Vectorization using Numpy and Pandas

🤔 Calculate average height and BMI

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

Vectorization: A better way to handle tabular data

`list` & `dict`: Not optimized for tabular data

```
import numpy as np

# Create numpy array from list of lists
data = np.array([
    [170, 68],
    [180, 70],
    [160, 60],
])

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

average_height = np.mean(height)

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

Numpy

Package that provides foundation for vectorized operations in Python

```python
import numpy as np
```

Numpy arrays: Data structures for vectorization

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

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

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

np_2da = np.array([[1, 2, 3, 4, 5]])
print(np_2da.shape) # (1, 5)

np_2db = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(np_2db.shape) # (3, 3)
```

Subsetting (1d array)

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

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

Subsetting (2d array)

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

```
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 (1d array)

```
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
```

Mathematical operations (2d array)

`axis` parameter specifies the dimension along which the operation is performed.

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

print(np.sum(np_2d, axis=0)) # by column. [5 7 9]
print(np.sum(np_2d, axis=1)) # by row. [ 6 15]

print(np.mean(np_2d, axis=0)) # by column. [2.5 3.5 4.5]
print(np.mean(np_2d, axis=1)) # by row. [2. 5.]

print(np.dot(np_2d, np_2d.T)) # [[14 32] [32 77]]
```

Broadcasting

Array with smaller shape is "broadcast" to match the shape of the larger array.

```
np_2d = np.array([[1, 2, 3], [4, 5, 6]])
np_1d = np.array([10, 20, 30])
```

```
print(np_2d + 10)
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + 10 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix} = \begin{bmatrix} 11 & 12 & 13 \\ 14 & 15 & 16 \end{bmatrix}$$

```
print(np_2d + np_1d)
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 20 & 30 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 20 & 30 \\ 10 & 20 & 30 \end{bmatrix} = \begin{bmatrix} 11 & 22 & 33 \\ 14 & 25 & 36 \end{bmatrix}$$

Pandas

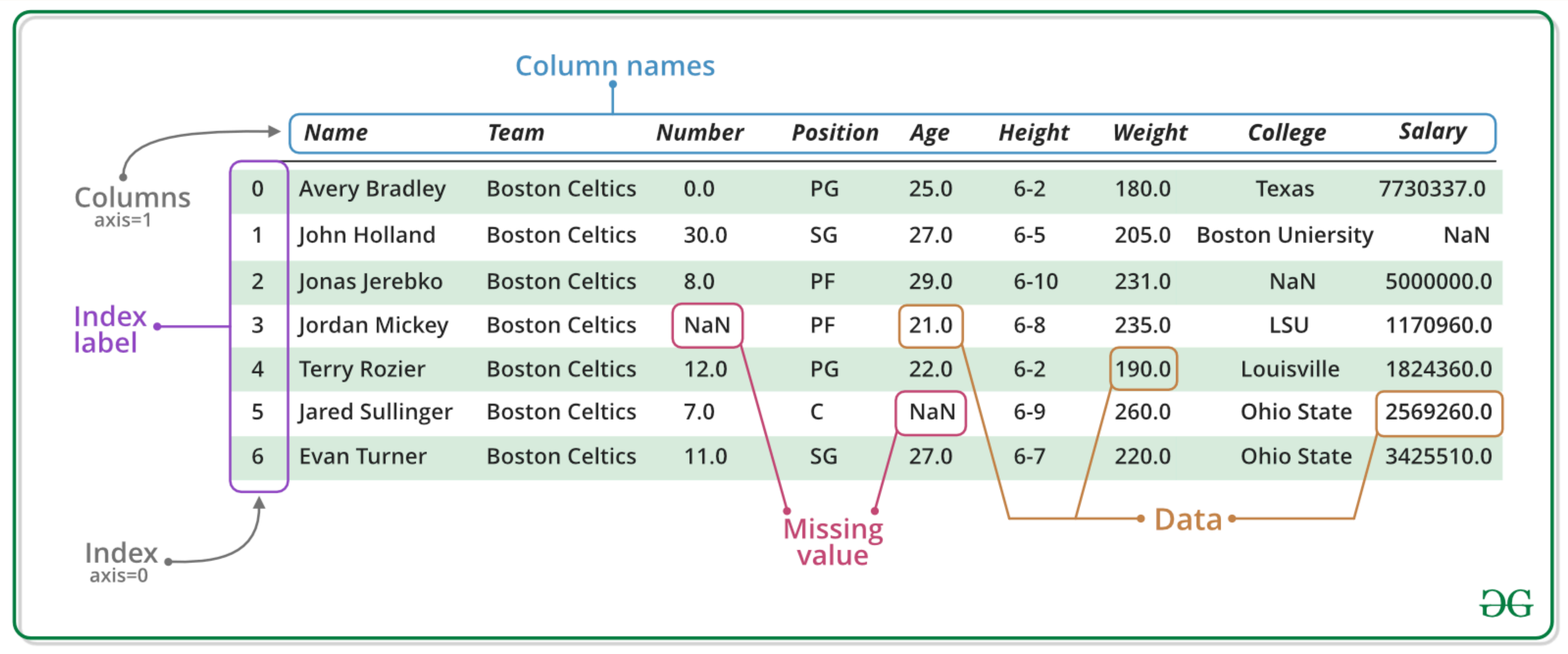
Built on top of NumPy

To support **user-friendly** vectorized operations for tabular data

```
import pandas as pd
```

Pandas `DataFrame`

Numpy array + Index & Column



	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

Pandas `Series` & `DataFrame`

`Series`: 1-d array
`DataFrame`: 2-d array

Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Pandas DataFrame for tabular data

```python
import pandas as pd

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

print(df.index)      # [0 1 2]
print(df.columns)    # ['name', 'age']
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 (using `loc`)

```
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)
```

💻 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.