# BSDS-202 : Advanced Python

## Course Instructor - Mr. Nitish Patil

Gauri Sharan - BSc Data Science, Semester 4

June 11, 2024

# Table of contents

1. OOP
2. Web Scraping
3. API Creation
4. Tensorflow & Keras
5. Pytorch
6. References
7. Thank You

# Object-oriented Programming (OOP) Concepts

OOP is a programming paradigm based on the concept of "objects", which can contain data and code to manipulate that data. Key concepts include:

- **Class**: A blueprint for creating objects.
- **Objects**: Instances of classes.
- **Inheritance**: Mechanism by which one class can inherit attributes and methods from another class.
- **Encapsulation**: Hiding the internal state of an object and requiring all interaction to be performed through an object's methods.
- **Polymorphism**: The ability to present the same interface for different underlying data types.

# Python code example: Class and Objects

```python
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        return f"{self.name} is barking."

my_dog = Dog("Buddy", 3)
print(my_dog.bark())
```

# Advanced Python Syntax and Concepts

Advanced features of Python include:

- **Iterators**: Objects that can be iterated upon.
- **Generators**: Functions that return an iterable set of items, one at a time, in a special way.
- **Decorators**: Functions that modify the behavior of another function.
- **Context Managers**: Allow you to allocate and release resources precisely when you want to.

# Python code example: Generators

```python
def countdown(n):
    while n > 0:
        yield n
        n -= 1

for count in countdown(5):
    print(count)
```

# Functions & Recursive Function

Functions are blocks of code that only run when called. Recursive functions are functions that call themselves.

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)

print(factorial(5))
```

# Web Scraping Libraries and Functionalities

Web scraping involves fetching and extracting data from websites. Python libraries used for web scraping include:

- **Beautiful Soup**: Parses HTML and XML documents.
- **Requests**: Sends HTTP requests.
- **Scrapy**: An open-source web-crawling framework.

# Python code example: Web Scraping with Beautiful Soup

```python
import requests
from bs4 import BeautifulSoup

url = 'https://example.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

for heading in soup.find_all('h2'):
    print(heading.text)
```

# Overview of Python Web Frameworks

Python web frameworks like Flask and Django help in building web applications quickly and efficiently. Flask is a lightweight WSGI web application framework.

# Python code example: Creating API with Flask

```python
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api', methods=['GET'])
def api():
    return jsonify({"message": "Hello, World!"})

if __name__ == '__main__':
    app.run(debug=True)
```

# Creating RESTful Web APIs

RESTful APIs are based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development.

# Python code example: Display Text with Streamlit

```python
import streamlit as st

st.write("Hello, Streamlit!")
```

# Python code example: Display Image with Streamlit

```python
import streamlit as st

st.image("path/to/image.jpg", caption="Sample Image")
```

# Tensorflow Basics

TensorFlow is an open-source framework for machine learning and deep learning. Key concepts include:

- **Tensors**: Multi-dimensional arrays.
- **Graphs**: Define the computation.
- **Sessions**: Execute the graph.

# Python code example: TensorFlow Basics

```python
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)

with tf.Session() as sess:
    print(sess.run(a + b))
```

# Python code example: Linear Regression with TensorFlow

```python
import tensorflow as tf

# Define the model
class LinearModel:
    def __init__(self):
        self.W = tf.Variable(tf.random.normal([1]), name='weight')
        self.b = tf.Variable(tf.random.normal([1]), name='bias')

    def __call__(self, x):
        return self.W * x + self.b

# Create the model
model = LinearModel()
# Define a loss function
def loss(predicted_y, target_y):
    return tf.reduce_mean(tf.square(predicted_y - target_y))
#continued in next slide
```

# Python code example: Linear Regression with TensorFlow

```python
# Training data
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]

# Define a training loop
def train(model, x, y, learning_rate):
    with tf.GradientTape() as t:
        current_loss = loss(model(x), y)
    dW, db = t.gradient(current_loss, [model.W, model.b])
    model.W.assign_sub(learning_rate * dW)
    model.b.assign_sub(learning_rate * db)

# Training the model
epochs = 100
for epoch in range(epochs):
    train(model, x_train, y_train, learning_rate=0.1)
    current_loss = loss(model(x_train), y_train)
    print(f"Epoch {epoch}: Loss: {current_loss.numpy()}")
```

# Python code example: Keras Layers

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([
    Dense(32, activation='relu', input_shape=(784,)),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

# TensorFlow & Keras Applications

Applications of TensorFlow and Keras include image recognition, natural language processing, and predictive analytics.

# Python code example: MNIST with Keras

```python
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical

# Load dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Preprocess the data
x_train = x_train.reshape((60000, 28 * 28)).astype('float32') / 255
x_test = x_test.reshape((10000, 28 * 28)).astype('float32') / 255
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Build the model
model = Sequential([
    Flatten(input_shape=(28*28,)),
    Dense(512, activation='relu'),
    Dense(10, activation='softmax')
])
```

# Python code example: MNIST with Keras

```
# Compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=128)

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

# PyTorch Basics

PyTorch is an open-source machine learning library based on the Torch library. Key concepts include tensors, automatic differentiation, and dynamic computation graphs.

# Python code example: Linear Regression with PyTorch

```
import torch
import torch.nn as nn
import torch.optim as optim

# Data
x_train = torch.tensor([[1.0], [2.0], [3.0], [4.0]])
y_train = torch.tensor([[0.0], [-1.0], [-2.0], [-3.0]])

# Model
class LinearModel(nn.Module):
    def __init__(self):
        super(LinearModel, self).__init__()
        self.linear = nn.Linear(1, 1)

    def forward(self, x):
        return self.linear(x)

model = LinearModel()
```

# Python code example: Linear Regression with PyTorch

```
# Loss and optimizer
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training loop
for epoch in range(100):
    model.train()
    optimizer.zero_grad()
    outputs = model(x_train)
    loss = criterion(outputs, y_train)
    loss.backward()
    optimizer.step()

    print(f'Epoch [{epoch+1}/100], Loss: {loss.item():.4f}')
```

# PyTorch Applications

PyTorch is used in various applications, including -

- Computer vision
- Natural language processing
- Reinforcement learning

# References

- Luciano Ramalho. *Fluent Python*. O'Reilly Media, 2015.
- David Beazley and Brian K. Jones. *Python Cookbook*. O'Reilly Media, 2013.
- Wes McKinney. *Python for Data Analysis*. O'Reilly Media, 2017.
- Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning*. Packt Publishing, 2019.
- *Python Documentation*. Available at: `https://docs.python.org/3/`
- *Python Programming Tutorials*. Available at: `https://www.w3schools.com/python/`
- *Real Python*. Available at: `https://realpython.com/`
- *Python Weekly*. Available at: `https://www.pythonweekly.com/`

# Thank You

Hope you liked this presentation.

Gauri Sharan
Student, School of Data Science
AAFT Noida (Shobhit University)
BSc Data Science 2022-25
Semester 4, 2024

- LinkedIn: **linkedin.com/in/gauri-sharan**
- GitHub: **github.com/gaurisharan**
- Mail: **gaurisharan123@gmail.com**