Python Programming Lab Module

Module Outline

Week 1:
https://colab.research.google.com/drive/1nGN6v-a_SXRYKlQlAzonCDFMQ-_7G7Eo?usp=sharing

Introduction to Python

- Overview
- Installation
- Basic Syntax

Data Types and Variables

- Numbers
- Strings
- Lists
- Tuples
- Sets
- Dictionaries

Week 2:

https://colab.research.google.com/drive/1f5RPFtDeGwpNMz32fOAbdjRhFSx9nweU?usp=sharing

Control Flow

- If Statements
- For Loops
- While Loops

Week 3:

https://colab.research.google.com/drive/1LcuDUrIoFAFatGLD5O0mCk_Fx48OOT-x?usp=sharing

Functions

- Defining Functions
- Function Arguments
- Lambda Functions

Week 4:

https://colab.research.google.com/drive/18llMvktMz2406pbqr4cwn_-De0PW4oBV?usp=sharing

Modules and Packages

- Importing Modules
- Standard Library
- Creating and Using Packages

File Handling

- Reading Files
- Writing Files
- Working with CSV and JSON

Working with Libraries

- NumPy
- Pandas
- Matplotlib

Week 5:
https://colab.research.google.com/drive/1xmFuGTKVdwt3nYbwp2USC4Z5P-k6bRfu?usp=sharing
Introduction to Machine Learning with Python using Image data

- Overview of Machine Learning
- Installation and Setup

Visualization

- Data Visualization with Matplotlib and Seaborn

Week 6:
https://colab.research.google.com/drive/1nw_7uhAD8rnqozgc6Nwt8gYdAFzGcVhM?usp=sharing

Image Transfer Learning and Model Deployment

- Transfer Learning Concepts
- Using Pre-trained Models
- Model Deployment with Flask

Fine-Tuning Models for Images

- Fine-Tuning Techniques
- Case Study: Fine-Tuning a Pre-trained Model

Week 7:
https://colab.research.google.com/drive/15wCo1OT0KMq1g7jAd503OSvWBY_DE6X2?usp=sharing

Audio Classification

- Overview of Audio Classification
- Preprocessing Audio Data
- Feature Extraction
- Building and Training a Neural Network for Audio Classification
- Evaluating the Model
- Making Predictions

Week 8:
https://colab.research.google.com/drive/1nE5oG_JjulvkrO01ltz5Nba3MdgYhLLS?usp=sharing

Audio Processing and Translator Generator

- Audio Data Handling
- Building a Speech-to-Text Model
- Building a Language Translator Model

Week 9:
https://colab.research.google.com/drive/1p4x9mvY9NAOdVBz8mQPbh9JEv43pH4TL?usp=sharing

Applied CNN: Object Detection and YOLO

- Introduction to Convolutional Neural Networks (CNN)
- Object Detection with YOLO

Week 10:
https://colab.research.google.com/drive/1e5LBt1yeiuM85jjw7QiT_pa8qrGSkoGq?usp=sharing

Working with Text and Music Generation

- Text Generation using RNNs and LSTMs
- Music Generation using AI

Week 1:
https://colab.research.google.com/drive/1nGN6v-a_SXRYKlQlAzonCDFMQ-_7G7Eo?usp=sharing

In the first week, students will get an overview of Python, a high-level programming language known for its readability and versatility. The session starts with instructions on how to install Python on different operating systems. The basics of Python syntax will be covered, including how to write and execute a simple "Hello, World!" program. Students will explore Python's fundamental data types and variables, including numbers (integers, floats, and complex numbers), strings, lists, tuples, sets, and dictionaries. Each concept will be illustrated with practical examples to ensure a solid foundation.

Introduction to Python

Overview

Python is a high-level, interpreted programming language known for its readability and versatility. It is widely used for web development, data analysis, artificial intelligence, scientific computing, and more.

Installation

To install Python, download it from python.org and follow the installation instructions for your operating system.

Basic Syntax

```
# Hello World Program
print("Hello, World!")
```

Data Types and Variables

Numbers

Python supports integers, floats, and complex numbers.

```
x = 5       # int
y = 3.14    # float
z = 1 + 2j  # complex
```

Strings

Strings are sequences of characters.

```
name = "Alice"
print(name)
```

Lists

Lists are ordered collections of items.

```
fruits = ["apple", "banana", "cherry"]
print(fruits)
```

Tuples

Tuples are immutable collections of items.

```
coordinates = (10, 20)
print(coordinates)
```

Sets

Sets are unordered collections of unique items.

```
numbers = {1, 2, 3, 4, 5}
print(numbers)
```

Dictionaries

Dictionaries are collections of key-value pairs.

```
person = {"name": "Alice", "age": 25}
print(person)
```

Week 2:

https://colab.research.google.com/drive/1f5RPFtDeGwpNMz32fOAbdjRhFSx9nweU?usp=sharing

Week 2 delves into control flow in Python, teaching students how to direct the execution of their programs. The module covers if statements for decision-making processes and demonstrates how to use for loops to iterate over sequences and while loops for repeated execution as long as a condition is true. Practical exercises will help students understand and apply these control structures effectively in their programs.

Control Flow

If Statements

```
age = 18
if age >= 18:
    print("You are an adult.")
else:
    print("You are a minor.")
```

For Loops

```
for i in range(5):
    print(i)
```

While Loops

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Week 3:

https://colab.research.google.com/drive/1LcuDUrloFAFatGLD5O0mCk_Fx48OOT-x?usp=sharing

The focus of Week 3 is on defining and using functions in Python. Students will learn how to create functions using the def keyword, pass arguments to functions, and return values. The concept of lambda functions, which allows the creation of small, anonymous functions, will also be introduced. Hands-on coding tasks will reinforce these concepts, enabling students to write modular and reusable code.

Functions

Defining Functions

```
def greet(name):
    print(f"Hello, {name}!")

greet("Alice")
```

Function Arguments

```
def add(a, b):
    return a + b

result = add(2, 3)
print(result)
```

Lambda Functions

```
square = lambda x: x ** 2
print(square(4))
```

Week 4:

https://colab.research.google.com/drive/18lIMvktMz2406pbqr4cwn_-De0PW4oBV?usp=sharing

In Week 4, students will explore modules and packages to understand how to organize and reuse code efficiently. They will learn to import modules, utilize Python's standard library, and create their own packages. The session will also cover file handling, including reading from and writing to files, and working with CSV and JSON data formats. Additionally, students will be introduced to essential libraries such as NumPy for numerical operations, Pandas for data manipulation, and Matplotlib for data visualization.

Modules and Packages

Importing Modules

```
import math
print(math.sqrt(16))
```

Standard Library

Python comes with a rich standard library. For example, the `datetime` module provides classes for manipulating dates and times.

```
from datetime import date
today = date.today()
print(today)
```

Creating and Using Packages

A package is a way of organizing related modules.

```
my_package/
    __init__.py
    module1.py
    module2.py

# module1.py
def hello():
    print("Hello from module1")

# In another script
```

```
from my_package import module1
module1.hello()
```

6. File Handling

Reading Files

```
with open("file.txt", "r") as file:
    content = file.read()
    print(content)
```

Writing Files

```
with open("file.txt", "w") as file:
    file.write("Hello, World!")
```

Working with CSV and JSON

```
import csv
import json

# CSV
with open("data.csv", "r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)

# JSON
data = '{"name": "Alice", "age": 25}'
parsed_data = json.loads(data)
print(parsed_data)
```

6. Working with Libraries

NumPy

```
import numpy as np

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

Pandas

```
import pandas as pd

data = {
    "name": ["Alice", "Bob"],
    "age": [25, 22]
}

df = pd.DataFrame(data)
print(df)
```

Matplotlib

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x, y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Line Plot')
plt.show()
```

Week 5:
https://colab.research.google.com/drive/1xmFuGTKVdwt3nYbwp2USC4Z5P-k6bRfu?usp=sharing

Week 5 introduces students to the basics of machine learning with Python, particularly using image data. They will gain an overview of machine learning concepts and the installation and setup of necessary libraries. Data visualization techniques will be covered using Matplotlib and Seaborn to help students understand and visualize datasets. The practical component will involve visualizing datasets and preparing them for machine learning tasks.

Introduction to Machine Learning with Python

Overview of Machine Learning

Machine learning is a subset of artificial intelligence that enables systems to learn and improve from experience without being explicitly programmed. It involves algorithms that can identify patterns and make decisions with minimal human intervention.

Installation and Setup

Ensure you have Python installed. Use `pip` to install essential libraries.

```
pip install numpy pandas matplotlib seaborn scikit-learn tensorflow
keras flask
```

Data Visualization with Matplotlib and Seaborn

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(data)
plt.show()
```

Week 6:
https://colab.research.google.com/drive/1nw_7uhAD8rnqozgc6Nwt8gYdAFzGcVhM?usp=sharing

This week focuses on transfer learning and model deployment. Students will learn about the principles of transfer learning and how to use pre-trained models for new tasks. They will also gain practical experience in deploying models using Flask, a lightweight web framework. The week includes a case study on fine-tuning a pre-trained model, where students will apply fine-tuning techniques to improve the performance of their models on specific tasks.

Transfer Learning and Model Deployment

Transfer Learning Concepts

Transfer learning involves using a pre-trained model on a new problem, leveraging the knowledge the model has gained from the original task.

Using Pre-trained Models

```
from tensorflow.keras.applications import VGG16

base_model = VGG16(weights='imagenet', include_top=False)
```

Model Deployment with Flask

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict(data)
    return jsonify(prediction.tolist())

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

Fine-Tuning Models

Fine-Tuning Techniques

Fine-tuning involves unfreezing a few top layers of the frozen model base used for feature extraction and jointly training both the newly added part and these top layers.

Fine-Tuning Models

Fine-Tuning Techniques

Fine-tuning involves unfreezing a few top layers of the frozen model base used for feature extraction and jointly training both the newly added part and these top layers.

Case Study: Fine-Tuning a Pre-trained Model

```
from tensorflow.keras.applications import VGG16

from tensorflow.keras import layers, models, optimizers


base_model = VGG16(weights='imagenet', include_top=False)

for layer in base_model.layers[:-4]:

    layer.trainable = False


model = models.Sequential([

    base_model,

    layers.Flatten(),

    layers.Dense(256, activation='relu'),

    layers.Dropout(0.5),

    layers.Dense(1, activation='sigmoid')

])


model.compile(optimizer=optimizers.Adam(lr=0.0001)
```

Week 7:
https://colab.research.google.com/drive/15wCo1OT0KMq1g7jAd503OSvWBY_DE6X2?usp=sharing

Week 7 is dedicated to audio classification, where students will learn to preprocess audio data and extract features for building machine learning models. They will build and train a neural network for audio classification tasks, evaluate the model's performance, and make predictions using new audio data. This hands-on approach will help students understand the unique challenges and techniques involved in working with audio data.

Overview of Audio Classification

Audio classification involves categorizing audio signals into predefined classes. This can be applied to tasks such as speech recognition, music genre classification, and environmental sound classification.

Preprocessing Audio Data

```
import librosa

audio_data, sr = librosa.load('audio_file.wav')
print(f'Sample rate: {sr}, Audio data shape: {audio_data.shape}')
```

Building the Model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,
MaxPooling2D

model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
input_shape=(X_train.shape[1], 1, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(np.unique(y_encoded)), activation='softmax')
])
```

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

Training the Model

```
history = model.fit(X_train, y_train, epochs=30, batch_size=32,
validation_data=(X_test, y_test))
```

Evaluating the Model

```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc}")
```

Making Predictions

```
def predict(file_path, model, le):
    features = extract_features(file_path)
    features = features[..., np.newaxis]
    features = np.expand_dims(features, axis=0)
    prediction = model.predict(features)
    predicted_label = le.inverse_transform([np.argmax(prediction)])
    return predicted_label

sample_audio_file = 'path_to_sample_audio.wav'
print(predict(sample_audio_file, model, le))
```

Week 8:
https://colab.research.google.com/drive/1nE5oG_JjulvkrO01Itz5Nba3MdgYhLLS?usp=sharing

In Week 8, students will delve into audio processing and build models for speech-to-text and language translation. They will handle audio data using libraries like Librosa and implement a speech-to-text model using speech recognition techniques. Additionally, they will build a language translator model using transformer-based architectures, gaining insight into advanced natural language processing tasks.

Audio Processing and Translator Generator

Audio Data Handling

```
import librosa

audio_data, sr = librosa.load('audio_file.wav')
```

Building a Speech-to-Text Model

```
import speech_recognition as sr

recognizer = sr.Recognizer()
with sr.AudioFile('audio_file.wav') as source:
    audio = recognizer.record(source)
    text = recognizer.recognize_google(audio)
    print(text)
```

Building a Language Translator Model

```
from transformers import MarianMTModel, MarianTokenizer

model_name = 'Helsinki-NLP/opus-mt-en-de'
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name)

text = "Hello, how are you?"
translated =
model.generate(**tokenizer.prepare_seq2seq_batch([text],
return_tensors="pt"))
print([tokenizer.decode(t, skip_special_tokens=True) for t in
translated])
```

Week 9:
https://colab.research.google.com/drive/1p4x9mvY9NAOdVBz8mQPbh9JEv43pH4TL?usp=sharing

Week 9 introduces Convolutional Neural Networks (CNNs) and their application in object detection using the YOLO (You Only Look Once) algorithm. Students will learn the fundamentals of CNNs and how they are used for image processing tasks. They will implement the YOLO algorithm to detect objects in images, understanding the process of object detection from data preparation to model inference.

Applied CNN: Object Detection and YOLO

Introduction to Convolutional Neural Networks (CNN)

CNNs are deep learning algorithms that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and differentiate one from the other.

Object Detection with YOLO

```
import cv2

net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in
net.getUnconnectedOutLayers()]

image = cv2.imread('image.jpg')
height, width, channels = image.shape
blob = cv2.dnn.blobFromImage(image, 0.00392, (416, 416), (0, 0, 0),
True, crop=False)
net.setInput(blob)
outs = net.forward(output_layers)

for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
```

```
            w = int(detection[2] * width)
            h = int(detection[3] * height)

            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255,
0), 2)

cv2.imshow('Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Week 10:
https://colab.research.google.com/drive/1e5LBt1yeiuM85jjw7QiT_pa8qrGSkoGq?usp=sharing

The final week focuses on generating text and music using AI models. Students will learn to build text generation models using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. They will also explore music generation techniques, applying AI to create new musical compositions. This week's activities will demonstrate the creative potential of machine learning and provide students with a comprehensive understanding of sequence modeling.

Working with Text and Music Generation

Text Generation using RNNs and LSTMs

```
from keras.models import Sequential
from keras.layers import LSTM, Dense
import numpy as np

data = "your text data"
chars = sorted(list(set(data)))
char_indices = dict((c, i) for i, c in enumerate(chars))
indices_char = dict((i, c) for i, c in enumerate(chars))

maxlen = 40
step = 3
sentences = []
next_chars = []
for i in range(0, len(data) - maxlen, step):
    sentences.append(data[i: i + maxlen])
    next_chars.append(data[i + maxlen])
X = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        X[i, t, char_indices[char]] = 1
    y[i, char_indices[next_chars[i]]] = 1

model = Sequential()
model.add(LSTM(128, input_shape=(maxlen, len(chars))))
model.add(Dense(len(chars), activation='softmax'))
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.fit(X, y, batch_size=128, epochs=20)

def sample(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)

start_index = np.random.randint(0, len(data) - maxlen - 1)
generated
```