Hands-On GenAI: LLMs, RAGs, and Agentic Systems for Beginners

Day 2

Adya Bhat and Tejas Venugopalan

Agenda

Addressing feedback

Building programming intuition

A quick revision of Python

Backend Development

Addressing Feedback

Difficulty level

More practice exercises

Tips

Machine learning concepts

Overall

Voice volume

Building Programming Intuition- How to start coding?

1. understand the problem: **read** thoroughly, read multiple times
2. simplify it: break the problem down to **simpler problems**/ simpler aspects
3. code it: **code it in parts**, for each of the simpler problems
4. e.g. coding exercises from day 1, etc.

For harder problems/ complex problems:

1. decide the level of **abstractions** in the problem: what **functions, classes, files** to have
2. e.g. designing a program for managing a bookstore, etc.

Good Programming Habits

1. writing **readable code**- choosing relevant variable names, function names, file names
    a. some naming ‘conventions’: i (temporary variable for counter), key (temporary variable for dictionary key), value (temporary variable for dictionary value)
    b. don’t use keywords as variable names
2. **ordering** and **grouping** related import statements, function definitions
3. writing crisp, comprehensive **comments** describing functionality of code snippets
4. **testing code** in parts (as you write)- by printing intermediate variable values, function outputs, etc.
5. separating logical code parts by **Enter**, and comments, following proper indentation (not only in python- improves readability)

Python- Data Structures, Conditions, Loops

Some quick questions!

- which data structure can be used to represent:
  a. a matrix of numbers?
  b. location coordinates for a place?
  c. a unique collection of car names?
  d. a store's stock inventory (item name, item price, item quantity, other item details)?

Python- Data Structures, Conditions, Loops

Answers:

- which data structure can be used to represent:
  a. a matrix of numbers? a list of lists
  b. location coordinates for a place? a tuple for each location
  c. a unique collection of car names? a set (unique)
  d. a store's stock inventory (item name, item price, item quantity, other item details)? a dictionary that looks like {item_name: (item_price, item_quantity, item_colour)}

Python- Functions

- reusable block of code
- optionally take parameters, optionally return values
- functions are **defined**, and then **called**
- function definition:

```
def find_difference(num1, num2):

    return num1 - num2
```

- function calling:

```
difference = find_difference(2, 3)
```

Let’s look at some code examples!

And discuss solutions to Assignment 1

Python- Modules, Packages, Libraries

Module: single file, containing reusable code, can use for built-in functions. E.g. math, random

Package: directory of modules. E.g. numpy

Library: set of modules and packages. E.g. NLTK

Python- Errors

Errors in Python:

- Syntax Errors

```
a = int(input())
if a < 5
  print("a is less than 5")
```

```
  File "/tmp/ipython-input-560118079.py", line 2
    if a < 5
            ^
SyntaxError: expected ':'
```

Errors are described in detail, source of error is pointed out. So read the errors carefully!

Python- Errors

Errors in Python:

- Exceptions

Errors are described in detail, source of error is pointed out. So read the errors carefully!

```
def is_divisible(num1, num2):
  if num1 % num2 == 0:
    return True
  return False

is_divisible(7, 0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
/tmp/ipython-input-1966518779.py in <cell line: 0>()
      4   return False
      5
----> 6 is_divisible(7, 0)

/tmp/ipython-input-1966518779.py in is_divisible(num1, num2)
      1 def is_divisible(num1, num2):
----> 2   if num1 % num2 == 0:
      3     return True
      4   return False
      5

ZeroDivisionError: integer modulo by zero
```

Python- Handling Exceptions

Try- except- finally block

- try block: to test a code that might raise an exception
- except block: to handle the exception. When the particular exception is raised, the code statements in the except block for the particular/ general exception. A program can have multiple except blocks, one for each exception.
- finally block: executes statements in the block irrespective of exceptions raised.

Practising Coding- Some tips

Easy:

1. Solving **textbook, question bank exercises** from pesuacademy
2. Exploring examples and questions from **online platforms** like W3Schools (https://www.w3schools.com/python/)
3. Implementing **built-in functions** (max, len, ... to string functions ... to matrix mult using 2d lists, etc.)
4. Can refer to the course **Introduction to Computer Science and Programming in Python on MIT OpenCourseWare**: https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016: for slides, code, course text, **assignments, mini-projects**

Practising Coding- Some tips

Medium

1. Easy questions on **leetcode, hackerrank** (some questions may require knowledge of data structures, algorithms)
2. Implementing **‘worlds-of-interest’** in python (music logic, games, math/ physics function/ equation solvers etc.). E.g.
    a. music raga identification, raga search
    b. games: hangman, wordle, other nyt/ linkedin games
    c. implementing a bookstore management system, etc. (can extend using classes)

INTRODUCTION TO BACKEND

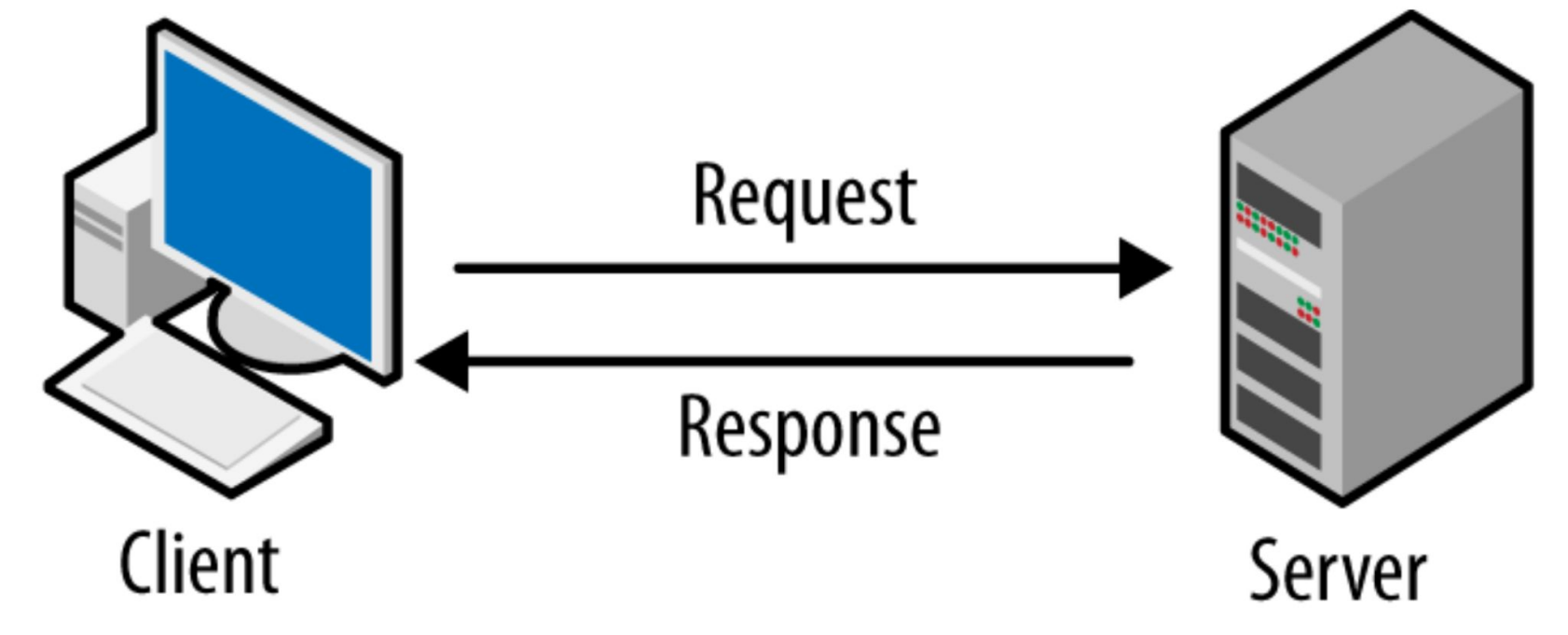
Application Programming Interface (API)

It’s basically a set of rules and definitions that allow two software systems to talk to each other.

Types:

- Web APIs: Used for communication over the internet using HTTP requests.
- Library APIs: Functions exposed by a programming library.
- OS APIs: Allow apps to interact with the operating system.

Client Server Architecture



INTRODUCTION TO FLASK

Flask

Flask is a lightweight Python web framework used to build web applications and REST APIs.

It’s called a microframework because it gives core functionality but leaves architecture choices to developers.

Features:

Minimal & flexible

Built-in development server & debugger

Supports extensions (ORMs, Auth, etc.)

RESTful request handling

Easy integration with databases

Conclusion

What we have covered today:

1. building programming intuition
2. built-in data structures, conditions, loops
3. functions
4. modules, packages, libraries
5. practising coding
6. Backend Development

Thank You!