Regular expressions and Exception Handling

# Exception Handling – 7 Questions

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

except Exception as e: catches only exceptions derived from the Exception class and gives access to the exception object e.

except: catches all exceptions, including system-exiting exceptions like KeyboardInterrupt, which may not be desirable.

except Exception as e is preferred because it avoids masking system-level exceptions and gives more control.

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

A: try:

num = int(input("Enter a number: "))

result = 100 / num

print("Result:", result)

except ValueError:

print("Invalid input. Please enter a number.")

except ZeroDivisionError:

print("Cannot divide by zero.")

except Exception as e:

print("An unexpected error occurred:", e)

1. 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).

finally ensures cleanup happens, even if an exception occurs.

Example: try:

f = open("data.txt", "r")

content = f.read()

print(content)

except FileNotFoundError:

print("File not found")

finally:

print("File closed in finally block")

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

class InvalidAgeError(Exception):

pass

age = int(input("Enter your age: "))

if age < 18:

raise InvalidAgeError("Age must be 18 or above")

else:

print("You are eligible")

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

1. 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.")

attempts += 1

if attempts == 3:

print("Too many invalid attempts. Exiting.")

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

raise is used to manually raise exceptions and assert checks if a condition is True, otherwise raises AssertionError.

# 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

pattern = r'^(?=.\*[A-Z])(?=.\*\d)(?=.\*[@#$%&]).{8,}$'

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

Txt =”hello python”

match - checks only at the beginning

print(re.match("Hello", text))

search - checks anywhere in string

print(re.search("Python", text))

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"

emails = re.findall(r'\b[\w.-]+@[\w.-]+\.\w+\b', text)

print(emails)

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

import re

pattern = r'^[6-9]\d{9}$'

number = input("Enter mobile number: ")

if re.match(pattern, number):

print("Valid")

else:

print("Invalid")

12. What does the following pattern do?  
  
 r"^[A-Za-z0-9\_]{3,15}$"  
   
Explain in plain English.

It matches:

Start ^ and end $of a string 3 to 15 length of the characters .Letters (A–Z, a–z), numbers (0–9), and underscores (\_) is used for valid usernames or IDs

13. Extract all the hashtags from the text:  
   
 text = "I love #Python and #MachineLearning! #AI"

Find\_res = re.findall(“#”,text)

Print(find\_res)

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

re.match() checks only at the start of the string.

Example:

import re

pattern = re.compile(r'^\d{4}$') # e.g., 4-digit PIN

print(pattern.match("1234"))

print(pattern.match("abcd"))

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

def is\_valid\_passward(password):

pattern = r"^(?=.\*[A-Z])(?=.\*[0-9])(?=.\*[!@#&%]).{8,}$"

return bool(re.match(pattern, password))

test\_password = ["Abc123!","ABcd!3","A@werc435","ABCDE!345", "Abc123!@","Abcdghi1!"]

print("password validation result is \n")

for pwd in test\_password:

result = "valid" if is\_valid\_passward(pwd) else "invalid"

print(f"password : {pwd} : {result}")