Python review 2

Vectorization using Numpy and Pandas

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

```
# height and weight
data = [
    [170, 68],
    [180, 70],
    [160, 60],
]

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

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

Vectorization

```
import numpy as np

# height and weight
data = np.array([
    [170, 68],
    [180, 70],
    [160, 60],
])

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

# 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

$$np_{1d} = [1 \quad 2 \quad 3 \quad 4 \quad 5]$$

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

print(np_1d[0])     # 1
print(np_1d[1:3])   # [2 3]
print(np_1d[1:])    # [2 3 4 5]
```

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]
```

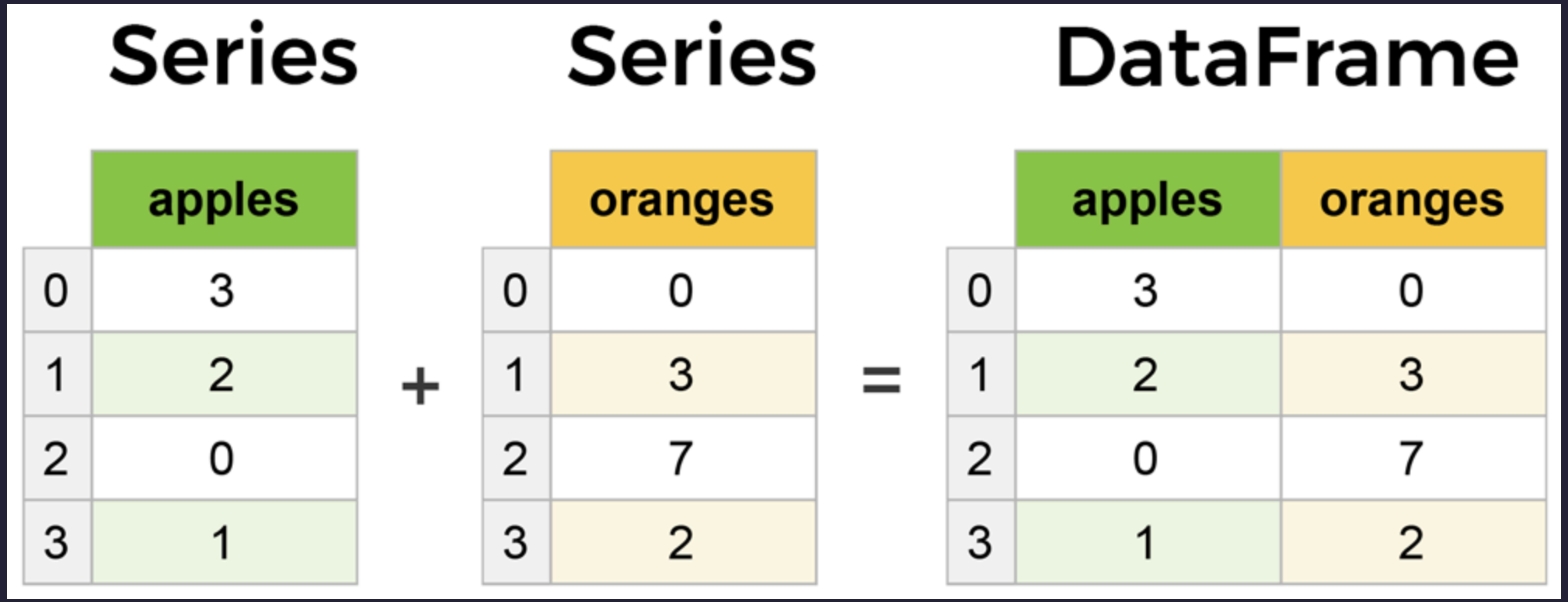
Mathematical operations

```
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 (1d array) & DataFrame (2d array)



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())
```

	name	age
0	John	25
1	Jane	30
2	Mary	27

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']]
```

	name	age
0	John	25
1	Jane	30
2	Mary	27

Filtering

```
cond = df['age'] > 25 # [False True True]
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
```

Convert between Numpy and Pandas

```
# Convert DataFrame to NumPy array
np_2d = df.to_numpy()
np_2d = df.values

# Convert NumPy array to DataFrame
df2 = pd.DataFrame(np_2d, columns=['name', 'age'])

# Convert Series to NumPy array
np_1d = df['age'].to_numpy()
np_1d = df['age'].values

# Convert NumPy array to Series
s = pd.Series(np_1d)
```

💻 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.

```python
import pandas as pd

# Step 1: Create a DataFrame with employee data
data = {
    'Name': ['John', 'Alice', 'Bob', 'Claire', 'Dan', 'Eva'],
    'Age': [28, 34, 45, 29, 40, 32],
    'Department': ['IT', 'HR', 'IT', 'Finance', 'IT', 'HR'],
    'Salary': [55000, 62000, 70000, 48000, 72000, 59000]
}

df = pd.DataFrame(data)

# Step 2: Subset employees from the IT department
it_department = df[df['Department'] == 'IT']
print("Employees in the IT department:")
print(it_department)

# Step 3: Filter employees with a salary greater than 60,000
high_salary = df[df['Salary'] > 60000]
print("\nEmployees with salary greater than 60,000:")
print(high_salary)

# Step 4: Calculate the average salary of all employees
average_salary = df['Salary'].mean()
print(f"\nThe average salary of all employees is: {average_salary}")

# Step 5: Calculate the average salary of IT employees
average_salary_it = it_department['Salary'].mean()
print(f"The average salary of IT employees is: {average_salary_it}")
```

```python
import numpy as np

# Step 1: Create a 2D NumPy array with employee data
# Employee data: [Name, Age, Department, Salary]
employee_data = np.array([
    ['John', 28, 'IT', 55000],
    ['Alice', 34, 'HR', 62000],
    ['Bob', 45, 'IT', 70000],
    ['Claire', 29, 'Finance', 48000],
    ['Dan', 40, 'IT', 72000],
    ['Eva', 32, 'HR', 59000]
], dtype=object)  # Use object dtype to hold both strings and numbers

# Step 2: Subset employees from the IT department using NumPy
it_department_mask = employee_data[:, 2] == 'IT'  # Mask for IT department
it_employees = employee_data[it_department_mask]
print("Employees in the IT department:")
print(it_employees)

# Step 3: Filter employees with a salary greater than 60,000 using NumPy
high_salary_mask = employee_data[:, 3] > 60000  # Mask for salary > 60,000
high_salary_employees = employee_data[high_salary_mask]
print("\nEmployees with salary greater than 60,000:")
print(high_salary_employees)

# Step 4: Calculate the average salary of all employees using NumPy
average_salary = np.mean(salaries)
print(f"\nThe average salary of all employees is: {average_salary}")

# Step 5: Calculate the average salary of IT employees using NumPy
average_salary_it = np.mean(it_employees[:, 3])
print(f"The average salary of IT employees is: {average_salary_it}")
```