Python review 2

Vectorization using Numpy and Pandas

`list` and `dict` are not ideal for data analysis

```python
# height and weight
data = [
    [170, 68],
    [180, 70],
    [160, 60],
    [150, 55],
    [175, 65]
]

# average height
total_height = 0
for row in data:
    total_height += row[0]
average_height = total_height / len(data)
print(average_height)

# bmi
for row in data:
    height = row[0]
    weight = row[1]
    bmi = weight / (height / 100) ** 2
    print(bmi)
```

Vectorization

```
import numpy as np

data = np.array([
    [170, 68],
    [180, 70],
    [160, 60],
    [150, 55],
    [175, 65]
])

# select height and weight
height = data[:, 0]
weight = data[:, 1]

# average height
average_height = np.mean(height)

# bmi
bmi = weight / (height / 100) ** 2
```

Numpy array for matrix

$$np_{1d} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

$$np_{2d} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```python
np_1d = np.array([1, 2, 3, 4, 5])
print(np_1d.shape) # (5,)

np_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(np_2d.shape) # (3, 3)
```

Subsetting

$$np_{2d} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
np_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(np_2d[0])     # [1 2 3]
print(np_2d[0, 1])  # 2
print(np_2d[:, 1])  # [2 5 8]
print(np_2d[0, :])  # [1 2 3]
print(np_2d[1:, :]) # [[4 5 6] [7 8 9]]
```

Filtering

```
np_1d = np.array([1, 2, 3, 4, 5])

cond = np_1d > 3   # [False False False  True  True]
print(np_1d[cond]) # [4 5]
```

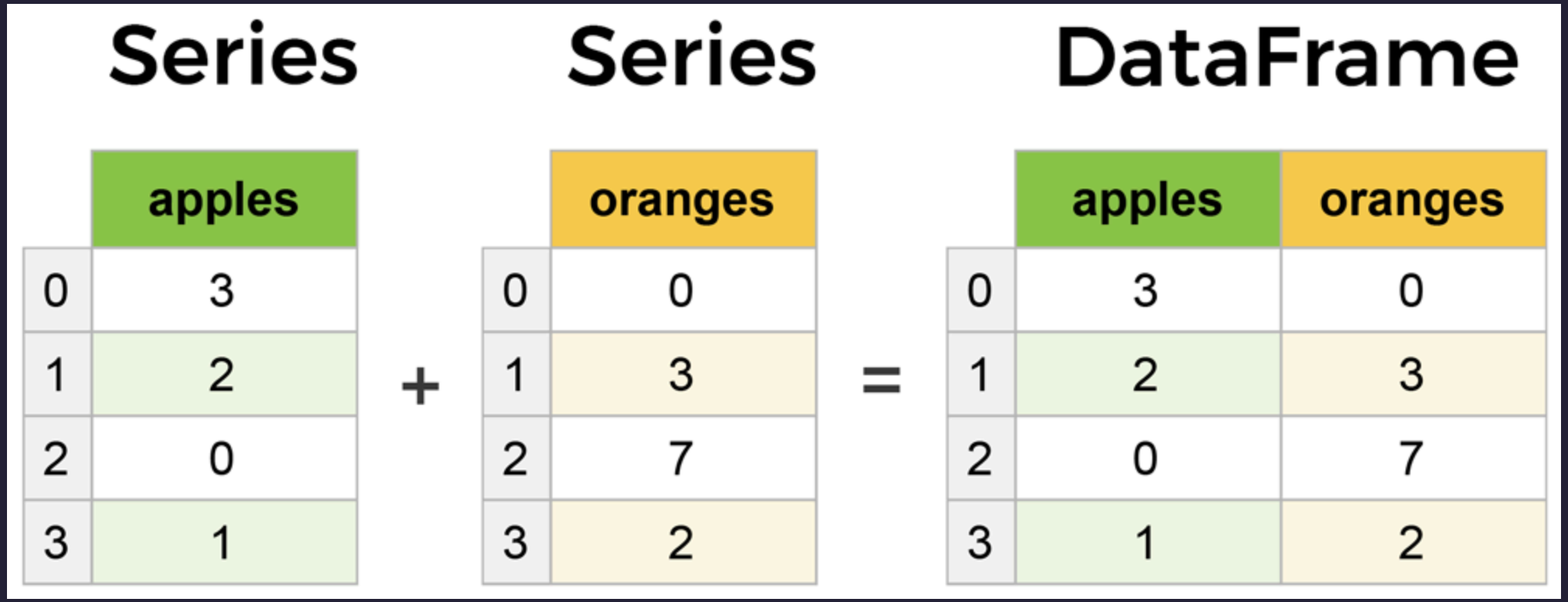
Mathematical operations

```
np_1da = np.array([1, 2, 3, 4, 5])
np_1db = np.array([6, 7, 8, 9, 10])

print(np_1da + np_1db) # [ 7  9 11 13 15]
print(np_1da - np_1db) # [-5 -5 -5 -5 -5]
print(np_1da * np_1db) # [ 6 14 24 36 50]

print(np.mean(np_1da)) # 3.0
print(np.sum(np_1da))  # 15
print(np.std(np_1da))  # 1.4142135623730951
```

Pandas Series and DataFrame



Pandas DataFrame for tabular data

```
import pandas as pd

data = {
    'name': ['John', 'Jane', 'Mary'],
    'age': [25, 30, 27]
}
df = pd.DataFrame(data)

print(df.index)
print(df.columns)
print(df.head())
```

Subsetting

Selecting columns

```
df['name']
df[['name', 'age']]
df.loc[:, 'name']
```

Selecting rows

```
df.loc[0]
df.loc[0:2]
```

Selecting rows and columns

```
df.loc[0, 'name']
df.loc[0:2, ['name', 'age']]
```

Filtering

```
cond = df['age'] > 25
df[cond]
df.loc[cond]

cond2 = (df['age'] > 25) & (df['name'] == 'John')
df.loc[cond2, 'name']
```

Mathematical operations

```
df['age'] + 5
df['age'] * 2

df['age'].mean()
df['age'].sum()

df['bmi'] = df['weight'] / (df['height'] / 100) ** 2
```

💻 5. HR Data Analysis (`pandas` or `numpy`)

1. Create a Pandas DataFrame (or Numpy array) from the employee data.
2. Use filtering to select employees from the “IT” department.
3. Use another filter to select employees with a salary greater than $60,000.
4. Calculate the average salary of all employees.
5. Calculate the average salary of the employees in the “IT” department.