

# “Cracking the Code”

Demystifying Programming Languages  
for Data Analytics

Kyle Ferris, John Maddrey

Advanced Analytics Research Group  
Tecalote Research, Inc.



# Presentation Outline



Unveiling the Shroud of Complexity



The “Problems” with Programming



Addressing the Problem



Tackling the Problem



Programming Languages: Give Me the BLUF!



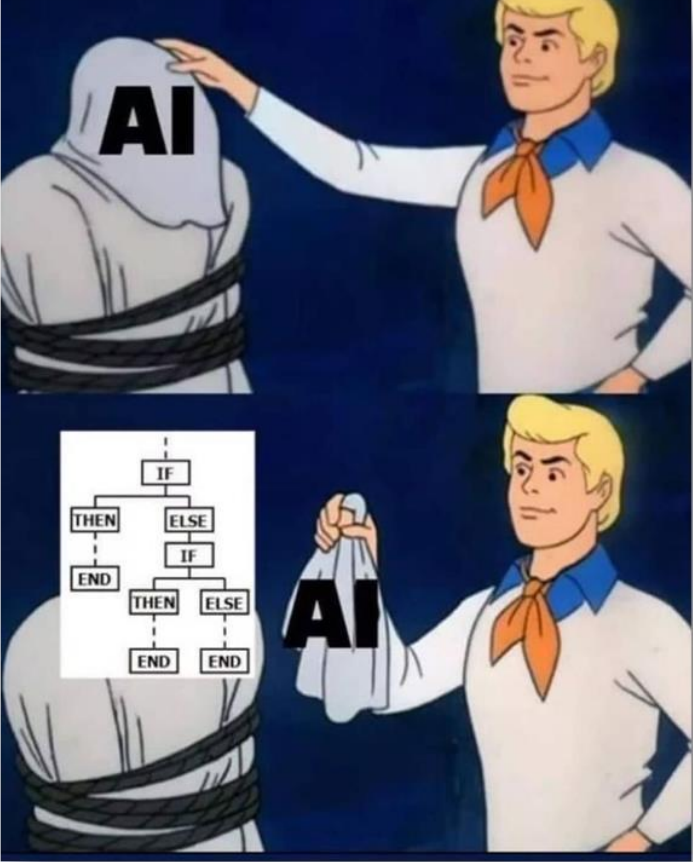
Conclusion



# Machine Learning



# Artificial Intelligence



# Big Data



# Cool Buzzwords...

- But really, how well do we understand these concepts?
  - Do we have a good grasp of their definitions, methodologies, and applicability to our work?
  - How many of us know how to code a simple function... much less develop a machine learning algorithm?
- As the acquisition community works to incorporate data science into its body of knowledge, acquisition professionals should establish a baseline understanding of their own programming capabilities





# Unveiling the Shroud of Complexity



# Why is Programming So Scary?

- From anecdotal experience:
  - “A computer *language*? I tried learning French in High School and that didn’t go very well...”
  - “What are all those random words in parentheses? Why are they all indented differently?”
  - “How long will it take me to memorize all these blocks of code?”
  - “Don’t I have to get a degree in computer science to *really* learn how to program? I don’t know if I have the time to become a software engineer”
- At some point in time, we have all thought along these lines

```
1 C A weird program for calculating Pi written in Fortran.
2 C From: Fink, D.G., Computers and the Human Mind, Anchor Books, 1966.
3
4 PROGRAM PI
5 DIMENSION TERM(100)
6 N=1
7 TERM(N)=((-1)**(N+1))*(4./(2.*N-1.))
8 N=N+1
9 IF (N-101) 3,6,6
10 N=1
11 SUM98 = SUM98+TERM(N)
12 WRITE(*,28) N, TERM(N)
13 N=N+1
14 IF (N-99) 7, 11, 11
15 SUM99=SUM98+TERM(N)
16 SUM100=SUM99+TERM(N+1)
17 IF (SUM98-3.141592) 14,23,23
18 IF (SUM99-3.141592) 23,23,15
19 IF (SUM100-3.141592) 16,23,23
20 AV89=(SUM98+SUM99)/2.
21 AV90=(SUM99+SUM100)/2.
22 COMANS=(AV89+AV90)/2.
23 IF (COMANS-3.1415920) 21,19,19
24 IF (COMANS-3.1415930) 20,21,21
25 WRITE(*,26)
26 GO TO 22
27 WRITE(*,27) COMANS
28 STOP
29 WRITE(*,25)
30 GO TO 22
31 FORMAT('ERROR IN MAGNITUDE OF SUM')
32 FORMAT('PROBLEM SOLVED')
33 FORMAT('PROBLEM UNSOLVED', F14.6)
34 FORMAT(I3, F14.6)
35 END
36
```



# Demystifying Computer Programming

- “A computer language?”
  - **Programming languages are easier to learn than human languages.** There is certainly a “vocabulary” (e.g., objects) and “grammar” (e.g., syntax). However, a programming language provides explicit commands to facilitate the execution of computational tasks, without all of the cultural or historical nuance you might expect from, say, French
- “How long will it take me to memorize all these blocks of code?”
  - **You don’t need to.** Programming isn’t centered around rote learning. It’s more important to understand the objective you are trying to accomplish, and the types of commands you can use to accomplish this objective. If you can’t remember the code syntax, just Google it!
- “Don’t I have to get a degree in computer science to really learn how to program?”
  - **Absolutely not.** This is akin to saying “Don’t I have to get an English degree to really learn how to write stories?”. A formal education in computer science will certainly help, but programming is a technical skill that anyone can learn!



# Demystifying Programming Languages

- A common misconception of would-be programmers is that it can take numerous years to develop proficiency in a specific language
  - Following this rationale, it's easy to believe that learning a *variety* of programming languages can take decades
  - After all, why else do software engineers or data scientists command such high salaries?
- Learning to program is certainly not “easy”, but it's also not as time-consuming as you might think:

Learning Style	Definition	Duration
Casual Learner	“I'm not in a rush, just learning at a consistent pace”	2-3 years
Career Advancer	“This will help me with my work”	1-2 years
Career Changer	“I need this to qualify for a new job”	6 months – 1 year
Coding Bootcamp	“I'm drinking water from a fire hose!!!”	3 months

*Some industry recognized “rules of thumb” for developing programming proficiency*





# The “Problems” with Programming



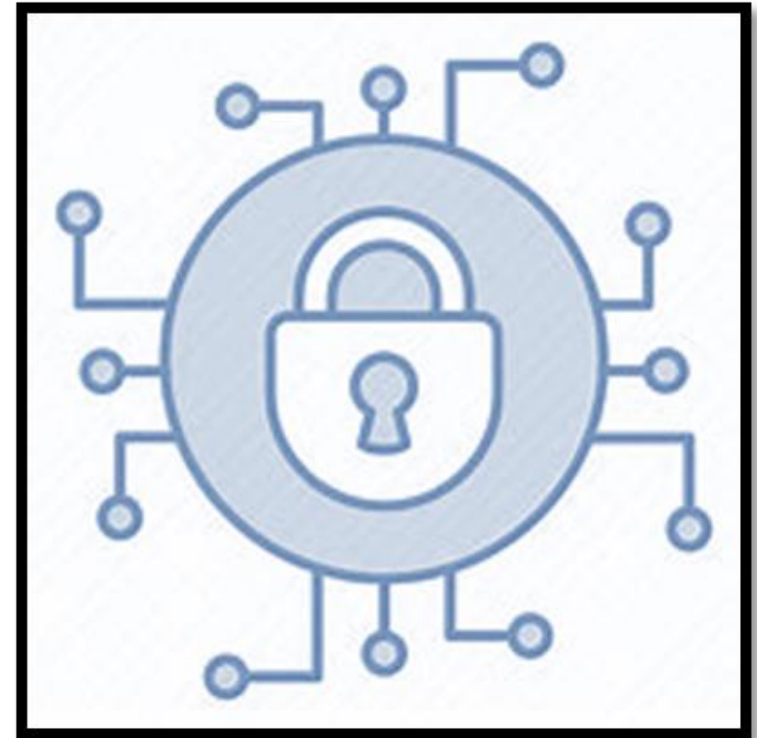
# Yes - Programming Can Be Difficult

- But... what actually makes programming languages difficult to learn?
- Several factors contribute to this:
  - Confusion about which learning approach to follow
  - Not understanding which available resources are relevant for you
  - Starting with a code base or a programming concept that's too advanced
  - Attempting to memorize code vs. understanding the underlying commands being executed
- In short, learning how to program can feel overwhelming!



# Programming in a Secure Environment

- Navigating enterprise IT requirements can be a challenge
  - Software requirements for an integrated development environment (IDE) might not align with approved software versions or configurations as defined by IT policies
  - Certain open-source packages/add-ons may not be allowed, which can limit analysis and/or development activities
  - Similarly, enterprise IT architecture might not be properly configured to facilitate developer privileges needed for open-source data collection
- Open-source accessibility vs. cybersecurity requirements can often be at odds





# Addressing the Problem



# A Framework to Address Anxiety

## Example Problem Statement:

I have unstructured data that I want to analyze and understand

I have customers that want me to “tell a story” with the data

I’m new to programming, and have anxiety around being able to use a programming language to structure the data and understand it

## High Level Solutions:

As with anything, structure your approach. What is your end goal for this data?

Determine what about your problem requires a programming language vs. what can be solved through other known tools

Rely on your skill sets, rather than the skill sets you think your customer wants.

Manage expectations – i.e., “under-promise and over-deliver”



# Addressing the Problem

- It's important to note that not all data analysis problems call for the utilization of a programming language
  - As analysts, remember that we are ultimately responsible for developing *practical* solutions and reports, with no obligation to develop fancy models that may end up being unnecessary
  - No matter what tools you end up using, always start with a question to answer
- One common pitfall of the budding data scientist is to provide awesome data driven answers to questions no one is asking
  - Generally, it is better to have a question in search of data, then have data in search of a question
  - Your analysis should always be centered on your ability to answer a given question. Let that be your anchor
- Understanding the problem and your ability to solve it is generally agnostic to any programming language or data you can use







# Tackling the Problem



# Get to Work!

- Lets assume you now have a well understood question, and you have access to unstructured datasets that can help you answer it
- You may think that structuring datasets and preparing your analysis using a programming language looks something like this:

1.	Code a program to scrape the data source for relevant data
2.	Run a standard tidying script of the data (based on data structure)
3.	Run a series of standard statistical algorithms
4.	Run standard unsupervised learning algorithms
5.	Compile your results into a nicely knitted HTML format
6.	After a solid hour of work, put a lid on it
7.	Pat yourself on the back for a job well done

- Looks efficient, right?
  - In practice, analytical programming is never this simple...



# ... But Work Smart

- Generally speaking, a programming language is not the silver bullet for your task
  - No matter what tool you use, “garbage in = garbage out”
  - Sometimes simple tools such as Excel are the best way to analyze your data
- If using a programming language makes sense, other options exist to help structure and analyze large datasets more efficiently
  - Don’t reinvent the wheel! A large variety of pre-developed packages can help you run various analyses quickly and efficiently
  - Despite your proficiency in a language, your patience to code will most likely be tested. Expect roadblocks, as a majority of time spent coding involves debugging
- Regarding programming syntax, don’t dwell on what you don’t know
  - If you’re unsure about a code’s syntax, remember that Google is your friend!
  - Online communities like Stack Overflow often have an answer to your question
  - The recent advent of Large Language Models (LLMs) such as ChatGPT likewise offer assistance with code templates and debugging



# Programming for Data Analytics

- Numerous programming languages offer similar tools for building, analyzing, and modeling datasets
  - Try not to fret over the right language to use. Again, focus on the question at hand
- Before choosing a language to use, start with a few basic steps:
  - Visualize your data. A scatterplot is worth a thousand data points!
  - Evaluate the statistics of your dataset (i.e., mean, standard deviation, coefficient of variation,  $R^2$ , etc.) to determine its usefulness
- At this point, we can choose the language that best suits our analytical requirements
  - Does data need to be structured to facilitate dynamic visualization?
  - Do we need to develop an automated model that calculates numerous iterations, and aggregate the results for predictive analytics?
  - Do we need to develop a dashboard or app that allows end-users to filter data inputs/outputs or upload unique datasets for automated analytics?



# Programming for Data Analytics

## Example Problem:

I have **unstructured cost data** on HVAC components that I want to normalize and analyze. My customer wants me to use this historical data to **develop a predictive model** for future HVAC component costs, complete with a risk calculation that **aggregates 5000 unique iterations** of defined uncertainty/risk parameters



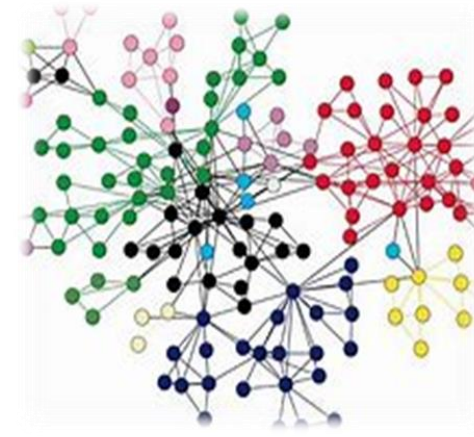
Design Loads		Equipment	
Design Heating Load:	48000 Btu/hr	Water Heater:	Natural Gas
Indoor Design Temperature:	70 °F	Heating:	90% Gas Furnace
Outdoor Design Temperature:	3 °F	Cooling:	13 SEER 2006 Minimum
Design Cooling Load:	24000 Btu/hr		
Indoor Design Temperature:	78 °F		
Outdoor Design Temperature:	90 °F		
Utility Rates		Operating Cost	
Heating Electric Rate:	0.12 \$/kwh	Heating	
Cooling Electric Rate:	0.12 \$/kwh	Natural Gas:	744 ccf
Natural Gas Rate:	0.80 \$/ccf	Electricity:	604 kWh
		Annual Cost:	\$668
		Hot Water	
		Natural Gas:	474 ccf
		Annual Cost:	\$379
		Cooling	
		Electricity:	1924 kWh
		Average Efficiency:	13.6 Btu/Watt
		Annual Cost:	\$231
		Constant Fan Annual Cost:	\$77
Total Annual Operating Costs			
		Heating:	\$668
		Cooling:	\$231
		Constant Fan:	\$77
		Hot Water:	\$379
		Total <sup>12,3</sup> :	\$1,354



# Programming for Data Analytics

## Example Solution – Step 1

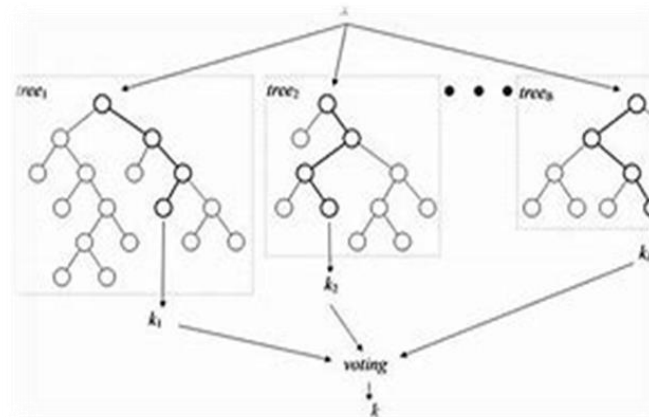
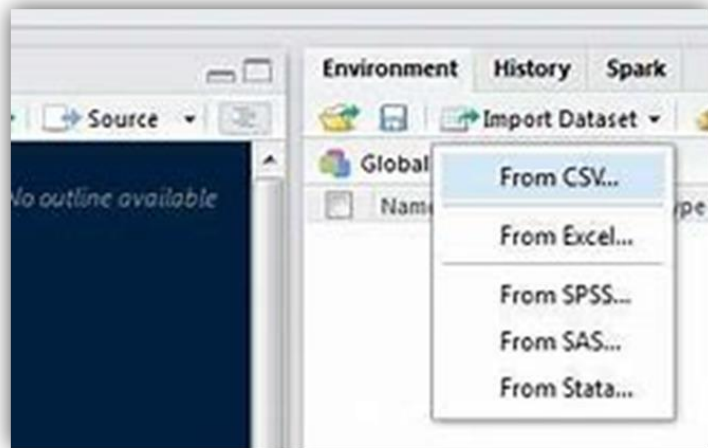
I realized I could use **Excel** to normalize and structure my data using a pivot table, which gave me a clear picture of historical component types, the year they were developed, and the associated development costs incurred



# Programming for Data Analytics

## Example Solution – Step 2

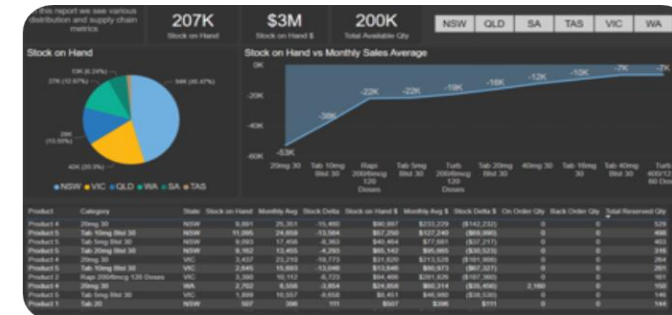
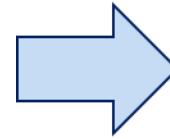
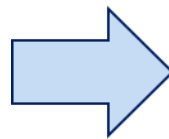
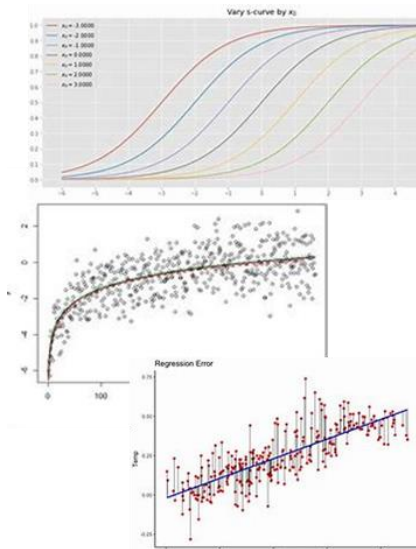
I came to the conclusion that a predictive model with 5000 automated uncertainty iterations required an algorithm too complex and time consuming for Excel. As a result, I imported my structured Excel data as a CSV file into **RStudio**, and developed an automated iteration algorithm using tools provided by R's "purrr" package



# Programming for Data Analytics

## Example Solution – Step 3

I visualized my predictive model with calculated risk using R's "ggplot2" package, but found the visualizations to be too "busy". As a result, I decided to pipeline my R script into **Power BI** to develop a dynamic dashboard, complete with a data slicer that visually compares all child-level historical and projected HVAC component costs by individual fiscal years



```
1 # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:
2
3 # dataset <- data.frame(Year, Quarter, Month, Day, BackupType, Database)
4 # dataset <- unique(dataset)
```







# Programming Languages: Give Me the BLUF!



# So Many Choices...

- We've discussed some challenges associated with learning to program, high level ways to address these challenges, and actionable steps we can take towards understanding how to incorporate programming into our analytical work
- This begs the question – ***based on my requirements, what language should I use?***
- Some related, and important, questions include:

What are the most popular programming languages?

What language is best suited for the data I work with?

What languages are my customer using and/or asking for?

Are there specific skills I need to become proficient in a certain language?

Which languages are open-source? Which are COTS?

Am I allowed to use certain open-source languages within my organization?



# The “Quick” Answers

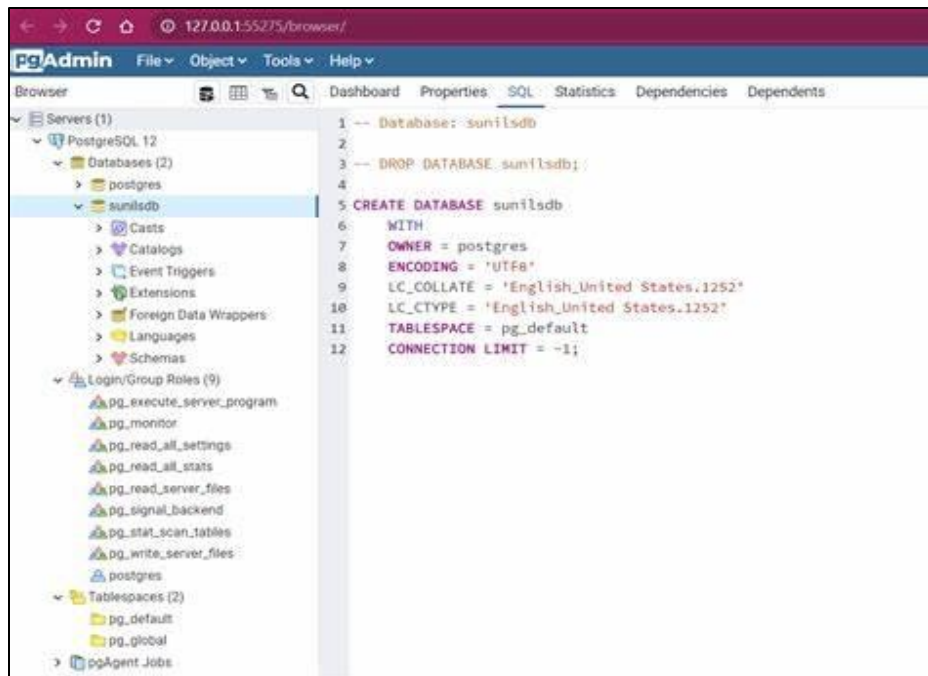
- There are a wide variety of programming languages, each generally comparable in capabilities yet well suited for uniquely specialized tasks
- For data analysis, the most popular are arguably SQL, Python, and R
  - **SQL** is designed for database management, analysis and reporting
  - **Python** is ideal for generalized workflow automation, data analytics and machine learning
  - **R** is ideal for mathematical/numerical analysis, modeling and algorithm development
- It is important to note, however, that ***all of these languages are practical and useful***, with each featuring unique benefits and limitations
  - In addition to your objective, the decision to use one language over the other often comes down to dedicated resources, availability, and customer preference or familiarity



# Structured Query Language (SQL)

When it comes to database management and reporting, SQL is the industry standard

- Dynamic relational databases are necessary for structuring large volumes of data, making a database language such as SQL very important
- SQL is a “non-procedural language”. This means that it does not require the use of traditional programming logic, making SQL *more straightforward and accessible compared to other languages*
- Anyone managing, manipulating, or analyzing data from a relational database should be proficient in using SQL



IF/ELSE statement in SQL to check if the text string “*Sample SQL text*” contains the substring “*SQL*”:

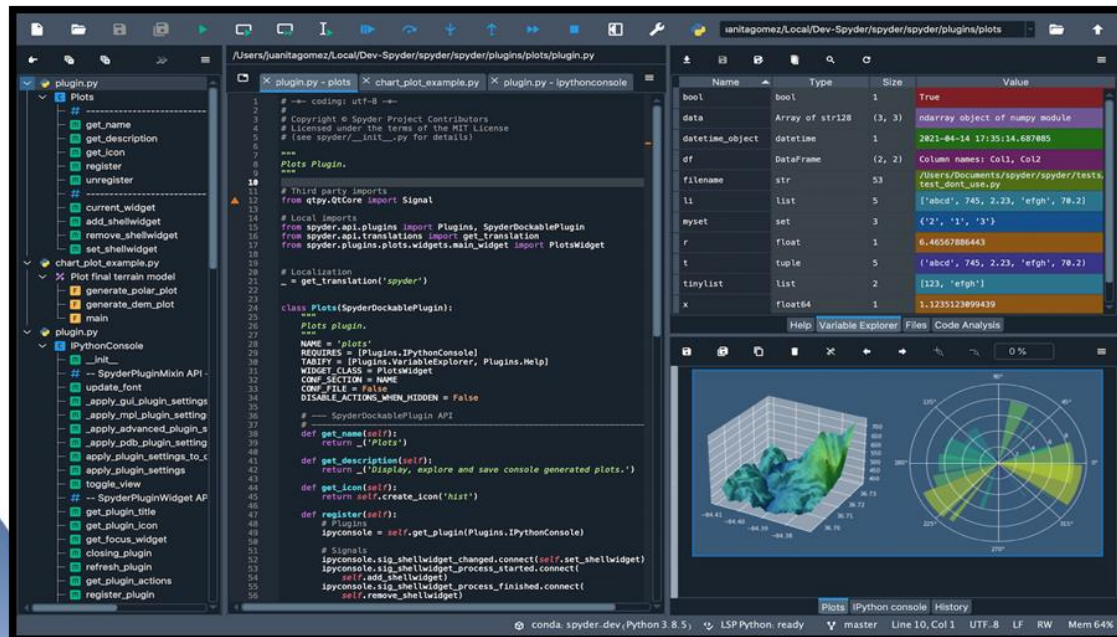
```
IF (CHARINDEX ('SQL',
'Sample SQL text') > 0)
    PRINT 'TRUE'
ELSE
    PRINT 'FALSE'
```



# Python

## Python is arguably the most popular programming language across all STEM fields

- Python is often the first choice for tasks including software development, data engineering, machine learning, and data analytics
- Python's appeal is in its **general usability, flexibility and simplicity – without sacrificing computational power**. Whether you need to develop web-based applications or perform predictive analytics, Python can help you
- Python is “object-oriented”, meaning functions and workflows are centered around defined objects with assigned values. This makes Python workflows incredibly flexible and intuitive



IF/ELSE statement in Python to check if the text string “**Sample Python text**” contains the substring “**Python**”:

```
if 'Python' in 'Sample Python text':  
    print ('TRUE')  
else:  
    print ('FALSE')
```



# R

## R is a comprehensive programming language that supercharges mathematical operations

- R specializes in large and complex numerical data sets, *especially powerful when performing mathematical analysis and modeling*
- Similar to Python, R is an object-oriented language with numerous benefits – including open-source availability, large community of users, high quality visualizations, and impressive algorithmic capabilities
- R is quickly rising the ranks as one of the most popular programming languages for numerical data analysis, most utilized within the mathematics, economics, physical engineering, and medical communities

The screenshot shows the RStudio interface. The script editor contains R code for data manipulation and visualization. The console shows an error message: "Error in ztable(as.data.frame(dfr\_dist), digits = 1, caption = 'Distribution of Gears by Cylinders') : não foi possível encontrar a função 'ztable'". Below the error, the user has installed the 'ztable' package. The Environment pane shows the loaded objects: df\_mtcars, dfr, dfr\_dist, dfr\_perc, dfr\_prop, m, and ztab. A preview of the 'ztab' object is shown as a table.

```
dfr_dist <- dfr %>%
  count(cyl) %>%
  mutate(c(\\W) = prop.table(n)*100) %>%
  left_join(dfr_perc, by = 'cyl')
# 1 digit to distinguish % from n and caption
ztab <- ztable(as.data.frame(dfr_dist), digits = 1
  , caption = 'Distribution of Gears by Cylinders')
# name and number column groups
ztab <- addcgroup(ztab, cgroup = c('Cylinders', 'Gear Distribution (\\W)')
  , n.cgroup = c(3, ncol(dfr_dist)-3)) # 3 columns & other
ztab
```

Cylinders	Gear Distribution (%)		
cyl n (%)	3	4	5
1 4 11 34.4	3.1	25.0	6.2
2 6 7 21.9	6.2	12.5	3.1
3 8 14 43.8	37.5		6.2

Vectorized IF/ELSE statement in R to check if the text string "Sample R text" contains the substring "R":

```
result <- ifelse(grepl('R', 'Sample R text'),
  'TRUE' , 'FALSE')
```

```
print(result)
```



# Conclusion



# Lessons Learned

- Learning to program can be challenging. Learning to apply programming skillsets can be even more challenging
  - *However*, don't let this discourage you!
  - The time you dedicate to developing programming proficiency will pay huge dividends through amazing opportunities for advanced analytics and modeling (... not to mention highly marketable skillsets!)
- Motivate yourself by aligning learning to the skillsets you need
  - When starting out, don't focus on trying to learn advanced programming concepts as those skillsets come with time and practice. Walk before you run!
  - Learn what is practical for you. For example, you don't need to learn how to develop complex machine learning algorithms if you're primarily concerned with data visualization
- Recognize small achievements that lead to growth over time
  - "Rome wasn't built in a day"
  - Don't set unrealistic goals. No one is expecting you to become a programmer overnight!





# Lessons Learned

- Learning how to work with unstructured data and new programming languages is a process
  - As you develop programming competencies, you'll gradually start to understand more advanced concepts as well as how to incorporate them into your work
  - Don't use high stress "hot tasks" to practice programming capabilities. Start with something low stakes, so you can take the time you need to develop a quality product
- Be mindful of your own interests and map out what you want to get out of programming
  - After all, the best way to get good at something is to feel fulfilled with the work involved
- Make sure to ask for help!
  - **Don't be afraid to "suck" at something new**
  - The only "bad" question is the one you don't ask. Also, search engines don't judge 😊
  - Online communities most likely have an answer to your questions, especially if you're a beginner





# THANK YOU



# References

Tom Dalling. *Programming for Beginners (2020 edition)*.

[https://www.programmingforbeginnersbook.com/blog/expand\\_your\\_programming\\_vocabulary/#:~:text=A%20List%20of%20Basic%20Programming%20Terms%201%20algorithm,often%20used%20to%20surround%20text.%20...%20More%20items](https://www.programmingforbeginnersbook.com/blog/expand_your_programming_vocabulary/#:~:text=A%20List%20of%20Basic%20Programming%20Terms%201%20algorithm,often%20used%20to%20surround%20text.%20...%20More%20items)

“How Long Does it Take to Learn Code?”. Code Academy.

<https://www.codecademy.com/resources/blog/how-long-does-it-take-to-learn-to-code/>

Nicholas Gallinelli. “10 Best Data Science Programming Languages”. Flatiron School.

<https://flatironschool.com/blog/data-science-programming-languages/>

