# Handling Multiple Exceptions raise

In Python, you can handle multiple exceptions in a single `try-except` block by specifying multiple `except` clauses, each handling a different type of exception. This allows you to provide specific handling for various error scenarios. Here's an explanation with examples:

### Handling Multiple Exceptions:

```python
try:
    # Code that might raise different exceptions
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"ZeroDivisionError: {e}")
except TypeError as e:
    print(f"TypeError: {e}")
except Exception as e:
    print(f"Caught an exception: {e}")
```

In this example:
- If a `ZeroDivisionError` occurs, the first `except` block will be executed.
- If a `TypeError` occurs, the second `except` block will be executed.
- The third `except` block, with the generic `Exception` type, serves as a catch-all for any other exceptions that may occur.

### Handling Multiple Exceptions in One Block:

You can also handle multiple exceptions in a single `except` block by using parentheses to specify a tuple of exception types.

```python
try:
    # Code that might raise different exceptions
    result = 10 / 0
except (ZeroDivisionError, TypeError) as e:
    print(f"Caught an exception: {e}")
except Exception as e:
    print(f"Caught an exception: {e}")
```

This is functionally equivalent to the previous example. The `except` block catches either a `ZeroDivisionError` or a `TypeError`, and the generic `Exception` block serves as a catch-all.

### Ordering of Except Blocks:

When handling multiple exceptions, it's important to order the `except` blocks from the most specific to the most general. If the order is reversed, the more general exception will catch everything, and the more specific ones will be unreachable.

```python
try:
    # Code that might raise different exceptions
    result = 10 / 0
except Exception as e:
    print(f"Caught a generic exception: {e}")
except ZeroDivisionError as e:
    # This block will never be reached due to the previous generic except block
    print(f"Caught a ZeroDivisionError: {e}")
```

In this example, the `ZeroDivisionError` block will never be executed because the more general `Exception` block comes first.

Handling multiple exceptions provides a way to create more robust error-handling mechanisms in your code, allowing you to respond differently to various exceptional situations.