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: it catches all expressions

Exception as e: catches only standard exception

Ex: try:

             name()

       except Exception as e

            print(“error”,e)

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(f"Result:{result}")

except ValueError as ve:

   print("Invalid input,plse enter correct number")

except ZeroDivisionError as zde:

   print("Number cannot be divided by zero",)

except Exception as ex:

   print("unexpected error",ex)

o/p:enter a number:25

Result:4.0

         enter a number:0

Number cannot be divided by zero

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).

    finally block runs under any condition,it is used for closing a file or db connection(for cleanup)

Ex:

try:

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

       file.write("good morning")

except Exception as e:

   print("Error",e)

finally:

   print("finally closed file")

o/p:finally closed file

4. 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("Your age should be less than 18")

else:

   print("Valid age,access granted")

o/p:Enter your age:14

Traceback (most recent call last):

  File "/Users/abhigna/PycharmProjects/PythonProject/day 7 python.py", line 61, in <module>

    raise InvalidAgeError("Your age should be less than 18")

\_\_main\_\_.InvalidAgeError: Your age should be less than 18

5. What will the following code output?  
  
   try:  
       print(1 / 0)  
   except ZeroDivisionError:  
       print("Divided by zero")  
   finally:  
       print("Done")

o/p:

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:

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

       print("your input number is:",n)

       break

   except ValueError:

       print("invalid input")

       attempts+=1

if attempts==3:

   print("Exit program, too many invalid attempts")

o/p:1.

Enter a number:2

your input number is: 2

2.

 Enter a number:ab

invalid input

Enter a number:2s

invalid input

Enter a number:w42

invalid input

Exit program, too many invalid attempts

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

 raise-> means it manually throws an exception.

assert-> is used for debugging and it raises  AssertionError if gvien condition is false.

Ex:

if num < 0:

    raise ValueError("num cannot be negative")

ex:

assert 2 + 2 == 4

assert 2 + 2 == 5  ->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)(?=.\*[@#$%&]).{8,}$'

password = input("Enter your password: ")

if re.match(pattern, password):

   print("Password is valid.")

else:

   print("Password is invalid")

o/p:Enter your password: abhi92

Password is invalid

Enter your password: Python8@hi

Password is valid.

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

re.match() ->checks if it matches and are similar,checks start of string.

re.search()-> search for any particular keyword or char in a string,searches complete string not just the start.

Ex:

txt= "My name is Abhigna,I am a BE graduate "

match\_result = re.match(r"abhi",txt)

print("1.match result:",match\_result.group() if match\_result else "no match")

search\_result = re.search(r"is",txt)

print("2.search result:",search\_result.group() if search\_result else "not found")

1.match result: no match

2.search result: is

10. Given a string: "Email me at test123@gmail.com or hr@openai.org"  
Extract all email addresses using regex.

text = "Email me at test123@gmail.com or hr@openai.org"

emails = re.findall(r"[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}", text)

print(emails)

o/p:['test123@gmail.com', 'hr@openai.org']

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

import re

def is\_valid\_mobile(number):

   pattern = r'^(?:\+91[\-\s]?|0)?[6-9]\d{9}$'

   if re.match(pattern, number):

       print("Valid Indian mobile number")

   else:

       print("Invalid mobile number")

is\_valid\_mobile("9876543210")

is\_valid\_mobile("233445")

o/p: Valid Indian mobile number

     Invalid mobile number

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

->it tells only letters, digits, or underscore (\_)

->Length: should be between 3 to 15

->there should be no special symbols, must be alphanumeric or underscore

->Matches the full string from start (^) to end ($)

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']

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

re.match()- is used to match the beginning character of a string.

import re

pattern = re.compile(r'\d{4}-\d{2}-\d{2}')

date1 = "2025-07-22:today's date."

m1 = re.match(pattern, date1)

date2 = "Today's date: 2025-07-22."

m2 = re.search(pattern, date2)

print("Match result:", m1.group() if m1 else "No match at beginning")

print("Search result:", m2.group() if m2 else "No match found")

o/p:Match result: 2025-07-22

Search result: 2025-07-22

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 is\_strong\_password(pwd):

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

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

password = input("Enter password: ")

if is\_strong\_password(password):

   print("it is a Strong password ")

else:

   print("it is a Weak password ")

o/p:Enter password: Abhi@123

      it is a Strong password