Intermediate Python - Filip Schouwenaars

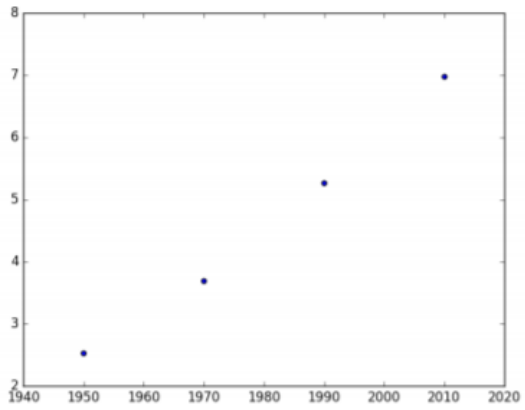
Note as of Dec 2019
Note Taker: Paris Zhang

Ch 1 - Matplotlib

Scatter Plot

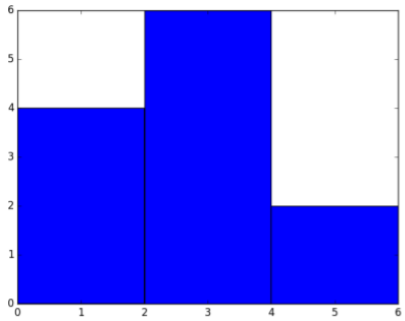
```
In [1]: import matplotlib.pyplot as plt
In [2]: year = [1950, 1970, 1990, 2010]
In [3]: pop = [2.519, 3.692, 5.263, 6.972]
In [4]: plt.scatter(year, pop)
In [5]: plt.show()
```

Adjust size using `s`.
Adjust color using `c` by creating color dictionary.



Histogram

```
In [3]: values = [0,0.6,1.4,1.6,2.2,2.5,2.6,3.2,3.5,3.9,4.2,6]
In [4]: plt.hist(values, bins = 3)
In [5]: plt.show()
```



Customization

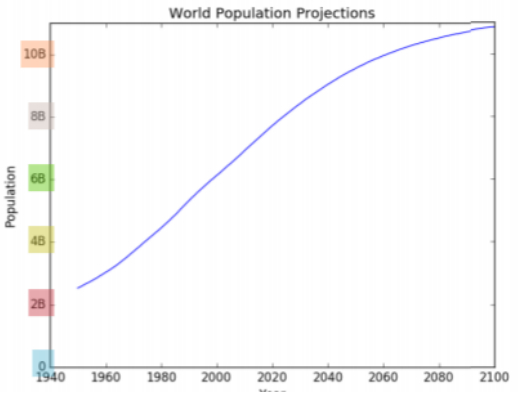
Axis labels, title, ticks

```
import matplotlib.pyplot as plt
year = [1950, 1951, 1952, ..., 2100]
pop = [2.538, 2.57, 2.62, ..., 10.85]

plt.plot(year, pop)

plt.xlabel('Year')
plt.ylabel('Population')
plt.title('World Population Projections')
plt.yticks([0, 2, 4, 6, 8, 10],
           ['0', '2B', '4B', '6B', '8B', '10B'])

plt.show()
```



Add historical data

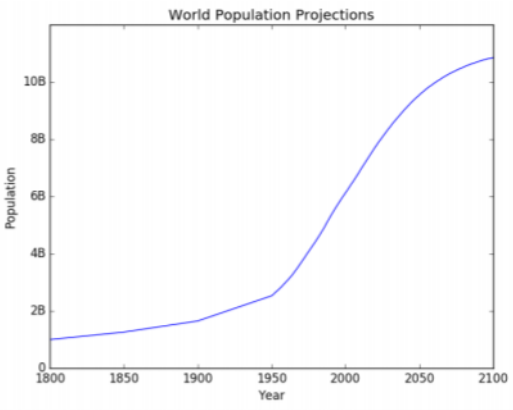
```
import matplotlib.pyplot as plt
year = [1950, 1951, 1952, ..., 2100]
pop = [2.538, 2.57, 2.62, ..., 10.85]

# Add more data
year = [1800, 1850, 1900] + year
pop = [1.0, 1.262, 1.650] + pop

plt.plot(year, pop)

plt.xlabel('Year')
plt.ylabel('Population')
plt.title('World Population Projections')
plt.yticks([0, 2, 4, 6, 8, 10],
           ['0', '2B', '4B', '6B', '8B', '10B'])

plt.show()
```



Customization Example

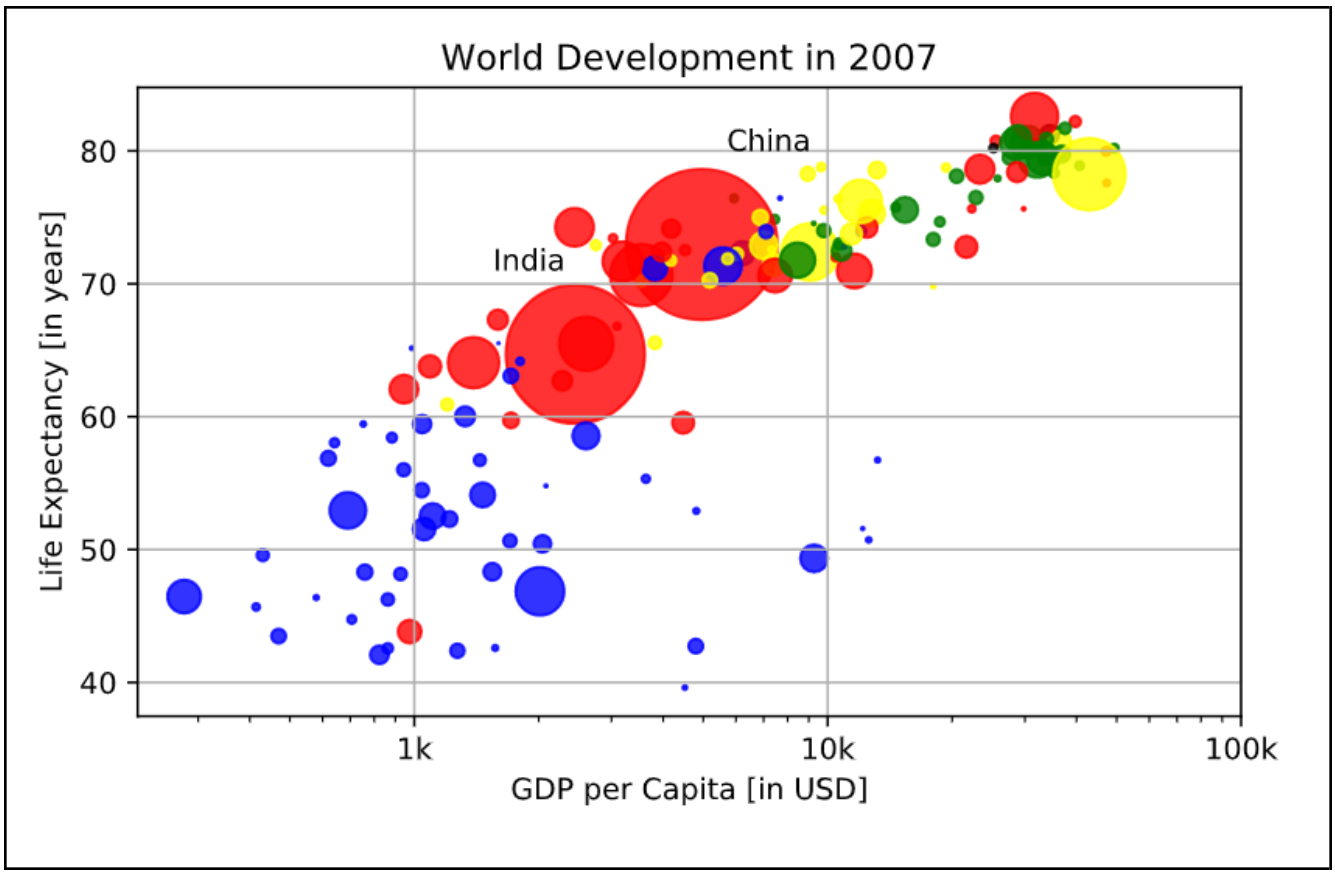
```
# Scatter plot
plt.scatter(x = gdp_cap, y = life_exp, s = np.array(pop) * 2,
c = col, alpha = 0.8)

# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
plt.xticks([1000,10000,100000], ['1k','10k','100k'])

# Additional customizations
plt.text(1550, 71, 'India')
plt.text(5700, 80, 'China')

# Add grid() call
plt.grid(True)

# Show the plot
plt.show()
```



Ch 2 - Dictionaries & Pandas

Dictionary

Define

```
In [1]: world = {"afghanistan":30.55, "albania":2.77, "algeria":39.21}

In [2]: world["albania"]
Out[2]: 2.77

In [3]: world = {"afghanistan":30.55, "albania":2.77,
                 "algeria":39.21, "albania":2.81}

In [4]: world
Out[4]: {'afghanistan': 30.55, 'albania': 2.81, 'algeria': 39.21}
```

keys have to be "immutable" objects

```
In [5]: {0:"hello", True:"dear", "two":"world"}
Out[5]: {0: 'hello', True: 'dear', 'two': 'world'}

In [6]: {["just", "to", "test"]: "value"}
TypeError: unhashable type: 'list'
```

Assign

```
In [8]: world["sealand"] = 0.000027

In [9]: world
Out[9]: {'afghanistan': 30.55, 'albania': 2.81,
                'algeria': 39.21, 'sealand': 2.7e-05}

In [10]: "sealand" in world
Out[10]: True

In [11]: world["sealand"] = 0.000028

In [12]: world
Out[12]: {'afghanistan': 30.55, 'albania': 2.81,
          'algeria': 39.21, 'sealand': 2.8e-05}

In [13]: del(world["sealand"])

In [14]: world
Out[14]: {'afghanistan': 30.55, 'albania': 2.81, 'algeria': 39.21}
```

Pandas

DataFrame from dictionary

```
In [2]:  dict = {
    "country":["Brazil", "Russia", "India", "China", "South Africa"],
    "capital":["Brasilia", "Moscow", "New Delhi", "Beijing", "Pretoria"],
       "area":[8.516, 17.10, 3.286, 9.597, 1.221]
 "population":[200.4, 143.5, 1252, 1357, 52.98] }
```

keys (column labels) **values (data, column by column)**

```
In [3]: import pandas as pd

In [4]: brics = pd.DataFrame(dict)
```

```
In [5]: brics
Out[5]:
     area    capital       country  population
0   8.516   Brasilia        Brazil      200.40
1  17.100     Moscow        Russia      143.50
2   3.286  New Delhi         India     1252.00
3   9.597    Beijing         China     1357.00
4   1.221   Pretoria  South Africa       52.98

In [6]: brics.index = ["BR", "RU", "IN", "CH", "SA"]
In [7]: brics
Out[7]:
     area    capital       country  population
BR   8.516   Brasilia        Brazil      200.40
RU  17.100     Moscow        Russia      143.50
IN   3.286  New Delhi         India     1252.00
CH   9.597    Beijing         China     1357.00
SA   1.221   Pretoria  South Africa       52.98
```

DataFrame from csv

brics.csv

```
,country,capital,area,population
BR,Brazil,Brasilia,8.516,200.4
RU,Russia,Moscow,17.10,143.5
IN,India,New Delhi,3.286,1252
CH,China,Beijing,9.597,1357
SA,South Africa,Pretoria,1.221,52.98
```

CSV = comma-separated values

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

In [2]: brics = pd.read_csv("path/to/brics.csv", index_col = 0)

In [3]: brics
Out[3]:
         country    capital    area  population
BR        Brazil   Brasilia   8.516      200.40
RU        Russia     Moscow  17.100      143.50
IN         India  New Delhi   3.286     1252.00
CH         China    Beijing   9.597     1357.00
SA  South Africa   Pretoria   1.221       52.98
```

Index and Select Data

1. Square brackets:
    a. Column access:
    `cars['cars_per_cap']` returns Pandas Series
    `cars[['cars_per_cap']]` returns Pandas DataFrame
    b. Row access:
    `cars[0:5]` returns the first 5 rows (may only use a slice, i.e., a range in integers)
2. Advanced methods: `loc` and `iloc`

Each of these pairs return the same results	

`cars.loc['RU']`
`cars.iloc[4]`
	`In [1]: cars.iloc[4]`
	`Out[1]:`
	`cars_per_cap 200`
	`country Russia`
	`drives_right True`
	`Name: RU, dtype: object`

`cars.loc[['RU']]`
`cars.iloc[[4]]`
	`In [2]: cars.iloc[[4]]`
	`Out[2]:`
	`cars_per_cap country drives_right`
	`RU 200 Russia True`

```
cars.loc[['RU', 'AUS']]
cars.iloc[[4, 1]]
```

```
In [3]: cars.iloc[[4, 1]]
Out[3]:
     cars_per_cap    country  drives_right
RU            200     Russia          True
AUS           731  Australia         False
```

Select both rows and columns from a DataFrame

```
cars.loc['IN', 'cars_per_cap']
cars.iloc[3, 0]
```

```
In [1]: cars.iloc[3, 0]
Out[1]: 18
```

```
cars.loc[['IN', 'RU'], 'cars_per_cap']
cars.iloc[[3, 4], 0]
```

```
In [2]: cars.iloc[[3, 4], 0]
Out[2]:
IN     18
RU    200
Name: cars_per_cap, dtype: int64
```

```
cars.loc[['IN', 'RU'], ['cars_per_cap',
'country']]
cars.iloc[[3, 4], [0, 1]]
```

```
In [3]: cars.iloc[[3, 4], [0, 1]]
Out[3]:
    cars_per_cap country
IN            18   India
RU           200  Russia
```

Select only columns

```
cars.loc[:, 'country']
cars.iloc[:, 1]
```

```
In [1]: cars.iloc[:, 1]
Out[1]:
US     United States
AUS        Australia
JPN            Japan
IN             India
RU            Russia
MOR          Morocco
EG             Egypt
Name: country, dtype: object
```

```
cars.loc[:,
['country','drives_right']]
cars.iloc[:, [1, 2]]
```

```
In [2]: cars.iloc[:, [1, 2]]
Out[2]:
           country  drives_right
US   United States          True
AUS      Australia         False
JPN          Japan         False
IN           India         False
RU          Russia          True
MOR        Morocco          True
EG           Egypt          True
```

Ch 3 - Logic, Control Flow and Filtering

Ch 4 - Loops

Ch 5 - Case Study: Hacker Statistics