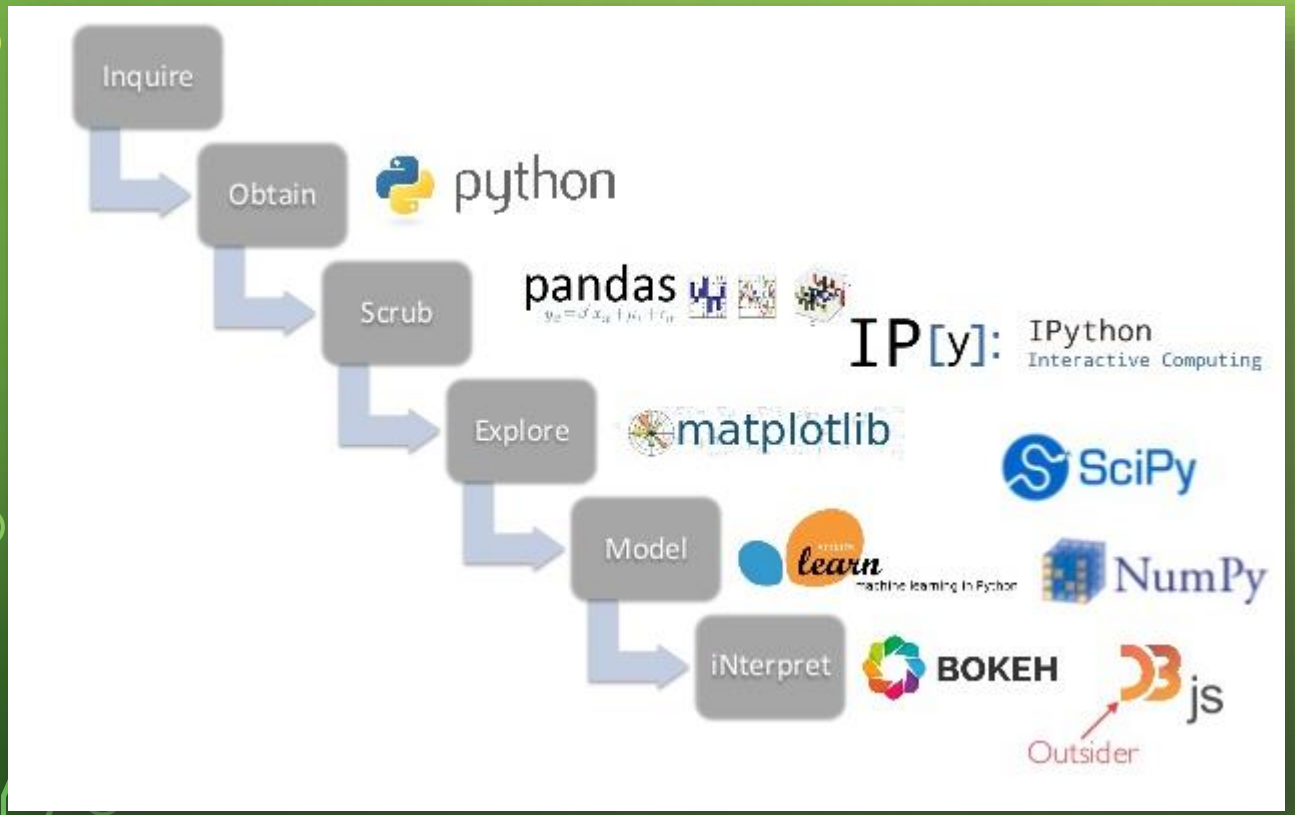
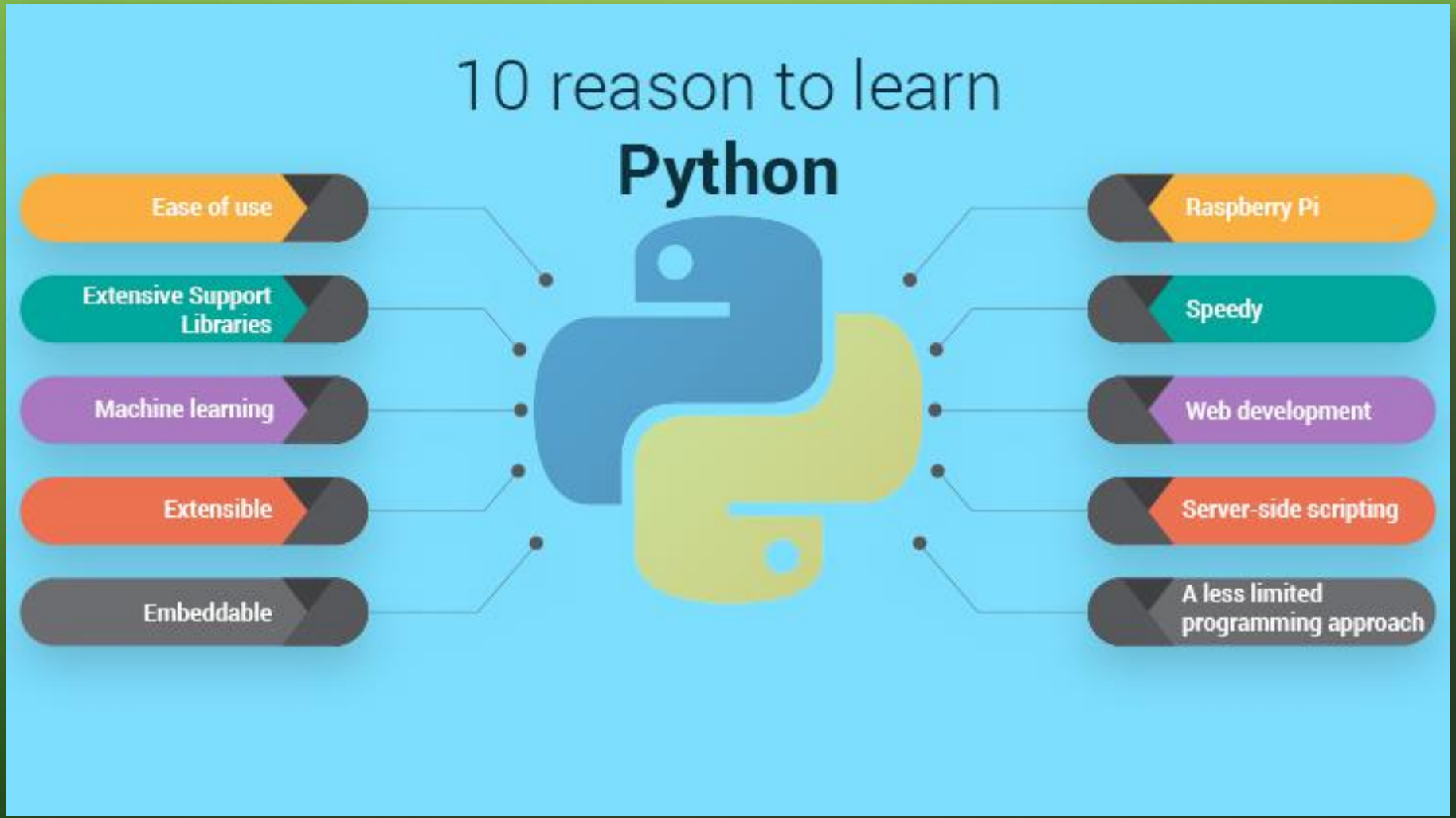
PYTHON

FOR ANALYTICS AND VISUALIZATION

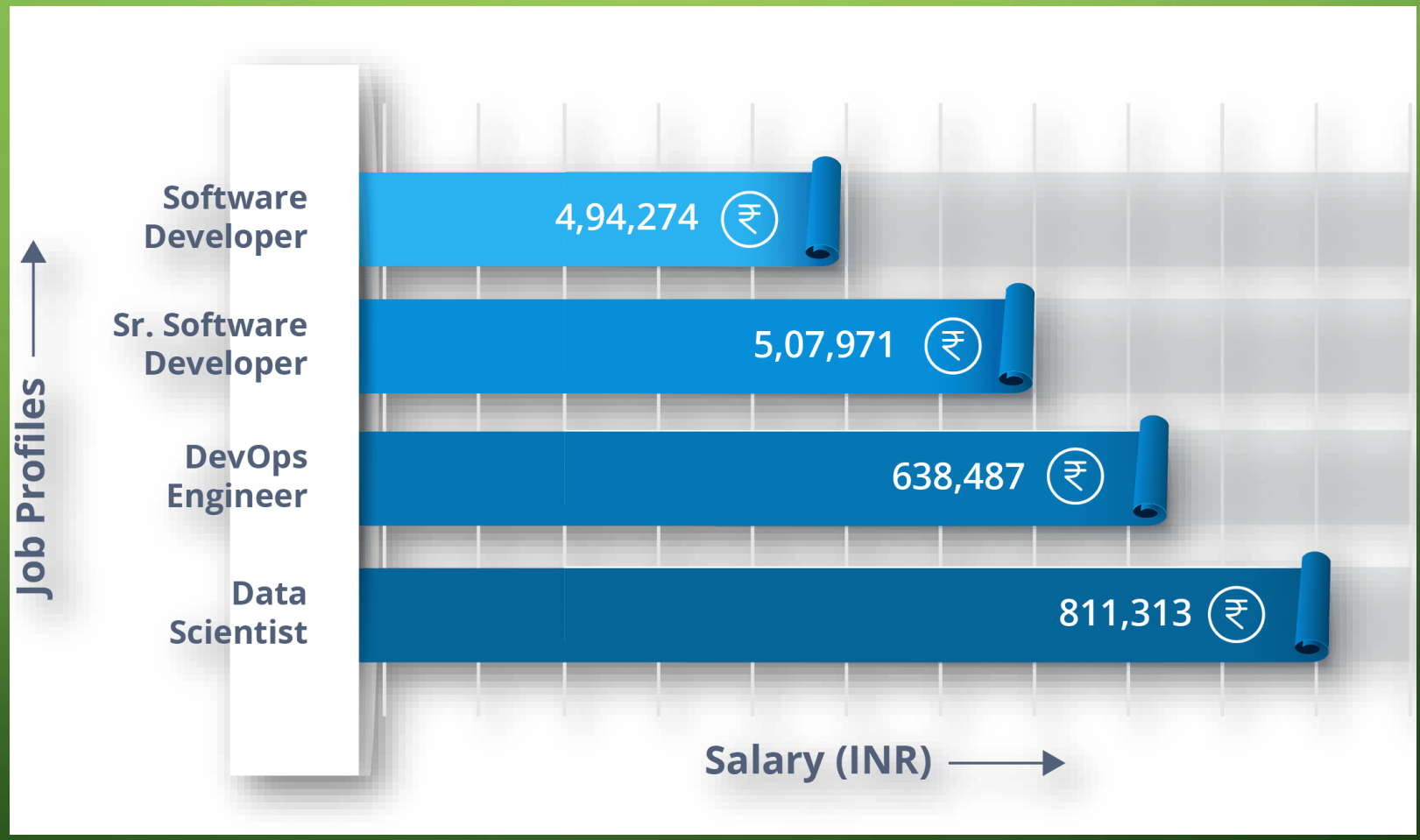
WHAT IS PYTHON?



WHY LEARN PYTHON?



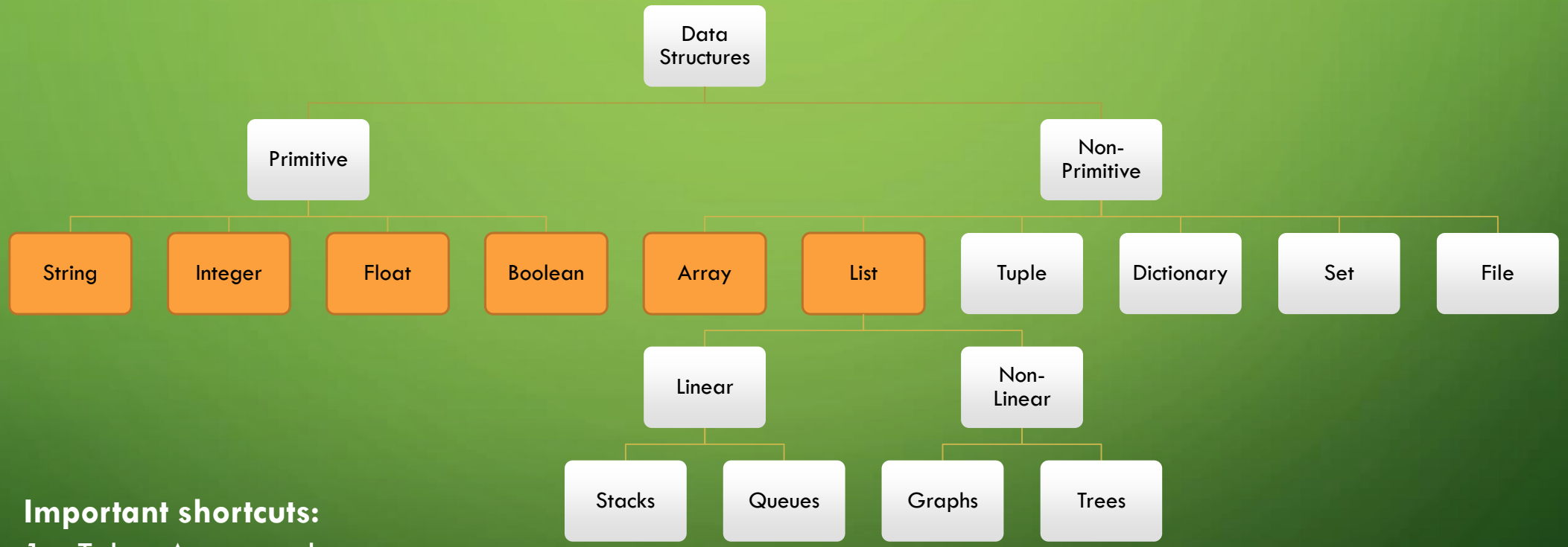
SOME SALARY INDICATORS



PYTHON BASICS

DATA TYPES

PYTHON DATA TYPES



Important shortcuts:

1. Tab – Autocomplete
2. Shift + Tab – Shows Function Arguments

INTEGER

- You can use an integer represent numeric data, and more specifically, whole numbers from negative infinity to infinity, like 4, 5, or -1.

Note that in Python, you do not have to explicitly state the type of the variable or your data. That is because it is a dynamically typed language. Dynamically typed languages are the languages where the type of data an object can store is mutable.

FLOAT

Important shortcuts:

1. Tab – Autocomplete
2. Shift + Tab – Shows Function Arguments

script.py

```
1   # Floats
2   x = 4.0
3   y = 2.0
4
5   # Addition
6   print(x + y)
7   # Subtraction
8   print(x - y)
9
10  # Multiplication
11  print(x * y)
12
13  # Returns the quotient
14  print(x / y)
15
16  # Returns the remainder
17  print(x % y)
18
19  # Absolute value
20  print(abs(x))
21
22  # x to the power y
23  print(x ** y)
```

IPython Shell

```
6.0
2.0
8.0
2.0
0.0
4.0
16.0

In [1]:
```

STRING

- Strings are collections of alphabets, words or other characters. In Python, you can create strings by enclosing a sequence of characters within a pair of single or double quotes. For example: 'cake', "cookie", etc.

```
In [3]: # String
        x = 'Fun'
        y = 'Excel'

        # Addition
        print(x + ' X ' + y)
```

```
Fun X Excel
```

Note that in Python, you do not have to explicitly state the type of the variable or your data. That is because it is a dynamically typed language. Dynamically typed languages are the languages where the type of data an object can store is mutable.

```
In [9]: # String
        # Range Slicing
        myName = 'Kunaal'
        z1 = myName[2:]

        print(z1)

        # Slicing
        z2 = myName[0] + myName[5]

        print(z2)
```

```
naal
Kl
```

```
In [11]: # String
         x = '2'
         y = '1'

         x + y
Out[11]: '21'

In [10]: str.capitalize('elephant')
Out[10]: 'Elephant'

In [18]: str1 = "We love Data Science"
         str2 = "404"
         len(str1)
Out[18]: 20
```

```
In [21]: str3 = 'like'
         str1.replace('love', str3)
Out[21]: 'We like Data Science'

In [23]: str1.find('love')
Out[23]: 3
```

```
In [16]: str1.isdigit()
Out[16]: False

In [17]: str2.isdigit()
Out[17]: True

In [21]: str3 = 'like'
         str1.replace('love', str3)
Out[21]: 'We like Data Science'
```

BOOLEAN

In [25]:
```
# Boolean
x = 4
y = 2
z = (x==y) # Comparison expression (Evaluates to false)
if z: # Conditional on truth/false value of 'z'
    print("We want to be a Data Scientists")
else: print("We DO not want to be a Data Scientists")
```

```
We DO not want to be a Data Scientists
```

IMPLICIT CONVERSIONS

In [26]:
```
# Implicit Conversions
# A float
x = 6.0

# An integer
y = 3

# Divide `x` by `y`
z = x/y

# Check the type of `z`
type(z)
```

Out[26]:
```
float
```

EXPLICIT CONVERSIONS

In [28]:
```
# Explicit Conversion
x = 8
y = "Game of Thrones: Season "
fav_season = y + x
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-773ae15bc933> in <module>()
      2 x = 8
      3 y = "Game of Thrones: Season "
----> 4 fav_season = y + x

TypeError: must be str, not int
```

In [29]:
```
x = 8
y = "Game of Thrones: Season "
fav_season = (y) + str(x)
print(fav_season)
```

```
Game of Thrones: Season 8
```

LISTS (STORES HETEROGENEOUS ITEMS)

```
In [2]: # List
        x1 = [1,2,3]
        type(x1)

Out[2]: list

In [3]: x2 = list([1,'apple',3])
        type(x2)

Out[3]: list

In [4]: print(x2[1])

        apple

In [5]: x2[1] = 'orange'
        print(x2)

        [1, 'orange', 3]

In [6]: list_num = [1,2,45,6,7,2,90,23,435]
        list_char = ['c','o','o','k','i','e']

        list_num.append(11) # Add 11 to the list, by default adds to the last position
        print(list_num)

        [1, 2, 45, 6, 7, 2, 90, 23, 435, 11]

In [7]: list_num.insert(0, 11)
        print(list_num)

        [11, 1, 2, 45, 6, 7, 2, 90, 23, 435, 11]

In [8]: list_char.remove('o')
        print(list_char)

        ['c', 'o', 'k', 'i', 'e']
```

Lists can store multiple data types.

ARRAYS (STORES HOMOGENEOUS ITEMS)

```
In [1]: import numpy as np

In [2]: my_list1 = [1,2,3,4]

In [3]: my_array1 = np.array(my_list1)

In [4]: my_array1
Out[4]: array([1, 2, 3, 4])
```

```
In [5]: my_list2 = [11,22,33,44]

In [7]: my_lists = [my_list1,my_list2]

In [8]: my_array2 = np.array(my_lists)

In [9]: my_array2
Out[9]: array([[ 1,  2,  3,  4],
               [11, 22, 33, 44]])

In [10]: my_array2.shape
Out[10]: (2L, 4L)
```

```
In [17]: np.eye(5)
Out[17]: array([[ 1.,  0.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.,  0.],
                [ 0.,  0.,  1.,  0.,  0.],
                [ 0.,  0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  0.,  1.]])

In [18]: np.arange(5)
Out[18]: array([0, 1, 2, 3, 4])

In [19]: np.arange(5,50,2)
Out[19]: array([ 5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37,
                39, 41, 43, 45, 47, 49])
```

ARRAYS

```
In [6]: arr1 = np.array([[1,2,3,4],[8,9,10,11]])

In [7]: arr1

Out[7]: array([[ 1,  2,  3,  4],
               [ 8,  9, 10, 11]])

In [8]: arr1*arr1

Out[8]: array([[  1,   4,   9,  16],
               [ 64,  81, 100, 121]])

In [9]: arr1 - arr1

Out[9]: array([[0, 0, 0, 0],
               [0, 0, 0, 0]])

In [10]: 1 / arr1

Out[10]: array([[ 1.        ,  0.5       ,  0.33333333,  0.25      ],
                [ 0.125     ,  0.11111111,  0.1       ,  0.09090909]])
```

ARRAYS

```
In [1]: import numpy as np

In [2]: arr = np.arange(0,11)

In [3]: arr
Out[3]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

In [4]: arr[8]
Out[4]: 8
```

```
In [5]: arr[1:5]
Out[5]: array([1, 2, 3, 4])

In [6]: arr[0:5]
Out[6]: array([0, 1, 2, 3, 4])

In [7]: arr[0:5] = 100

In [8]: arr
Out[8]: array([100, 100, 100, 100, 100,   5,   6,   7,   8,   9,  10])
```

```
In [9]: arr = np.arange(0,11)

In [10]: arr
Out[10]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

In [11]: slice_of_arr = arr[0:6]

         slice_of_arr
Out[11]: array([0, 1, 2, 3, 4, 5])
```

```
In [12]: slice_of_arr[:] = 99

In [13]: slice_of_arr
Out[13]: array([99, 99, 99, 99, 99, 99])

In [14]: arr
Out[14]: array([99, 99, 99, 99, 99, 99,  6,  7,  8,  9, 10])
```

```
In [15]: arr_copy = arr.copy()

In [16]: arr_copy
Out[16]: array([99, 99, 99, 99, 99, 99,  6,  7,  8,  9, 10])
```

Python stores only one copy of the array.
To copy and make modifications, we need to use **“copy”**

ARRAYS VS LISTS

- Can hold Homogeneous items
- Can perform operations to all its items easily
- Since Python does have to remember the data type, it is faster
- Uses less memory

- Can hold Heterogeneous items
- May not perfrom operations to all its item
- Has to remember individual data type, hence slower
- Uses more memory

```
In [13]: # Arrays vs Lists
         array_char = array.array("u",["c","a","t","s"])
         array_char.tostring()
         print(array_char)

            array('u', 'cats')
```

Tostring() applied to array char since Python knows the data type

INTRODUCTION TO PANDAS

INTRODUCTION TO PANDAS, SERIES, DATAFRAMES, MISSING DATA, GROUP BY, MERGING, JOINING, CONCATENATING, OPERATIONS, DATA INPUT AND OUTPUT

SERIES

- The first main data type we will learn about for pandas is the Series data type. Let's import Pandas and explore the Series object.

- A Series is very similar to a NumPy array (in fact it is built on top of the NumPy array object). What differentiates the NumPy array from a Series, is that a Series can have axis labels, meaning it can be indexed by a label, instead of just a number location. It also doesn't need to hold numeric data, it can hold any arbitrary Python Object.

DATAFRAMES

- DataFrames are the workhorse of pandas and are directly inspired by the R programming language. We can think of a DataFrame as a bunch of Series objects put together to share the same index. Let's use pandas to explore this topic!

INDEX OBJECTS

- Index is a way to create an ordered, sliceable set. Just like Rows/Columns in Excel, in Python we can create our own Index

MISSING DATA

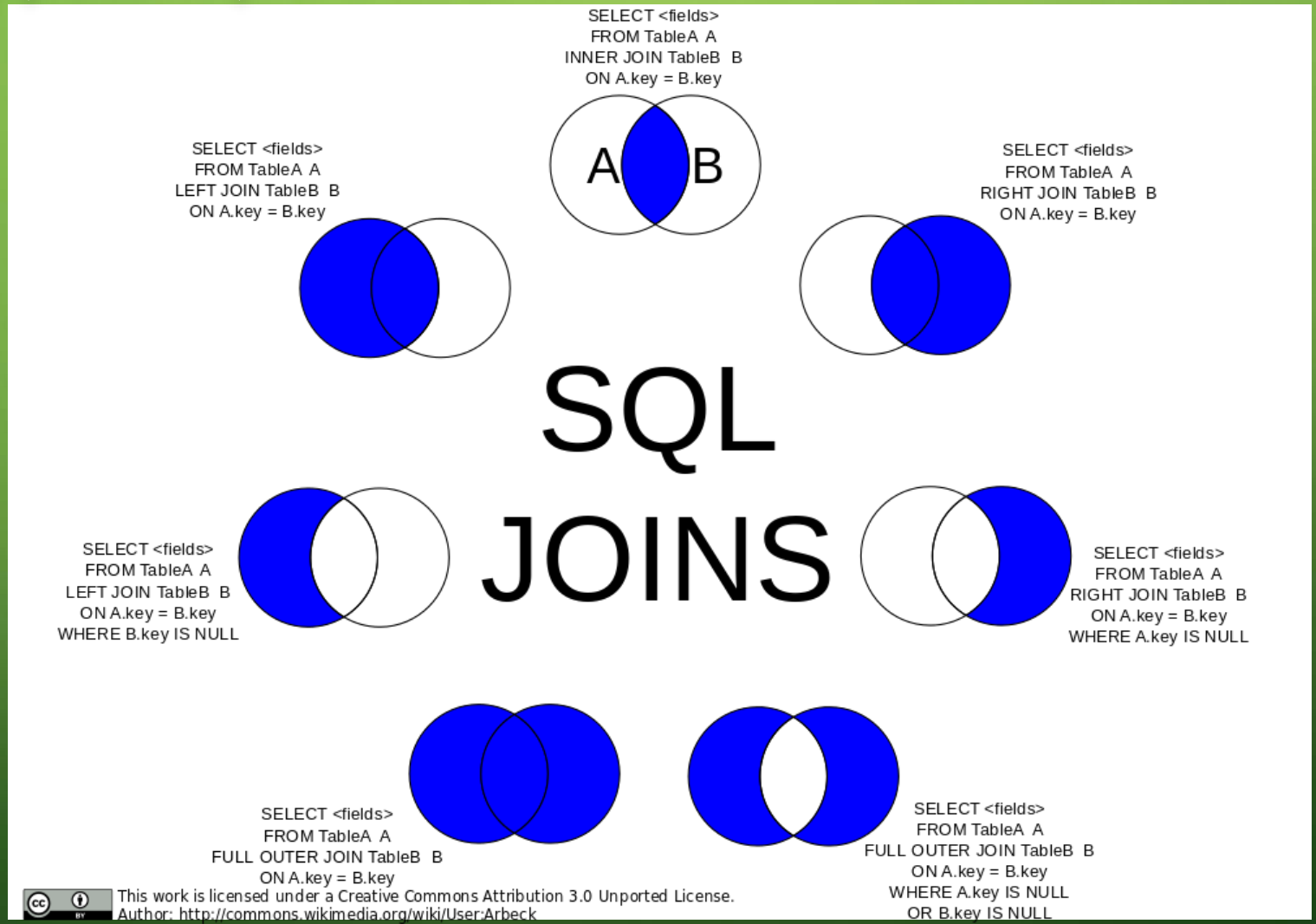
- Handling missing information form various datasets is key. Often we do not get perfect data. So it is essential we know how to handle them.
- To be used to clean missing information in the Titanic Dataset

GROUP BY

Date	City	Sales
1-Jan	Hyderabad	100
1-jan	Bangalore	200
2-Jan	Hyderabad	100
2-Jan	Bangalore	200
3-Jan	Hyderabad	200
3-Jan	Bangalore	50
4-Jan	Hyderabad	100

City	Sales
Hyderabad	500
Bangalore	450

MERGE, JOIN, CONCATENATE



EXAMPLE

trade

	dt	sym	price	size
0	2015-01-01	`C	10	10
1	2015-01-02	`C	10.5	100
2	2015-01-03	`MS	260	15
3	2015-01-04	`C	11	200
4	2015-01-04	`DBK	35.6	55
5	2015-01-05	`AAPL	1,010	20
6	2015-01-06	`AAPL	1,020	300
7	2015-01-07	`MS	255	200
8	2015-01-07	`MS	254	400

stock

	sym	sector	employees
0	`AAPL	`Tech	72800
1	`C	`Financial	262000
2	`FB	`Tech	4331
3	`MS	`Financial	57726

trade lj stock

	dt	sym	price	size	sector	employees
0	2015-01-01	`C	10	10	`Financial	262000
1	2015-01-02	`C	10.5	100	`Financial	262000
2	2015-01-03	`MS	260	15	`Financial	57726
3	2015-01-04	`C	11	200	`Financial	262000
4	2015-01-04	`DBK	35.6	55	`	
5	2015-01-05	`AAPL	1,010	20	`Tech	72800
6	2015-01-06	`AAPL	1,020	300	`Tech	72800
7	2015-01-07	`MS	255	200	`Financial	57726
8	2015-01-07	`MS	254	400	`Financial	57726

OPERATIONS

- Selecting Data
- Applying Functions
- Remove Columns
- Get slices
- Sort
- Pivot

IMPORT/EXPORT IN PANDAS

```
In [1]: import numpy as np
        import pandas as pd
```

CSV

CSV Input

```
In [25]: df = pd.read_csv('example')
         df
```

Out[25]:

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

CSV Output

```
In [24]: df.to_csv('example',index=False)
```

Excel Input

```
In [35]: pd.read_excel('Excel_Sample.xlsx',sheetname='Sheet1')
```

Out[35]:

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Excel Output

```
In [33]: df.to_excel('Excel_Sample.xlsx',sheet_name='Sheet1')
```

SF SALARIES EXERCISE CASE STUDY

USING PANDAS AND FUNCTIONS

EXERCISE 1

** Import pandas as pd.**

In [6]:

** Read Salaries.csv as a dataframe called sal.**

In [7]:

** Check the head of the DataFrame. **

In [8]:

Out[8]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011	NaN	San Francisco	NaN
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279.91	335279.91	2011	NaN	San Francisco	NaN
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343.61	332343.61	2011	NaN	San Francisco	NaN
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373.19	326373.19	2011	NaN	San Francisco	NaN

SOLUTION

** Import pandas as pd.**

In [6]:
```
import pandas as pd
```

** Read Salaries.csv as a dataframe called sal.**

In [7]:
```
sal = pd.read_csv('Salaries.csv')
```

** Check the head of the DataFrame. **

In [8]:
```
sal.head()
```

Out[8]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011	NaN	San Francisco	NaN
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279.91	335279.91	2011	NaN	San Francisco	NaN
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343.61	332343.61	2011	NaN	San Francisco	NaN
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373.19	326373.19	2011	NaN	San Francisco	NaN

EXERCISE 2

** Use the .info() method to find out how many entries there are.**

In [9]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
Id                 148654 non-null int64
EmployeeName       148654 non-null object
JobTitle           148654 non-null object
BasePay            148045 non-null float64
OvertimePay        148650 non-null float64
OtherPay           148650 non-null float64
Benefits           112491 non-null float64
TotalPay           148654 non-null float64
TotalPayBenefits   148654 non-null float64
Year               148654 non-null int64
Notes              0 non-null float64
Agency             148654 non-null object
Status             0 non-null float64
dtypes: float64(8), int64(2), object(3)
memory usage: 14.7+ MB
```

What is the average BasePay ?

In [10]:

Out[10]: 66325.44884050643

** What is the highest amount of OvertimePay in the dataset ? **

In [11]:

Out[11]: 245131.88

SOLUTION

** Use the .info() method to find out how many entries there are.**

In [9]:
```
sal.info() # 148654 Entries
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
Id                 148654 non-null int64
EmployeeName       148654 non-null object
JobTitle           148654 non-null object
BasePay            148045 non-null float64
OvertimePay        148650 non-null float64
OtherPay           148650 non-null float64
Benefits           112491 non-null float64
TotalPay           148654 non-null float64
TotalPayBenefits   148654 non-null float64
Year               148654 non-null int64
Notes              0 non-null float64
Agency             148654 non-null object
Status             0 non-null float64
dtypes: float64(8), int64(2), object(3)
memory usage: 14.7+ MB
```

What is the average BasePay ?

In [10]:
```
sal['BasePay'].mean()
```

Out[10]:
```
66325.44884050643
```

** What is the highest amount of OvertimePay in the dataset ? **

In [11]:
```
sal['OvertimePay'].max()
```

Out[11]:
```
245131.88
```

EXERCISE 3

** What is the job title of JOSEPH DRISCOLL ? Note: Use all caps, otherwise you may get an answer that doesn't match up (there is also a lowercase Joseph Driscoll). **

In [12]:

```
Out[12]: 24    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

** How much does JOSEPH DRISCOLL make (including benefits)? **

In [13]:

```
Out[13]: 24    270324.91
Name: TotalPayBenefits, dtype: float64
```

** What is the name of highest paid person (including benefits)?**

In [14]:

Out[14]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.0	400184.25	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN

** What is the name of lowest paid person (including benefits)? Do you notice something strange about how much he or she is paid?**

In [15]:

Out[15]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
148653	148654	Joe Lopez	Counselor, Log Cabin Ranch	0.0	0.0	-618.13	0.0	-618.13	-618.13	2014	NaN	San Francisco	NaN

SOLUTION

** What is the job title of JOSEPH DRISCOLL ? Note: Use all caps, otherwise you may get an answer that doesn't match up (there is also a lowercase Joseph Driscoll). **

In [12]:
```
sal[sal['EmployeeName']=='JOSEPH DRISCOLL']['JobTitle']
```

Out[12]:
```
24    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

** How much does JOSEPH DRISCOLL make (including benefits)? **

In [13]:
```
sal[sal['EmployeeName']=='JOSEPH DRISCOLL']['TotalPayBenefits']
```

Out[13]:
```
24    270324.91
Name: TotalPayBenefits, dtype: float64
```

** What is the name of highest paid person (including benefits)?**

In [14]:
```
sal[sal['TotalPayBenefits']== sal['TotalPayBenefits'].max()] #['EmployeeName']
# or
# sal.loc[sal['TotalPayBenefits'].idxmax()]
```

Out[14]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.0	400184.25	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN

** What is the name of lowest paid person (including benefits)? Do you notice something strange about how much he or she is paid?**

In [15]:
```
sal[sal['TotalPayBenefits']== sal['TotalPayBenefits'].min()] #['EmployeeName']
# or
# sal.loc[sal['TotalPayBenefits'].idxmax()]['EmployeeName']

## ITS NEGATIVE!! VERY STRANGE
```

Out[15]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
148653	148654	Joe Lopez	Counselor, Log Cabin Ranch	0.0	0.0	-618.13	0.0	-618.13	-618.13	2014	NaN	San Francisco	NaN

EXERCISE 4

** What was the average (mean) BasePay of all employees per year? (2011-2014) ? **

In [16]:

```
Out[16]: Year
         2011    63595.956517
         2012    65436.406857
         2013    69630.030216
         2014    66564.421924
         Name: BasePay, dtype: float64
```

** How many unique job titles are there? **

In [17]:

```
Out[17]: 2159
```

** What are the top 5 most common jobs? **

In [18]:

```
Out[18]: Transit Operator                7036
         Special Nurse                   4389
         Registered Nurse                3736
         Public Svc Aide-Public Works    2518
         Police Officer 3                2421
         Name: JobTitle, dtype: int64
```

** How many Job Titles were represented by only one person in 2013? (e.g. Job Titles with only one occurence in 2013?) **

In [19]:

```
Out[19]: 202
```

SOLUTION

** What was the average (mean) BasePay of all employees per year? (2011-2014) ? **

In [16]:
```
sal.groupby('Year').mean()['BasePay']
```

Out[16]:
```
Year
2011    63595.956517
2012    65436.406857
2013    69630.030216
2014    66564.421924
Name: BasePay, dtype: float64
```

** How many unique job titles are there? **

In [17]:
```
sal['JobTitle'].nunique()
```

Out[17]:
```
2159
```

** What are the top 5 most common jobs? **

In [18]:
```
sal['JobTitle'].value_counts().head(5)
```

Out[18]:
```
Transit Operator                7036
Special Nurse                   4389
Registered Nurse                3736
Public Svc Aide-Public Works    2518
Police Officer 3                2421
Name: JobTitle, dtype: int64
```

** How many Job Titles were represented by only one person in 2013? (e.g. Job Titles with only one occurence in 2013?) **

In [19]:
```
sum(sal[sal['Year']==2013]['JobTitle'].value_counts() == 1) # pretty tricky way to do this...
```

Out[19]:
```
202
```

LAST ONE! TRICKY ONE!

** How many people have the word Chief in their job title? (This is pretty tricky) **

In [20]:

In [21]:

Out[21]: 477

** Bonus: Is there a correlation between length of the Job Title string and Salary? **

In [22]:

In [23]:

Out[23]:

	title_len	TotalPayBenefits
title_len	1.000000	-0.036878
TotalPayBenefits	-0.036878	1.000000

SOLUTION

** How many people have the word Chief in their job title? (This is pretty tricky) **

In [20]:
```
def chief_string(title):
    if 'chief' in title.lower():
        return True
    else:
        return False
```

In [21]:
```
sum(sal['JobTitle'].apply(lambda x: chief_string(x)))
```

Out[21]: 477

** Bonus: Is there a correlation between length of the Job Title string and Salary? **

In [22]:
```
sal['title_len'] = sal['JobTitle'].apply(len)
```

In [23]:
```
sal[['title_len','TotalPayBenefits']].corr() # No correlation.
```

Out[23]:

	title_len	TotalPayBenefits
title_len	1.000000	-0.036878
TotalPayBenefits	-0.036878	1.000000

INTRODUCTION TO MATPLOTLIB

BASICS, DATA HANDLING, MORE DATA HANDLING

MATPLOTLIB

- Matplotlib is the most popular plotting library for Python
- It gives you control over every aspect of a figure
- It was designed to have a similar feel to MatLab's graphical plotting
- Link: https://matplotlib.org/
  - Check out the Gallery; search for the
- Installation
  - Pip install matplotlib

READING LINKS

- www.matplotlib.com
- www.github.com/matplotlib/matplotlib
- www.matplotlib.org/gallery.html
- https://www.labri.fr/perso/nrougier/teaching/matplotlib/
- http://www.scipy-lectures.org/intro/matplotlib/matplotlib.html

INTRODUCTION TO SEABORN

MORE HELPFUL FOR STATISTICAL PLOTTING

LOGISTIC REGRESSION

SOLVING A KAGGLE COMPETITION