1. Mutable vs Immutable Data Types

Mutable data types can be changed after creation (list, set, dict).

Immutable data types cannot be changed (int, float, string, tuple).

Example: list [1,2] can be modified.

String "abc" cannot be modified.

Change creates a new object in immutable types.

2. Factorial using loop

Factorial of n is product of numbers from 1 to n.

Use for loop to multiply values.

Initialize fact = 1.

Loop from 1 to n.

Print factorial value.

3. append() vs extend()

append() adds one element at the end.

extend() adds multiple elements.

append adds list as single element.

extend merges elements individually.

Both are list methods.

4. Exception Handling

It handles runtime errors.

Uses try, except blocks.

Prevents program crash.

Example: divide by zero error.

Ensures smooth execution.

5. Modules in Python

Module is a file with Python code.

Used to reuse code.

Imported using import.

Example: import math.

math module provides functions.

6. Mutable and Immutable (Again)

Mutable: list, set, dictionary.

Immutable: int, float, tuple, string.

Mutable objects can change.

Immutable objects cannot change.

Memory differs for both.

7. Polymorphism

Same function name, different behavior.

Works with methods and operators.
Example: + for int and string.
Supports flexibility.
Used in OOP.

8. break, continue, pass
break exits loop.
continue skips current iteration.
pass does nothing.
Used in loops.
Control loop flow.

9. File Modes
File mode specifies operation.
r – read mode.
w – write mode.
a – append mode.
Used in file handling.

10. Relational vs Non-Relational DB
Relational uses tables.
Non-relational uses documents/collections.
Relational uses SQL.
Non-relational uses NoSQL.
Examples: MySQL vs MongoDB.

11. List vs Tuple
List is mutable.
Tuple is immutable.
List uses [ ].
Tuple uses ( ).
Tuple is faster.

12. Exception Handling – Why
Handles runtime errors.
Prevents abnormal termination.
Improves reliability.
Separates error logic.
Makes code robust.

13. try, except, finally
try contains risky code.
except handles error.
finally always executes.

Used for cleanup.
Ensures safety.

14. Recursion
Function calling itself.
Used for repetitive tasks.
Has base condition.
Example: factorial.
Simplifies code.

15. Slicing
Extracts part of sequence.
Uses [start:end].
Works on string/list.
Example: a[1:4].
Returns new sequence.

16. Inheritance
Child class gets parent properties.
Promotes code reuse.
Types: single, multiple, multilevel.
Uses class Child(Parent).
OOP concept.

17. Lambda Functions
Anonymous functions.
Written in one line.
Uses lambda keyword.
Example: lambda x:x*x.
Used for short tasks.

18. File Handling
Used to store data permanently.
open() opens file.
read() reads data.
write() writes data.
close() closes file.

19. Dictionary
Stores key-value pairs.
Uses {} braces.
Keys are unique.
Fast access.
Example: {1:"a"}

20. Local vs Global Variables
Local defined inside function.
Global defined outside function.
Local scope limited.
Global accessible everywhere.
global keyword used.

21. List, Tuple, Set, Dictionary
List: ordered, mutable.
Tuple: ordered, immutable.
Set: unordered, unique values.
Dictionary: key-value pairs.
Different use cases.

22. Inheritance in Python
Child class inherits parent.
Uses class B(A).
Reduces redundancy.
Improves maintenance.
Supports OOP.

23. 8 String Methods
upper(), lower()
strip(), replace()
split(), join()
find(), len()
Used for string handling.

24. 6 List Methods
append()
extend()
insert()
remove()
pop()
sort()

25. Data Types in Python
int, float – numbers.
str – text.
list, tuple – collections.
set – unique values.
dict – key-value.

26. Factorial Program
Uses loop or recursion.
Multiply numbers.
Initialize variable.
Loop till n.
Print result.

27. Conditional Statements
Used for decision making.
if, elif, else.
Checks conditions.
Executes block.
Example: even/odd.

28. Looping Statements
Used for repetition.
for and while loops.
Reduce code length.
Control execution.
Used with break.

29. OOP Concepts
Class and object.
Encapsulation.
Inheritance.
Polymorphism.
Abstraction.

30. Prime Number Program
Check divisibility.
Loop from 2 to n-1.
If divisible → not prime.
Else prime.
Display result.

31. String Manipulation
Modify strings.
Using methods.
Slicing, replace.
Concatenation.
Formatting strings.

32. Reverse a String
Use slicing [::-1].

Or loop method.
Store reversed string.
Display output.
Simple logic.

33. Functions
Block of reusable code.
Defined using def.
Can take arguments.
Returns value.
Improves modularity.

34. Largest of Three Numbers
Compare using if-else.
Check conditions.
Store max value.
Print result.
Simple logic.

35. List Comprehension
Short syntax for lists.
Uses loop in one line.
Readable and fast.
Example: [x*x for x in a].
Efficient.

36. CRUD in MySQL
Create – INSERT.
Read – SELECT.
Update – UPDATE.
Delete – DELETE.
Basic DB operations.

37. Abstraction & Encapsulation
Abstraction hides details.
Encapsulation binds data & methods.
Uses classes.
Improves security.
Supports OOP.

38. Git with VS Code
Install Git.
Initialize repository.
Stage changes.

Commit code.
Push/Pull repo.

39. File Operations
open() opens file.
read() reads content.
write() writes data.
close() closes file.
Used for storage.

40. Conditionals & Loops
Conditionals make decisions.
Loops repeat execution.
if-else checks logic.
for/while repeat tasks.
Core programming tools.

41. Exception Handling Mechanism
Detects runtime errors.
Uses try-except.
Handles specific errors.
Avoids crash.
Improves reliability.

42. Inheritance & Polymorphism
Inheritance reuses code.
Polymorphism allows many forms.
Method overriding.
Same interface, different behavior.
OOP feature.

43. Types of Databases
Relational DB.
Non-relational DB.
Centralized DB.
Distributed DB.
Each has pros & cons.

44. MySQL Commands
CREATE, DROP.
INSERT, SELECT.
UPDATE, DELETE.
ALTER.
Used to manage DB.

45. Python Architecture

Interpreter based.

Uses bytecode.

Portable language.

Dynamic typing.

Easy to use.

46. Types of Functions

Built-in functions.

User-defined functions.

Lambda functions.

Recursive functions.

Modular programming.

47. Memory Management

Handled automatically.

Uses garbage collection.

Reference counting.

Efficient memory use.

No manual freeing.

48. File Handling Modes

r – read.

w – write.

a – append.

r+ – read/write.

Used in files.

49. Git & GitHub Workflow

Git is version control.

GitHub is hosting platform.

Clone repository.

Commit changes.

Push to GitHub.

50. Python Libraries for Data Science

NumPy – numerical computing.

Pandas – data analysis.

Matplotlib – visualization.

Seaborn – advanced plots.

Scikit-learn – ML.