# Python Language

- ❖ Objects, attributes, and methods
- ❖ Functions vs. object methods
- ❖ Object references
- ❖ Mutable and immutable objects

# Objects, attributes, and methods

- **Everything in Python is an object.**
  - Scalars, sequences, dictionaries, functions, DataFrames, modules, and more
- Each type of object has a set of
  - **Attributes**: Characteristics of the object
  - **Methods**: Functions that operate on the object (and possibly other objects)
- Atrtributes and methods are accessible by:
  - `obj.attr_name` or `getattr(obj, ‘attr_name’`
  - `obj.method_name()`

# Functions vs. Object Methods

- Functions and object methods are essentially the same…
  - One or more bundled steps performed on some input
  - In some cases, there will be a function and an object method that do the same thing (e.g., sum)
- …BUT, they differ in how they are used
  - Functions are called on zero or more objects and return result(s) that can be assigned to a variable
  - Object methods are called by an object and can either update the calling object or return results

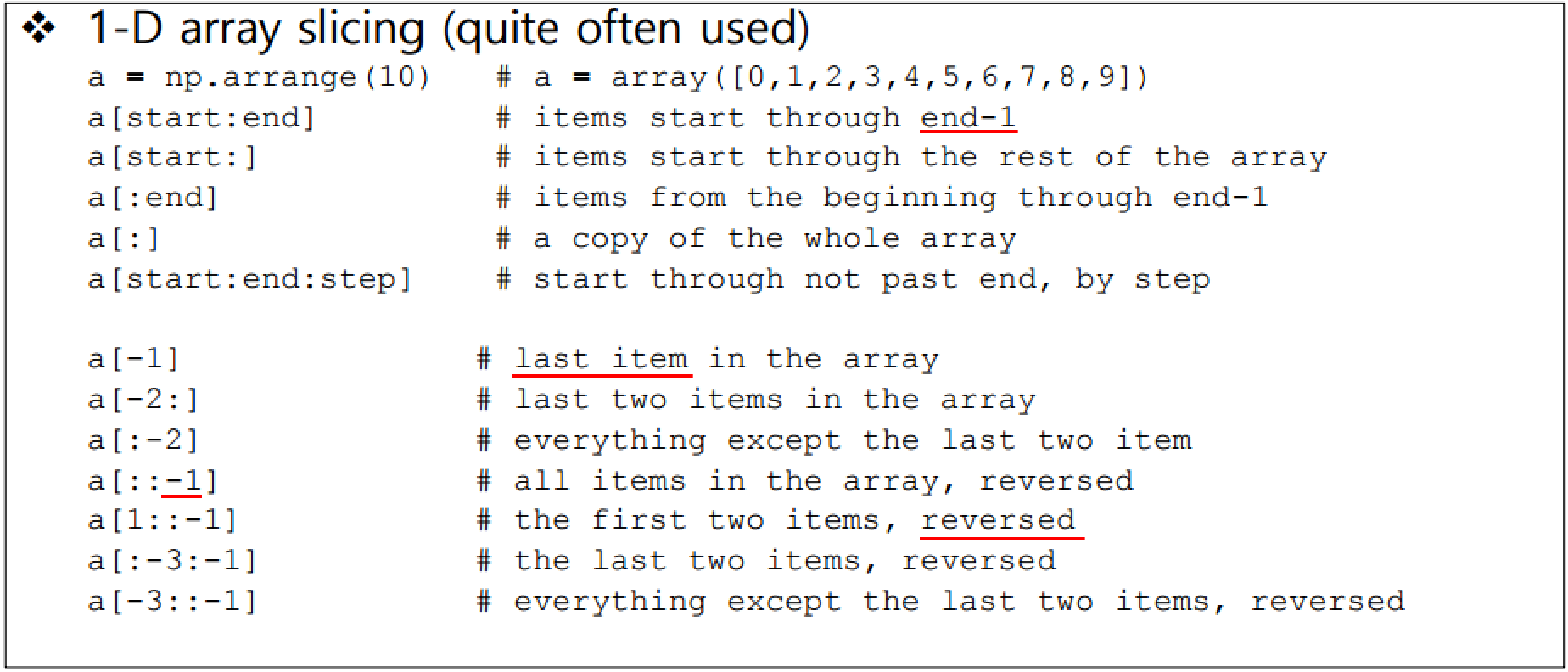
# Mutable and Immutable Objects

- Mutable Objects
  - Can be modified via assignment or a function/method
  - Lists, dictionaries, ndarrays, class instances
- Immutable Objects
  - Can not be modified
  - Strings, tuples, sets

# Importing Modules and Scripts

- Modules and Python scripts are loaded in the same manner. For a module or Python script P (.py):
  - (ex) **import P [as p]**
    - Loads the module or script into the workspace, with an optional shorter name
    - Can use any functionality in an OOP fashion (e.g., P.method())
  - (ex) **from python_module import ***
    - Imports all of the functionality directly into workspace
  - (ex) **from python_module import f, g, h**
    - Imports specific functions

# Slicing: list and array



❖ 1-D array slicing (quite often used)

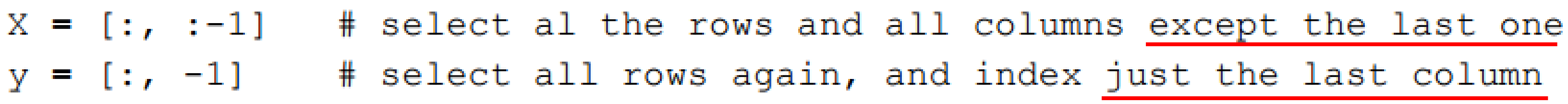
```
a = np.arrange(10)    # a = array([0,1,2,3,4,5,6,7,8,9])
a[start:end]          # items start through end-1
a[start:]             # items start through the rest of the array
a[:end]               # items from the beginning through end-1
a[:]                  # a copy of the whole array
a[start:end:step]     # start through not past end, by step

a[-1]                # last item in the array
a[-2:]               # last two items in the array
a[:-2]               # everything except the last two item
a[::-1]              # all items in the array, reversed
a[1::-1]             # the first two items, reversed
a[:-3:-1]            # the last two items, reversed
a[-3::-1]            # everything except the last two items, reversed
```

- a[start::-1] : start(포함)부터 제일 앞까지, <end 생략 : 제일 앞>
- a[:end:-1] : 제일 마지막에서 end(이전까지), <start 생략 : 제일 마지막>

# Slicing: list and array

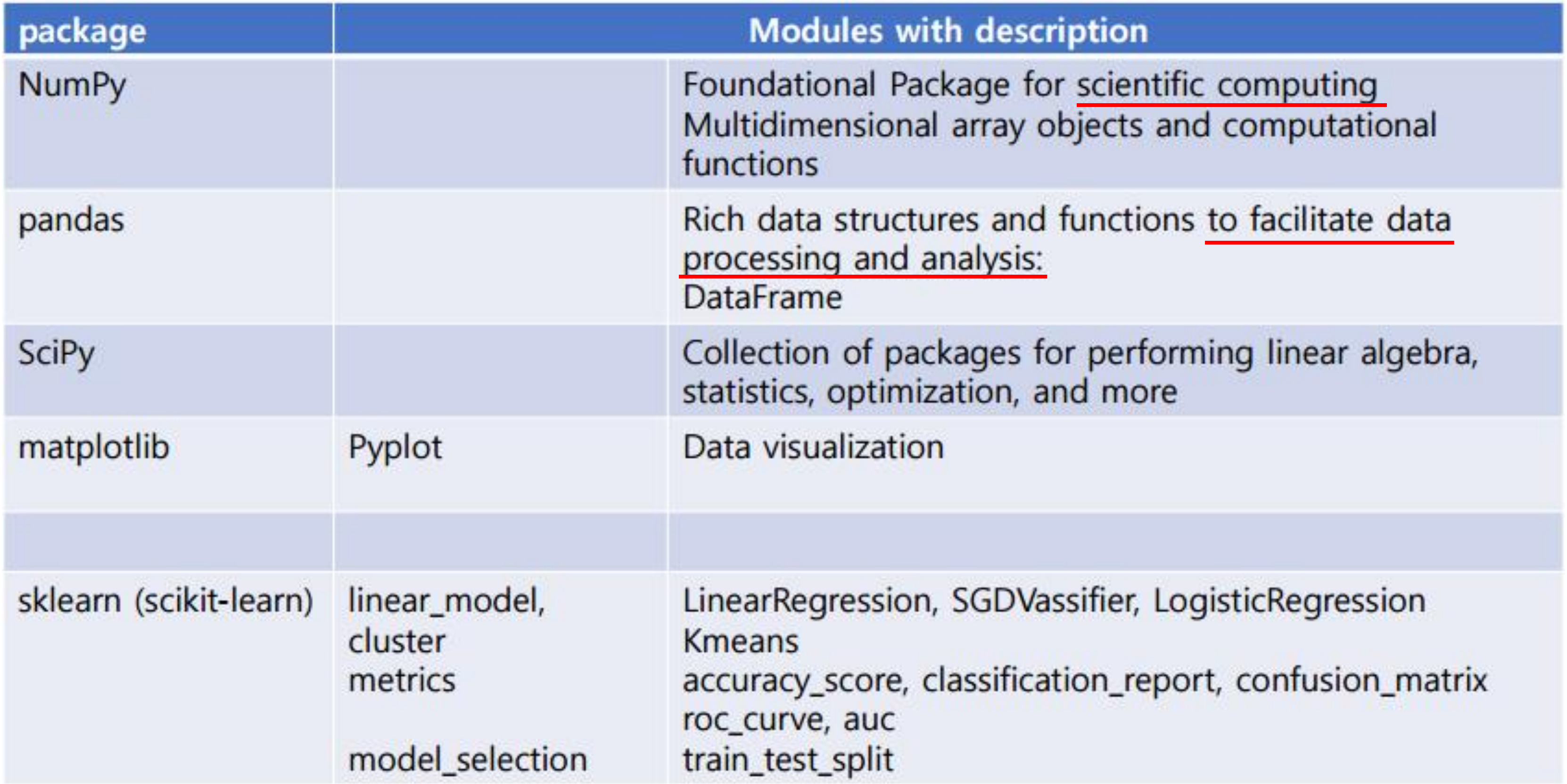
- 2-D array slicing (to split loaded data into input(X) and the output(y))



```
X = [:, :-1]    # select al the rows and all columns except the last one
y = [:, -1]     # select all rows again, and index just the last column
```

# Python 라이브러리

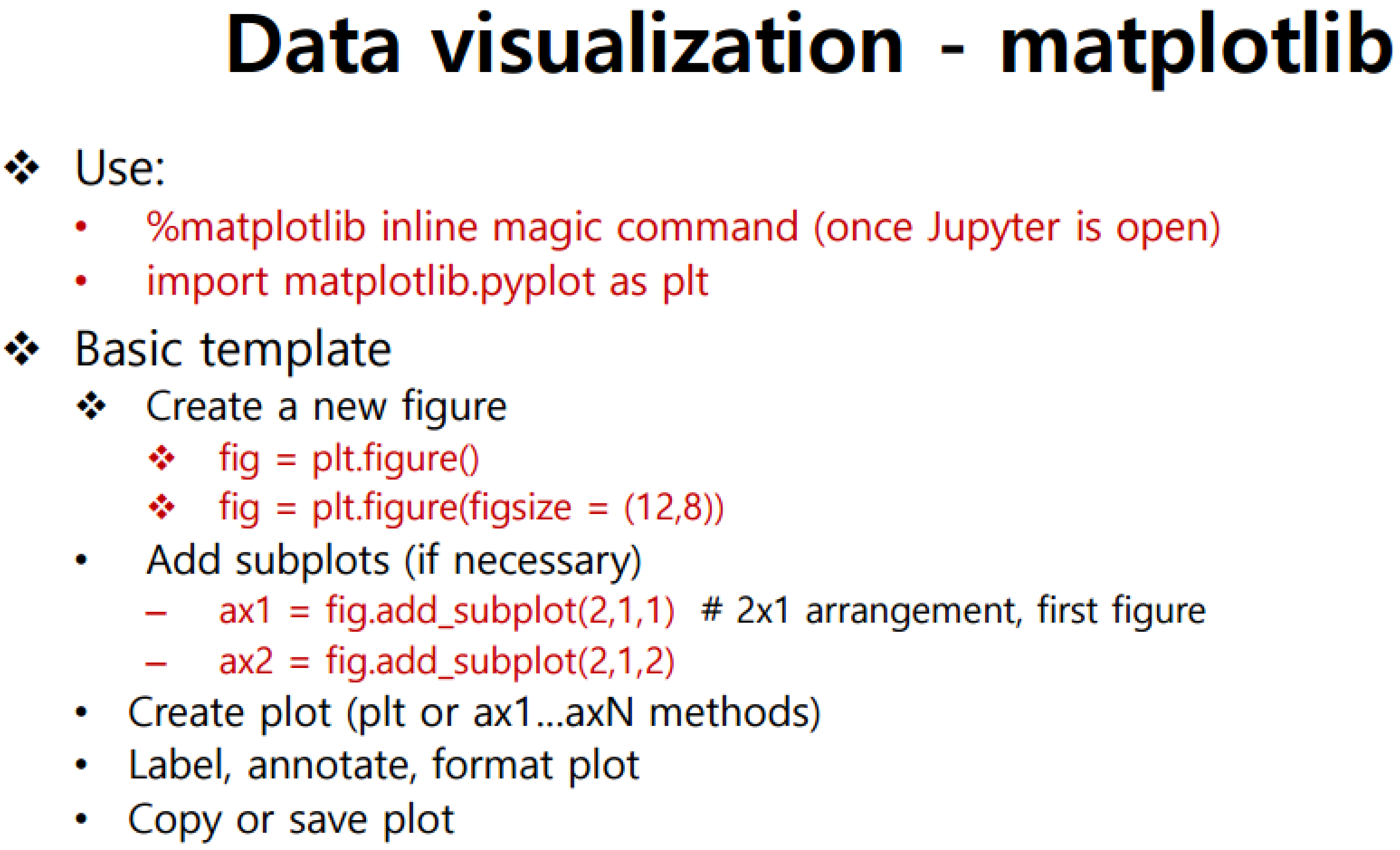
- **Python의 주요 장점 : 유용한 라이브러리가 많다**
  - **기본 라이브러리 : Numpy, pandas, matplotlib, SciPy, sklearn(scikit-learn) 등**



| package | Modules with description | |
|---|---|---|
| NumPy | | Foundational Package for scientific computing<br>Multidimensional array objects and computational functions |
| pandas | | Rich data structures and functions to facilitate data processing and analysis:<br>DataFrame |
| SciPy | | Collection of packages for performing linear algebra, statistics, optimization, and more |
| matplotlib | Pyplot | Data visualization |
| | | |
| sklearn (scikit-learn) | linear_model,<br>cluster<br>metrics<br><br>model_selection | LinearRegression, SGDVassifier, LogisticRegression<br>Kmeans<br>accuracy_score, classification_report, confusion_matrix<br>roc_curve, auc<br>train_test_split |

  - **딥러닝 모델을 이용 : Tensorflow, Keras 등을 추가 설치해야**

# Data visualization – matplotlib



## Data visualization - matplotlib

- Use:
  - %matplotlib inline magic command (once Jupyter is open)
  - import matplotlib.pyplot as plt
- Basic template
  - Create a new figure
    - fig = plt.figure()
    - fig = plt.figure(figsize = (12,8))
  - Add subplots (if necessary)
    - ax1 = fig.add_subplot(2,1,1) # 2x1 arrangement, first figure
    - ax2 = fig.add_subplot(2,1,2)
  - Create plot (plt or ax1…axN methods)
  - Label, annotate, format plot
  - Copy or save plot

# Data visualization – matplotlib

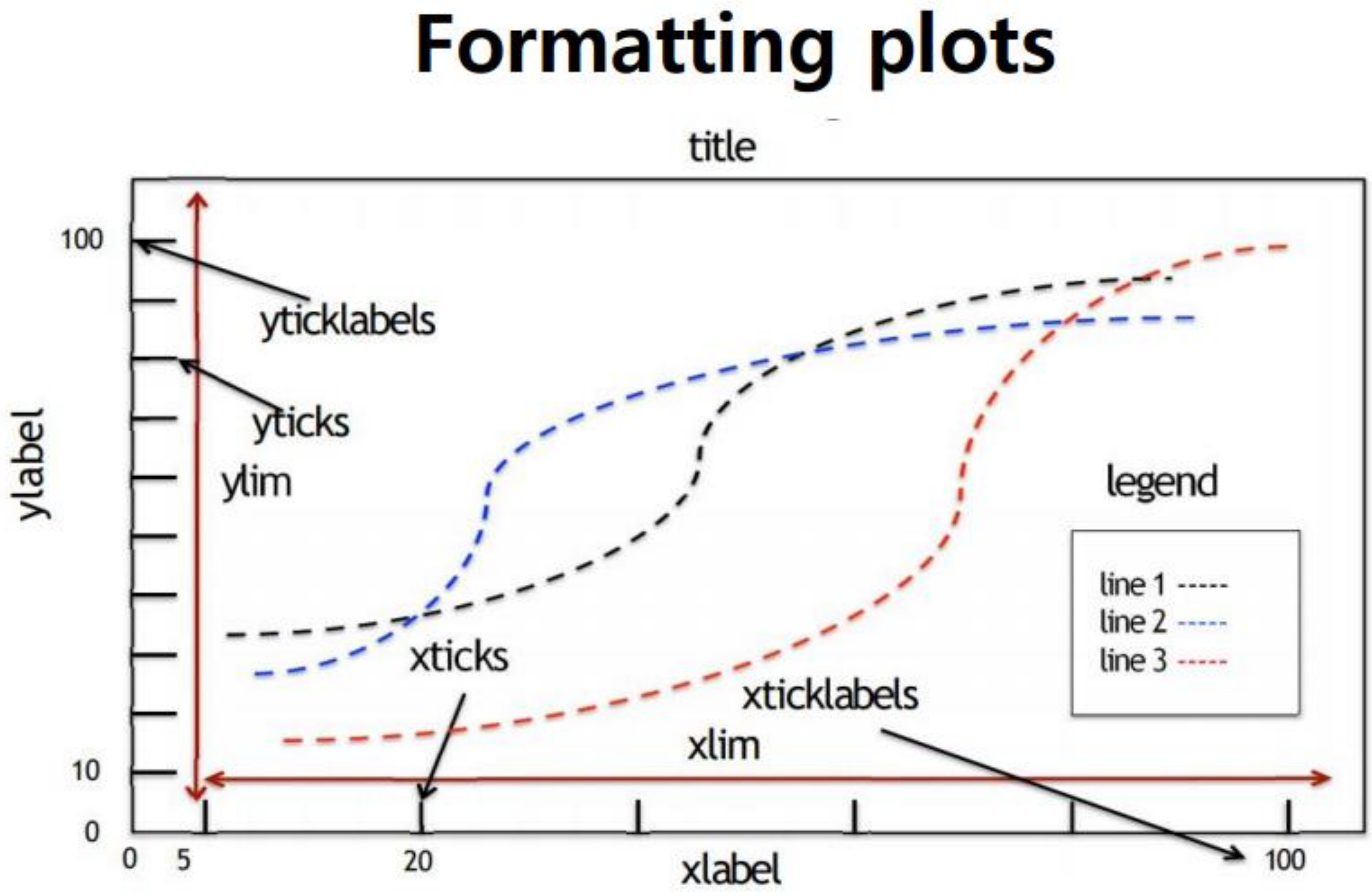
## Matplotlib - Common plot types

- Line plots – trends:
  - plt.plot (x, y, '-')
- Scatter plots – comparison between lots of data
  - plt.plot (x, y, '.')
- Bar plots – comparison between few data
  - Bar (horizontal): plt.barh (x, y, width)
  - Column (vertical): plt.bar (x, y, width)
- Histogram plots – single distributions
  - plt.hist (x, bins)
- Boxplots – one or more distributions
  - plt.boxplot (x)

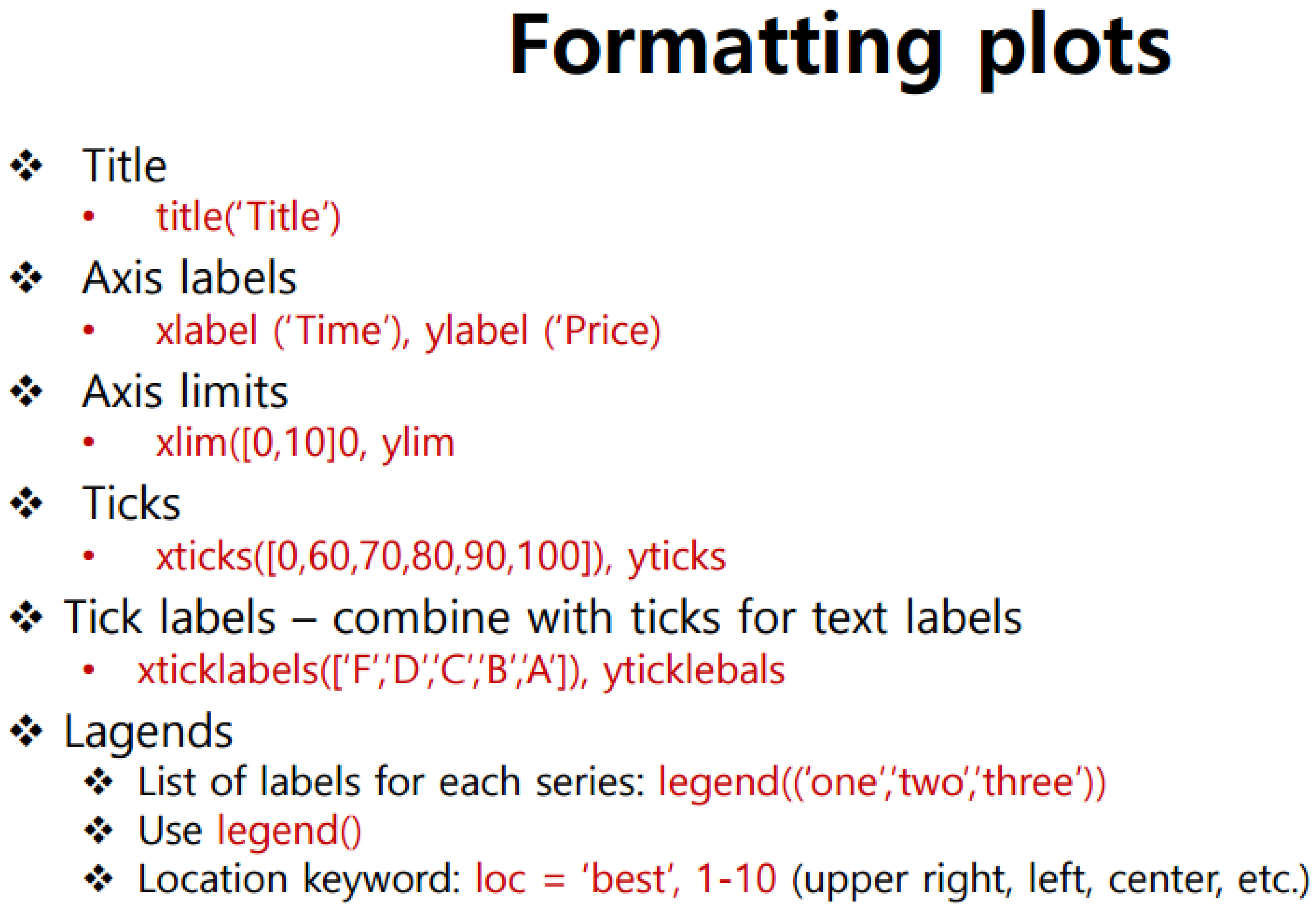
## Matplotlib - Colors, Markers, and Line Styles

- All specified as special string characters in plot call
- Colors - Many plot types
  - Basic colors: g(reen), r(ed), b(lue), (blac)k, m(agenta), y(ellow), c(yan), w(hite)
  - For more, see http://matplotlib.org/api/colors_api.html
- Markers and Line Styles - Mostly relate to plt.plot
  - Markers: ., o, +, * (star), 1, 2, 3, 4 (triangles), s(quare), D(iamond)
  - Line styles: solid (-), dashed (--), dotted (:), dash-dot (-.)
  - linewidth keyword (float value)
- Usage
  - Style string: Combines all three (e.g., 'k.', 'g--', 'ro-')
  - Separate keyword arguments: color, linestyle, marker

# Data visualization – matplotlib



# Data visualization – matplotlib



## Formatting plots

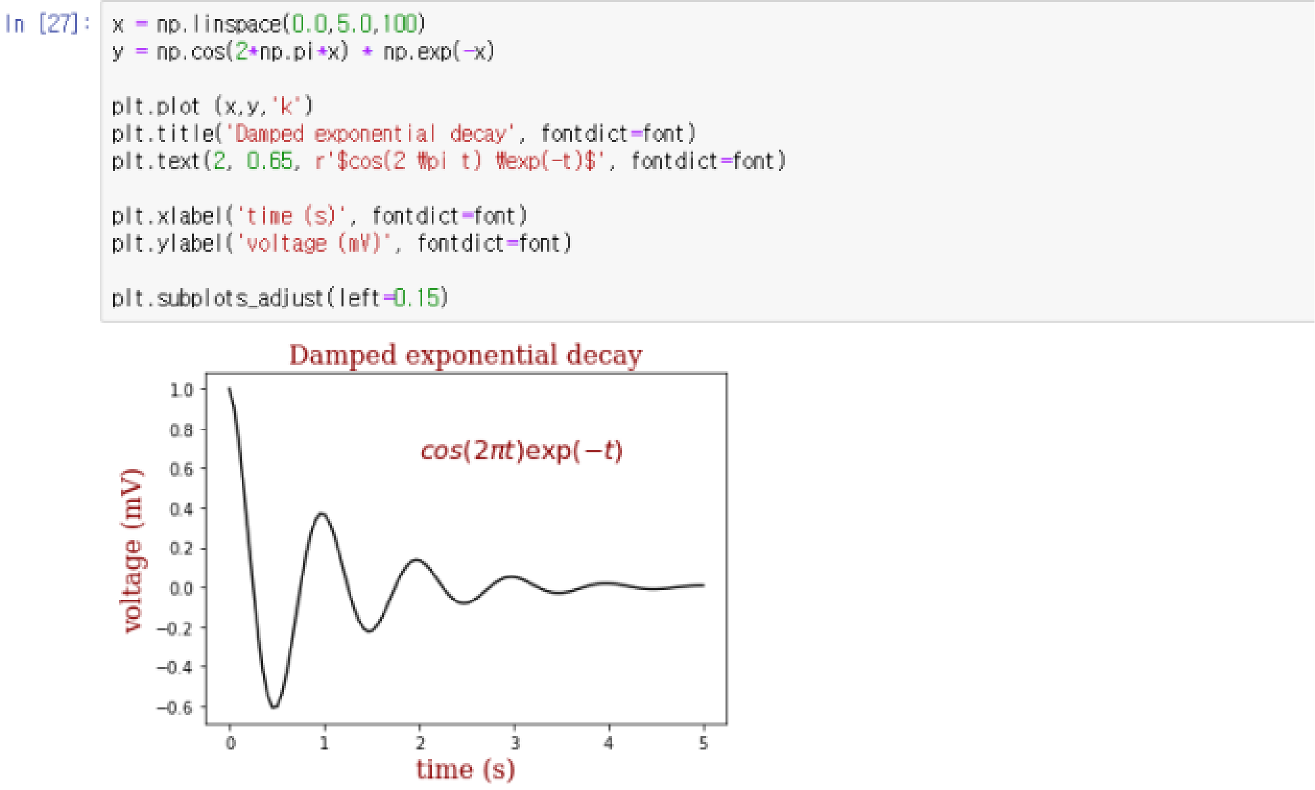
- Title
  - title(‘Title’)
- Axis labels
  - xlabel (‘Time’), ylabel (‘Price)
- Axis limits
  - xlim([0,10]0, ylim
- Ticks
  - xticks([0,60,70,80,90,100]), yticks
- Tick labels – combine with ticks for text labels
  - xticklabels([‘F’,‘D’,‘C’,‘B’,‘A’]), yticklebals
- Lagends
  - List of labels for each series: legend((‘one’,‘two’,‘three’))
  - Use legend()
  - Location keyword: loc = ‘best’, 1-10 (upper right, left, center, etc.)

# Annotating plots

- Text
  - text(x, y, text, fontsize)
  - arrow(x, y, dx, dy)  # draws arrow from (x,y) to (x+dx, y+dy)
  - annotate (text, xy, xytext)  # annotate the xy point with text positioned at xytext
- shapes
  - Rectangles, circles, polygons
  - Location, size, color, transparency (alpha)

# Data visualization – matplotlib

## Matplotlib - Example(1)



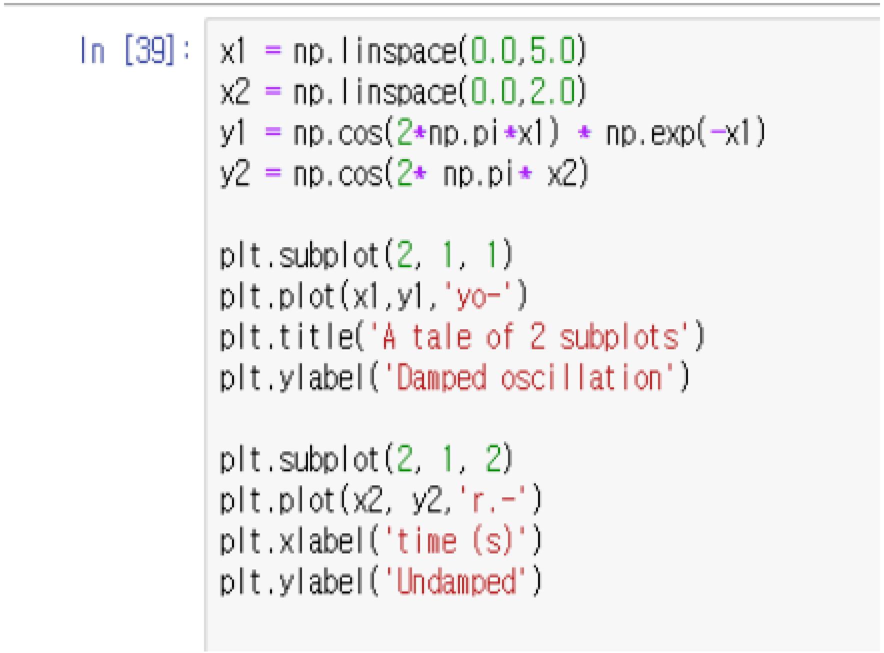
```
In [27]: x = np.linspace(0.0,5.0,100)
         y = np.cos(2*np.pi*x) * np.exp(-x)

         plt.plot (x,y,'k')
         plt.title('Damped exponential decay', fontdict=font)
         plt.text(2, 0.65, r'$cos(2 ₩pi t) ₩exp(-t)$', fontdict=font)

         plt.xlabel('time (s)', fontdict=font)
         plt.ylabel('voltage (mV)', fontdict=font)

         plt.subplots_adjust(left=0.15)
```

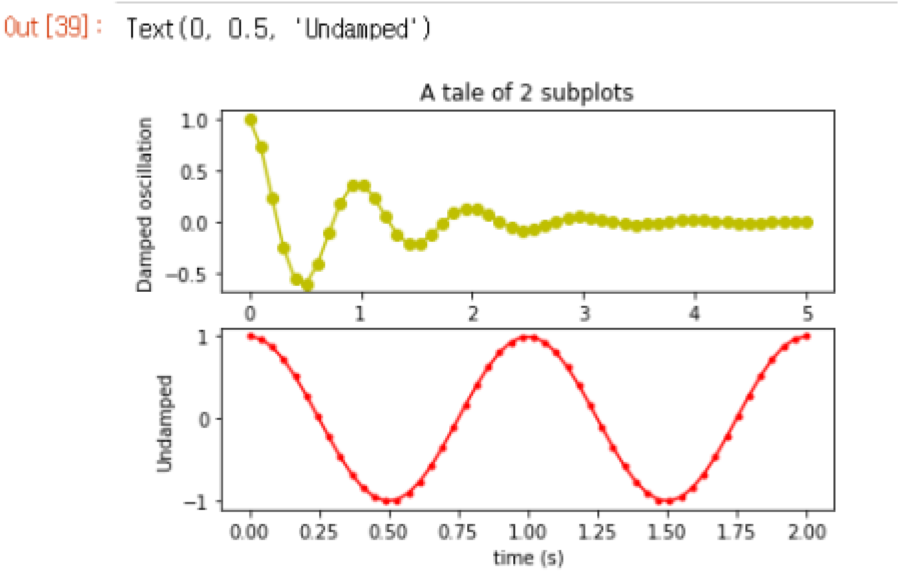
# Matplotlib - Example(2)



```
In [39]: x1 = np.linspace(0.0,5.0)
         x2 = np.linspace(0.0,2.0)
         y1 = np.cos(2*np.pi*x1) * np.exp(-x1)
         y2 = np.cos(2* np.pi* x2)

         plt.subplot(2, 1, 1)
         plt.plot(x1,y1,'yo-')
         plt.title('A tale of 2 subplots')
         plt.ylabel('Damped oscillation')

         plt.subplot(2, 1, 2)
         plt.plot(x2, y2,'r.-')
         plt.xlabel('time (s)')
         plt.ylabel('Undamped')
```

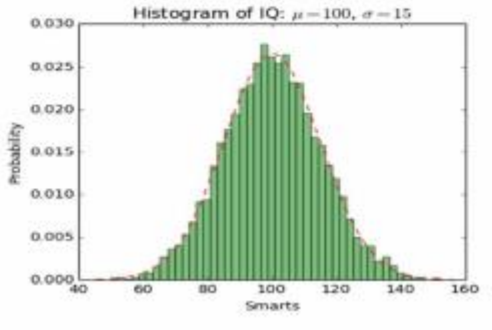


```
Out[39]: Text(0, 0.5, 'Undamped')
```

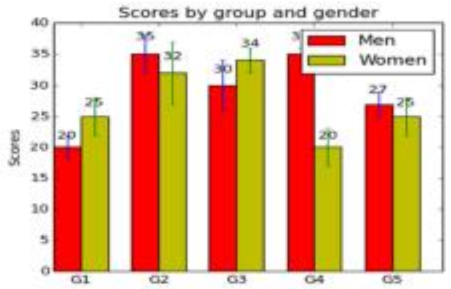
# Data visualization – matplotlib

## Many more examples...

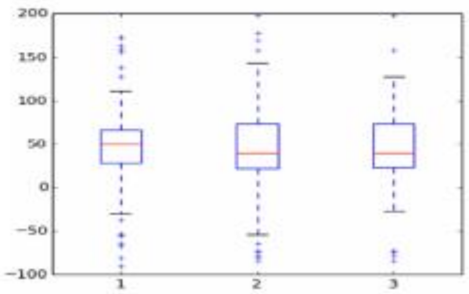
Histogram

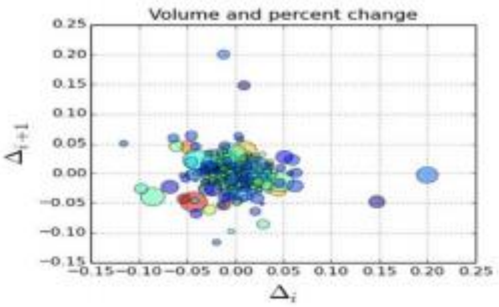


Bar Chart (with error bars and legend)

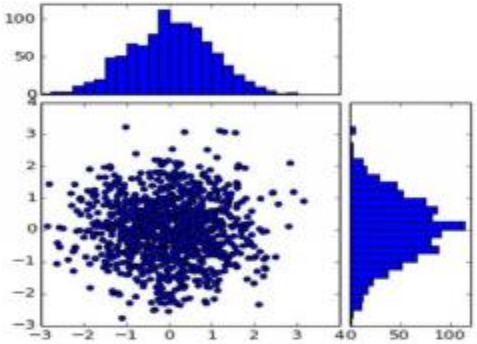


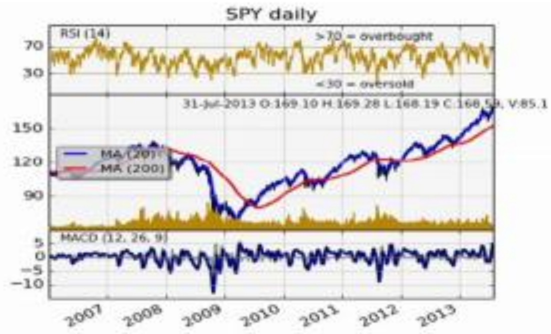
Boxplots





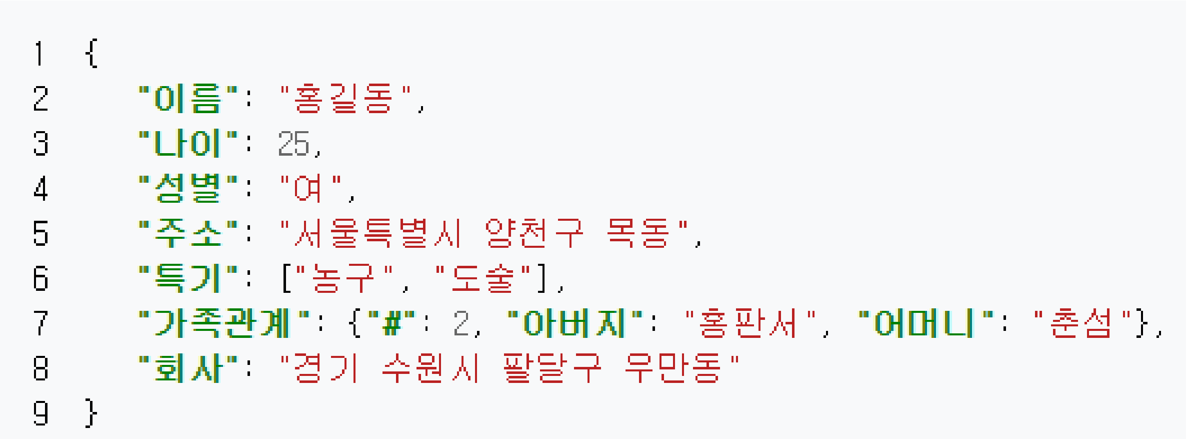
Scatter + Histogram





# JSON(JavaScript Object Notation)

- JSON
  - 속성-값 쌍(attribute-value pair & array data types (or any other serializable value)) 또는 “키-값 쌍” 으로 이루어진 데이터 object를 전달하기 위해 인간이 읽고 쓰기 쉽게 텍스트를 사용하는 개방형 표준 포맷이다.
  - 비동기 브라우저/서버 통신(AJAX)을 위해 넓게는 XML(AJAX가 사용)을 대체하는 주요 데이터 포맷.
  - 특히 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있고 자료의 종류에 큰 제한은 없으며, 컴퓨터 프로그램의 변수 값을 표현하는데 적합
- Example



```
{
    "이름": "홍길동",
    "나이": 25,
    "성별": "여",
    "주소": "서울특별시 양천구 목동",
    "특기": ["농구", "도술"],
    "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},
    "회사": "경기 수원시 팔달구 우만동"
}
```

# JSON(JavaScript Object Notation)

- JSON 패키지
  - JSON Encoding : Python Object(dict, list, tuple 등) -> JSON 문자열
    (ex) json.dumps(result)
  - JSON Decoding : JSON 문자열 -> Python Object(dict, list, tuple 등)
    (ex) json.loads(obj)

- JSON format normalize
  - Pandas.io.json_normalize()
  - Normalize semi-structured JSON data into a flat table
  - for문을 사용하지 않고도 JSON 데이터를 손쉽게 DataFrame형태로 전환 가능

# 수고하셨습니다.

Q & A