Python Projects

Contents:

Python Projects

Contents:

Python Projects

Contents:

1. Basic level(keywords):

Python Projects

Contents:

Python Projects

Contents:

Python Projects

Contents:

Python Projects

Contents:

Beginner Level

Print statement

Qno.1) Make a basic python code which print "Hello World on screen."

Qno.2) Make a basic python code which print the function of print statement.

Qno.3) Make a basic python code which print some python advantages and disadvantages.

Input statement - operators

Qno.4) Make a basic python code which input two numbers and print their sum.

Qno.5) Make a basic python code which input two numbers and print their difference.

Qno.6) Make a basic python code which input two numbers and print their product.

Qno.7) Make a basic python code which input two numbers and print their reminder.

Qno.8) Make a basic python code which input a number and print it on the output terminal\screen.

If - elif - else statement

Qno.9) Make a basic python code which check if the input number is odd or even and print result on screen.

Qno.10) Make a basic python code which check if the input number is prime or not and print result on screen.

Beginner Level

If - elif - else statement

Qno.11) Make a basic python code which check if the input number is palindrome or not and print result on screen.

Qno.12) Make a basic python code which check if the input number is an Armstrong number or not and print result on screen.

Qno.13) Make a basic python code which check if the input number is perfect or not and print result on screen.

Qno.14) Make a basic python code which check if the input number is Fibonacci sequence or not and print result on screen.

Qno.15) Make a basic python code which check if the input number is strong number or not and print result on screen.

Loops

Qno.16) Make a basic python code which print 1-100 numbers on screen by loop.

Qno.17) Make a basic python code which print ********** by loop.

Qno.18) Make a basic python code which input 5 number and print them on screen by loop.

Array

Qno.19) Make a basic python code which input 5 numbers and store them in an array and print their reverse by loop.

Qno.20) Make a basic python code which check the array length.

Beginner Level

Array

Qno.21) Make a basic python code which store the data in memory in reverse order and print them in reverse order.

Qno.22) Make a basic python code which store the data by using array of size 10, if data is missing, I show the error.

Functions

Qno.23) Make a basic python code which have functionality of (+=*/), and solve them as user input data.

Qno.24) Make a basic python code which check if the user_input data is e, and print (You enter e) if true.

Qno.25) Make a basic python code which check if the e is in user_input, print (e is here) if condition true.

Global

Qno.26) Make a basic python code which store username (temporary) in a data.

Qno.27) Make a basic python code which save password in a data by global with in function.

Qno.28) Make a basic python code which input 5 number and print them in reverse order in a function.

Return

Qno.29) Make a basic python code which return a if user_input is a if not, print b.

Qno.30) Make a basic python code which return l as user_input length and print it and them multiply length out of function.

Beginner Level

Return

Qno.21) Make a basic python code which return username and password as user input.

Qno.22) Make a basic python code which store the data by using array of size 10, if data is missing, I return the error.

Qno.23) Make a basic python code which return the sum of two numbers.

Qno.24) Make a basic python code which return the sum of two numbers and also return the average of two numbers.

Qno.25) Make a basic python code which return the sum of two numbers and also return the average of two numbers and also return the multiplication of two numbers.

Make your own python progects you want!

Basic Level -- Keywords

Continue.

Qno.1) Make a basic python code which input and contiune inputs till user input l.

Qno.2) Make a basic python code which Loop from 1 to 20 and print only odd numbers using continue.

Qno.3) Make a basic python code which Ask the user to input a string and print the string with vowels skipped using continue.

Qno.4) Make a basic python code which Loop from 1 to 100 and skip numbers that are divisible by both 3 and 5.

Qno.5) Make a basic python code which Loop through a list of passwords, skip weak ones (len < 6 or contains only letters) using continue.

Qno.6) Make a basic python code which Loop through a list of numbers and skip even numbers using continue.

Qno.7) Make a basic python code which Loop through a list of names and skip names that start with 'A' using continue.

Qno.8) Make a basic python code which Loop through a list of numbers and skip numbers that are less than 10 using continue.

Qno.9) Make a basic python code which Loop through a list of words and skip words that are less than 4 characters long using continue.

Qno.10) Make a basic python code which Loop through a list of numbers and skip numbers that are greater than 50 using continue.

Basic Level -- Boolean, in

Continue.

Qno.11) Make a basic python code which Ask the user their age and check if they're old enough to vote (e.g., age >= 18 → True)

Qno.12) Make a basic python code which Create a simple username/password checker using and to validate both.

Qno.13) Make a basic python code which Check if a number is within a certain range, like between 10 and 50 using and

Qno.14) Make a basic python code which Simulate a traffic light system with Boolean conditions (e.g., if light == "green" and not pedestrian:)

Qno.15) Make a basic python code which For each question, return True if the answer is correct, otherwise False.

Qno.16) Make a basic python code which Check if an email contains @ and . using in and and.

Qno.17) Make a basic python code which Check if a year is a leap year

Qno.18) Make a basic python code which Simulate a basic alarm system with booleans like motion_detected = True and alarm_on = not password_entered.

Note: In question 1-10, you can also use pass.

Basic Level -- Boolean, in, ,break,agrements-->Function

Continue.

Qno.19) Make a basic python code which Write a function that returns None if no input is provided

Qno.20) Make a basic python code which Use None as a default parameter in functions when no argument is passed

Qno.21) Make a basic python code which Initialize empty to-do tasks with None until filled.

Qno.22) Make a basic python code which Create a dictionary of user data with fields that can be None (e.g., no phone number yet).

Qno.23) Make a basic python code which Make a function that searches a list and returns None if the item isn't found.

Qno.24) Make a basic python code which Let users "clear" an entry by setting its value to None.

Qno.25) Make a basic python code which Check if any required field is None, and return an error if so.

Qno.26) Make a basic python code which Keep asking for a password until the correct one is entered, then break.

Basic Level -- Boolean, in, ,break,agrements-->Function

Continue.

Qno.27) Make a basic python code which Write a function that returns None if no input is provided

Qno.28) Make a basic python code which Loop through a list of numbers and use break when the first even number is found.

Qno.29) Make a basic python code which Ask for words in a loop. If the user enters "stop", break the loop.

Qno.30) Make a basic python code which Show a menu repeatedly until the user chooses "Exit", then break.

Qno.31) Make a basic python code which Check if a number is prime; break if a divisor is found.

Qno.32) Make a basic python code which Loop through a string to find a specific character, and break when it's found.

Qno.33) Make a basic python code which Give the user 3 login attempts; break if login is successful.

Qno.34) Make a basic python code which Scan a list of numbers, break when a negative number is found.

Qno.35) Make a basic python code which Loop through names until a certain name is found, then break.

Basic Level -- file reader, try, expect and prevoius all.

Continue.

Qno.36) Make a basic python code which Ask the user for two numbers and divide them. Use except ZeroDivisionError to handle division by zero.

Qno.37) Make a basic python code which Prompt for a number using int(input()), and use except ValueError to catch non-numeric input.

Qno.38) Make a basic python code which try to read a file.

Qno.39) Make a basic python code which have login/signup function, if user input login, ask for username and password, and search in a file line by line.

Qno.40) Update Qno.39 that it save username and password in a that file.

Qno.41) Make a basic python code which Access a specific index in a list, and catch IndexError if it's out of range.

Qno.42) Make a basic python code which Access a key in a dictionary, and use except KeyError if the key doesn't exist.

Qno.43) Make a basic python code which Make a calculator that handles bad input with try-except.

Qno.44) Make a basic python code which Try converting strings to floats, catch errors if the input is invalid.

Basic Level -- file reader, try, expect and prevoius all.

Continue.

Qno.45) Make a basic python code which Try to open several files, and skip ones that don't exist using except.

Qno.46) Make a basic python code which Loop through a list and divide 100 by each value, catching ZeroDivisionError or TypeError

Qno.47) Make a basic python code which Try to open a file that doesn't exist, and catch the exception.

keywords--in, for, and prevoius

Qno.48) Make a basic python code which try to read a specific line in a file.

Qno.49) Make a basic python code which try to save file in a specific dircetory.

keywords--raise

Qno.50) Make a basic python code to capture output as in file.

Qno.51) Make a basic python code which Create a function that raises a ValueError if the user's age is less than 0.

Qno.52) Make a basic python code which Raise an exception if a username or password is empty.

Qno.53) Make a basic python code which Write a function to input a grade (0–100), and raise an error if the grade is out of range.

Qno.54) Make a basic python code which Raise a custom exception if the password is too short or lacks numbers/letters.

Basic Level -- keywords--raise.

Raise

Qno.55) Make a basic python code which Raise an error if a file name doesn't end with .txt or .csv.

Qno.56) Make a basic python code which Raise an exception if someone tries to withdraw more than their balance.

Qno.57) Make a basic python code which Create a custom InvalidEmailError and raise it if an email is missing @.

Qno.58) Make a basic python code which Raise an exception if input temperature is below absolute zero (e.g., -273.15°C).

Qno.59) Make a basic python code which Raise an exception if the task description is empty or too long.

Qno.60) Make a basic python code Raise TypeError if someone sends a string instead of a number to a math function.

keywords--class

Qno.61) Make a basic python code which Create a Person class with attributes like name, age, and a method to say hello.

Qno.62) Make a basic python code which Create a class to simulate a bank account with methods like deposit(), withdraw(), and check_balance().

Qno.63) Make a basic python code which Define a Car class with make, model, and drive() and stop() methods.

Qno.64) Make a basic python code which Make a Calculator class with methods like add(), subtract(), multiply(), and divide().

Basic Level -- keywords--finally.

finally

Qno.65) Make a basic python code which Open a file, read it, and close it using finally to ensure it closes no matter what.

Qno.66) Make a basic python code which Build a calculator that handles errors and prints "Goodbye!" in the finally block

Qno.67) Make a basic python code which Use finally to always print a message after a user inputs data, even if they mess up.

Qno.68) Make a basic python code which Try dividing two numbers, handle zero division, and log "division attempt finished" in finally.

Qno.69) Make a basic python code which Use finally to stop or reset a timer even if an error occurs during timing.

Qno.70) Make a basic python code Simulate a download and always show "Download finished" with finally.

keywords--is

Qno.71) Make a basic python code whichCheck if a variable is None (not just equal to None).

Qno.72) Make a basic python code which Use is True or is False to validate flags (e.g., logged_in is True).

Qno.73) Make a basic python code which Create two variables with the same value and check if they point to the same object (a is b).

Qno.74) Make a basic python code which Compare different data types using is to explore Python's memory behavior.

Basic Level -- keywords--lambda.

lambda

Qno.75) Make a basic python code which Use a lambda function for basic operations like addition, subtraction, etc.

Qno.76) Make a basic python code which Sort a list of tuples by the second item using a lambda function.

Qno.77) Make a basic python code which Use map() with a lambda function to multiply each element in a list by 2.

Qno.78) Make a basic python code which Use lambda to create a power function (e.g., x^y) and test it with different values.

Qno.79) Make a basic python code which Sort a dictionary by its values using sorted() and a lambda function.

Qno.70) Make a basic python code Find the smallest or largest number in a list using min() or max() with a lambda.

keywords--as-with

Qno.81) Make a basic python code Use with open() to read files, ensuring the file is automatically closed.

Qno.82) Make a basic python code which Use as to assign the result of a database query to a variable and handle it.

Qno.83) Make a basic python code which Catch an exception and use as to give it a name, allowing access to the error message.

Qno.84) Make a basic python code which Raise a custom exception and use as to capture it, allowing easy debugging.

Basic Level -- keywords-- Nonlocal

Nonlocal

Qno.85) Make a basic python code which Use nonlocal to modify a variable in a nested function.

Qno.86) Make a basic python code which Create a counter using nonlocal to maintain the count between function calls.

Qno.87) Make a basic python code which Use nonlocal to modify a variable from the outer function inside a nested loop.

Qno.88) Make a basic python code which Track the event count with nonlocal in a nested function scenario.

Qno.89) Make a basic python code which Create a function that modifies variables from the outer function using nonlocal

Qno.80) Make a basic python code Create a function that retains state using nonlocal and returns it in different calls.

keywords-- Assert

Qno.91) Make a basic python code Use assert to make sure the user input is positive.

Qno.92) Make a basic python code which Use assert to check if a list has at least 5 elements.

Qno.93) Make a basic python code which Use assert to make sure user input is within a specific range.

Qno.94) Make a basic python code which Use assert to check if two variables are equal, especially in unit tests.

Basic Level -- keywords-- Del

Del

Qno.95) Make a basic python code whichUse del to remove an item from a list by its index.

Qno.96) Make a basic python code which Use del to remove a specific key-value pair from a dictionary.

Qno.97) Make a basic python code which Use del to remove a variable from memory.

Qno.98) Make a basic python code which Use del to remove an object from a set.

Qno.99) Make a basic python code which Use del to delete a slice of a list.

Qno.100) Make a basic python code Use del to remove an object from a set.

keywords-- Async

Qno.101) Make a basic python code Write an async function that waits for a specified time before completing.

Qno.102) Make a basic python code which Read a file asynchronously to avoid blocking the program.

Qno.103) Make a basic python code which Simulate downloading files asynchronously to demonstrate speed improvements

keywords--Yield

Qno.104) Make a basic python code which Use yield to create a generator function that produces a sequence of numbers.

Qno.105) Make a basic python code which Use yield to create an infinite generator that produces Fibonacci numbers.

Qno.106) Make a basic python code which Use yield to create a generator that reads lines from a file lazily.

Qno.107) Make a basic python code which Use yield to create a generator that produces prime numbers.

Note: The above code is a basic level code. You can modify it as per your requirements.

Note: There are not all projects, you can also make your own, it just practise.

Advance Level -- keywords-- Imports

Built-in

1. os
2. sys
3. math
4. datetime
5. time
6. random
7. json
8. re
9. statistics
10. itertools
11. functools
12. collections
13. csv
14. subprocess
15. threading
16. asyncio
17. http
18. unittest
19. logging
20. socket

Advanced Level — Python Imports

Module: os

1. File Organizer – Sort files into folders by extension.
2. Directory Tree Viewer – Print the structure of a folder.
3. Batch Renamer – Rename all files in a folder at once.
4. Disk Usage Checker – Display sizes of files and folders.
5. Temp File Cleaner – Remove temporary files automatically.
6. Environment Variable Viewer – Show system environment vars.
7. Permission Modifier – Change read/write/execute modes.
8. Auto Folder Creator – Make project folder structures fast.
9. Log Archiver – Move old log files to backup folders.
10. Startup Script Generator – Build platform-specific startup scripts.

Advanced Level — Python Imports

Module: sys

1. Command-line Calculator – Take input via CLI args and do math.
2. Custom Error Logger – Print error messages to `stderr`.
3. Python Version Checker – Display current version using `sys.version`.
4. Path Explorer – Explore and edit Python import paths using `sys.path`.
5. Module Import Checker – Check if a module is available to import.
6. Memory Usage Tracker – Use `sys.getsizeof()` to inspect object sizes.
7. Script Timer – Use `sys.argv` to pass custom durations to a timer script.
8. Interactive Shell Detector – Detect whether script runs from shell or IDLE.
9. Command Logger – Save all command-line arguments to a file for auditing.
10. Exit Code Tester – Exit a script with custom codes and handle them externally.

Advanced Level — Python Imports

Module: math

1. Area Calculator – Compute area of circles, triangles, and rectangles.
2. Pythagorean Theorem Solver – Find missing side of a right triangle.
3. Simple Trig Tool – Compute sin, cos, tan for angles in degrees.
4. Logarithm Converter – Convert between log bases (e, 10, custom).
5. Factorial Finder – Calculate factorials using `math.factorial()`.
6. Prime Number Estimator – Estimate primes using `sqrt()` and division tests.
7. Decimal to Radian Converter – Convert degrees to radians and vice versa.
8. Compound Interest Calculator – Use exponentiation for financial growth.
9. Hypotenuse Finder – Apply `math.hypot()` directly to two sides.
10. Scientific Constants App – Display values like pi, e, and tau from the math module.

Advanced Level — Python Imports

Module: datetime

1. Date Difference Calculator – Find the difference between two dates.
2. Time Zone Converter – Convert time between different time zones.
3. Age Calculator – Calculate age from a birthdate.
4. Day of the Week Finder – Find the day of the week for any date.
5. Countdown Timer – Show remaining time until a future date.
6. Leap Year Checker – Check if a given year is a leap year.
7. Date Formatter – Format date/time in various styles.
8. Time Logger – Log events with timestamps to a file.
9. Event Scheduler – Schedule tasks based on dates and times.
10. Localized Date Display – Show dates in the local format of the user's location.

Advanced Level — Python Imports

Module: time

1. Stopwatch – Create a simple stopwatch using `time.time()`.
2. Delayed Message – Print messages with delays using `time.sleep()`.
3. Timer – Set a timer that counts down and alerts when done.
4. Current Time Display – Continuously show the current system time.
5. Interval Logger – Log data at specified time intervals.
6. Time Spent Calculator – Calculate how long a process took to run.
7. Time Zone Checker – Check current time in different time zones using `time.gmtime()`.
8. Countdown Timer – Countdown to a specific future time.
9. Performance Benchmarker – Measure execution time of code blocks.
10. Daylight Saving Checker – Check if current time is in daylight saving time.

Advanced Level — Python Imports

Module: random

1. Dice Roller – Simulate rolling dice with random numbers.
2. Random Password Generator – Create secure, random passwords.
3. Lottery Number Picker – Randomly select lottery numbers.
4. Random Sentence Generator – Generate random sentences from a word list.
5. Card Shuffling – Shuffle a deck of cards randomly.
6. Random Joke Generator – Fetch a random joke from a list.
7. Random Color Picker – Pick a random color from a set.
8. Random Name Picker – Pick a random name from a list of names.
9. Random Recipe Generator – Select a random recipe from a collection.
10. Random Quiz Generator – Pick random questions from a list for quizzes.

Advanced Level — Python Imports

Module: json

1. JSON File Reader – Read and load JSON data from a file.
2. JSON Data Validator – Validate JSON format and structure.
3. JSON-to-CSV Converter – Convert JSON data into CSV format for analysis.
4. JSON to Dictionary – Convert JSON strings into Python dictionaries.
5. JSON Formatter – Format and pretty-print JSON data.
6. API Response Logger – Log API responses in JSON format.
7. Nested JSON Reader – Parse nested JSON objects.
8. JSON Search – Search for specific keys or values in a JSON file.
9. JSON File Writer – Write Python data to a JSON file.
10. JSON Merger – Merge multiple JSON files into one.

Advanced Level — Python Imports

Module: re

1. Email Validator – Validate email addresses using regular expressions.
2. Phone Number Formatter – Format phone numbers to a standard style.
3. Password Strength Checker – Use regex to check password strength based on criteria.
4. Text Searcher – Search for a specific pattern (word, phrase) in a large text.
5. Date Finder – Extract dates (in various formats) from text.
6. HTML Tag Extractor – Extract all HTML tags from a given string of HTML code.
7. IP Address Extractor – Extract IP addresses from a text document.
8. Word Counter – Count occurrences of words in a sentence using regex.
9. Log File Parser – Parse log files to identify error patterns.
10. Data Masker – Mask sensitive information like credit card numbers in a string.

Advanced Level — Python Imports

Module: statistics

1. Mean and Median Calculator – Calculate mean, median, and mode of a dataset.
2. Standard Deviation Finder – Compute standard deviation for a list of numbers.
3. Data Outlier Detection – Identify outliers in a dataset using standard deviation.
4. Percentile Calculator – Calculate specific percentiles in a dataset.
5. Variance Finder – Compute the variance of a dataset.
6. Data Range – Calculate the range (difference between max and min values).
7. Data Normalization – Normalize data using z-scores or min-max scaling.
8. Random Sampling – Generate random samples and analyze mean/median.
9. Population vs Sample – Learn the difference and calculate statistics for both.
10. Moving Average – Calculate a moving average over a window of values.

Advanced Level — Python Imports

Module: itertools

1. Permutation Generator – Generate all permutations of a list.
2. Combinations Finder – Find all combinations of a given size from a list.
3. Infinite Counter – Create an infinite counting generator using `itertools.count()`.
4. Circular Iteration – Use `itertools.cycle()` for repeating a sequence infinitely.
5. Group By – Group elements of a sequence by a common key using `itertools.groupby()`.
6. Cartesian Product – Compute the cartesian product of multiple sequences.
7. Chunks of Data – Split a list into fixed-size chunks using `itertools.islice()`.
8. Running Totals – Generate a running total for a list of numbers.
9. Collapsing Sequences – Collapse a sequence by skipping elements using `itertools.dropwhile()`.
10. Slice Generator – Create slices of data based on a condition using `itertools.compress()`.

Advanced Level — Python Imports

Module: collections

1. Frequency Counter – Use `Counter()` to count occurrences of items in a list.

2. Default Dictionary – Use `defaultdict()` to handle missing keys in dictionaries.

3. Named Tuple – Create a named tuple for cleaner, readable data structures.

4. Ordered Dict – Maintain insertion order with `OrderedDict()`.

5. Queue Implementation – Use `deque()` for fast appends and pops from both ends of a list.

6. Multiset Operations – Use `Counter()` to perform set operations like union, intersection, etc.

7. Chain Iterators – Use `chain()` to combine multiple iterables into a single iterator.

8. Itertools with Collections – Use collections with itertools for efficient data processing.

9. Circular Buffer – Use `deque()` as a circular buffer to store fixed-size data.

10. Data Grouping – Use `groupby()` to group data based on a specified condition.

Advanced Level — Python Imports

Module: csv

1. CSV File Reader – Read data from a CSV file and print it in tabular form.
2. CSV Writer – Write data from a Python list to a CSV file.
3. CSV Data Cleaner – Clean and filter data from a CSV file.
4. CSV to Dictionary – Read a CSV file and convert it into a dictionary.
5. CSV Data Summarizer – Summarize data by aggregating statistics from a CSV.
6. CSV Row Updater – Modify specific rows in a CSV file based on conditions.
7. CSV to JSON Converter – Convert CSV data into JSON format.
8. Column Filter – Filter and select specific columns from a CSV file.
9. CSV Reader with Error Handling – Handle errors when reading malformed CSV data.
10. CSV Data Exporter – Export data to a CSV file from a database or external source.

Advanced Level — Python Imports

Module: subprocess

1. Run External Commands – Use `subprocess.run()` to execute shell commands.
2. Pipe Output – Capture the output of a shell command and use it in your script.
3. Process Management – Launch and manage processes asynchronously.
4. File Operations – Use subprocess to manipulate files through command-line tools.
5. Command Argument Parser – Parse arguments and options passed to a subprocess.
6. Background Process – Run commands in the background and track their execution.
7. Handle Errors – Capture error codes and output from failed subprocess calls.
8. Execute Shell Scripts – Use `subprocess` to run shell scripts from Python.
9. Parallel Task Execution – Execute multiple tasks concurrently with subprocess.
10. External Command Timer – Track the execution time of external commands.

Advanced Level — Python Imports

Module: threading

1. Threaded Web Scraper – Scrape web data using multiple threads to improve speed.
2. Multithreaded File Downloader – Download multiple files concurrently using threading.
3. Thread Pool – Use a thread pool to manage a set of worker threads.
4. Background Task Runner – Run background tasks concurrently while the main program continues.
5. Threaded Timer – Run a timer in a separate thread to allow non-blocking execution.
6. Shared Resource Locking – Prevent race conditions when accessing shared resources with threading locks.
7. Producer-Consumer Problem – Solve the producer-consumer problem using threading.
8. Parallel Processing of Data – Split a large data set into chunks and process in parallel.
9. GUI Event Loop – Use threading to run a GUI event loop alongside other tasks.
10. Thread Synchronization – Ensure proper synchronization between threads to avoid conflicts.

Advanced Level — Python Imports

Module: asyncio

1. Simple Asynchronous Web Scraper – Scrape multiple web pages asynchronously.
2. Asynchronous File Downloader – Download files asynchronously from multiple sources.
3. Background Task Scheduler – Run background tasks while waiting for user input.
4. WebSocket Server – Build an asynchronous WebSocket server for real-time communication.
5. Async Database Query – Perform database queries asynchronously to prevent blocking.
6. Asynchronous Email Sender – Send emails asynchronously in a queue.
7. Async HTTP Server – Build an HTTP server that handles requests asynchronously.
8. Async File I/O – Perform non-blocking file read/write operations.
9. Periodic Task Scheduler – Run tasks at specific intervals using `asyncio`.
10. Coroutine Timeout – Set time limits for coroutines and manage cancellations.

Advanced Level — Python Imports

Module: http

1. Simple HTTP Server – Create a basic HTTP server using the `http.server` module.
2. HTTP Request Handler – Handle GET and POST requests using `BaseHTTPRequestHandler`.
3. Web API Client – Build a client to send HTTP requests to a REST API.
4. HTTP Response Status – Parse HTTP status codes and handle responses.
5. File Server – Serve files over HTTP using `http.server` for file sharing.
6. Custom HTTP Headers – Create custom HTTP headers for your requests and responses.
7. Redirect Server – Implement an HTTP redirect server using `http`.
8. HTTPS Request Handler – Handle secure HTTPS requests with SSL support.
9. HTTP Proxy Server – Build a basic HTTP proxy server to route requests.
10. Cookie Handling – Send and receive cookies using the `http.cookies` module.

Advanced Level — Python Imports

Module: unittest

1. Basic Test Case – Create a simple unit test for a Python function.
2. Test Setup and Teardown – Use `setUp()` and `tearDown()` methods for test preparation.
3. Mocking Objects – Use `unittest.mock` to mock objects in your tests.
4. Test Suite – Group multiple tests into a suite using `unittest.TestSuite()`.
5. Parameterized Tests – Write tests that run with different inputs using `unittest.subTest()`.
6. Test Assertions – Use different assertions to verify the behavior of your code (e.g., `assertEqual()`).
7. Test Coverage – Measure and improve test coverage using `unittest`.
8. Test Result Handling – Customize test results and report generation.
9. Integration Testing – Use unittest to perform integration testing of multiple modules.
10. Continuous Testing – Implement continuous testing in a CI pipeline using `unittest`.

Advanced Level — Python Imports

Module: logging

1. Basic Logging – Set up basic logging to log messages to a file.

2. Log Level Customization – Customize log levels (e.g., DEBUG, INFO, WARNING, ERROR, CRITICAL).

3. Rotating Logs – Use `logging.handlers.RotatingFileHandler()` for log rotation.

4. Logging to Multiple Destinations – Log to multiple outputs like files, console, and remote servers.

5. Error Logging – Log exceptions and errors with stack traces using `logging.exception()`.

6. Logging Filters – Create custom filters for logs using `logging.Filter()`.

7. Email Notifications – Set up email alerts when critical errors are logged using `logging.handlers.SMTPHandler()`.

8. Log Formatting – Format log messages with timestamps, log level, and other info.

9. Performance Monitoring – Log performance metrics and execution time of functions.

10. Debugging Application – Use detailed logging for debugging an application in production.

Advanced Level — Python Imports

Module: socket

1. Simple TCP Server – Build a basic TCP server that listens for client connections.
2. TCP Client – Create a client that connects to the server and sends messages.
3. UDP Server – Build a simple UDP server to send and receive datagrams.
4. Port Scanner – Scan open ports on a target server using `socket`.
5. Chat Application – Develop a real-time chat application using sockets and threads.
6. Remote File Transfer – Send and receive files between a client and server.
7. Reverse DNS Lookup – Perform a reverse DNS lookup on an IP address.
8. Network Performance Monitor – Measure and log network performance metrics.
9. Web Server – Build a basic HTTP server that responds to requests.
10. Secure Socket Layer (SSL) Server – Set up a secure SSL server using `ssl` module with `socket`.

Python Module Projects

OS Module

Project 1: Make a python-code programme for advance calculator.

Project 2: Make a python-code programme for advance Numberguessing game.

Project 3: Make a python-code programme which handle files.

Project 4: Make a python-code programme which have AI.

Project 5: Make a basic text editor with some advance features.

Project 6: Make a python-code programme which install videos online.

Project 7: Make a python-code programme which can handle os-system easy.

Project 8: Make a secure login\signup function.