Ans. 4.

Python has several built-in exceptions that cover common error scenarios. Here are some of the most commonly used built-in exceptions:

SyntaxError: Raised when there is a syntax error in the code.

IndentationError: Raised when the indentation in the code is incorrect.

NameError: Raised when a local or global name is not found.

TypeError: Raised when an operation or function is applied to an object of an inappropriate type.

ValueError: Raised when a function receives an argument of the correct type but an inappropriate value.

ZeroDivisionError: Raised when division or modulo operation is performed with zero as the divisor.

IndexError: Raised when a sequence subscript is out of range.

KeyError: Raised when a dictionary key is not found.

FileNotFoundError: Raised when a file or directory is requested but cannot be found.

IOError: Raised when an input/output operation fails.

AssertionError: Raised when an assert statement fails.

OverflowError: Raised when the result of an arithmetic operation is too large to be expressed.

ImportError: Raised when an import statement fails to find the specified module.

TypeError: Raised when an incorrect type is passed to a function.

AttributeError: Raised when an attribute reference or assignment fails.

These are just a few examples of built-in exceptions. Python provides a wide range of exceptions to handle various error scenarios that might occur during program execution.

Ans. 5.

In Python, logging is a built-in module that provides a flexible and configurable way to record messages, events, and errors that occur during the execution of a program. It is an essential tool for developers to collect useful information about the behavior of their software while it is running.

The logging module allows developers to create log records with different levels of severity, such as DEBUG, INFO, WARNING, ERROR, and CRITICAL. These log records can be saved to various destinations, such as a file, the console, or a remote server. The module also supports log rotation, filtering, and formatting, enabling developers to control what information is logged and how it is presented.

The importance of logging in software development lies in the following aspects:

Debugging and Troubleshooting

Error Reporting and Monitoring

Performance and Optimization

Auditing and Compliance

Understanding User Behavior

Long-Term Maintenance

Ans. 9.

In Python, both logging and print statements are used for displaying information during the program's execution. However, they serve different purposes and have distinct advantages depending on the context of their use.

Print Statements:

* print is a built-in Python function used to display output on the standard output stream (usually the console).
* It is simple to use and is often used for quick debugging or getting immediate feedback during development.
* Print statements are typically used for temporary debugging purposes and should be removed or commented out once the debugging phase is complete. Leaving print statements in the code can clutter the output and may have performance implications, especially in large-scale applications.

Logging:

* Logging is a more advanced and versatile mechanism for recording messages during the program's execution.
* The logging module in Python provides a full-featured logging system with various log levels (DEBUG, INFO, WARNING, ERROR, CRITICAL), allowing you to control the verbosity of the output.
* Log messages can be directed to different output streams (e.g., console, files), making it easier to manage and analyze logs in a real-world application.
* Logging is designed to be persistent and is not removed during production, allowing developers to monitor and diagnose issues in a deployed application.
* It offers flexibility in formatting log messages, capturing stack traces, and including contextual information.
* With logging, you can differentiate between different log levels, so you can log only important events or errors in production while keeping detailed logs during development and testing.

Overall, using logging instead of print statements in real-world applications improves maintainability, provides better control over the output, and offers a more structured and informative approach to monitoring and debugging.