Assignment-11

Ans-1. Assert statement to check if `spam` is a negative integer:

```
assert spam >= 0, 'spam should not be a negative integer'
```

Ans-2. Assert statement to check if `eggs` and `bacon` are case-insensitively the same:

```
assert eggs.lower() != bacon.lower(), 'eggs and bacon should not be the same'
```

Ans-3. Assert statement that throws an `AssertionError` every time:

```
assert False, 'This assertion always triggers'
```

Ans-4. Two lines required to call `logging.debug()`:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

Ans-5. Two lines required for `logging.debug()` to log messages to `programLog.txt`:

```
import logging
logging.basicConfig(filename='programLog.txt', level=logging.DEBUG)
```

Ans-6. The five levels of logging:

- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL

Ans-7. Line of code to disable all logging messages:

```
logging.disable(logging.CRITICAL)
```

Ans-8.

- Logging allows you to record messages with varying severity levels (e.g., DEBUG, INFO, ERROR).
- You can configure logging to write messages to a file, display them in the console, or suppress them entirely.
- It provides better control over where and how messages are outputted.
- Logging can be disabled or set to record only certain levels without changing the codebase, unlike `print()`.

Ans-9.

- Step Over: Executes the current line of code but doesn't step into any function calls.
- Step In: Steps into a function call and allows debugging inside it.
- Step Out: Finishes executing the current function and returns to the calling function.

Ans-10. The debugger will stop when it hits the next breakpoint or when the program finishes execution.

Ans-11. <u>Concept of a breakpoint:</u> A breakpoint is a marker you set in your code where the debugger will pause execution, allowing you to inspect the state of the program at that point.