FUTURE VISION™
COMPUTER INSTITUTE

HANDLING EXCEPTIONS IN PYTHON:

@futurevisioncomputers

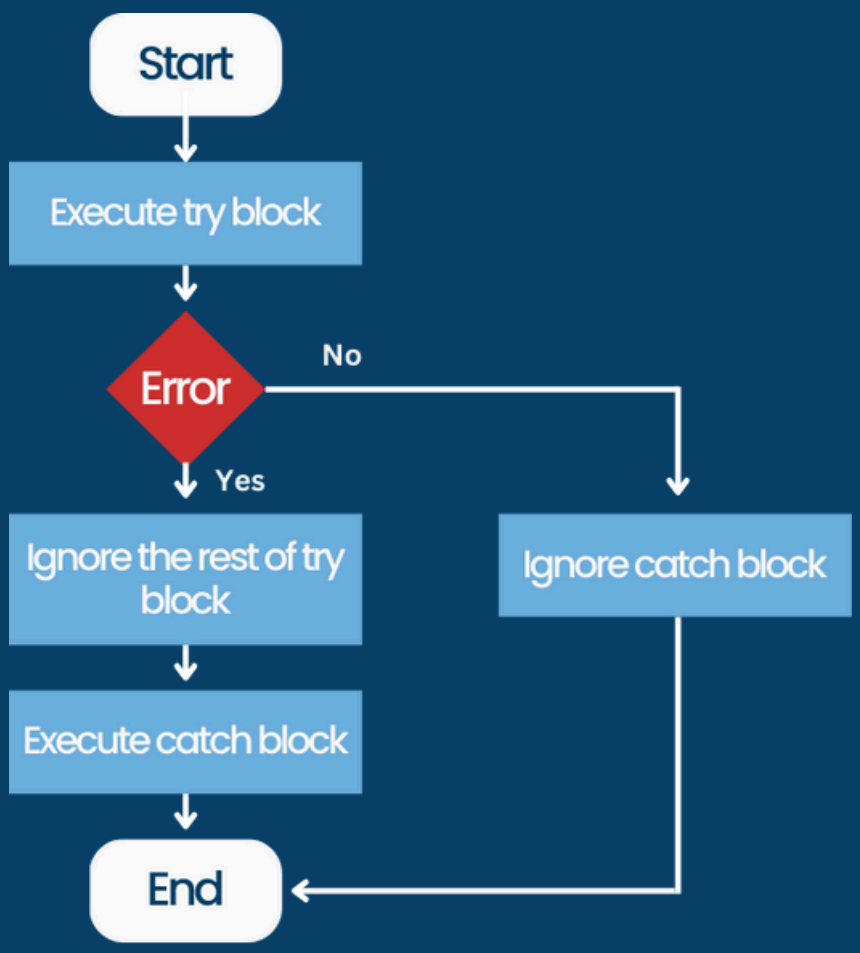
```python
try:
    # Attempting a risky operation
    result = 10 / 0  # This will raise a ZeroDivisionError
except ZeroDivisionError:
    print("An error occurred: Division by zero is not allowed.")
finally:
    print("This will execute no matter what.")
```

HANDLING EXCEPTIONS IN PYTHON:

A Crash Course Ever had your Python code crash unexpectedly? That's where exception handling comes in! It's like having a safety net for your code.

FUTURE VISION

@futurevisioncomputers



HOW IT WORKS:

- Try block: This is where you put your code that might raise exceptions.
- Except block: This is where you handle the exceptions that might occur.
- Finally block: (Optional) This code always runs, whether an exception occurs or not.

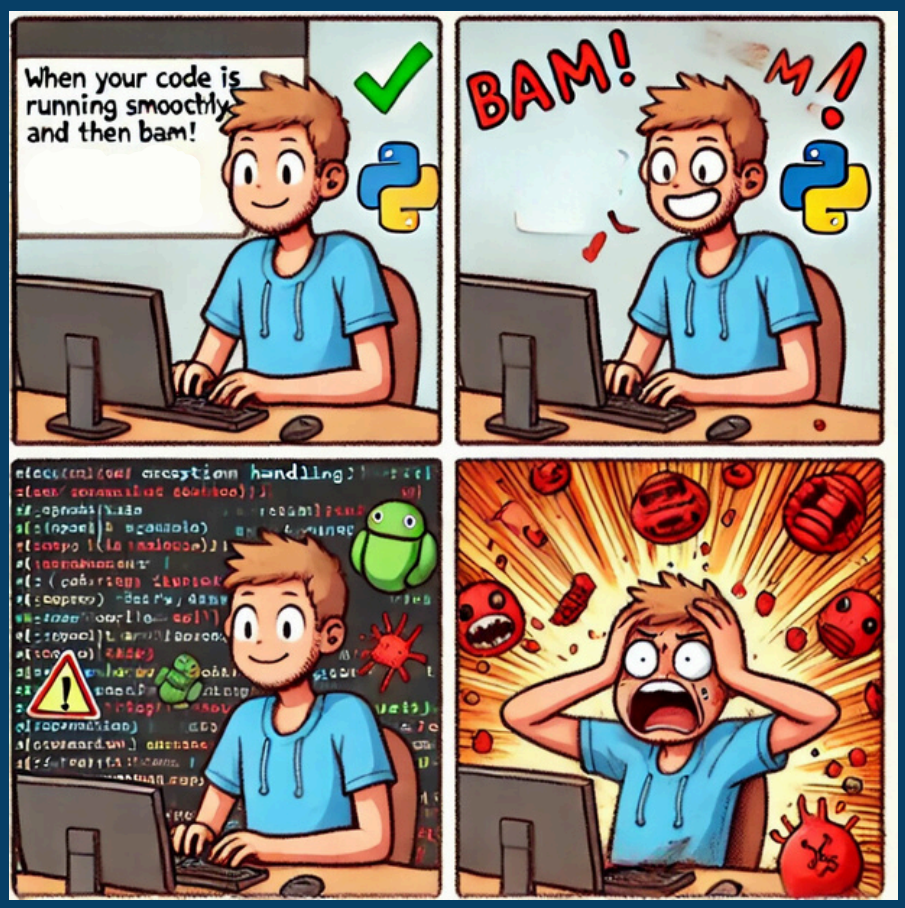
@futurevisioncomputers

```python
try:
    # Code that may raise an exception
    num = int(input("Enter a number: "))
    result = 10 / num
    print("Result:", result)

except ValueError:
    print("Error: Invalid input. Please enter a valid integer.")

except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")

finally:
    print("Execution completed.")
```

COMMON EXCEPTIONS:

- ZeroDivisionError: Trying to divide by zero.
- TypeError: Using an incorrect data type.
- ValueError: An invalid value is used.
- IndexError: Accessing an index out of range.
- KeyError: Trying to access a key that doesn't exist in a dictionary.
- And many more!

@futurevisioncomputers



WHEN YOUR CODE IS RUNNING SMOOTHLY AND THEN BAM! AN UNEXPECTED EXCEPTION HITS.

Remember: Exception handling is essential for robust and reliable Python programs. By anticipating potential errors and gracefully handling them, you can prevent your code from crashing and ensure a better user experience.

SINCE 2006

FUTURE VISION

DID YOU LIKE THE POST?

FOLLOW FOR MORE!

Comment

Save

Share