Exception Handling in Python

Exception Handling in Python

This note is a compilation of information from three videos on exception handling in Python.

The `try-except` block

The `try-except` block is a fundamental method for handling errors and preventing a program from crashing. The `try` block contains the code that might produce an error, and the `except` block specifies what to do if an error occurs. You can also use multiple `except` blocks to handle specific types of errors, such as `ValueError` or `IndexError`.

Code Example: `try-except`

```
# Basic example
try:
    # Code that might cause an error
    num = int(input("Enter a number: "))
except ValueError:
    # Code to run if a ValueError occurs
    print("Invalid input. Please enter a number.")

# Example with multiple except blocks
try:
    # Code that might cause an error
    my_list = [1, 2, 3]
    print(my_list[int(input("Enter an index: "))])
except ValueError:
    print("Invalid input. Please enter a number.")
except IndexError:
    print("Index out of range.")
```

The `finally` keyword

The `finally` block is used to ensure a certain piece of code is always executed, regardless of whether an error occurs. This is especially useful for "clean-up" operations, such as closing files or database connections.

Code Example: `finally`

```
try:
    # Code that might cause an error
    f = open("my_file.txt", "r")
    content = f.read()
except FileNotFoundError:
    print("The file was not found.")
finally:
    # This block will always run
    if 'f' in locals():
        f.close()
    print("Cleanup is complete.")
```

Raising Custom Errors

You can raise custom errors using the `raise` keyword to intentionally stop a program when invalid input is provided or an unexpected condition is met.

Code Example: `raise`

```
# Raising a custom ValueError
num = int(input("Enter a number between 5 and 9: "))
if num < 5 or num > 9:
    raise ValueError("Number must be between 5 and 9")

# Defining a custom exception class
class InvalidAgeError(Exception):
    pass

def check_age(age):
    if age < 0:
```

```
        raise InvalidAgeError("Age cannot be negative")
    return age

check_age(-5)
```