



# Learning Bytes By Byte Academy

Webinar Series

**Introduction To Python For Data Analysis**

**March 19, 2019**

With Interactive Brokers

# Introduction to Byte Academy

Industry focused coding school headquartered in NYC with campuses in Houston, Texas and Bangalore, India

**1st Python Fullstack Full-Time Program in NYC** also known for Data Science, Blockchain, and FinTech programs

Blockchain program featured in Bloomberg

Offers **Intro Python Foundation & Python related workshops** in the evening

Check out [byteacademy.co](https://byteacademy.co)



# Questions during the webinar?

Use the Questions section of the Control Panel

We'll try the best to answer your questions after the webinar

For those tuning in, first tell us where you are from using the Control Panel



# Now, A Little Background

- Named after “Monty Python” the movie
- Based on simplicity
- Great for analyzing data
- Named “highest paid coding language for recent bootcamp grads” by Course Report
- Used by companies such as Bank of America, Google, Goldman Sachs, Dropbox and more
- Popular with non-programmers too! Citi even uses it in training their analysts!



# Python: Some features

- Python is a programming language, as are C, JavaScript, PHP, etc.
- Some specific features of Python are as follows:
  - 1) an interpreted language- code is not compiled before execution
  - 2) Python can be used interactively
  - 3) Human-readable
  - 4) Easily extensible with a massive library of modules



# Python: Some features, continued

5) Very easy to interface with other languages, in particular C and C++

6) Python is an object-oriented language, with dynamic typing (the same variable can contain objects of different types during the course of a program)



# Python Numeric Data Types

## -Integer

*e.g (1,4,2)*

## - Floating point (decimal)

*e.g (1.32,15.234324)*

## - Complex numbers

*e.g (a+bi = 1 + .5j)*

## -Booleans

*only two values (True or False)*



# Built-in Collection Types



- Lists: A list is an ordered, **mutable** collection of zero or more references to Python data objects. Lists are written as comma-delimited values enclosed in square brackets. The empty list is simply [ ].
- Lists are heterogeneous, meaning that the data objects need not all be from the same class
- `list = [1, 2, 3, 'Greg', 45.9, [12, 4]]`



# Operations on Lists



Operation Name	Operator	Explanation
indexing	[ ]	Access an element of a sequence
concatenation	+	Combine sequences together
repetition	*	Concatenate a repeated number of times
membership	in	Ask whether an item is in a sequence
length	len	Ask the number of items in the sequence
slicing	[ : ]	Extract a part of a sequence

# List Methods

- A method is a function that “belongs to” an object

Method Name	Use	Explanation
<code>append</code>	<code>alist.append(item)</code>	Adds a new item to the end of a list
<code>insert</code>	<code>alist.insert(i,item)</code>	Inserts an item at the ith position in a list
<code>pop</code>	<code>alist.pop()</code>	Removes and returns the last item in a list
<code>pop</code>	<code>alist.pop(i)</code>	Removes and returns the ith item in a list
<code>sort</code>	<code>alist.sort()</code>	Modifies a list to be sorted
<code>reverse</code>	<code>alist.reverse()</code>	Modifies a list to be in reverse order
<code>del</code>	<code>del alist[i]</code>	Deletes the item in the ith position
<code>index</code>	<code>alist.index(item)</code>	Returns the index of the first occurrence of <code>item</code>
<code>count</code>	<code>alist.count(item)</code>	Returns the number of occurrences of <code>item</code>
<code>remove</code>	<code>alist.remove(item)</code>	Removes the first occurrence of <code>item</code>



# Strings

- Strings are sequential, immutable, collections of zero or more letters, numbers and other symbols (characters). String values are differentiated from identifiers by using quotation marks (either single or double).
- The operations on Lists and strings are exactly the same EXCEPT that strings are immutable and cannot be modified. To modify a string, the `.replace()` method must be used and must be set to another variable.
- `string = "Greg"`



# String Methods



Method Name	Use	Explanation
<code>center</code>	<code>astring.center(w)</code>	Returns a string centered in a field of size <code>w</code>
<code>count</code>	<code>astring.count(item)</code>	Returns the number of occurrences of <code>item</code> in the string
<code>ljust</code>	<code>astring.ljust(w)</code>	Returns a string left-justified in a field of size <code>w</code>
<code>lower</code>	<code>astring.lower()</code>	Returns a string in all lowercase
<code>rjust</code>	<code>astring.rjust(w)</code>	Returns a string right-justified in a field of size <code>w</code>
<code>find</code>	<code>astring.find(item)</code>	Returns the index of the first occurrence of <code>item</code>
<code>split</code>	<code>astring.split(schar)</code>	Splits a string into substrings at <code>schar</code>

# Tuples



- Tuples are very similar to lists in that they are heterogeneous sequences of data. The difference is that a tuple is **immutable**, like a string. A tuple cannot be changed. Tuples are written as comma-delimited values enclosed in parentheses.
  
- `tuple = (1, 2, 3, 'Greg')`

# Dictionaries

Dictionaries are unordered collections of associated pairs of items where each pair consists of a key and a value. This key-value pair is typically written as `key:value`.

- `dict = {'first' : 1, 'second' : 2, 'third' : 'Byte Academy'}`



Operator	Use	Explanation
<code>[]</code>	<code>myDict[k]</code>	Returns the value associated with <code>k</code> , otherwise its an error
<code>in</code>	<code>key in adict</code>	Returns <code>True</code> if key is in the dictionary, <code>False</code> otherwise
<code>del</code>	<code>del</code> <code>adict[key]</code>	Removes the entry from the dictionary

# Dictionary Methods



Method Name	Use	Explanation
<code>keys</code>	<code>dict.keys()</code>	Returns the keys of the dictionary in a <code>dict_keys</code> object
<code>values</code>	<code>dict.values()</code>	Returns the values of the dictionary in a <code>dict_values</code> object
<code>items</code>	<code>dict.items()</code>	Returns the key-value pairs in a <code>dict_items</code> object
<code>get</code>	<code>dict.get(k)</code>	Returns the value associated with <code>k</code> , <code>None</code> otherwise
<code>get</code>	<code>dict.get(k, alt)</code>	Returns the value associated with <code>k</code> , <code>alt</code> otherwise

# Jupyter Notebooks

- Open source web-based environment for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, visualization, and documentation
- Frequently used through Anaconda, a popular statistical, scientific, and analytical computing package for Python





# Jupyter Notebooks



Chrome File Edit View History Bookmarks People Window Help 2:29 120°F 1312rpm 71% Tue 11:05 AM

Home March 11 Intro

localhost:8889/notebooks/March%2011%20Intro.ipynb

Apps Bookmarks Byte Word of the Day | ... Watch Live Sports... Other Bookmarks

jupyter March 11 Intro Last Checkpoint: 13 hours ago (autosaved) Python 3

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
In [4]: 4
Out[4]: 4

In [5]: a = 8
In [6]: a + 5
Out[6]: 13

In [9]: a
Out[9]: 13

In [8]: a = a + 5
In [10]: type(8)
Out[10]: int

In [11]: type(8.0)
Out[11]: float
```

# Jupyter Notebooks



```
In [21]: a = 'Greg'
```

```
In [23]: def example_function(a):  
         return a  
         example_function()
```

```
Out[23]: 'Greg'
```

```
In [24]: output = example_function()
```

```
In [27]: output
```

```
Out[27]: 'Greg'
```

# NumPy

- Open-source scientific computing library for Python
  - Provides array data structure
  - Written using C as well as Python for efficiency
  - Includes support for some linear algebra not natively supported in Python



```
[ ]: import numpy as np
```

# Pandas

- Open-source data analysis library for Python
  - Provides data structures that are optimized for processing data over core Python data structures
  - Provides methods optimized for performance and ease of use
  - Methods for reading many file types (csv, excel, etc.)



```
In [ ]: import pandas as pd
```

# Pandas

## Creating a DataFrame and importing libraries

```
In [57]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [62]: df = pd.read_excel('filepath.xls', sheet_name=12)

df = pd.read_csv('filepath.csv', index_col='Index')
```

```
In [177]: df.head(4)
```

Out[177]:

	0	1	2	3	4	5	6	7
0	-1.013561	0.566544	0.883230	1.136468	0.112616	-0.883783	-1.456444	-1.135527
1	-0.445597	-0.278217	-0.496617	-1.026302	-2.024133	-1.270216	2.144927	-0.185745
2	-1.128715	0.195105	0.545059	0.186170	0.300642	-1.299699	0.008669	-1.185713
3	-1.567792	0.912488	0.356792	-0.894444	0.542930	-1.065870	1.685032	-0.445394



# Pandas

## Basic DataFrame description

```
In [66]: len(df)
```

```
Out[66]: 8310
```

```
In [178]: df.describe()
```

```
Out[178]:
```

	0	1	2	3	4	5	6	7
<b>count</b>	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000
<b>mean</b>	-0.209697	-0.073281	0.223873	-0.412984	0.108555	-0.263308	0.212968	-0.438705
<b>std</b>	0.966282	0.740817	0.701356	1.210622	0.963532	0.989303	1.189682	1.274968
<b>min</b>	-1.746061	-1.675882	-1.125919	-2.084090	-2.024133	-1.395783	-1.456444	-2.728300
<b>25%</b>	-1.042350	-0.317371	-0.296964	-1.084657	-0.271376	-1.116956	-0.581051	-1.148073
<b>50%</b>	-0.045584	0.019576	0.317405	-0.830723	0.314904	-0.536402	-0.117379	-0.644569
<b>75%</b>	0.595490	0.287965	0.676852	0.402436	0.530627	0.603414	1.096842	0.261729
<b>max</b>	1.071345	0.912488	1.199513	1.862520	1.596675	1.476468	2.144927	1.548319



# Pandas

## Dealing with missing/null values



```
In [69]: placeholder = np.nan
```

```
In [73]: type(placeholder)
```

```
Out[73]: float
```

```
In [91]: df.isna().any()    # Checks for missing data  
df.isnan().any()    # Checks for missing data (same as above)  
df = df.fillna(0)    # Fills missing data with 0
```

# Pandas

## Basic DataFrame description

```
In [185]: df['a'].unique()
```

```
Out[185]: array([ 0.40933963,  0.58555969, -0.22702401,  0.69847859,  0.39432712,
                0.00861505, -0.35100341, -1.74303414, -0.07008838,  1.99548627,
                -1.26922685, -0.19447854])
```

```
In [186]: df.sort_values(by=['c', 'd', 'a'], ascending=False, inplace=True)
df.head(2)
```

```
Out[186]:
```

	a	b	c	d	e	f	g	i
7	-1.743034	-1.858912	2.154776	0.028116	-0.022373	0.424424	-0.558149	0.022039
0	0.409340	-0.068977	1.950157	-0.567336	0.457463	1.320088	1.081865	-0.258598

```
In [187]: df.iloc[2]
```

```
Out[187]: a    -1.269227
b    -0.570937
c     1.721024
d     0.593840
e     0.341630
f    -0.624017
g     0.893498
i    -1.691865
Name: 10, dtype: float64
```





# Pandas

## Additional Pandas DataFrame methods

```
In [167]: # Saves DataFrame to a .csv file
df.to_csv('different_title.csv')
```

```
In [ ]: # takes a list of DataFrames and concatenates them together
new_df = pd.concat(dataframe_list)
```

```
In [ ]: # Apply a function to ever value in a column
df['New_Column'] = df['Old_Column'].apply(function, axis=1)
```

```
In [ ]: # Generate dates for time series data
dates = pd.date_range('20190304', periods=6)
```

```
In [ ]: # Set a different column as our index
df.set_index('Column Name', inplace=True)
```

```
In [ ]: # Rename the index column of our DataFrame
df.index.rename('Name', inplace=True)
```



# Want to more Python or Data Science?

Check out:

- **Python Foundation:** Intro Evening Workshop, no experience required. Next class starts April 1.
- **Intro To Python For Excel & Data Analysis:** Become more efficient at the office. Evening intro workshop. Next class is May 8.
- **Full and Part -Time Python, FinTech & Data Science bootcamps:** Start monthly, rolling admissions



# Want to learn more about Byte Academy?

Check our meetup groups to register for upcoming events:

- **March 21:** Virtual Open House with our Head of Admissions
- **March 26:** Lunch Break Open House

**Can't wait?**

Set-up an admissions call: [ba10minchat.youcanbook.me](https://ba10minchat.youcanbook.me)

**Ask questions on our community Slack:** [byteacademy2.slack.com](https://byteacademy2.slack.com)

Stop by! 295 Madison Avenue (1 block from Grand Central)




# Contact

295 Madison Ave | NYC | 10017

[www.byteacademy.co](http://www.byteacademy.co) [info@byteacademy.co](mailto:info@byteacademy.co)

 [byteacademyco](https://twitter.com/byteacademyco)

[@byteacademy](https://twitter.com/byteacademy)

 [Byte-Academy-Finance-and-Technology-community](https://medium.com/Byte-Academy-Finance-and-Technology-community)

 [Byte-Academy-Finance-and-Technology-community](https://byteacademy2.slack.com) [byteacademy2.slack.com](https://byteacademy2.slack.com)

Greg: [greg.s@byteacademy.co](mailto:greg.s@byteacademy.co)

We welcome questions about our curriculum, hiring developers + partner opportunities





THANK YOU