Why Python for Machine Learning?

- **Simple & Readable:** Easy to learn and understand
- **Rich Ecosystem:** NumPy, Pandas, Scikit-learn, PyTorch, TensorFlow
- **Interactive Development:** Jupyter notebooks
- **Community:** Extensive documentation and support

```
# Python's simplicity
numbers = [1, 2, 3, 4, 5]
squared = [x**2 for x in numbers]
print(f"Original: {numbers}")
print(f"Squared: {squared}")
```

Python Basics Review

Variables and Data Types:

```
# Numbers
x = 42          # integer
y = 3.14        # float
z = 2 + 3j      # complex

# Collections
my_list = [1, 2, 3, 4]       # mutable
my_tuple = (1, 2, 3, 4)      # immutable
my_dict = {'a': 1, 'b': 2}  # key-value pairs
```

Control Flow:

```
# Loops and conditionals
for i in range(5):
    if i % 2 == 0:
        print(f"{i} is even")
```

NumPy: The Foundation

Why NumPy?

- **Performance:** 50-100x faster than pure Python
- **Memory Efficient:** Contiguous memory layout
- **Broadcasting:** Efficient operations on arrays
- **Vectorization:** Apply operations to entire arrays

```
import numpy as np

# List vs NumPy comparison
python_list = [1, 2, 3, 4]
numpy_array = np.array([1, 2, 3, 4])

# Vectorized operations
result = numpy_array * 2  # All elements multiplied at once
```

Creating NumPy Arrays

```
import numpy as np

# Different ways to create arrays
a = np.array([1, 2, 3, 4])           # From list
b = np.zeros(5)                       # Array of zeros
c = np.ones((2, 3))                   # 2x3 array of ones
d = np.arange(0, 10, 2)               # [0, 2, 4, 6, 8]
e = np.linspace(0, 1, 5)              # 5 points from 0 to 1
f = np.random.random((3, 3))          # Random 3x3 matrix

# Array properties
print(f"Shape: {a.shape}")
print(f"Data type: {a.dtype}")
print(f"Number of dimensions: {a.ndim}")
```

NumPy Array Operations

```
import numpy as np

# Create sample arrays
a = np.array([1, 2, 3, 4])
b = np.array([5, 6, 7, 8])
matrix = np.array([[1, 2], [3, 4]])

# Element-wise operations
print(a + b)         # [6, 8, 10, 12]
print(a * b)         # [5, 12, 21, 32]
print(a ** 2)        # [1, 4, 9, 16]

# Mathematical functions
print(np.sqrt(a))    # Square root
print(np.exp(a))     # Exponential
print(np.sin(a))     # Sine function
```

Basic Mathematical Objects

Scalar: Simple numerical values

```
scalar = 42
temperature = 23.5
```

Vectors: Ordered list of scalar values

```
vector = np.array([3, 4, 5, 6])
position = np.array([10.2, 15.7, 8.3])
```

Basic Mathematical Objects

Sets: Unordered collection of unique elements

```
unique_values = {1, 3, 5, 7}
```

Matrices: 2D arrays of numbers

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

Array Indexing and Slicing

```
import numpy as np

# 1D array indexing
arr = np.array([10, 20, 30, 40, 50])
print(arr[0])         # 10 (first element)
print(arr[-1])        # 50 (last element)
print(arr[1:4])       # [20, 30, 40] (slice)

# 2D array indexing
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(matrix[0, 1])       # 2 (row 0, column 1)
print(matrix[:, 0])       # [1, 4, 7] (first column)
print(matrix[1, :])       # [4, 5, 6] (second row)

# Boolean indexing
mask = arr > 25
print(arr[mask])          # [30, 40, 50] (elements > 25)
```

Broadcasting in NumPy

Broadcasting: Performing operations on arrays of different shapes

```
import numpy as np

# Scalar with array
arr = np.array([1, 2, 3, 4])
result = arr + 10          # [11, 12, 13, 14]

# Different shaped arrays
matrix = np.array([[1, 2, 3], [4, 5, 6]])
vector = np.array([10, 20, 30])
result = matrix + vector  # Each row of matrix + vector

# Visual representation of broadcasting
print("Matrix shape:", matrix.shape)    # (2, 3)
print("Vector shape:", vector.shape)     # (3,)
print("Result shape:", result.shape)     # (2, 3)
```

Visualizing Mathematical Objects

```
import matplotlib.pyplot as plt
import numpy as np

# Visualize vectors as arrows
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# Vector visualization
vectors = np.array([[3, 2], [1, 4], [2, -1]])
colors = ['red', 'blue', 'green']

for i, (v, color) in enumerate(zip(vectors, colors)):
    ax1.quiver(0, 0, v[0], v[1], angles='xy', scale_units='xy',
               scale=1, color=color, width=0.005)
    ax1.text(v[0]+0.1, v[1]+0.1, f'v{i+1}', fontsize=12)

ax1.set_xlim(-1, 4)
ax1.set_ylim(-2, 5)
ax1.grid(True)
ax1.set_title('Vector Visualization')
```

Matrix Visualization

```
# Matrix as heatmap
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

plt.figure(figsize=(8, 6))
plt.imshow(matrix, cmap='viridis')
plt.colorbar()
plt.title('Matrix Heatmap')

# Add text annotations
for i in range(matrix.shape[0]):
    for j in range(matrix.shape[1]):
        plt.text(j, i, matrix[i, j], ha="center", va="center",
                 color="white", fontsize=14)

plt.show()
```

Applications in ML:

- **Feature vectors:** Each row represents a data sample

Data Distribution Visualization

```
import matplotlib.pyplot as plt
import numpy as np

# Generate sample data
np.random.seed(42)
data = np.random.normal(50, 15, 1000)  # Normal distribution

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# Histogram
ax1.hist(data, bins=30, alpha=0.7, color='skyblue', edgecolor='black')
ax1.set_title('Histogram of Sample Data')
ax1.set_xlabel('Value')
ax1.set_ylabel('Frequency')

# Box plot
ax2.boxplot(data)
ax2.set_title('Box Plot of Sample Data')
ax2.set_ylabel('Value')

plt.tight_layout()
plt.show()
```

NumPy for Machine Learning

Common ML Tasks with NumPy:

```
import numpy as np

# Create sample dataset (features and labels)
X = np.random.randn(100, 3)  # 100 samples, 3 features
y = np.random.randint(0, 2, 100)  # Binary labels

# Data preprocessing
X_normalized = (X - X.mean(axis=0)) / X.std(axis=0)  # Standardization
X_minmax = (X - X.min()) / (X.max() - X.min())      # Min-max scaling

# Basic statistics
print(f"Mean of each feature: {X.mean(axis=0)}")
print(f"Standard deviation: {X.std(axis=0)}")
print(f"Correlation matrix:\n{np.corrcoef(X.T)}")

# Train-test split
indices = np.random.permutation(len(X))
train_size = int(0.8 * len(X))
```

Thank You!

Next: Fundamental Algorithms