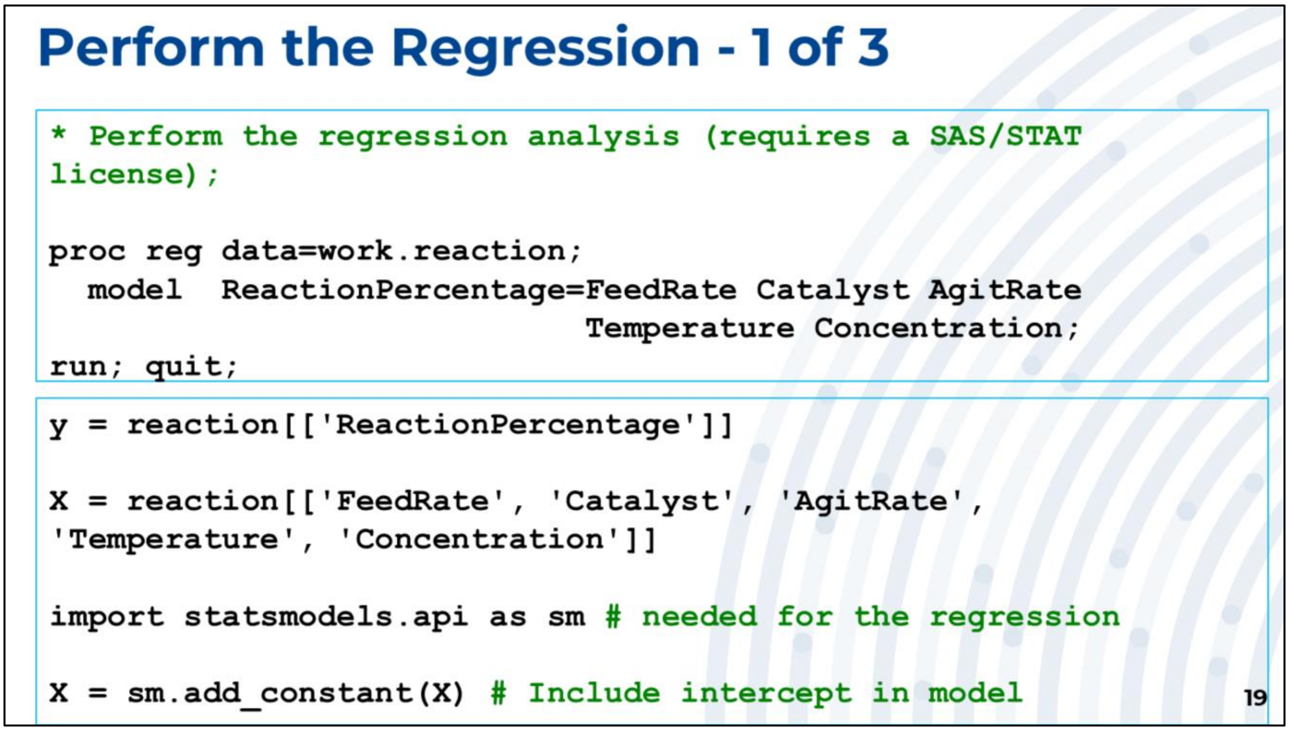
	count	mean	std	min	25%	50%	75%	max
FeedRate	12.0	12.500000	2.132007	10.0	10.0	12.5	15.000	15.0
Catalyst	12.0	1.500000	0.426401	1.0	1.0	1.5	2.000	2.0
AgitRate	12.0	110.000000	8.528029	100.0	100.0	110.0	120.000	120.0
Temperature	12.0	160.000000	17.056057	140.0	140.0	160.0	180.000	180.0
Concentration	12.0	4.500000	1.279204	3.0	3.0	4.5	6.000	6.0
ReactionPercentage	12.0	41.658333	9.657636	28.5	38.4	40.0	44.675	64.5

Here are screenshots of some cells in a Jupyter Notebook.

The "info" method is similar to PROC CONTENTS, and provides this information about the data in the DataFrame:

- 12 rows, with an index from 0 to 11
- 6 columns, with indexes from 0 to 5
- There are no missing values in any of the columns
- All columns are numeric, 2 have integer values and 4 have floating point values

The output from the "head" and "describe" methods are very similar to the PRINT and MEANS procedure output, respectively. Note the zero-based index displayed in the "head" method output. This is in contrast to SAS, which is one-based.



Perform the Regression - 1 of 3

```
* Perform the regression analysis (requires a SAS/STAT
license);

proc reg data=work.reaction;
  model  ReactionPercentage=FeedRate Catalyst AgitRate
                            Temperature Concentration;
run; quit;
```

```
y = reaction[['ReactionPercentage']]

X = reaction[['FeedRate', 'Catalyst', 'AgitRate',
'Temperature', 'Concentration']]

import statsmodels.api as sm # needed for the regression

X = sm.add_constant(X) # Include intercept in model
```

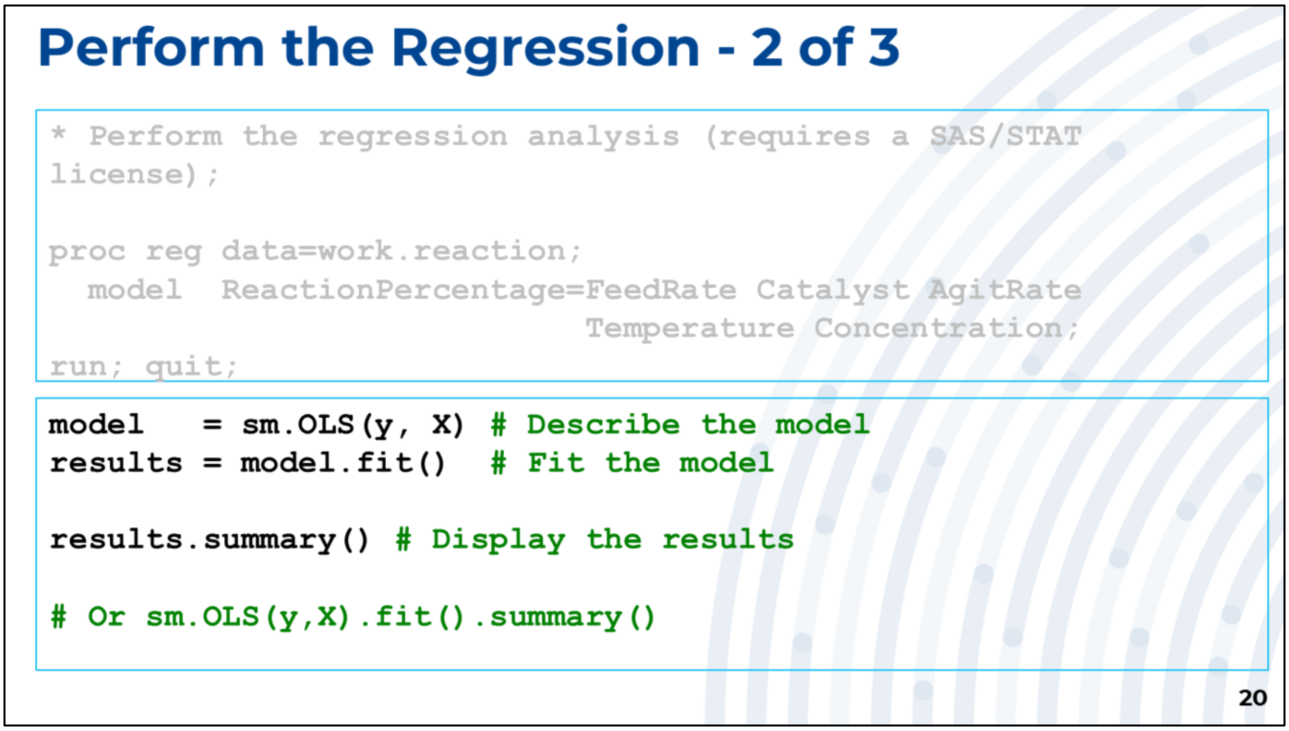
The dependent and regressor variables are specified in the MODEL statement of PROC REG.

We create DataFrames "y" and "X" (note case) with the columns from the "reaction" DataFrame that we want to use as the dependent and regressor data, respectively. Creating the "y" DataFrame is similar to this SAS code:

```
data work.y;
set work.reaction;
keep ReactionPercentage;
run;
```

The SAS code to create the "X" DataFrame is similar, with the difference being the variables specified in the KEEP statement.

The statsmodels method that we use later to perform the regression analysis does not include the intercept in the model, which is like running PROC REG and specifying the NOINT option. We specify that we want the intercept to be included by adding a constant to the regressor variables. This provides regression results that are comparable to running PROC REG with default options.



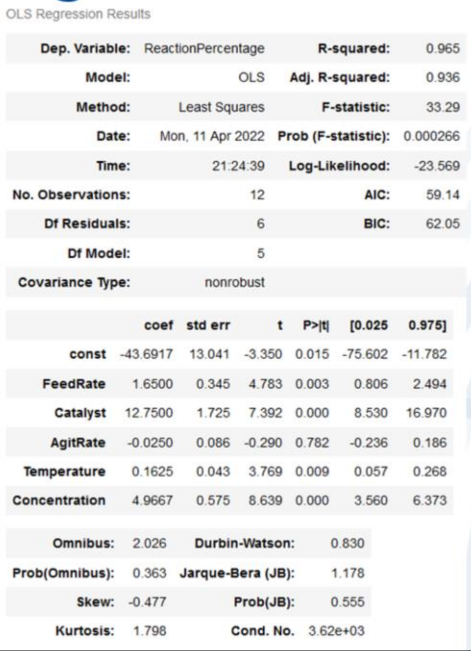
Perform the Regression - 2 of 3

```
* Perform the regression analysis (requires a SAS/STAT
license);

proc reg data=work.reaction;
  model  ReactionPercentage=FeedRate Catalyst AgitRate
                            Temperature Concentration;
run; quit;
```

```
model   = sm.OLS(y, X) # Describe the model
results = model.fit()  # Fit the model

results.summary() # Display the results

# Or sm.OLS(y,X).fit().summary()
```

The statsmodels "OLS" method creates an Ordinary Least Squares model object based on the specified dependent and regressor variables. The "model" object's "fit" method performs the regression and stores the results in the "results" object. We use the "summary" method of the "results" object to display the results.

Perform the Regression - 3 of 3



OLS Regression Results

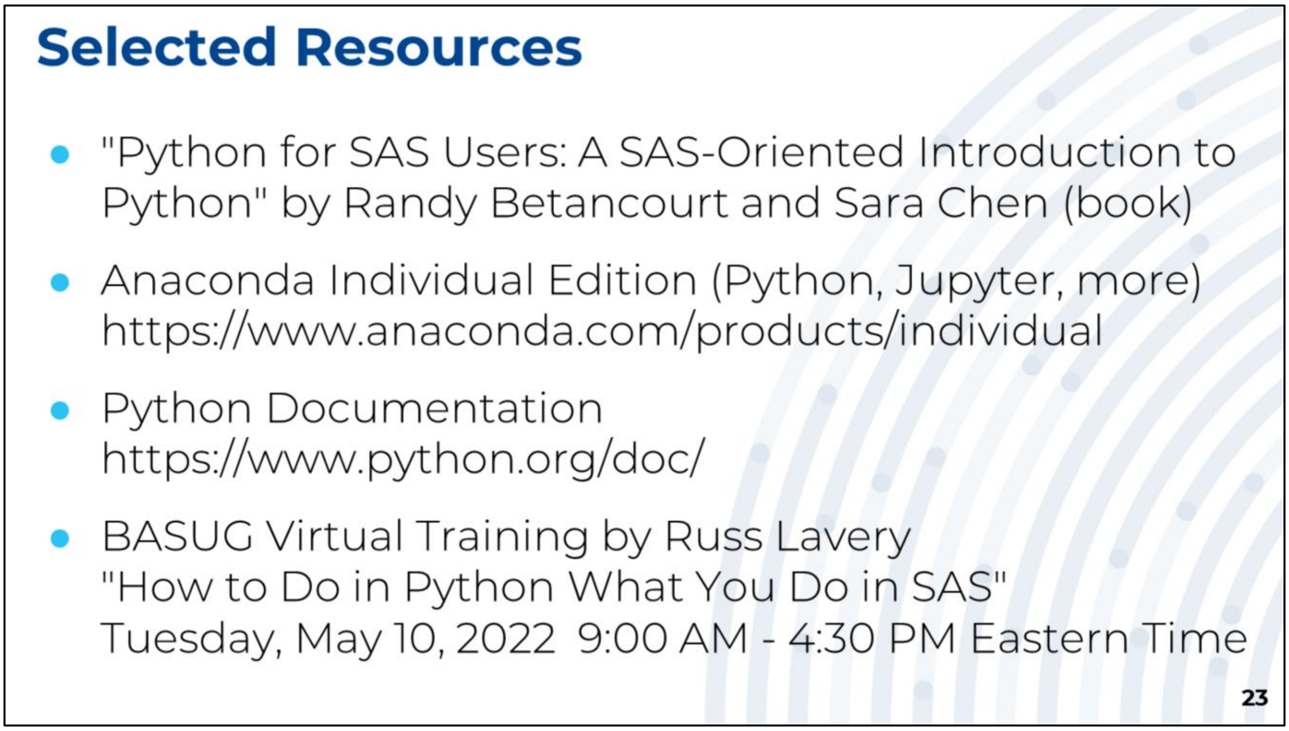
Dep. Variable:	ReactionPercentage	**R-squared:**	0.965
Model:	OLS	**Adj. R-squared:**	0.936
Method:	Least Squares	**F-statistic:**	33.29
Date:	Mon, 11 Apr 2022	**Prob (F-statistic):**	0.000266
Time:	21:24:39	**Log-Likelihood:**	-23.569
No. Observations:	12	**AIC:**	59.14
Df Residuals:	6	**BIC:**	62.05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-43.6917	13.041	-3.350	0.015	-75.602	-11.782
FeedRate	1.6500	0.345	4.783	0.003	0.806	2.494
Catalyst	12.7500	1.725	7.392	0.000	8.530	16.970
AgitRate	-0.0250	0.086	-0.290	0.782	-0.236	0.186
Temperature	0.1625	0.043	3.769	0.009	0.057	0.268
Concentration	4.9667	0.575	8.639	0.000	3.560	6.373

Omnibus:	2.026	**Durbin-Watson:**	0.830
Prob(Omnibus):	0.363	**Jarque-Bera (JB):**	1.178
Skew:	-0.477	**Prob(JB):**	0.555
Kurtosis:	1.798	**Cond. No.**	3.62e+03

Summary and Conclusion

- Python is a very popular language.
- Lots of useful functionality.
- It's used in your organization or likely will be used soon.
- Good idea to add it to your skill set to help you to do your job better, and make you more marketable.
- Different programming style than SAS, but you can learn it.



Selected Resources

- "Python for SAS Users: A SAS-Oriented Introduction to Python" by Randy Betancourt and Sara Chen (book)
- Anaconda Individual Edition (Python, Jupyter, more) https://www.anaconda.com/products/individual
- Python Documentation https://www.python.org/doc/
- BASUG Virtual Training by Russ Lavery "How to Do in Python What You Do in SAS" Tuesday, May 10, 2022 9:00 AM - 4:30 PM Eastern Time

"Python for SAS Users: A SAS-Oriented Introduction to Python" by Randy Betancourt and Sara Chen (book)

Anaconda Individual Edition (Python, Jupyter, more)
https://www.anaconda.com/products/individual

Python Documentation
https://www.python.org/doc/

BASUG Virtual Training by Russ Lavery
"How to Do in Python What You Do in SAS"
Tuesday, May 10, 2022 9:00 AM - 4:30 PM Eastern Time
https://www.basug.org/event-details/how-to-do-in-python-what-you-do-in-sas-by-russ-lavery

Contact Information

- vincent.delgobbo@gmail.com
- https://www.linkedin.com/in/vince-delgobbo/

24

About the Presenter

After a successful career at SAS Institute, Vince DelGobbo decided to explore new opportunities as an independent consultant. In this role, he is especially interested in the areas of using SAS to develop Business Intelligence/Business Analytics reports and applications, migrating SAS 9 applications to SAS Viya, and interoperability or migration between SAS and Microsoft Excel, Python, and Microsoft Power BI. While at SAS he worked on developing the SAS/IntrNet, Stored Process, and Viya Job Execution Web-enablement technologies, as well as SAS and Microsoft Excel integration. He has been awarded 3 software patents, and has received numerous invitations to speak about SAS technologies at domestic and international user group conferences and customer meetings.

Contact Information

vincent.delgobbo@gmail.com

https://www.linkedin.com/in/vince-delgobbo/