# Unit 4

## 1 Mark Questions & Answers

### File Handling, File modes, File methods

1. **Q:** What is the default mode when opening a file in Python?
   **A:** `"r"` (read mode).
2. **Q:** Which file mode is used to add content at the end of a file without overwriting it?
   **A:** `"a"` (append mode).
3. **Q:** Which method is used to read one line at a time from a file?
   **A:** `readline()`.
4. **Q:** What does the `tell()` method do?
   **A:** Returns the current position of the file pointer.
5. **Q:** Which mode is used to open a file for both reading and writing?
   **A:** `"r+"`.

---

### Working on CSV and JSON files:

1. **Q:** What does CSV stand for?
   **A:** CSV stands for *Comma-Separated Values*.
2. **Q:** Which module in Python is used to handle CSV files?
   **A:** The `csv` module.
3. **Q:** Which method is used to write a row into a CSV file?
   **A:** `writerow()` method.
4. **Q:** Which function is used to load data from a JSON file?
   **A:** `json.load()` function.
5. **Q:** Which function converts a Python dictionary into a JSON string?
   **A:** `json.dumps()` function.
6. **Q:** What does `newline=""` do in `open()` while writing CSV files?
   **A:** It prevents extra blank lines from being added between rows in Windows.
7. **Q:** In JSON, what are the data types equivalent to Python's dictionary and list?
   **A:** Dictionary → JSON Object, List → JSON Array.
8. **Q:** Which JSON function is used for pretty printing with indentation?
   **A:** `json.dump(data, file, indent=4)`.

## Exceptions, Exception Handling, Raising Exceptions:

1) **Q:** What is an exception in Python?
   **A:** An exception is an error that occurs during program execution and disrupts normal flow.
2) **Q:** Which keywords are used in exception handling in Python?
   **A:** `try`, `except`, `else`, `finally`, `raise`.
3) **Q:** What is the purpose of the `else` block in exception handling?
   **A:** The `else` block runs only if no exception occurs in the `try` block.
4) **Q:** Name two built-in exceptions in Python.
   **A:** `ZeroDivisionError`, `ValueError`.
5) **Q:** What is the use of the `finally` block?
   **A:** It always executes, whether an exception occurs or not (commonly used for cleanup like closing files).

## Regular Expressions:

## Q1. What is a regular expression in Python?

A1. A regular expression is a sequence of characters that defines a search pattern, often used for string matching and validation.

**Q2.** Which module in Python supports regular expressions?
**A2.** The `re` module.

**Q3.** Which function is used to check if a regex matches the whole string?
**A3.** `re.fullmatch()`.

**Q4.** What does the `.` (dot) metacharacter in regex represent?
**A4.** It matches any single character except newline (`\n`).

**Q5.** What does `*` mean in regex?
**A5.** It matches zero or more repetitions of the preceding element.

**Q6.** Which regex function returns all non-overlapping matches as a list?
**A6.** `re.findall()`.

**Q7.** What is the use of `\d` in regex?
**A7.** It matches any digit (0–9).

**Q8.** What symbol is used to represent the start of a string in regex?
**A8.** `^` (caret symbol).

# 2 Mark Questions & Answers

## File Handling, File modes, File methods:

1. **Q:** Differentiate between `"w"` and `"a"` file modes.
   **A:** `"w"` overwrites the file if it exists or creates a new one.

   `"a"` appends data to the end of the file, creating it if it does not exist.

2. **Q:** What is the difference between `read()` and `readlines()`?
   **A:** `read()` returns the entire file content as a string.

   `readlines()` returns all lines as a list of strings.

3. **Q:** What is the purpose of `seek()` method? Give an example.
   **A:** It moves the file pointer to a specific position.
   Example: `f.seek(0)` moves the pointer to the beginning of the file.

4. **Q:** Write a Python statement to open a file named `"data.txt"` in binary read mode.
   **A:** `f = open("data.txt", "rb")`.

---

## Working on CSV and JSON files:

1. **Q:** Differentiate between `csv.reader()` and `csv.DictReader()`.
   **A:** `csv.reader()` reads each row as a list of strings.

   `csv.DictReader()` reads each row as a dictionary, using the first row as keys.

2. **Q:** Write a Python statement to open a file `data.csv` in append mode.
   **A: f = open("data.csv", "a", newline="")**

   **3. Q:** How do `json.dump()` and `json.dumps()` differ?
   **A:** `json.dump()` writes JSON data directly to a file.

   `json.dumps()` converts a Python object to a JSON string.

   **4. Q:** Write one line Python code to read JSON data from a file `info.json`.
   **A: data = json.load(open("info.json", "r"))**

   **5. Q:** State one advantage of JSON over CSV.
   **A:** JSON supports **nested and hierarchical data**, whereas CSV is limited to tabular data.

## Exceptions, Exception Handling, Raising Exceptions

1) **Q: Differentiate between syntax error and runtime error.**
   **A:** Syntax error: Errors in code structure, detected before execution (e.g., missing colon).

   Runtime error (exception): Errors occurring during execution (e.g., division by zero).

2) **Q: Write a Python program using `try-except` to handle division by zero.**
   **A:**
```
try:
    a = 10 / 0
except ZeroDivisionError:
    print("Division by zero is not allowed")
```

3) **Q: What is exception propagation?**
   **A:** If an exception is not handled in a function, it propagates back to the caller function until it is handled or program terminates.

4) **Q: Give an example of raising an exception manually.**
   **A:**
```
x = -1
if x < 0:
    raise ValueError("Negative values not allowed")
```

## Regular Expressions:

**Q1. Differentiate between `re.search()` and `re.match()`.**
**A1.**

- `re.match()` checks for a match **only at the beginning** of the string.
- `re.search()` checks for a match **anywhere** in the string.

---

**Q2. What is the difference between `re.findall()` and `re.finditer()`?**
**A2.**

- `re.findall()` returns all matches as a **list of strings**.
- `re.finditer()` returns an **iterator of match objects**, providing detailed match info.

---

**Q3. Write a regex pattern to validate an Indian mobile number starting with 7, 8, or 9 and having 10 digits.**
**A3. pattern = r'^[789]\d{9}$'**

**Q4. Explain the use of `re.split()` with an example.**
**A4.** `re.split()` splits a string by the occurrences of a pattern.
Example:

import re

print(re.split(r'\s+', 'Python is fun'))

# Output: ['Python', 'is', 'fun']

**Q5. How does `re.sub()` work? Give an example.**
**A5.** `re.sub(pattern, repl, string)` replaces all matches of the pattern with `repl`.

### Example:

import re

print(re.sub(r'\d', '*', 'Phone: 98765'))

# Output: Phone: *****

# 5 Mark Questions & Answers

1. **Q:** Explain different file modes in Python with examples.
   **A:**
   - `"r"`: Read file (error if not exists).
   - `"w"`: Write file (overwrite/create).
   - `"a"`: Append data.
   - `"x"`: Create new file (error if exists).
   - `"r+"`: Read + write existing file.
   - `"w+"`: Write + read (overwrite/create).
   - `"a+"`: Append + read.
     Example:

     ```
     f = open("test.txt", "w")
     f.write("Hello")
     f.close()
     ```

**2. Q:** Explain any five file methods with examples.
**A:**

- `read()`: Reads file content.

- `readline()`: Reads one line.
- `write()`: Writes data.
- `seek()`: Moves pointer.
- `tell()`: Returns pointer position.
  Example:

```
f = open("test.txt", "w")
f.write("Python\n")
f.close()

f = open("test.txt", "r")
print(f.readline())
f.close()
```

**3. Q:** Write a Python code to merge two given file contents into a third file.

**4. Q:** Write a program that reads a file and displays the number of words, number of vowels, blank spaces, lower case letters and uppercase letters.

**5. Q:** Write a Python code to Read text from a text file, find the word with most number of occurrences.

**6. Q:** Write a Python code to open a given file and to check for given words present in it and display on found.

## Exceptions, Exception Handling, Raising Exceptions:

**7. Q: Explain the flow of execution in `try`, `except`, `else`, and `finally` blocks with an example.**
**A:**

- `try`: Contains code that may raise an exception.
- `except`: Handles the exception.
- `else`: Executes only if no exception occurs.
- `finally`: Always executes.

**Example:**

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
except ZeroDivisionError:
    print("Division by zero error!")
```

```
    else:
        print("Result is:", result)
    finally:
        print("Execution completed")
```

**Explanation:** If `num=0`, exception occurs → `except` runs → `finally` runs.
If `num=5`, no error → `else` runs → `finally` runs.

**8. Q: List and explain any four built-in exceptions in Python with examples.**

**A: i)** ZeroDivisionError: Raised when dividing by zero.
Eg: print(5/0)
ii) **ValueError:** Raised when invalid value is passed.
Eg: int("abc")
iii) **IndexError:** Raised when accessing an invalid list index.
Eg: lst = [1,2,3]
print(lst[5])
iv) **TypeError:** Raised when operation is applied to an invalid type.
Eg: print("5" + 2)

**9 Q: Explain except clause with multiple exceptions.**

```
def check_number(n):
    if n < 0:
        # raising ValueError with custom message (acting as custom exception)
        raise ValueError("Negative numbers are not allowed")
    return n

try:
    num = -5
    print(check_number(num))
except ValueError as e:
    print("Caught exception:", e)
```

**10 Q: Explain how to create user-defined exceptions.**

**11Q: Write a program that raises and handles a custom exception.**

**12Q: What happens if except clause is written without any Exception type? Explain with an example.**

Answer: If you write an **`except` clause without specifying any exception type**, it will **catch all exceptions** (no matter what error occurs).

Eg:

```
try:
   x = int("abc")   # This will raise ValueError
   y = 10 / 0       # This would raise ZeroDivisionError
except:
   print("An error occurred!")
```

**13Q: Discuss with an example Exceptions with arguments.**

**14Q: Write about Errors and Exception handling in python.**

**15 Q: Define error and exception. Differentiate them.**

**A. Error**

- An **error** indicates a problem in the program that makes it impossible to run.
- Errors are mostly **syntax-related** or **logical mistakes** made by the programmer.
- Errors stop the execution of the program **before it runs properly**.

**Exception**

- An **exception** is an **event that occurs during program execution** and disrupts the normal flow of instructions.
- Exceptions usually happen due to **runtime issues** (like division by zero, accessing an undefined variable, file not found, etc.).
- Exceptions can be **handled** using `try-except`.

## Differences between Error and Exception

| Aspect | Error | Exception |
|---|---|---|
| **Definition** | Mistakes in program code (syntax/logic) | Events that occur during program execution |
| **When it occurs** | During compilation/parsing (before execution) | During execution (runtime) |
| **Examples** | `SyntaxError`, `IndentationError` | `ZeroDivisionError`, `ValueError`, `FileNotFoundError` |
| **Can it be handled?** | **No**, must be fixed in code | **Yes**, can be handled with `try-except` |
| **Cause** | Programmer mistakes (typos, wrong syntax) | Invalid operations or unexpected runtime conditions |
| **Recovery** | Program cannot continue unless fixed | Program can continue if handled properly |

# Regular Expressions:

**16 Q: Describe the core functions and methods of re module**
A. Core functions: match, search, findall, split, sub

Methods: group, start, end, span

**17 Q: Does regular expression pattern enable matching of multiple strings. Justify.**

A. Yes, **regular expression (regex) patterns enable matching of multiple strings** because they are not fixed string literals – they are **patterns** that describe a set of strings that share a common structure.

## Justification

1. A **regex pattern** defines **rules** (using special characters like `. * + |` etc.) rather than exact text.
2. These rules allow a single regex pattern to match **different variations of strings** at once.
3. This makes regex powerful for **search, validation, and text extraction**.

**18 Q: Explain various string pattern matching functions in python.**

**19 Q: How to find whether an email Id entered by the user is valid or not?**

```
import re

# Regex pattern for a valid email

pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

# Take user input

email = input("Enter your email ID: ")


# Check if it matches the pattern

if re.fullmatch(pattern, email):

    print("Valid Email ID")

else:

    print("Invalid Email ID")
```

**20 Q: Illustrate common regular expression symbols and special characters used in Python for string matching.**