**Assignment – 11**

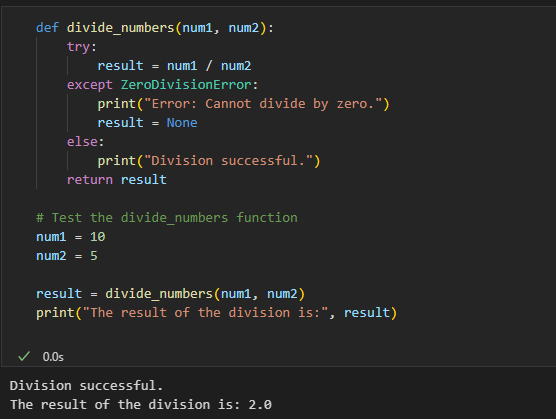
**18th June 2023**

1. **What is the role of the 'else' block in a try-except statement? Provide an example scenario where it would be useful.**

**Ans:** The “else” block in a “try-except” statement in Python is used to specify a block of code that should be executed if no exceptions occur in the corresponding “try” block. In other words, the “else” block is executed only when the “try” block runs successfully without raising any exceptions.

The “else” block is optional. If present, it must follow all the “except” blocks, and it should not contain any code that can potentially raise an exception.

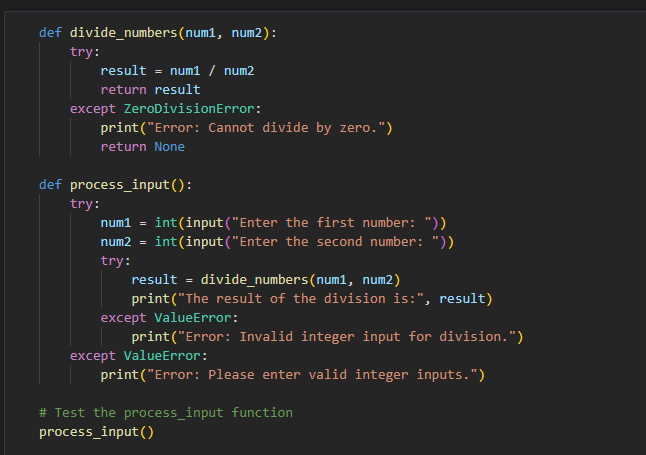
Example:



1. **Can a try-except block be nested inside another try-except block? Explain with an example.**

**Ans:** Yes, a “try-except” block can be nested inside another “try-except” block in Python. Nesting try-except blocks allow to handle exceptions at different levels of granularity, providing more fine-grained error handling in the code.

Here's an example of a nested try-except block:

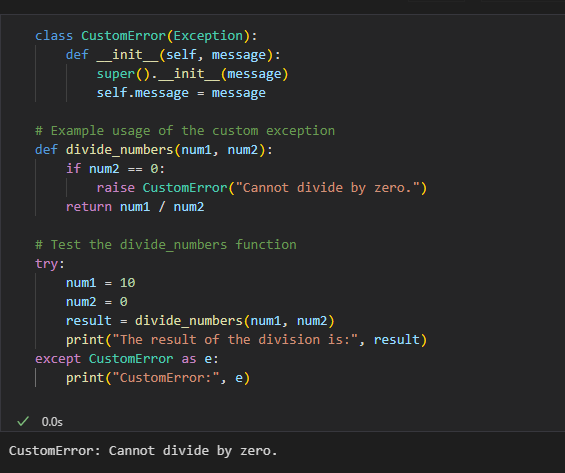


In this example, we have two functions: divide\_numbers() and process\_input(). The divide\_numbers() function takes two numbers as input and attempts to perform division. It has its own try-except block to handle the ZeroDivisionError that may occur during division.

1. **How can you create a custom exception class in Python? Provide an example that demonstrates its usage.**

**Ans:** In Python, you can create a custom exception class by defining a new class that inherits from the built-in Exception class or one of its subclasses. By creating a custom exception class, you can define specific exception types tailored to your application's needs and provide more meaningful error messages to users or developers when exceptional situations occur.

Here's an example of creating a custom exception class:



1. **What are some common exceptions that are built-in to Python?**

**Ans:** Python provides several built-in exception classes to handle common error scenarios. Some of the most common exceptions that are built-in to Python include:

* **SyntaxError:** Raised when there is a syntax error in the Python code, indicating that the code is not properly formatted.
* **IndentationError:** Raised when there is a problem with the indentation of the code, such as inconsistent use of tabs and spaces.
* **NameError:** Raised when a variable or name is not defined in the current scope.
* **TypeError:** Raised when an operation or function is applied to an object of an inappropriate data type.
* **ValueError:** Raised when a function or operation receives an argument of the correct type, but the value is not suitable for the operation.
* **ZeroDivisionError:** Raised when division or modulo operation is performed with a divisor of zero.
* **IndexError:** Raised when trying to access an element from a sequence (e.g., list, tuple, or string) using an invalid index or an index that is out of range.
* **KeyError:** Raised when trying to access a non-existent key in a dictionary.
* **FileNotFoundError:** Raised when trying to access a file that does not exist.
* **IOError:** Raised when an I/O operation (e.g., reading or writing to a file) fails.
* **ImportError:** Raised when an import statement fails to find or load a module.
* **AssertionError:** Raised when an assert statement fails.

1. **What is logging in Python, and why is it important in software development?**

**Ans:** Logging in Python is the process of recording events, messages, or status information during the execution of a program to a log file, the console, or other output destinations. The Python standard library provides a powerful logging module that allows developers to log various types of messages with different severity levels, making it easier to monitor and debug applications effectively.

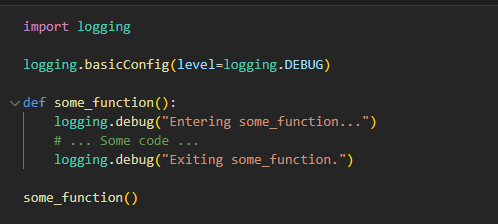
Importance of Logging in Software Development:

* + 1. **Debugging and Troubleshooting:** Logging is essential for debugging and troubleshooting applications. It provides valuable insights into the flow of the program, the values of variables, and the occurrence of critical events, making it easier to identify and fix issues.
    2. **Monitoring and Maintenance:** In production environments, logging helps monitor the health of applications by capturing errors, warnings, and critical events. It aids in maintaining the application and quickly responding to potential problems.
    3. **Auditing and Compliance:** Logging can be used for auditing purposes and to meet compliance requirements in various industries.
    4. **Performance Optimization:** By logging execution times and resource utilization, developers can identify performance bottlenecks and optimize the application accordingly.
    5. **Historical Analysis:** Log files can be analyzed historically to understand patterns, trends, and potential issues that may have occurred in the past.

1. **Explain the purpose of log levels in Python logging and provide examples of when each log level would be appropriate.**

**Ans:** The Python logging module defines several log levels, each serving a distinct purpose. The most commonly used log levels are (in increasing order of severity):

* **DEBUG:** The lowest log level used for debugging information. These messages are typically used during development to track the flow of the program, log variable values, or trace specific actions.



* **INFO:** Used to provide general information about the application's operation. These messages are informative and typically used to track significant milestones or key events during program execution.

A screen shot of a computer program

Description automatically generated

* **WARNING:** Used to indicate potential issues or non-critical problems. These messages highlight situations that might lead to errors or unexpected behavior.

A screen shot of a computer program

Description automatically generated

* **ERROR:** Used to indicate errors that are recoverable but require immediate attention. These messages highlight problems that may cause the application to fail or produce incorrect results.

A computer screen shot of a program

Description automatically generated

* **CRITICAL:** The highest log level, used for severe errors or critical failures that can lead to the application's termination or significant data loss.

A screen shot of a computer code

Description automatically generated

1. **What are log formatters in Python logging, and how can you customize the log message format using formatters?**

**Ans:** In Python logging, log formatters are used to specify the format of log messages when they are output to the log destination. The logging module provides a set of pre-defined formatters, and you can also customize the log message format to suit your specific needs.

A log formatter is an instance of the logging.Formatter class, which is responsible for converting log records into formatted log messages. The Formatter class has methods to customize the log message format, including options to include the timestamp, log level, module name, message, etc

1. **How can you set up logging to capture log messages from multiple modules or classes in a Python application?**

**Ans:** To set up logging to capture log messages from multiple modules or classes in a Python application, you need to follow these steps:

* Create a logger for each module or class that requires logging.
* Set the desired log level and log message format for each logger.
* Add handlers (e.g., FileHandler, StreamHandler) to each logger to specify where the log messages will be output (e.g., log files, the console).

1. **What is the difference between the logging and print statements in Python? When should you use logging over print statements in a real-world application?**

**Ans:** The logging module and print statements in Python serve different purposes and have distinct characteristics.

The logging module is designed specifically for generating log messages and capturing them in a configurable manner. It provides log levels (e.g., DEBUG, INFO, WARNING, ERROR, CRITICAL), custom log message formats, and various output handlers (e.g., log files, the console, remote servers). Logging is primarily used for debugging, monitoring, and maintaining the application.

While the print statement is a built-in function used for basic output to the console. It is typically used for temporary or ad-hoc debugging during development and provides less control over log levels and message formatting compared to the

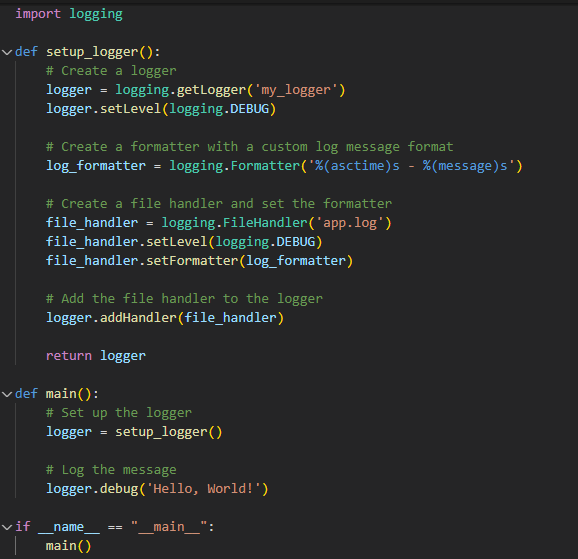
*When to use logging over print statements in a real-world application:*

* **Debugging and Maintenance:** For debugging and maintaining a complex application, using the logging module is more appropriate. It allows you to selectively log messages at different log levels, track the flow of the program, capture errors, and monitor important events.
* **Production Use:** In a production environment, print statements are generally discouraged as they can clutter the console and may expose sensitive information to end-users. The logging module, on the other hand, allows you to log essential information while keeping the output well-structured and controlled.
* **Long-Term Development:** In larger projects, using the logging module allows consistent logging practices throughout the codebase. It makes it easier to manage logs, track issues, and analyze application behavior in a systematic manner.
* **Log Analysis and Monitoring:** The logging module enables you to log data in a standardized format that can be analyzed and monitored more effectively using log analysis tools.

**10. Write a Python program that logs a message to a file named "app.log" with the following requirements:**

**● The log message should be "Hello, World!"**

**Ans:**

****

**● The log level should be set to "INFO."**

**Ans:**

**A computer screen shot of text

Description automatically generated**

**● The log file should append new log entries without overwriting previous ones.**

**Ans:**

**A computer screen shot of text

Description automatically generated**

**11. Create a Python program that logs an error message to the console and a file named "errors.log" if an exception occurs during the program's execution. The error message should include the exception type and a timestamp.**

**Ans: refer below.**

**A screenshot of a computer code

Description automatically generated**