동국대학교
dongguk university

Numpy, Pandas, Matplotlib

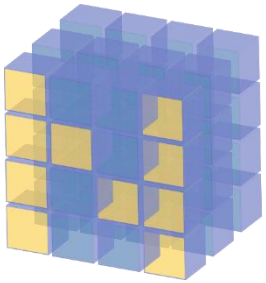
Machine Learning

Numpy, Pandas, Matplotlib

- Contents
  - Numpy
  - Pandas
  - Matplotlib
  - Scikit-learn

What is Numpy?

- Numpy
  - One of the library that provides efficient and easy operation of the large-scale and multi-dimensional matrix in the python
  - It can make us to implement matrix operation, generate random numbers and fourier transform more easily than C, C++, Fortran
  - It is partially implemented in C, Fortran internally so, its speed of running is fast
  - Mostly, it is used with other python libraries (Scipy, Sympy, Matplotlib, Pandas) to implement various statistical and numerical analysis techniques



NumPy

Array Creation

- array()
  - Example Code

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

→ Display full output in Jupyter cell, not only last result.

```
import numpy as np

a = np.array([[0, 1, 2, 3],
              [5, 6, 7, 8],
              [10, 11, 12, 13]])
a

a.ndim # 2 dimensions
a.shape # 3x4
```

```
array([[ 0, 1, 2, 3],
       [ 5, 6, 7, 8],
       [10, 11, 12, 13]])

2

(3, 4)
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Array Creation

1. np.arange() & np.reshape()

```
a = np.arange(6)
a # np.array([0, 1, 2, 3, 4, 5])
```

```
array([0, 1, 2, 3, 4, 5])
```

```
b = a.reshape(2,3)
b # np.array([[0, 1, 2], [3, 4, 5]])
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

2. np.zeros()

```
# initialized all 0
x = np.zeros((3, 4))
x
```

```
array([[ 0., 0., 0., 0.],
       [ 0., 0., 0., 0.],
       [ 0., 0., 0., 0.]])
```

3. np.ones()

```
# initialized all 1
y = np.ones((2, 3, 4), dtype = np.float64)
y
```

```
array([[[ 1., 1., 1., 1.],
        [ 1., 1., 1., 1.],
        [ 1., 1., 1., 1.]],
       [[ 1., 1., 1., 1.],
        [ 1., 1., 1., 1.],
        [ 1., 1., 1., 1.]]])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Array Creation

4. np.identity()

```
# identity matrix of size 3 x 3
i = np.identity(n=3)
i
```

```
array([[ 1., 0., 0.],
       [ 0., 1., 0.],
       [ 0., 0., 1.]])
```

5. np.random.randn()

```
# Return samples from the "standard normal" distribution.
r = np.random.randn(3,3)
r
```

```
array([[-0.03318424, 0.07825921, -0.59816703],
       [-0.91686982, -0.18075674, 1.92220989],
       [ 0.84470512, -1.42788256, 0.44475968]])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Array Operations

- +, -, *, matrix multiplication, matrix transpose

```
a = np.array([[1, 1], [0, 1]])
b = np.array([[2, 0], [3, 4]])

a
b

a * 10 # matrix * scalar
a + b # elementwise addition
a * b # elementwise product
```

```
array([[1, 1],
       [0, 1]])

array([[2, 0],
       [3, 4]])

array([[10, 10],
       [ 0, 10]])

array([[3, 1],
       [3, 5]])

array([[2, 0],
       [0, 4]])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Array Operations

- +, -, *, matrix multiplication, matrix transpose

```
a @ b    # matrix product
a.dot(b) # matrix product

a.T      # matrix transpose
```

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

array([[5, 4],
       [3, 4]])

array([[1, 0],
       [1, 1]])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Indexing

- Indexing, slicing

```
a = np.arange(20).reshape(4, 5)
a
```

```
array([[ 0, 1, 2, 3, 4],
       [ 5, 6, 7, 8, 9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

```
# indexing
a[2, 2]

# slicing
a[0:2, 0:2]
a[0:2, :]
a[:, 0:2]
```

```
12

array([[0, 1],
       [5, 6]])
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
array([[ 0, 1],
       [ 5, 6],
       [10, 11],
       [15, 16]])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Stacking

- vstack(), hstack()

```
a = np.array([[1, 1],[0, 1]])
b = np.array(10*np.random.random((2, 2)))
a
B

np.vstack((a, b)) # append row
np.hstack((a, b)) # append column
```

```
array([[1, 1],
       [0, 1]])
array([[ 8.95081961, 8.17967375],
       [ 8.91370118, 6.80621455]])


array([[ 1. , 1. ],
       [ 0. , 1. ],
       [ 8.95081961, 8.17967375],
       [ 8.91370118, 6.80621455]])
array([[ 1. , 1. , 8.95081961, 8.17967375],
       [ 0. , 1. , 8.91370118, 6.80621455]])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Axis

- For array of shape M x N,
  - axis = 0 : M rows
  - axis = 1 : N columns

```
a = np.arange(6).reshape(2,3)
a
a.sum(axis = 0) # axis = 0, x axis(row)
a.sum(axis = 1) # axis = 1, y axis(colum)
```

```
array([[0, 1, 2],
       [3, 4, 5]])

array([3, 5, 7])

array([ 3, 12])
```

```
a = np.array([[20, 10], [2, 1]])
a
np.sort(a, axis = 0)
np.sort(a, axis = 1)
```

```
array([[20, 10],
       [ 2, 1]])

array([[ 2, 1],
       [20, 10]])

array([[10, 20],
       [ 1, 2]])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

Basic Statistics

- max(), min()

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

np.max(a) # max
np.min(a) # min
```

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

- mean(), std(), var(), …

```
np.mean(a) # mean
np.mean(a, axis=0)
np.mean(a, axis=1)

np.std(a) # standard deviation
np.std(a, axis=0)
np.std(a, axis=1)

np.var(a) # variance
np.var(a, axis=0)
np.var(a, axis=1)
```

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

0.4330127018922193
array([0.5, 0. ])
array([0. , 0.5])

0.1875
array([0.25, 0. ])
array([0. , 0.25])
```

https://docs.scipy.org/doc/numpy/user/quickstart.html

What is Pandas?

- Pandas
  - One of the python libraries in python to analyze data
  - It provides data structure ‘Dataframe’, so that it makes data analysis efficiently by the various operation
  - It provides a variety of data structure to sort data by the name of the axis
  - It can make us to handle both time series and non-time series data
  - It provides flexible handling of missing data(NA, etc.)
  - It is also able to perform relation calculation in SQL

Object Creation

- Series, DataFrame

```
import numpy as np
import pandas as pd

s = pd.Series([1, 3, 5, 7, 9])
s
```

```
0 1
1 3
2 5
3 7
4 9
dtype: int64
```

```
data = {'Name':['John', 'Bill', 'Tom'],
        'height':[172, 168, 185],
        'weight':[67, 72, 88]}
df = pd.DataFrame(data)
df
```

	Name	height	weight
0	John	172	67
1	Bill	168	72
2	Tom	185	88

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#min

Getting Data In

- read_csv()

```
# Read data from csv file
toycars = pd.read_csv('toycars.csv')
toycars
```

	angle	distance	car
0	1.3	0.43	1
1	1.3	0.37	2
2	1.3	0.27	3
3	4.0	0.84	1
4	4.0	0.92	2
5	4.0	0.69	3
6	2.7	0.58	1
7	2.7	0.64	2
8	2.7	0.55	3

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#min

Viewing Data

- head() & tail()

```
# head() & tail()
toycars.head(2)
toycars.tail(2)
```

	angle	distance	car
0	1.3	0.43	1
1	1.3	0.37	2

	angle	distance	car
7	2.7	0.64	2
8	2.7	0.55	3

- index & columns

```
# index & columns
toycars.index
toycars.columns
```

```
RangeIndex(start=0, stop=9, step=1)
Index(['angle', 'distance', 'car'], dtype='object')
```

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#min

Viewing Data

- describe()

```
# describe()
toycars.describe()
```

	angle	distance	car
count	9.000000	9.000000	9.000000
mean	2.666667	0.587778	2.000000
std	1.169402	0.212707	0.866025
min	1.300000	0.270000	1.000000
25%	1.300000	0.430000	1.000000
50%	2.700000	0.580000	2.000000
75%	4.000000	0.690000	3.000000
max	4.000000	0.920000	3.000000

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#min

Selection

- by labels

```
# 'loc'ation : [row index, [column name]]
toycars.loc[0:3, ['angle', 'distance']]
```

	angle	distance
0	1.3	0.43
1	1.3	0.37
2	1.3	0.27
3	4.0	0.84

- by position

```
# 'i'nteger 'loc'ation : select via position, numpy style
toycars.iloc[[2, 4, 6], [1, 2]]
```

	distance	car
2	0.27	3
4	0.92	2
6	0.58	1

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#min

Selection

- Boolean indexing

```
# Select only samples with anlgle value more than 0.6
toycars[toycars.angle > 0.6]
```

	angle	distance	car
0	1.3	0.43	1
1	1.3	0.37	2
2	1.3	0.27	3
3	4.0	0.84	1
4	4.0	0.92	2
5	4.0	0.69	3
6	2.7	0.58	1
7	2.7	0.64	2
8	2.7	0.55	3

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#min

What is Matplotlib?

- Matplotlib
  - One of the library in the python to visualize the data
  - It provides Object-Oriented API that can embed various data an numerical data in application through general GUI-toolkits
  - Pylab in Matplotlib includes most of interfaces provided in MATLAB
  - Usually, it is utilized with Scipy, Numpy and Pandas as well as to visualize the results of training of the data from Scikit-learn and Tensorflow

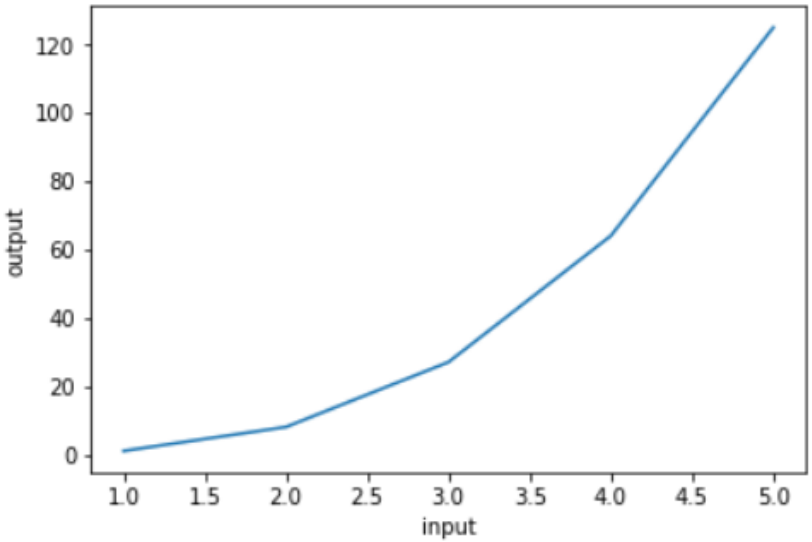
Plot

- plt.plot(x, y)

```
import matplotlib.pyplot as plt
import numpy as np
%pylab inline
%matplotlib inline

x = [1, 2, 3, 4, 5]
y = [1, 8, 27, 64, 125]

plt.plot(x, y)
plt.ylabel('output')
plt.xlabel('input')
plt.show()
```



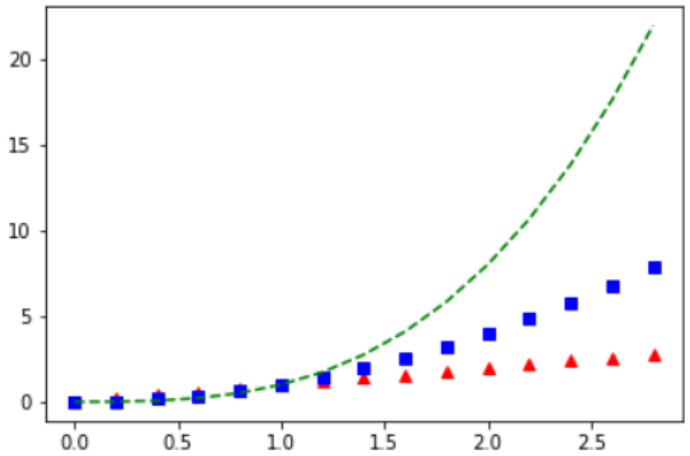
https://matplotlib.org/tutorials/index.html

Plot

- Color & shapes
  - Colors(c=)
    - Red : r
    - Green : g
    - Blue : b
  - Marker Shapes(marker=)
    - Triangle : ^
    - Circle : o
    - Star : *
    - Square : s
  - Line Styles(linestyle=)
    - Normal line : -
    - Dashed line : --

```
# evenly sampled time at 200ms intervals
t = np.arange(0, 3, 0.2)
t
# red triangles, blue squares and green dashes
plt.plot(t, t, 'r^', t, t**2, 'bs', t, t**3, 'g--')
plt.show()
```

```
[0.  0.2 0.4 0.6 0.8 1.  1.2 1.4 1.6 1.8 2.  2.2 2.4 2.6 2.8]
```



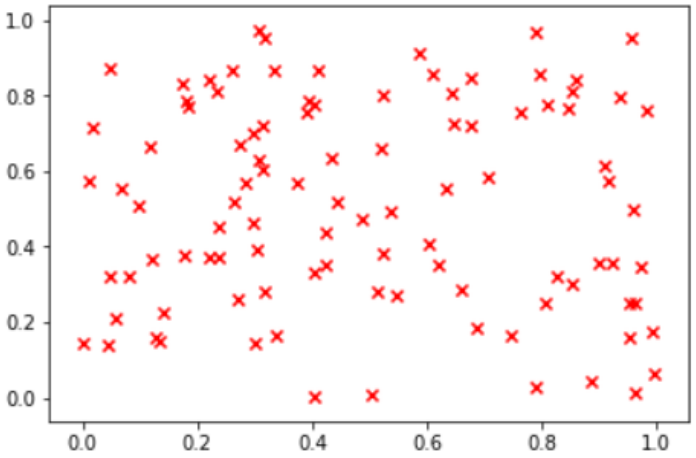
https://python-graph-gallery.com/cheat-sheets/
https://matplotlib.org/tutorials/index.html

Scatter Plot

- plt.scatter(x, y)

```
data = np.random.rand(100, 2)

plt.scatter(data[:,0], data[:,1],
            color='red', marker='x')
plt.show()
```



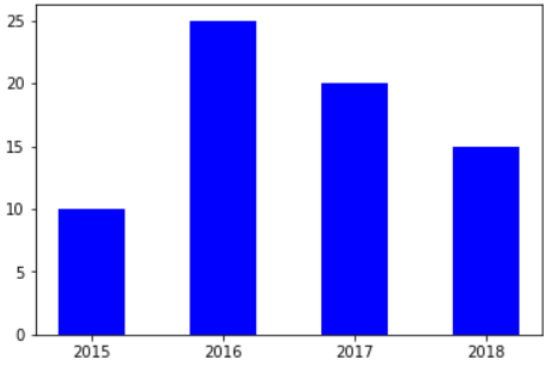
https://matplotlib.org/tutorials/index.html

Bar Plot

- plt.bar()

```
# old version
x = range(4)
data = [10.0, 25.0, 20.0, 15.0]
index = ['2015','2016','2017', '2018']

plt.bar(x, data, color='b', width=0.5)
plt.xticks(x, index)
plt.show()
```




```
# current version (without xticks)
data = [10.0, 25.0, 20.0, 15.0]
index = ['2015','2016','2017', '2018']

plt.bar(index, data, color='b', width=0.5)
plt.show()
```

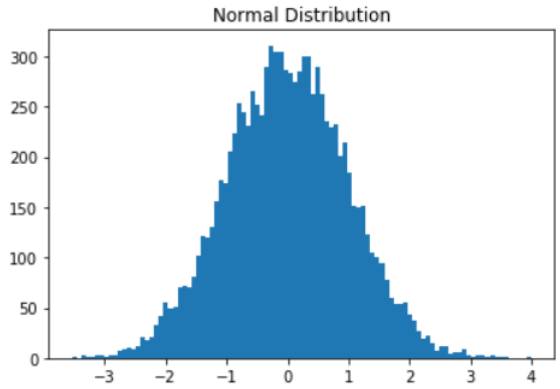
https://matplotlib.org/tutorials/index.html

Histogram

- plt.hist()

```
x = np.random.randn(10000)

plt.hist(x, bins=100)
plt.title('Normal Distribution')
plt.show()
```



https://matplotlib.org/tutorials/index.html

Matplotlib

- Too many tutorials. See the documents!

https://matplotlib.org/tutorials/index.html

Scikit-learn

- Scikit-learn
  - Simple and efficient python library for Datamining, Data Analysis, and Machine Learning
  - It does not deal with large-scale deep learning or reinforcement learning, but provides various algorithms necessary for machine learning such as classification, regression, clustering, and dimension reduction.
  - It provides various machine learning methods such as hyper-parameter tuning and model optimization.
  - It is very easy to learn because it is compatible with other libraries in Python and has an internally unified interface
  - Optimal framework for beginner of machine learning

https://scikit-learn.org/stable/

Training a Model

- Task
  - Classify x = [x1, x2, x3] to y = 1 or 0
- Training datatset
  - 5 instances (example data) with known labels

```
X = np.array([[0, 1, 1],
              [1, 0, 1],
              [1, 1, 1],
              [0, 1, 1],
              [0, 0, 1]])

y = np.array([1, 0, 1, 1, 0])
```

features label

5 instances

x1	x2	x3	y
0	1	1	1
1	0	1	0
1	1	1	1
0	1	1	1
0	0	1	0

Training a Model

- Training (learning) – fit
  - Learning Decision Tree Classifier model with training dataset

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf = clf.fit(X, y)
clf
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
      max_features=None, max_leaf_nodes=None,
      min_impurity_decrease=0.0, min_impurity_split=None,
      min_samples_leaf=1, min_samples_split=2,
      min_weight_fraction_leaf=0.0, presort=False, random_state=None,
      splitter='best')
```

Predicting using the Model

- Test dataset
  - 2 new instances

```
X_test = np.array([[0, 0, 0],
                   [1, 1, 0]])

y_test = np.array([0, 1])
```

new instances

x1	x2	x3	y
0	0	0	
1	1	0	

Predicting using the Model

- Predicting labels – predict
  - Predict the label of new x using the learned model

```
predicted = clf.predict(X_test)
predicted
```

```
array([0, 1])
```

- Evaluation : Accuracy

```
# Accuracy
acc = 100 * np.sum(y_test == predicted)/len(y_test)
acc
```

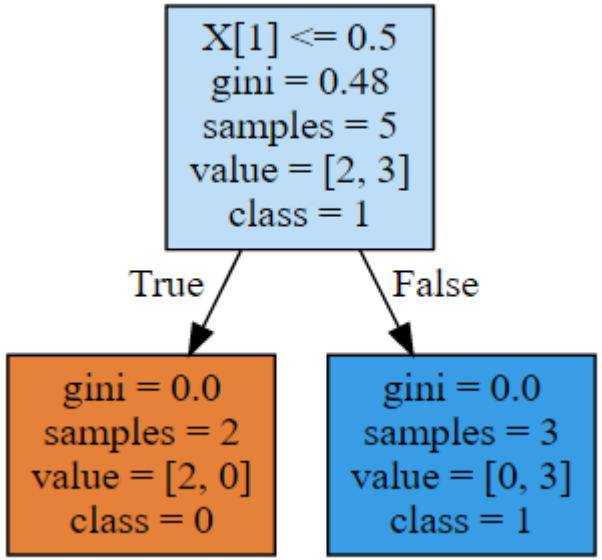
```
100.0
```

Visualize the Model

- Visualizing Decision Tree model using graphviz

```
from sklearn import tree
import graphviz

dot_data = tree.export_graphviz(clf,
                                filled=True,
                                out_file=None,
                                class_names=['0','1'])
graph = graphviz.Source(dot_data)
graph
```



Submit

- To make sure if you have completed this practice,
  Submit your practice file(Week03_givencode.ipynb) to e-class.
- **Deadline : tomorrow 11:59pm**
- Modify your ipynb file name as “Week03_StudentNum_Name.ipynb”
  Ex) **Week03_2020123456_홍길동.ipynb**
- You can upload this file without taking the quiz, but homework will be provided like a quiz every three weeks, so it is recommended to take the quiz as well.

Quiz 1

- 2D array
  - Make 10 x 10 matrix A using arange(1,101), and print A
  - Make 2 x 6 matrix B by slicing 0-1 rows 0-5 columns of A, and print B, B transpose
  - Compute matrix C = 0.1 (B x B transpose) (x : dot product)
- Desired Output

Matrix A →

```
[[  1  2  3  4  5  6  7  8  9 10]
 [ 11 12 13 14 15 16 17 18 19 20]
 [ 21 22 23 24 25 26 27 28 29 30]
 [ 31 32 33 34 35 36 37 38 39 40]
 [ 41 42 43 44 45 46 47 48 49 50]
 [ 51 52 53 54 55 56 57 58 59 60]
 [ 61 62 63 64 65 66 67 68 69 70]
 [ 71 72 73 74 75 76 77 78 79 80]
 [ 81 82 83 84 85 86 87 88 89 90]
 [ 91 92 93 94 95 96 97 98 99 100]]
```

Matrix B →

```
[[ 1  2  3  4  5  6]
 [11 12 13 14 15 16]]
```

```
[[ 1 11]
 [ 2 12]
 [ 3 13]
 [ 4 14]
 [ 5 15]
 [ 6 16]]
```

← Transposed matrix of B

```
[[  9.1  30.1]
 [ 30.1 111.1]]
```

← Matrix C

Quiz 2

- Vector computation
  - From $\mathbf{x}$ =[$x1$,$x2$,$x3$] and w=[$w1$,$w2$,$w3$] and b, y is computed as follows:

$$y=\mathbf{w}\cdot\mathbf{x}+b=w1x1+w2x2+w3x3+b$$

  - For following w, b and X, compute y for each row of X

- Desired Output

```
[10.1 20.1 30.1 40.1 50.1]
```

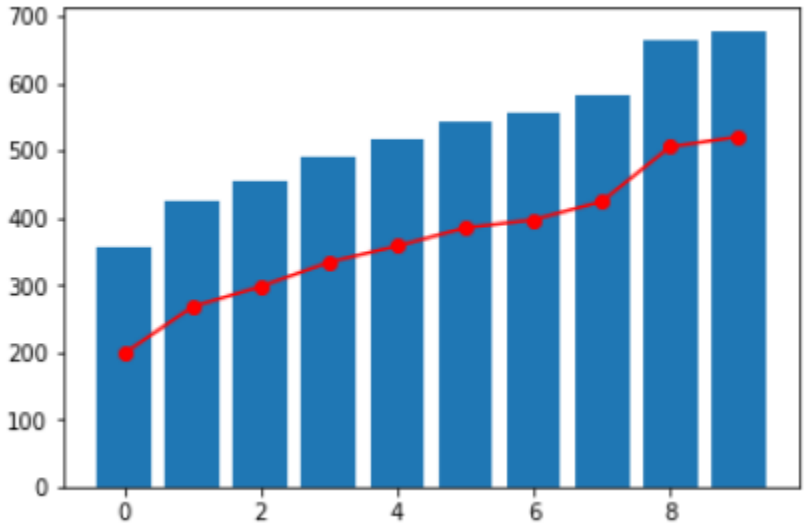
Quiz 3

- DataFrame and plot
  - The temperature (temp) and iced americano sales (sale) have the following relation
    - sale = 30.08 * temp - 159.47
  - Make a 1D array X from following 'temps' list
  - Compute 1D array Y for sales using above equation
  - Make 2D array C such that fist column is X, second column is Y, using vstack() and transpose
  - Make a DataFrame from C with column names "Temp" and "Sale" (use column = ["Temp", "Sale"])
  - Bar plot 30 * Temp values, and plot Sale values on it as a red line and marker "o" (use color="red", marker="o")

Quiz 3

- Desired output

	Temp	Sale
0	11.9	198.482
1	14.2	267.666
2	15.2	297.746
3	16.4	333.842
4	17.2	357.906
5	18.1	384.978
6	18.5	397.010
7	19.4	424.082
8	22.1	505.298
9	22.6	520.338



Quiz 4

- Visualizing Boston Housing Price Dataset
  - Load the dataset as a DataFrame (given)
  - Visualize the relationship between two variables using scatter plot
    - RM and MEDV
    - CRIM and MEDV
- Desired output