# Notes on Error Handling in Python

### 🔹 What is Error Handling?

Errors happen when your program doesn’t run as expected. Instead of crashing, Python allows you to **handle errors gracefully** using try, except, else, finally.

Common Errors:

* **IndexError** → When you try to access an invalid index in a list.

•   orders = ["masala", "ginger"]

•   print(orders[2])  # ❌ IndexError: list index out of range

* **KeyError** → When you try to access a dictionary key that doesn’t exist.
* **ZeroDivisionError** → When dividing a number by zero.
* **TypeError** → When operations are applied to wrong data types.
* **NameError** → When you use a variable that hasn’t been defined.

### 🔹 Basic Try–Except

chai\_menu = {"masala": 30, "ginger": 40}

try:

    chai\_menu["elaichi"]   # ❌ KeyError

except KeyError:

    print("The key you are trying to access does not exist")

print("hello chai code")   # ✅ Program continues

### 🔹 Try–Except–Else–Finally

* **try** → Code that might cause an error.
* **except** → Code that runs if an error occurs.
* **else** → Code that runs if no error occurs.
* **finally** → Code that always runs (cleanup, closing files).

def serve\_chai(flavour):

    try:

        print(f"Preparing {flavour} chai...")

        if flavour == "unknown":

            raise ValueError("I don't know this flavour")  # custom error

    except ValueError as e:

        print("Error:", e)

    else:

        print(f"{flavour} chai is served")

    finally:

        print("Next customer please")

serve\_chai("Masala")

print("----------------")

serve\_chai("unknown")

### 🔹 Catching Multiple Exceptions

def process\_order(item, quantity):

    try:

        price = {"masala": 20}[item]   # ❌ KeyError if item not found

        cost = price \* quantity        # ❌ TypeError if quantity not number

        print(f"Total cost is {cost}")

    except KeyError:

        print("Sorry, that chai is not on menu")

    except TypeError:

        print("Quantity must be a number")

process\_order("ginger", 2)   # KeyError

process\_order("masala", "two")  # TypeError

### 🔹 Raising Your Own Errors (raise)

def brew\_chai(flavour):

    if flavour not in ["masala", "ginger", "elaichi"]:

        raise ValueError("Unsupported chai flavour...")

    print(f"Brewing {flavour} chai...")

brew\_chai("masala")

# brew\_chai("mint")   # ❌ Raises ValueError

### 🔹 Creating Custom Exceptions

class OutOfIngredientsError(Exception):

    pass

def make\_chai(milk, sugar):

    if milk == 0 and sugar == 0:

        raise OutOfIngredientsError("Missing Milk or Sugar")

    print("Chai is ready")

make\_chai(0, 0)  # ❌ Custom exception raised

### 🔹 Mini Project (Billing System with Exceptions)

class InvalidChaiError(Exception):

    pass

def bill(flavour, cups):

    menu = {"masala": 20, "ginger": 40}

    try:

        if flavour not in menu:

            raise InvalidChaiError("That chai is not available")

        if not isinstance(cups, int):

            raise TypeError("Number of cups must be an integer")

        total = menu[flavour] \* cups

        print(f"Your bill for {cups} cups of {flavour} chai: Rs. {total}")

    except Exception as e:

        print("Error:", e)

    finally:

        print("Thank you for visiting chai code!")

bill("mint", 2)         # ❌ Invalid flavour

bill("masala", "three") # ❌ Wrong type

bill("ginger", 3)       # ✅ Works fine

# 📘 Notes on File Handling in Python

### 🔹 What is File Handling?

File handling lets us **store, read, and update data** permanently in files. Python provides built-in functions like open(), read(), write(), and uses **modes**:

* "r" → Read (default)
* "w" → Write (overwrites file)
* "a" → Append (adds to file)
* "x" → Create new file
* "b" → Binary mode (images, videos, etc.)

### 🔹 Using try-except with Files

try:

    file = open("order.txt", "w")

    file.write("Masala chai - 2 cups")

except Exception as e:

    print("Error writing file:", e)

finally:

    file.close()

### 🔹 Using with (Best Practice)

with automatically closes the file, even if errors occur.

with open("order.txt", "w") as file:

    file.write("Ginger tea - 4 cups")  # No need to close

✅ **Key Takeaways for Notes**:

1. Errors can be handled gracefully using try-except.
2. Use else for successful runs, finally for cleanup.
3. Raise built-in or custom errors using raise.
4. File handling is safer with with instead of open + close.