Introduction to Python for Data Science

Microsoft Reactor | *Belkacem BEKKOUR*

#INELEC

Microsoft

Reactor

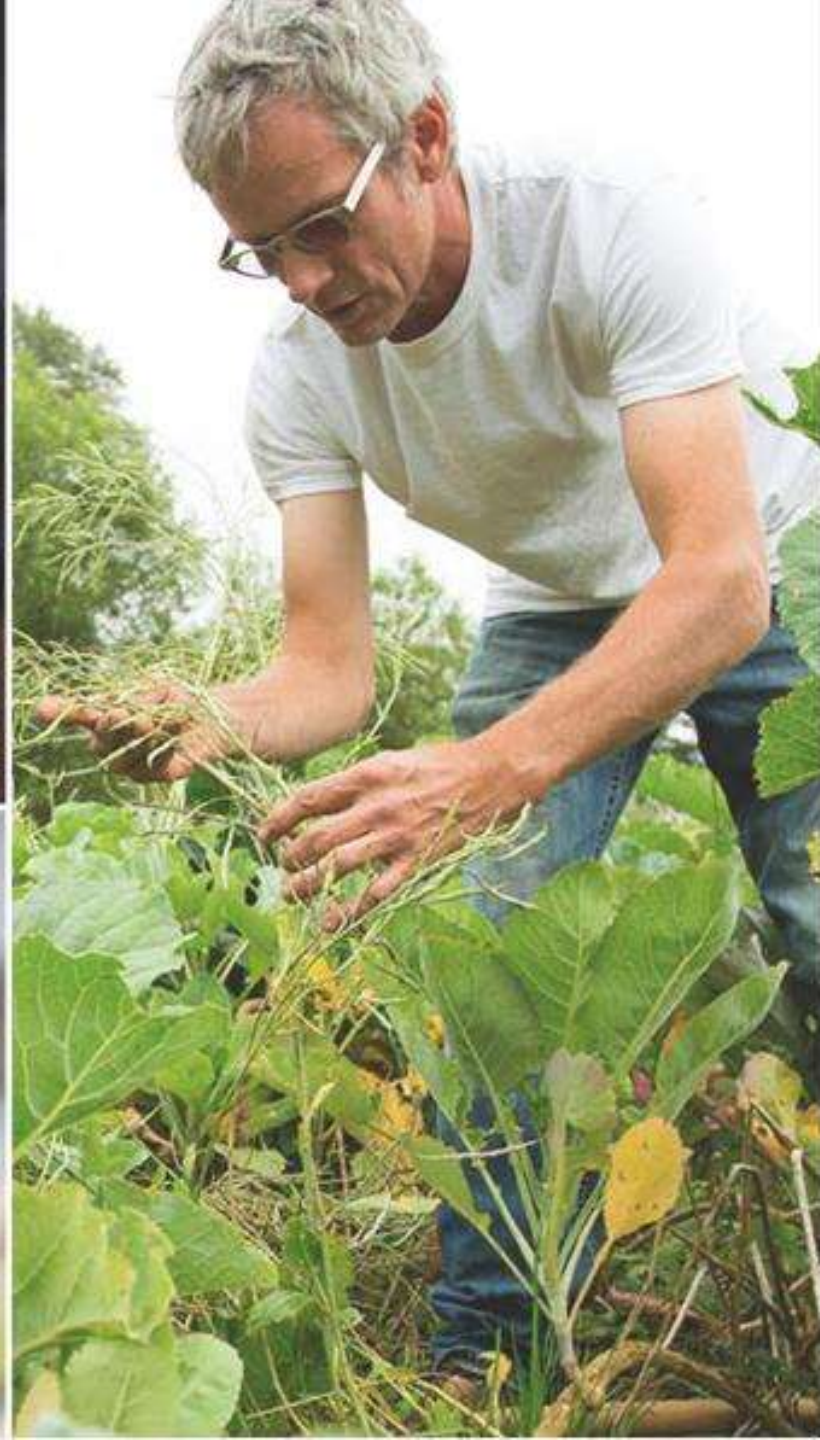
Map


Stockholm
COMING SOON
London
Redmond
Toronto
New York
San Francisco
Tel Aviv
Shanghai
Abu Dhabi
Bangalore
Sao Paulo
Sydney

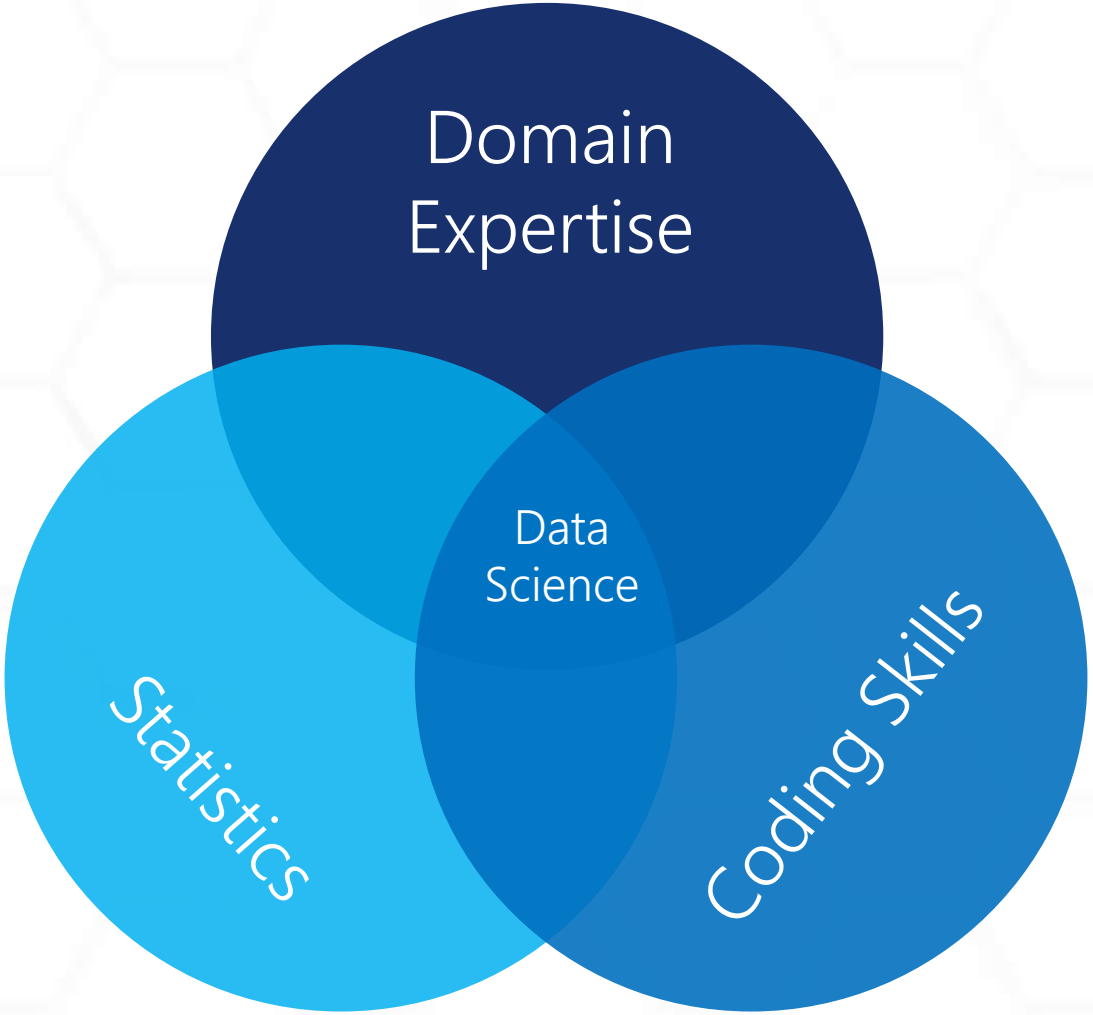
Agenda

1	Introduction to Data Science Keynote
2	Python Basics
3	NumPy: Your Local Data Friend
4	
5	Pandas Are More than Bears: How to Import, Clean, and Store Data
6	Data Science 101: Getting your Data Ready
7	Wrap Up and Next Steps
8	
9	
10	

Each module will have presentation & interactive labs; take a break whenever you like!

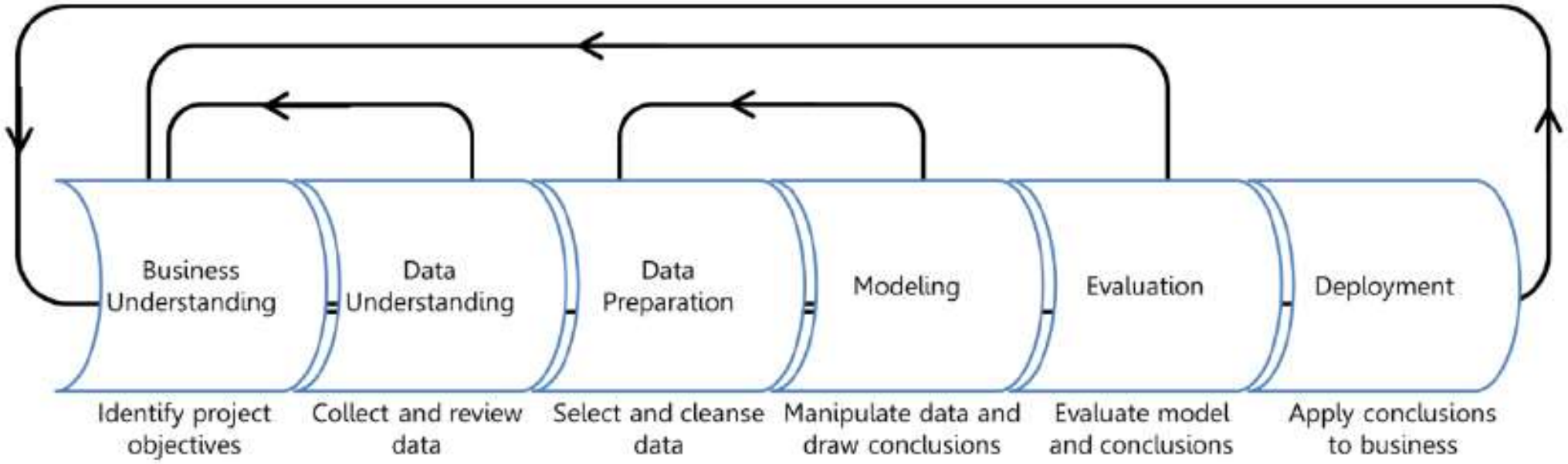


What is Data Science?



Data Science Process

Extracting value from large amounts of data **and making human sense of it** is the primary challenge of data science.



- From *Introduction to Data Science* on Microsoft Learn: https://docs.microsoft.com/learn/modules/intro-to-data-science-in-azure/2-data-science-process
- Helpful beginners' series: https://docs.microsoft.com/en-us/azure/machine-learning/studio/data-science-for-beginners-the-5-questions-data-science-answers

Specialized Roles in Data Science

Data Analyst

Data Engineer

Data Scientist

Data Architect

Developer

Data Scientist

#1 in 2018 and 2019

Median base salary (USA):

$130,000

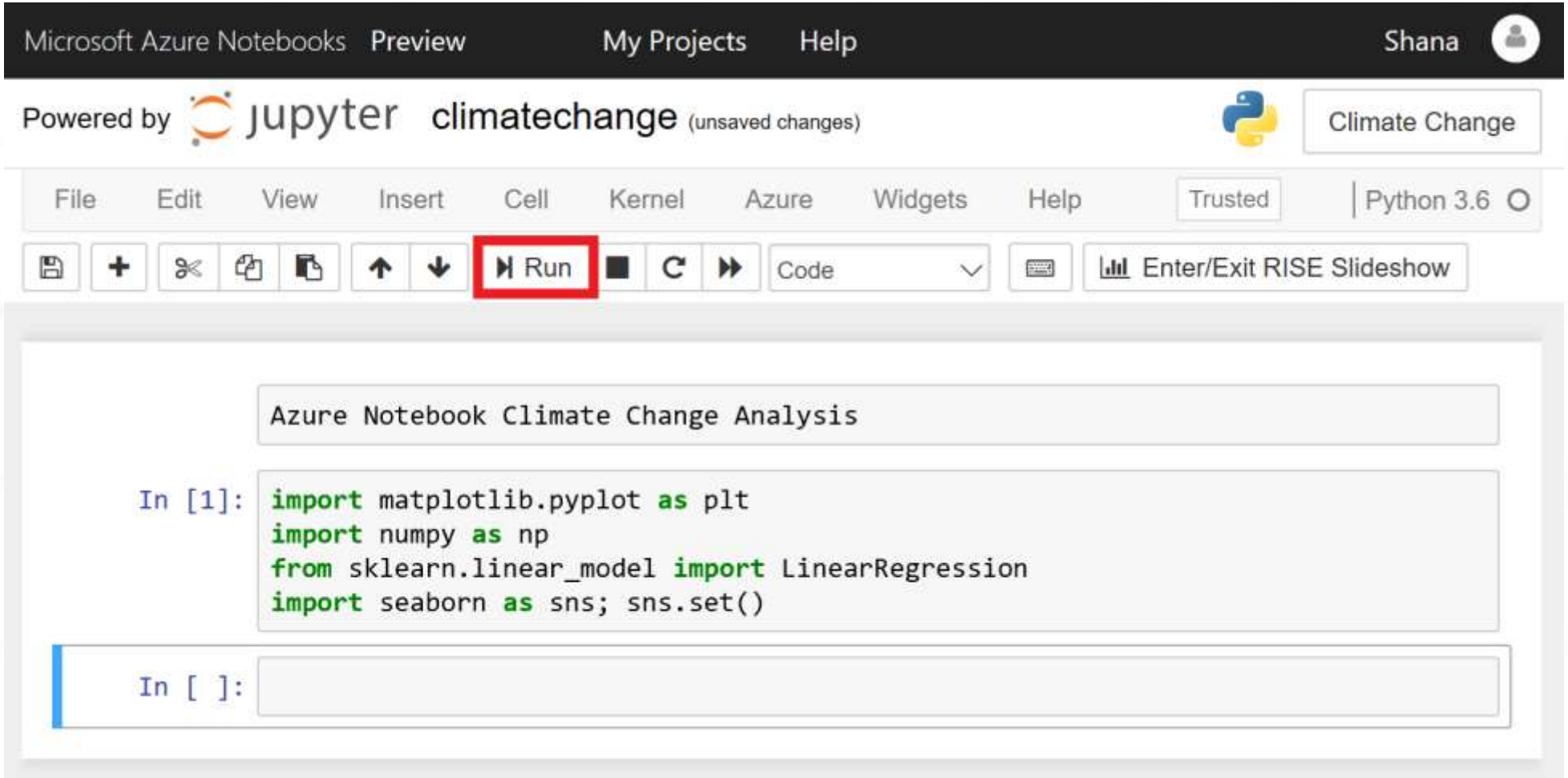
https://blog.linkedin.com/2019/january/10/linkedins-most-promising-jobs-of-2019

Let's Begin by Cloning Our Workshop Notebook

https://notebooks.azure.com/sguthals/projects/data-science-1

If you don't already have a
Microsoft Account, start here:
account.microsoft.com

How to Navigate Azure Notebooks



https://docs.microsoft.com/en-us/learn/modules/analyze-climate-data-with-azure-notebooks/1-create-an-azure-notebook

Microsoft

Reactor

Python Basics

Microsoft

Reactor

Python Basics Overview

- Arithmetic and numeric types in Python
- Strings, strings, and more strings
- Other data types
  - Lists
  - Tuples
  - Dictionaries
- Membership testing
- List comprehensions
- Importing modules

Python Basics Overview

Why Python for data science?

- Easy to learn
- Flexible
- Powerful libraries

Leading social network, video streaming, and search engine companies built some of their core technologies using Python.

They also use Python for data science.

Adjusting List Levels

Python numeric operators: + - * / // ** %

Variables:

```
length = 15
width = 3 * 5
length * width
```

```
225
```

Expressions:

```
1 < 2 or 1 > 2
```

```
True
```

Strings

String literals:

```
'"Isn\'t," she said.'
```

```
'"Isn\'t," she said.'
```

Concatenating strings:

```
3 * 'un' + 'ium'
```

```
'unununium'
```

Strings (Continued)

String indices:

```
word = 'Python'
word[0]  # Character in position 0.
```

```
'P'
```

Slicing strings:

```
word[0:2]  # Characters from position 0 (included) to 2 (excluded).
```

```
'Py'
```

Other Data Types

Lists:

```
squares = [1, 4, 9, 16, 25]
```

- List-object methods: append(), extend(), index(), count(), remove(), pop(), insert(), reverse(), sort()

Tuples:

```
t = (1, 2, 3)
```

Dictionaries:

```
capitals = {'France': ('Paris', 2140526)}
capitals['Nigeria'] = ('Lagos', 6048430)
capitals
```

```
{'France': ('Paris', 2140526), 'Nigeria': ('Lagos', 6048430)}
```

Membership Testing

'in':

```
tup = ('a', 'b', 'c')
'b' in tup
```

```
True
```

'not in':

```
lis = ['a', 'b', 'c']
'a' not in lis
```

```
False
```

List Comprehensions

Programmatically create lists:

```
numbers = [x for x in range(1,11)]
numbers
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
odd_squares = [x*x for x in range(1,11) if x % 2 != 0]
odd_squares
```

```
[1, 9, 25, 49, 81]
```

Importing Modules

```
import math
math.factorial(5)
```

```
120
```

```
from math import factorial
factorial(5)
```

```
120
```

Microsoft

Reactor

Intro to NumPy

Microsoft

Reactor

NumPy Overview

- Built-in help
- NumPy arrays
  - Creating
  - Attributes
  - Indexing
  - Reshaping
  - Splitting and joining
  - Fancy indexing
  - Sorting
- Aggregations

NumPy Overview

Why NumPy?

- Faster, pre-compiled functions
- Standard numeric library for data science using Python
- Foundation for other popular libraries (such as pandas)

The City of Los Angeles uses NumPy to predict illegal eviction of apartment tenants.

IPython and Built-in Help

- Use the tab key to explore package contents.
- Use "?" for help info related to any method.

```
import numpy as np
```

```
# Place your cursor after the period and press <TAB>:
np.
```

```
np.ALLOW_THREADS
np.alltrue
np.amax
np.amin
np.angle
np.any
np.append
np.apply_along_axis
np.apply_over_axes
np.arange
```

```
In [4]: np.array?
```

NumPy Arrays

Lists in Python:

```
myList2 = [True, "2", 3.0, 4]
[type(item) for item in myList2]
```

```
[bool, str, float, int]
```

- NumPy data types: floating point, integer, Boolean, string, general Python object

Fixed-type arrays in Python:

```
# Create an integer array:
np.array([1, 4, 2, 5, 3])
```

```
array([1, 4, 2, 5, 3])
```

Working with NumPy Arrays: Array Attributes

```
a3 = np.random.randint(10, size=(3, 4, 5))  # Three-dimensional array
```

```
# Change the values in this code snippet to look at the attributes for a1, a2, and a3:
print("a3 ndim: ", a3.ndim)
print("a3 shape:", a3.shape)
print("a3 size: ", a3.size)
```

```
a3 ndim:  3
a3 shape: (3, 4, 5)
a3 size:  60
```

Working with NumPy Arrays: Indexing

```
np.random.seed(0)
a1 = np.random.randint(10, size=6)
a1
```

```
array([5, 0, 3, 3, 7, 9])
```

```
a1[0]
```

```
5
```

```
a1[4]
```

```
7
```

```
a1[-1]
```

```
9
```

Working with NumPy Arrays: Slicing Arrays

```
a = np.arange(10)
a
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
a[4:7]  # middle sub-array
```

```
array([4, 5, 6])
```

Working with NumPy Arrays: Reshaping

```
grid = np.arange(1, 10).reshape((3, 3))
print(grid)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Working with NumPy Arrays: Joining and Splitting

```
a = np.array([1, 2, 3])
b = np.array([3, 2, 1])
np.concatenate([a, b])
```

```
array([1, 2, 3, 3, 2, 1])
```

```
a = [1, 2, 3, 99, 99, 3, 2, 1]
a1, a2, a3 = np.split(a, [3, 5])
print(a1, a2, a3)
```

```
[1 2 3] [99 99] [3 2 1]
```

Working with NumPy Arrays: Fancy Indexing

```
rand = np.random.RandomState(42)

arr = rand.randint(100, size=10)
print(arr)
```

```
[51 92 14 71 60 20 82 86 74 74]
```

```
ind = [3, 7, 4]
arr[ind]
```

```
array([71, 86, 60])
```

Working with NumPy Arrays: Sorting

```
a = np.array([2, 1, 4, 3, 5])
np.sort(a)
```

```
array([1, 2, 3, 4, 5])
```

```
a.sort()
print(a)
```

```
[1 2 3 4 5]
```

Aggregations

Summing the values in arrays:

```
myList = np.random.random(100)
sum(myList)
```

```
52.12818058833704
```

Minimum and maximum values:

```
print(large_array.min(), large_array.max(), large_array.sum())
```

```
1.4057692298008462e-06 0.9999994392723005 500202.5348847683
```

Microsoft

Reactor

Intro to pandas

Microsoft

Reactor

Intro to pandas Overview

- Data structures in pandas
  - Series
  - DataFrames
- Manipulating data
  - In Series
  - In DataFrames
- Data operations
  - Index preservation
  - Index alignment
  - Operations between Series and DataFrames

Intro to pandas Overview

Why pandas?

- Standard Python library for handling and manipulating general data
- Intuitive tabular data structure (DataFrame)
- Fast data calculation and transformation using NumPy
- Close integration with visualization libraries to easily explore data

The leading online video-streaming service uses pandas and Jupyter notebooks for data exploration, preparation, and validation.

Fundamental pandas Data Structures: Series

```
series_example = pd.Series([-0.5, 0.75, 1.0, -2])
series_example
```

```
0   -0.50
1    0.75
2    1.00
3   -2.00
dtype: float64
```

```
series_example.values
```

```
array([-0.5 ,  0.75,  1.  , -2.  ])
```

```
series_example.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

Pandas Data Structures: DataFrames

Creating a DataFrame from a dictionary:

```
population_dict = {'France': 65429495,
                   'Germany': 82408706,
                   'Russia': 143910127,
                   'Japan': 126922333}
population = pd.Series(population_dict)
population
```

```
France       65429495
Germany      82408706
Japan       126922333
Russia      143910127
dtype: int64
```

Pandas Data Structures: DataFrames

Creating a DataFrame from two series:

```
area = pd.Series({'Albania': 28748,
                  'France': 643801,
                  'Germany': 357386,
                  'Japan': 377972,
                  'Russia': 17125200})
population = pd.Series ({'Albania': 2937590,
                         'France': 65429495,
                         'Germany': 82408706,
                         'Russia': 143910127,
                         'Japan': 126922333})
countries = pd.DataFrame({'Area': area, 'Population': population})
countries
```

	Area	Population
Albania	28748	2937590
France	643801	65429495
Germany	357386	82408706
Japan	377972	126922333
Russia	17125200	143910127

Manipulating Data: Data Selection in Series

```
series_example2 = pd.Series([-0.5, 0.75, 1.0, -2], index=['a', 'b', 'c', 'd'])
series_example2
```

```
a   -0.50
b    0.75
c    1.00
d   -2.00
dtype: float64
```

```
series_example2['b']
```

```
0.75
```

```
series_example2[0:2]
```

```
a   -0.50
b    0.75
dtype: float64
```

Manipulating Data: Data Selection in DataFrames

```
countries.iloc[:3, :2]
```

	Area	Population
Albania	28748	2937590
France	643801	65429495
Germany	357386	82408706

```
countries.loc[:'Germany', :'Population']
```

	Area	Population
Albania	28748	2937590
France	643801	65429495
Germany	357386	82408706

```
countries['Area']
```

```
Albania        28748
France        643801
Germany       357386
Japan         377972
Russia      17125200
Name: Area, dtype: int64
```

Operating on Data in pandas: Index Alignment

```
series1 = pd.Series([2, 4, 6], index=[0, 1, 2])
series1
```

```
0    2
1    4
2    6
dtype: int64
```

```
series2 = pd.Series([3, 5, 7], index=[1, 2, 3])
series2
```

```
1    3
2    5
3    7
dtype: int64
```

```
series1 + series2
```

```
0     NaN
1     7.0
2    11.0
3     NaN
dtype: float64
```

```
series1.add(series2, fill_value=0)
```

```
0     2.0
1     7.0
2    11.0
3     7.0
dtype: float64
```

Microsoft

Reactor

Manipulating and Cleaning Data

Microsoft

Reactor

Manipulating and Cleaning Data Overview

- Exploring information in DataFrames
- Working with missing data values
  - Identifying
  - Removing
  - Filling
- Combining datasets
- Exploratory statistics and visualizations
- Example: Boston Housing dataset

About 80% of project time is usually spent cleaning and preparing data for the actual analysis.

Exploring DataFrame information

DataFrame.info

```
iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
dtypes: float64(4)
memory usage: 4.8 KB
```

DataFrame.head

```
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

DataFrame.tail

```
iris_df.tail()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

Dealing with Missing Data

Python null value: None

NumPy/pandas null value: NaN

```
example = pd.DataFrame([0, np.nan, '', None])
```

```
example
```

	0
0	0
1	NaN
2	
3	None

```
example.isnull()
```

	0
0	False
1	True
2	False
3	True

```
example.dropna()
```

	0
0	0
2	

```
example.fillna(0)
```

	0
0	0
1	0
2	
3	0

Removing Duplicate Data

Identifying duplicates

Dropping duplicates

```
example
```

	letters	numbers
0	A	1
1	B	2
2	A	1
3	B	3
4	B	3

```
example.duplicated()
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

```
example.drop_duplicates()
```

	letters	numbers
0	A	1
1	B	2
3	B	3

Combining Datasets

Categories of joins:

```
df3 = pd.merge(df1, df2)
```

- one-to-one, many-to-one, and many-to-many

Concatenation in NumPy:

```
np.concatenate([x, y, z])
```

Concatenation in pandas:

```
pd.concat([ser1, ser2])
```

Concatenation with joins:

```
pd.concat([df10, df11], join='inner')
```

Exploratory Statistics and Visualization

```
df = pd.read_csv('/Data/housing_dataset.csv')
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	5.33	36.2

```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.593761	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043
std	8.596783	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	3.647423	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

```
df.shape
```

```
(506, 13)
```

```
df['MEDV'].mean()
```

```
22.532806324110698
```

Microsoft

Reactor

Additional Learning Resources and Next Steps

Microsoft AI Platform

Azure AI Services

PRE-BUILT AI

- Cognitive Services

CONVERSATIONAL AI

- Bot Service

CUSTOM AI

- Azure Machine Learning

Azure Infrastructure

AI ON DATA

- Cosmos DB
- SQL DB
- SQL DW
- Data Lake

AI COMPUTE

- Spark
- DSVM
- Batch AI
- ACS
- IoT Edge

CPU, FPGA, GPU

Tools

CODING & MANAGEMENT TOOLS

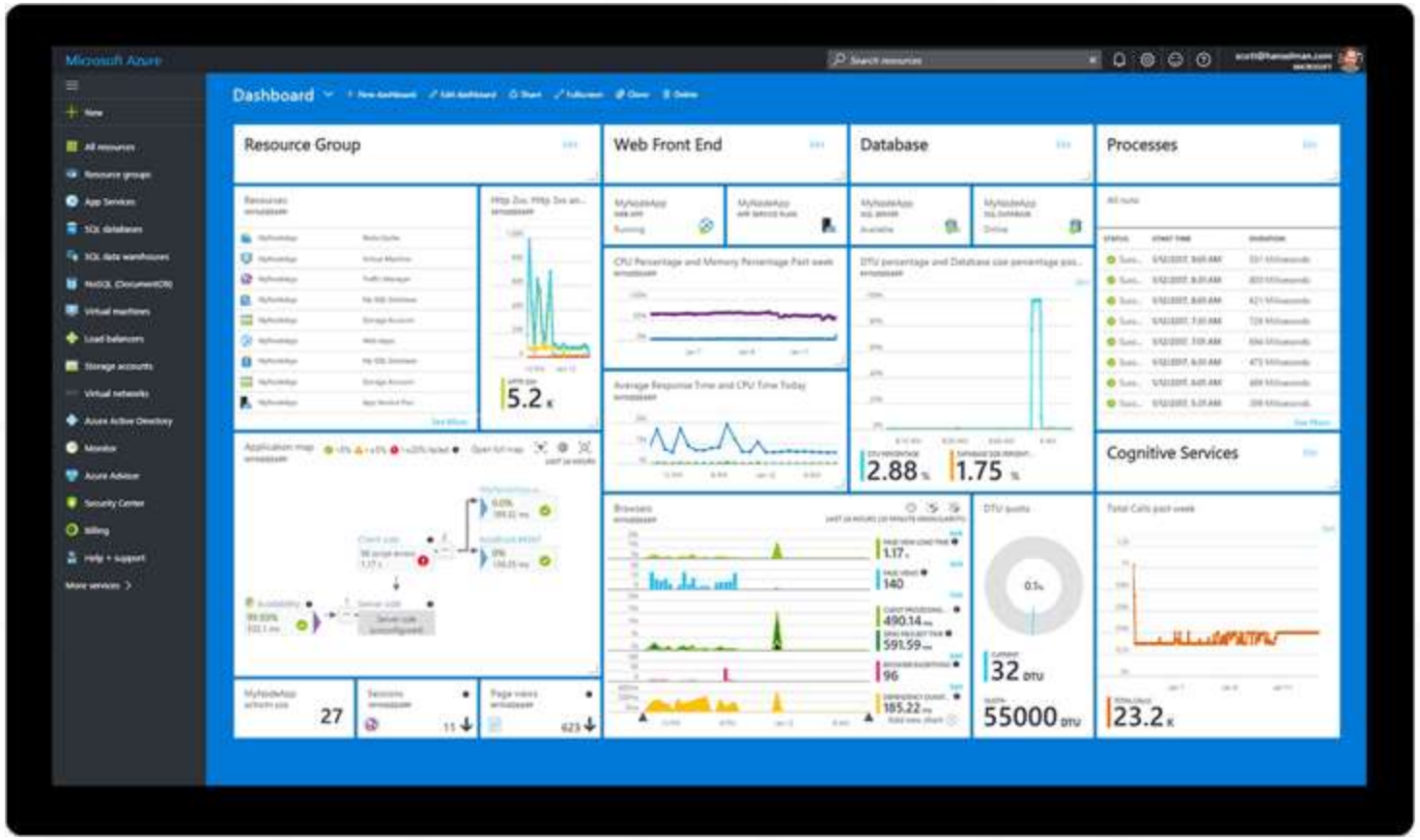
- VS Tools for AI
- Azure ML Studio
- Azure Notebooks

DEEP LEARNING FRAMEWORKS

- PyTorch
- TensorFlow
- Sci-kit Learn

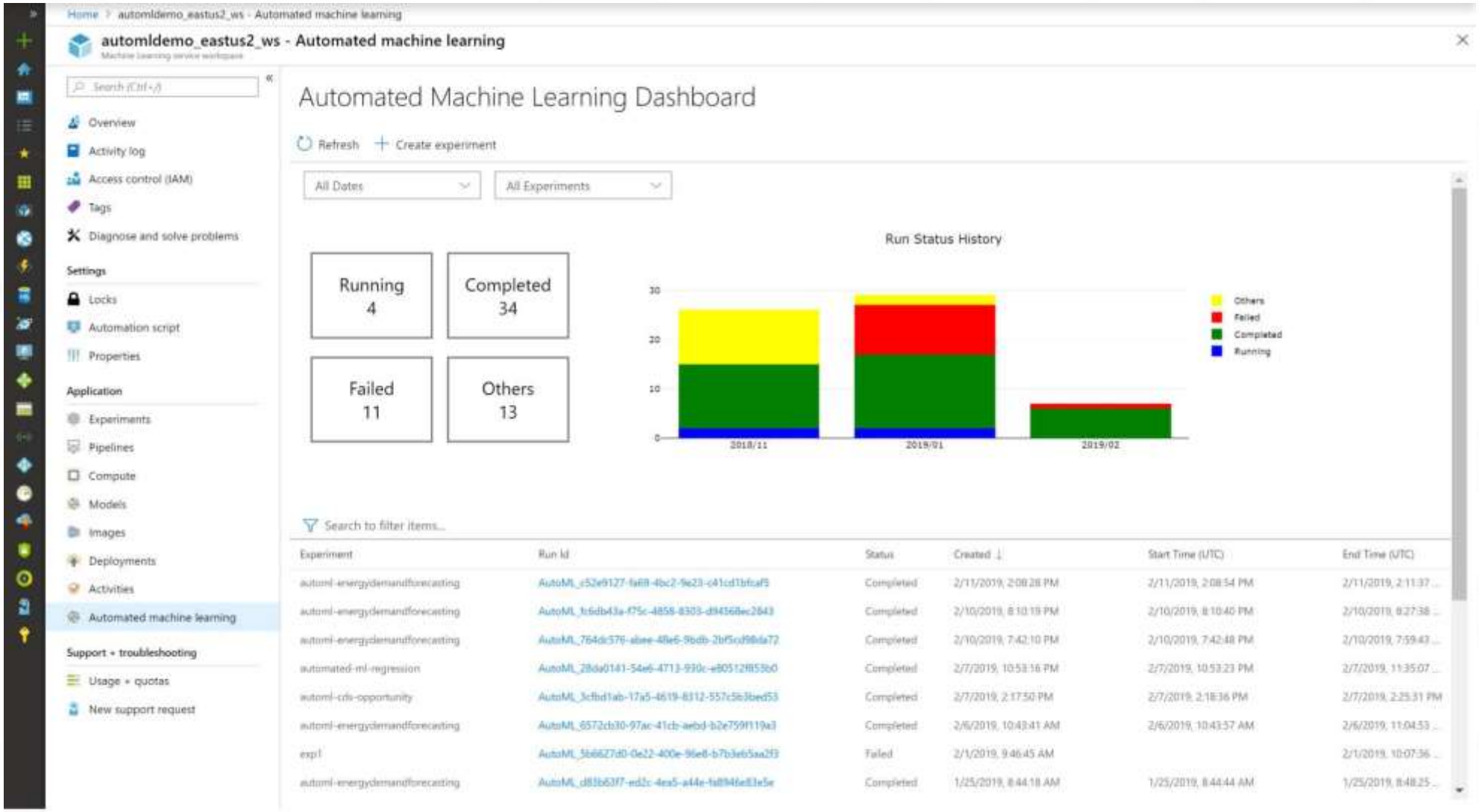
https://docs.microsoft.com/learn/modules/choose-data-science-option-in-azure/2-ml-options-on-azure

Free Azure for Your Next Experiment…

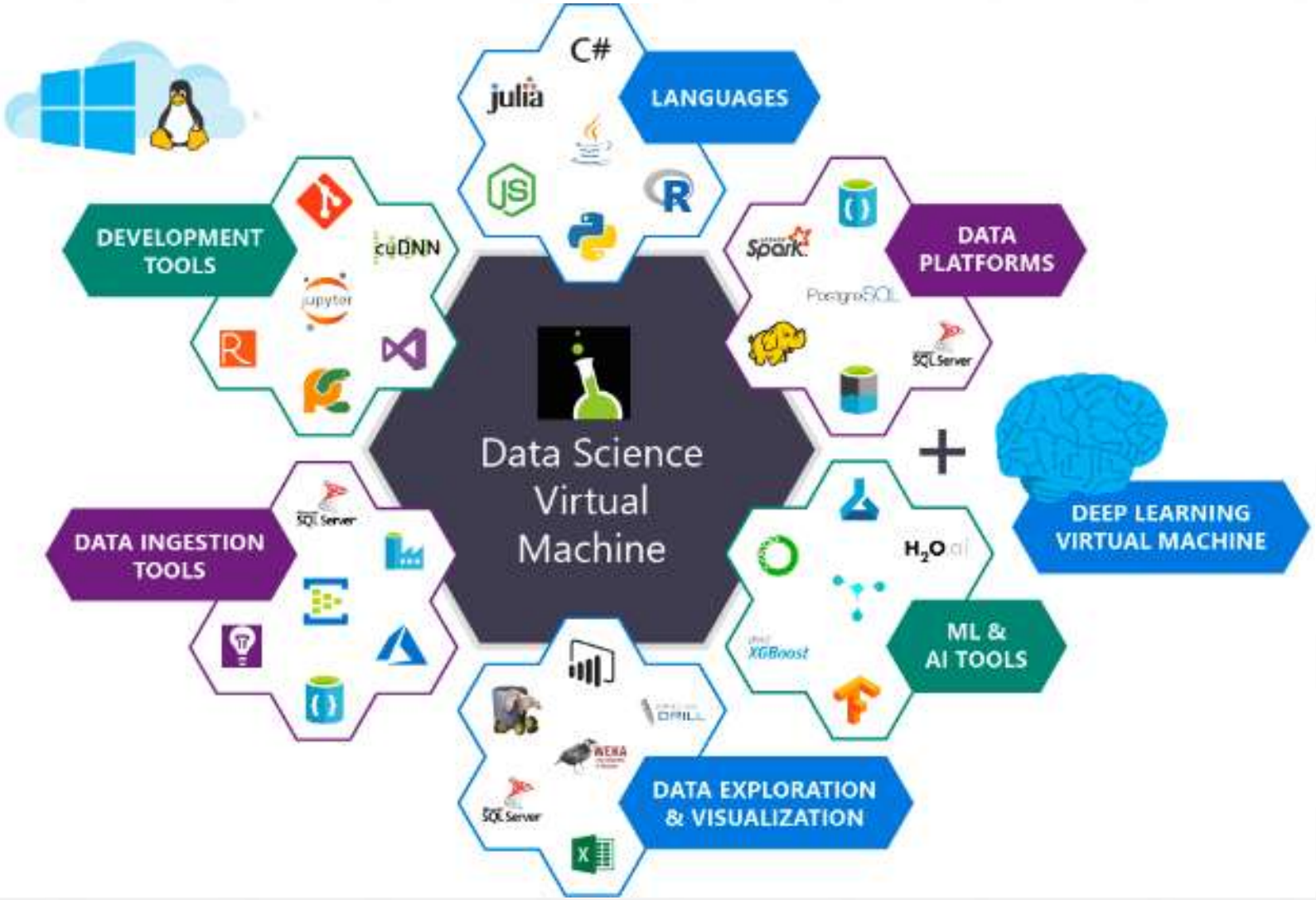


azure.microsoft.com/free/azure.microsoft.com/free/students/

Other Tools: Azure Machine Learning Services



Other Tools: Data Science Virtual Machine



Other Tools: Azure Cognitive Services
(ready-to-go, pre-trained AI models)

Decision
Build apps that surface recommendations for informed and efficient decision-making

Vision
From faces to feelings, allow your apps to understand images and videos

Speech
Hear and speak to your users by filtering noise, identifying speakers, and understanding intent

Language
Process text and learn how to recognize what users want

Knowledge
Tap into rich knowledge amassed from the web, academia, or your own data

Search
Access billions of webpages, images, videos, and news articles with the power of Bing APIs

Further Learning in Data Science

Microsoft Learn has free learning in a cloud sandbox:

- Good refresher on what we covered in this workshop:

  docs.microsoft.com/learn/paths/intro-to-ml-with-python/

- More on the data science methodology and best practices:

  docs.microsoft.com/learn/paths/explore-data-science-tools-in-azure/

- Take the next step in building ML models:

  docs.microsoft.com/learn/paths/build-ai-solutions-with-azure-ml-service/

- Experiment with the Data Science Virtual Machine:

  docs.microsoft.com/learn/paths/get-started-with-azure-dsvm/

- Learn more about what Azure can do:

  docs.microsoft.com/learn/paths/azure-fundamentals/

Role-Based Microsoft Certs

In the data sciences:

Azure Data Engineer Associate
Azure Data Scientist Associate
Azure Developer Associate

Also:

Azure Fundamentals
Azure AI Engineer Associate

Microsoft

Reactor

Belkacem.bekkour@studentambassadors.com
Belkacem BEKKOUR

Microsoft