Python Course - Part Three

1. Introduction

In this part of the Python course, we focus on files, errors, and modules. These concepts are crucial for building real-world applications because they help us handle data persistence, manage errors gracefully, and organize code into reusable components.

2. Exception Handling

Python exceptions help us handle errors gracefully.

Basic structure: try / except / else / finally

Example:

```
def safe_divide(a, b):
    try:
        result = a / b
    except ZeroDivisionError as e:
        print("Division by zero error:", e)
        return None
    except TypeError as e:
        print("Type error:", e)
        return None
    else:
        return result
    finally:
        print("Execution complete")
```

3. File Handling

Python provides support for working with files such as txt, csv, and json.

TXT Example:

```
with open("notes.txt", "w", encoding="utf-8") as f:
    f.write("Hello World\n")
```

CSV Example:

```
import csv
with open("contacts.csv", "w", newline='', encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(["id", "name", "email"])
    writer.writerow([1, "Alice", "alice@example.com"])
```

JSON Example:

```
import json
data = {"name": "Alice", "email": "alice@example.com"}
with open("data.json", "w", encoding="utf-8") as f:
    json.dump(data, f, indent=2)
```

The open() function in Python is a built-in function used to open files and return a **file object**. This object can then be used to **read**, **write**, or **append** to the file, depending on the mode you specify.

🔑 Syntax

```
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)
```

📄 Parameters

Parameter	Description
file	Path to the file (absolute or relative).
mode	Optional. Mode in which the file is opened. Default is 'r'.
buffering	Optional. Buffering policy (-1 = default, 0 = unbuffered, 1 = line buffered, >1 = buffer size).
encoding	Encoding (like 'utf-8', 'ascii', etc.). Needed for text files.
errors	How to handle encoding/decoding errors.
newline	Controls how universal newlines mode works (useful in text mode).

Parameter Description

closefd If False, file descriptor is not closed (only works if file is a file descriptor).

opener A custom opener. (Advanced use)

📂 Common File Modes

Mode Meaning

'r' Read (default). File must exist.

'w' Write. Creates file or truncates if exists.

'x' Write. Fails if file exists.

'a' Append. Creates file if not exists.

'b' Binary mode (add to other modes like 'rb' or 'wb').

't' Text mode (default).

'+' Read and write (e.g., 'r+', 'w+').

🧪 Examples

1. Read a file

```
with open('example.txt', 'r') as f:
    contents = f.read()
    print(contents)
```

2. Write to a file

```
with open('example.txt', 'w') as f:
    f.write("Hello, world!")
```

3. Append to a file

```
with open('example.txt', 'a') as f:
```

```
    f.write("\nAnother line.")
```

4. Read lines into a list

```
with open('example.txt', 'r') as f:
    lines = f.readlines()
```

5. Read line by line

```
with open('example.txt', 'r') as f:
    for line in f:
        print(line.strip())
```

6. Binary read

```
with open('image.png', 'rb') as f:
    data = f.read()
```

✅ **Best Practice: Use with Statement**

Using with open(...) automatically closes the file, even if an error occurs. This is better than manually calling f.close().

⚠️ **Common Errors**

- FileNotFoundError: File doesn't exist in 'r' mode.
- PermissionError: You don't have permission to access the file.
- UnicodeDecodeError: Wrong encoding when reading text.

4. Importing & Creating Modules

Modules let us organize code into reusable pieces.

mymath.py:

```
def add(a, b):
    return a + b
```

use_module.py:
import mymath
print(mymath.add(2, 3))

5. Python Standard Library

Key libraries include os, sys, and datetime.

OS Example:
import os
print(os.getcwd())

SYS Example:
import sys
print(sys.argv)

Datetime Example:
from datetime import datetime
print(datetime.now())

6. Capstone Project – Contacts CLI

The Contacts CLI app ties together exception handling, file handling, modules, and standard libraries. It supports JSON and SQLite storage, uses argparse for CLI, and allows adding, listing, removing, and exporting contacts.

Usage Examples:

python -m app.main add "Alice" alice@example.com +2348012345678

python -m app.main list

python -m app.main --backend sqlite export contacts.csv

7. Practice & Assignments

1. Write a program that reads a text file and counts the number of words.

2. Export JSON data into a CSV file.

3. Create a module that provides math functions and import it in another script.

4. Extend the Contacts CLI to include a search function.

5. Handle errors gracefully when a user provides an invalid email.