- Part 1
    1. **Scripts and Shell Scripts:**
        - A script is a command-line program containing a series of commands.
        - Shell scripts are executed by a shell interpreter and are great for automating tasks.
    2. **Execution of Shell Scripts:**
        - Before executing a shell script, ensure it has executable permissions (`chmod 755`).
        - You can execute a script by typing its name if it's in your path, or by specifying its relative or full path.
    3. **Shebang (`#!`) Line:**
        - The shebang line specifies the interpreter to use for executing the script.
        - It starts with `#!` followed by the path to the interpreter.
    4. **Variables in Shell Scripts:**
        - Variables in shell scripts are name-value pairs.
        - Use syntax like `VARIABLE_NAME="Value"` to assign values to variables.
        - Variables are case-sensitive and conventionally written in uppercase.
        - Access variable values using `$VARIABLE_NAME` or `${VARIABLE_NAME}` syntax.
    5. **Commands in Shell Scripts:**
        - Commands can be executed directly in shell scripts.
        - Output of commands can be captured in variables using `$()` or backticks.
    6. **Tests and Conditionals:**
        - Conditionals can be used to make decisions in shell scripts.
        - Tests are performed using conditional expressions enclosed in square brackets `[ ]`.
        - Various tests include file existence (`e`), readability (`r`), writability (`w`), and others.
    7. **String and Number Comparisons:**
        - String comparisons include equality (`=`, `!=`), empty string (`z`, `n`), and others.
        - Number comparisons include equality (`eq`, `ne`), less than (`lt`, `le`), greater than (`gt`, `ge`).
    8. **Additional Information:**
        - Python scripts can also be executed using shebang lines.
        - It's essential to be explicit about the interpreter used in the shebang line.
    - Flags for tests
        1. **Existence Tests:**
            - `e`: Returns true if the file exists.
            - `f`: Returns true if the file exists and is a regular file (not a directory or symlink).
        2. **Permission Tests:**
            - `r`: Returns true if the file is readable by you.
            - `w`: Returns true if the file is writable by you.
            - `x`: Returns true if the file is executable.
        3. **Size and Content Tests:**

        - `s`: Returns true if the file exists and is not empty.
        - `z`: Returns true if the string is empty.
        - `n`: Returns true if the string is not empty.
    4. **Directory Test:**
        - `d`: Returns true if the given path is a directory.

    When these tests are performed using the `test` command or its equivalent `[ ]`, they return either true (exit status 0) or false (exit status 1) based on the condition being tested. These tests are commonly used in conditional statements (`if`, `elif`, `else`) in shell scripts to make decisions based on file attributes or string values.

- Part 2
    1. **Conditional Statements:**
        - You can determine if a condition is true or false using conditional tests.
        - Conditional statements, like the 'if' statement, allow you to make decisions in your scripts based on these conditions.
    2. **Syntax of the 'if' Statement:**
        - The 'if' statement begins with the keyword 'if' followed by a test.
        - It is followed by the keyword 'then' and a command or series of commands to execute if the condition is true.
        - The 'if' statement ends with 'fi', which is 'if' spelled backward.
    3. **Handling False Conditions with 'else':**
        - You can execute commands when the condition is false using the 'else' keyword followed by the corresponding commands.
    4. **Testing Multiple Conditions with 'elif':**
        - 'elif' (else if) allows you to test multiple conditions in sequence.
    5. **Looping Constructs:**
        - 'for' loops allow you to perform actions on a list of items.
        - They iterate over each item in the list, executing commands for each iteration.
        - Behavior: The first item in the list is assigned to the variable and the commands are executed, then the next item on the list is assigned to the variable and so on
    6. **Positional Parameters:**
        - Positional parameters ($1, $2, ..., $9) contain the contents of the command line.
        - These parameters help scripts to process information passed during execution.
    7. **Comments and Shebang Lines:**
        - Shebang lines (#!) specify the interpreter to use for executing scripts.
        - Comments, denoted by the pound sign (#), are ignored by the interpreter and provide human-readable explanations within the script.
    8. **Accepting User Input:**
        - The 'read' command accepts input from standard input, typically from the keyboard.
        - You can prompt users for input and store their responses in variables.
    9. **Best Practices:**
        - Variables should be enclosed in quotes to avoid unexpected side effects.

- Be explicit about the interpreter used in the shebang line for better script portability.