Theoretical Questions (Python Basics)

1) What is Python, and why is it popular?

Answer: Python is a high-level, interpreted programming language known for its clean syntax and vast ecosystem of libraries.

Why popular?

- Readable, beginner-friendly syntax
- Large standard library and third-party packages (data, web, ML, automation)
- Cross-platform, open-source, strong community
- Supports multiple paradigms: procedural, object-oriented, functional

2) What is an interpreter in Python?

Answer: The interpreter reads Python source code and executes it directly, translating it into machine-understandable actions at runtime (as opposed to compiling ahead of time into a separate binary). CPython is the default reference interpreter.

3) What are pre-defined keywords in Python?

Answer: Keywords are reserved words that have special meaning in Python syntax (e.g., `if`, `for`, `True`, `None`) and cannot be used as identifiers (variable/function names).

4) Can keywords be used as variable names?

Answer: No. Keywords are reserved by the language. Attempting to use one as a variable name causes a `SyntaxError`.

5) What is mutability in Python?

Answer: Mutability is whether an object's value can change **in place**. Mutable objects (e.g., `list`, `dict`, `set`) can be modified after creation; immutable objects (e.g., `int`, `float`, `str`, `tuple`) cannot.

6) Why are lists mutable, but tuples are immutable?

Answer: By design: `list` is intended for dynamic collections where elements may be added/removed/changed. `tuple` provides a fixed-size, hashable (when elements are immutable) sequence useful for constants, dictionary keys, or protecting data from modification.

7) What is the difference between `==` and `is` operators in Python?

Answer: `==` checks **value equality** (do two objects have equal content?). `is` checks **identity** (are they the exact same object in memory?).
Example: two separate lists with equal elements are `==` but not `is`.

8) What are logical operators in Python?

Answer: `and`, `or`, `not`. They perform boolean logic and use short-circuit evaluation:

- `A and B` → evaluates `B` only if `A` is truthy
- `A or B` → evaluates `B` only if `A` is falsy
- `not A` → boolean negation of `A`

9) What is type casting in Python?

Answer: Converting a value from one data type to another, such as `int('10')`, `float('3.14')`, `bool(0)`, or `str(123)`.

10) What is the difference between implicit and explicit type casting?

Answer: Implicit casting happens automatically (e.g., `int + float → float`). **Explicit** casting is done intentionally by the programmer using constructors like `int()`, `float()`, `str()`, `bool()`.

11) What is the purpose of conditional statements in Python?

Answer: Conditionals (`if`/`elif`/`else`) allow branching logic—executing different code paths based on whether conditions evaluate to true or false.

12) How does the `elif` statement work?

Answer: `elif` is “else if”. It’s evaluated only if all previous conditions were false. The first `if`/`elif` whose condition is true runs; remaining branches are skipped.

13) What is the difference between `for` and `while` loops?

Answer:

- `for`: iterate over a known sequence/iterator for a fixed number of steps.
- `while`: repeat as long as a condition remains true—useful when the number of iterations isn’t predetermined.

14) Describe a scenario where a `while` loop is more suitable than a `for` loop.

Answer: Reading user input until they type `quit`, retrying network requests until success, or looping until a sensor value crosses a threshold—where the number of iterations is unknown beforehand.