The objective of these tasks are NOT to see what you know but to see how you solve/ write code for a completely unknown problem given to you, how you use different packages and documentation, and your own data structure skillset. That is why we have provided sources to help you. And you can take help from the web to solve the problem.

TASK 1:

1. Write a script test_server.py. To run nohup python test_server.py . The process will run on port 8080 or port 5000 and localhost. This server script will have an HTTP POST endpoint "/sendval/". It will receive requests and from the request body, it will fetch values of some key "val".

   The demo HTTP POST request body looks like this:

   {

   "val": 20,

   "text": "some free text in English"

   }

   It receives this value from the request and every time it gets a request, it inserts a node in a BST with the value and nouns of the text received from the request. Once a new node is inserted in the BST, it saves the BST in the disk (in some file path).

   So, a single node from the BST has the following attributes:

   key = which is the value e.g. 20

   nouns_list = ["English"], the word_tokens of the text which are of POS tag Noun or NP (Noun Phrase)

   Notes: You may consider the BST to be a class, Node to be a class, and insert a node in a BST to be a function of the BST class. Then you will be able to save the BST instance that you are building using pickle or joblib library. You may design the approach in this way.

   Source: https://www.tutorialspoint.com/python-pickling

   For Nouns: http://www.nltk.org/book/ch05.html

   You may use Flask/Fastapi for writing the API endpoint. Remember that the endpoint should be able to handle asynchronous requests.

   Source: https://flask.palletsprojects.com/en/1.1.x/quickstart/#a-minimal-application

   https://fastapi.tiangolo.com/

2. Write a script test_client.py to send HTTP POST request to "/sendval/" of the server that you are running with the demo request body:
   {

   "val": 20,

   "text": "some free text in English"

   }

   Write a client function that sends 10 asynchronous requests to this endpoint with some dummy/distinct random integer value and random text to the said endpoint. Then waits for 3 seconds and then again sends 5 asynchronous requests. The random text should be generated by a random text generator function and the random integer value should be generated by a random integer function.

3. Write a Dockerfile for the 1 and 2 using which we can dockerize the solution and run on the said ports. Please include the instructions in Readme.md to build and run the docker image. [environment variable to include if any etc.]

TASK 2:

Solve any one of the three given problems below [even if not wholly, partially] :

a. https://www.kaggle.com/shivamb/netflix-shows [kaggle problem. You may use notebook or colab]

b. https://www.kaggle.com/dataturks/resume-entities-for-ner [kaggle problem. You may use notebook or colab]

c. Implement the TASK 1 using socketio in python for event-driven flow. It will be better if you can do the socketio implementation on an asynchronous server framework [e.g. aiohttp]
Source: https://python-socketio.readthedocs.io/en/latest/index.html

COPY-PASTING a solution from the web is strictly prohibited. More weightage will be given to unique solutions. Also, we would look at your choice of frameworks, approach and code for solving the problem, how would you determine the accuracy, precision, recall, F-score, AUC etc.

TASK 3:

You may do either of the two problems given below:

a. Take one predefined XML document, and show how you can use ANTLR to parse all such XML documents having similar grammar. You may use any python package for ANTLR.
b. Write a program A that continuously runs and outputs the CPU usage of the machine for 20s. Then write another program [let's call it receiver] that checks the status of the program A i.e. by taking the process id of the program as input, the receiver checks whether the process for program A is still running or finished. Modify this receiver program to take multiple process ids as input [Suppose 5 different processes are running] and output the status FINISHED with the

corresponding process id whenever any of the given input process id is finished. You may take a look at process queue and other utilities in python to give a solution design for this problem.

SUBMISSION

You may use the web for tutorials, documentation, or youtube as you need to learn to do this sample tasks.
Once the tasks are done, please upload/push it to the main branch of a repository of your GitHub account. If you do not have a GitHub account, create one, and then create a git repository. Then commit and push three folders for three tasks.
Then share the link to the repo. We will evaluate it.