Exception Handling – 7 Questions

1. What is the difference between `except Exception as e:` and `except:`? Which is preferred and why?

except: catches all exceptions

except Exception as e: only catches exceptions derived from Exception

‘except Exception as e’ is preferred.

2. Write a program that reads a number from the user and divides 100 by that number. Handle:

- `ValueError` if input is not a number
- `ZeroDivisionError` if input is 0
- Any other unexpected error

```
try:
    num = int(input("Enter a number: "))
    result = 100 / num
    print("Result:", result)
except ValueError:
    print("Invalid input! Not a number.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
except Exception as e:
    print("Unexpected error:", e)
```

3. What is the use of the `finally` block in Python? Give an example where `finally` is essential (e.g., closing a file or DB connection).

The finally keyword is used in try...except blocks. It defines a block of code to run when the try...except...else block is final. The finally block will be executed no matter if the try block raises an error or not.

```
try:
    f = open("sample.txt", "r")
    content = f.read()
    print(content)
except FileNotFoundError:
    print("File not found.")
finally:
    print("Closing file...")
    f.close()
```

4. Create a custom exception class `InvalidAgeError` and raise it if the age is less than 18.

```
class InvalidAgeError(Exception):
    def __init__(self, message):
        self.message = message
    def __str__(self):
        return f"InvalidAgeError: {self.message}"
try:
    age = int(input("Enter your age: "))
    if age < 18:
        raise InvalidAgeError("Age must be atleast 18 to continue.")
    else:
        print("Access granted.")
except InvalidAgeError as e:
    print(e)
```

5. What will the following code output?

```
  try:
      print(1 / 0)
  except ZeroDivisionError:
      print("Divided by zero")
  finally:
      print("Done")
```

Output :

Divided by zero

Done

6. Modify the program to retry 3 times if user enters an invalid number (handle `ValueError`). After 3 failures, exit the program.

```
attempts = 0
while attempts < 3:
    try:
        num = int(input("Enter a number: "))
        print("You entered:", num)
        break
    except ValueError:
        print("Invalid input. Try again.")
        attempts += 1
else:
    print("Too many attempts. Exiting.")
```

7. What is the difference between `raise` and `assert`? Give an example of each.

raise: Manually raise an exception.

Assert is used to check a condition is true, else it raises an AssertionError.

Example of raise:

raise ValueError("Invalid value!")

Example of assert:

age = 17

assert age >= 18, "Age must be 18 or above"

Regular Expressions – 8 Questions

8. Write a regex pattern to match:

- At least one uppercase letter
- At least one digit
- At least one special character from `@#$%&`
- Minimum 8 characters

```
import re
password = "Abcdef!@1234"
pattern = r'^(?=.*[A-Z])(?=.*\d)(?=.*[@#$%&]).{8,}$'
if re.match(pattern, password):
    print("Strong password")
else:
    print("Weak password")
```

9. Explain the difference between `re.match()` and `re.search()` with code examples.

#Example 1: Using re.match()

```
import re
text = "Hello world"
result = re.match("Hello", text)
if result:
    print("Match found:", result.group())
else:
    print("No match")
```

Example 2: Using re.search()

```
text = "Say Hello world"
result = re.search("Hello", text)
if result:
    print("Search found:", result.group())
else:
    print("No match")
```

10. Given a string: "Email me at test123@gmail.com or hr@openai.org"

Extract all email addresses using regex.

```
import re
text = "Email me at test123@gmail.com or hr@openai.org"
em = re.findall(r'\b[\w.%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b', text)
print(em)
```

11. Validate if a string is a valid Indian mobile number (10 digits starting with 6-9).

```
import re
number = "7876543210"
if re.match(r'^[6-9]\d{9}$', number):
   print("Valid mobile number")
else:
   print("Invalid")
```

12. What does the following pattern do? Explain in plain English.

```
r"^[A-Za-z0-9_]{3,15}$"
```

- ^ : start of string
- **[A-Za-z0-9_]** : any alphanumeric or underscore
- **{3,15}** : between 3 to 15 characters
- **$** : end of string

13. Extract all the hashtags from the text:

```
text = "I love #Python and #MachineLearning! #AI"
import re
text = "I love #Python and #MachineLearning! #AI"
hashtags = re.findall(r"#\w+", text)
print(hashtags)
```

O/p : ['#Python', '#MachineLearning', '#AI']

OR

```
text = "I love #Python and #MachineLearning! #AI"
hashtags = re.findall(r'#', text)
print(hashtags)
```

O/p : ['#', '#', '#']

14. What is the purpose of `re.match()`? Show how it improves performance when using the same pattern multiple times.

re.match() checks if the given pattern exists at the beginning of the string. If the pattern is not right at the start, it returns None.

```
import re

names = ["Dr Smith", "Dr John", "Mr Kumar", "Dr Meena", "Professor"]

for name in names:
    if re.match("Dr", name):
        print("Valid:", name)
    else:
        print("Invalid:", name)
```

15. Write a Python function to:

- Read a string from user input
- Validate if it is a strong password using regex
  - At least one uppercase letter
  - At least one lowercase letter
  - At least one number
  - At least one special character
  - At least 8 characters

```
import re

def validate_password():
    password = input("Enter password: ")
    pattern = r'^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[@#$%^&+=]).{8,}$'
    if re.match(pattern, password):
        print("Strong password")
    else:
        print("Weak password")
validate_password()
```