Syllabus for Becoming an AI Developer
Format: Topic-based, progress at your own pace
Objective: Master the foundational, technical, and deployment skills necessary to build AI solutions.

Module 1: Programming Foundations for AI

Topics:

- Python Fundamentals: Data types, loops, functions
- Introduction to Libraries: NumPy, pandas, Matplotlib
- Basic Data Structures: Lists, dictionaries, arrays
- Coding Practices: Debugging, version control (Git/GitHub)

Assignments:

Assignment 1: Python Fundamentals

1. Write a Python program to:
   - Create and manipulate lists and dictionaries.
   - Write functions to perform basic arithmetic and string manipulations.
2. Implement loops to process a dataset (e.g., calculate the average of a list of numbers).

Assignment 2: Libraries and Data Visualization

1. Using NumPy and pandas:
   - Create and manipulate arrays and DataFrames.
   - Compute basic statistics like mean, median, and standard deviation.
2. Using Matplotlib:
   - Create a bar chart and scatter plot based on a small dataset (e.g., sales data for 5 products).

Bonus: Push your assignments to a GitHub repository and document the steps.

Resources:

- Python Crash Course
- GitHub tutorials

Module 2: Data Handling and Preprocessing

Topics:

- Understanding Data: Structured vs. Unstructured Data
- Data Cleaning: Handling missing values, duplicates
- Feature Engineering: Scaling, encoding, feature selection
- Exploratory Data Analysis (EDA): Insights through visualizations

Tools:

- pandas, NumPy, Seaborn, Scikit-learn

Assignments:

Assignment 1: Data Cleaning

1. Select a public dataset (e.g., Kaggle Titanic dataset).
2. Perform data cleaning:
   - Handle missing values (e.g., fill or drop).
   - Remove duplicate rows.

Assignment 2: Exploratory Data Analysis (EDA)

1. Use pandas and Seaborn to:
   - Visualize the distribution of numerical variables.
   - Analyze relationships using pair plots and correlation heatmaps.
2. Document your findings with markdown or comments.

Module 3: Fundamentals of Machine Learning

Topics:

- Supervised Learning: Regression and classification
- Unsupervised Learning: Clustering techniques
- Model Evaluation Metrics: Accuracy, precision, recall, F1-score
- Overfitting and Underfitting: Regularization techniques

Tools:

- Scikit-learn

Assignments:

Assignment 1: Supervised Learning

1. Build a regression model to predict house prices using the Boston Housing dataset.
2. Evaluate the model using metrics like RMSE, MAE, and R^2.

Assignment 2: Unsupervised Learning

1. Implement k-Means clustering to group data from a small dataset (e.g., customer purchase patterns).
2. Visualize the clusters using matplotlib or Seaborn.

Module 4: Deep Learning Basics

Topics:

- Neural Networks: Architecture, activation functions
- Gradient Descent and Backpropagation
- Building Deep Learning Models with Keras and TensorFlow
- Introduction to CNNs and RNNs

Assignments:

Assignment 1: Neural Networks

1. Using TensorFlow or Keras:
    - Build a feedforward neural network to classify images from the MNIST dataset.
    - Experiment with different activation functions and document their impact on performance.

Assignment 2: Sequential Data Analysis

1. Build an RNN to perform sentiment analysis on a text dataset (e.g., IMDB reviews).

Resources:

- TensorFlow Documentation
- Keras Tutorials

Module 5: Natural Language Processing (NLP)

Topics:

- Text Preprocessing: Tokenization, stemming, stop words
- Sentiment Analysis: Using pre-trained models
- Introduction to Transformer Models: BERT, GPT
- Text Generation and Summarization
- **Vector Representations and Vector Databases:** Semantic embeddings, similarity search

Tools:

- NLTK, spaCy, Hugging Face Transformers, Pinecone, Milvus

Assignments:

Assignment 1: Text Preprocessing

1. Write a Python script using NLTK or spaCy to:
    - Tokenize text.
    - Remove stop words.
    - Perform stemming or lemmatization.

Assignment 2: Fine-Tuning Transformers

1. Fine-tune a pre-trained BERT model using the Hugging Face library for text classification (e.g., spam vs. non-spam emails).

Module 6: Computer Vision

Topics:

- Image Preprocessing: Filters, transformations
- Convolutional Neural Networks (CNNs): Feature extraction, pooling layers
- Object Detection and Segmentation
- Generative Models: GANs (Generative Adversarial Networks)
- **Vector Databases for Visual Search:** Embedding storage and retrieval

Tools:

- OpenCV, TensorFlow, PyTorch, Pinecone

Assignments:

Assignment 1: Image Classification

1. Train a CNN using TensorFlow or PyTorch to classify images from the CIFAR-10 dataset.

Assignment 2: Visual Search System

1. Create a visual search system by:
    - Extracting embeddings using a pre-trained model.
    - Storing embeddings in a vector database (e.g., Pinecone).
    - Querying the database with an image and retrieving similar images.

Module 7: Model Optimization and Deployment

Topics:

- Hyperparameter Tuning: Grid search, random search
- Model Saving and Loading: Checkpoints, serialization
- Deployment Frameworks: Flask/Django, FastAPI
- Deployment Tools: Docker, AWS, Google Cloud
- **Vector Database Deployment:** Scaling and querying embeddings in production

Assignments:

Assignment 1: Hyperparameter Tuning

1. Use grid search or random search to optimize the hyperparameters of a deep learning model.

Assignment 2: Model Deployment

1. Deploy a trained model as a web app using Flask or FastAPI.
2. Containerize the app with Docker for deployment.

Resources:

- Flask Documentation
- Docker Tutorials

Module 8: Advanced Topics and Capstone Project

Topics:

- Reinforcement Learning: Q-learning, Deep Q Networks
- Ethics in AI: Fairness, transparency, accountability
- Capstone Project: Solve a real-world problem with AI (e.g., predicting stock prices, building a recommendation system)

Deliverables:

- A fully documented AI project with code, results, and deployment, optionally leveraging vector databases for advanced functionality

Capstone Project

1. Choose a real-world problem to solve using AI (e.g., fraud detection, recommendation system).
2. Deliverables:
    - A Python program implementing your solution.
    - Visualizations and insights from your analysis.
    - Documentation detailing your approach, challenges, and results.
    - Optional: Integrate a vector database for advanced functionality (e.g., semantic search).

Resources:

- Research papers and case studies
- Online platforms like Kaggle or AI Challenges