# DAY 7: MORNING ASSESSMENT

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

except Exception as e: Catches: Only exceptions that inherit from the built-in Exception class. Stores the exception in variable e, so you can inspect/log it. Considered safe and preferred for most use cases.

try:

x = 1 / 0

except Exception as e:

print("Caught:", e)

except: Catches: All exceptions, including critical system-exiting exceptions.Includes: KeyboardInterrupt, SystemExit, etc. Often used for a quick catch-all, but dangerous because it can hide bugs or prevent expected shutdown behavior.

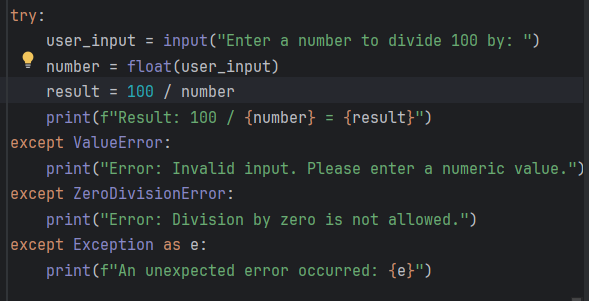
try:

x = 1 / 0

except:

print("Caught something, but not sure what.")

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



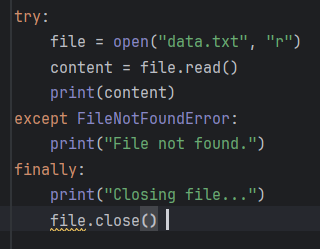
output:



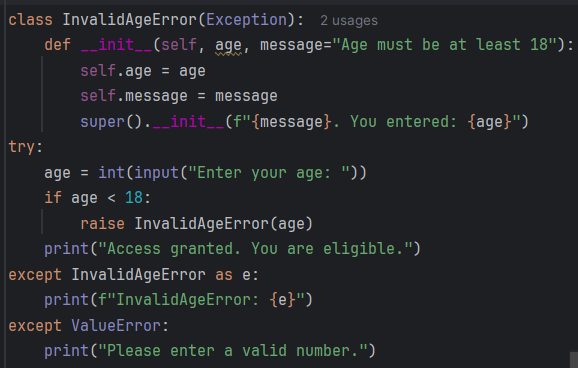
3. What is the use of the `finally` block in Python? Give an example where `finally` is essential (e.g., closing a file or DB connection).

The finally block is used to ensure that certain cleanup code is always executed, regardless of whether an exception occurred or not.

It runs after the try and except blocks. It always executes, even if: An exception is raised and not caught. A return, break, or continue is used in the try or except



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



Output:  



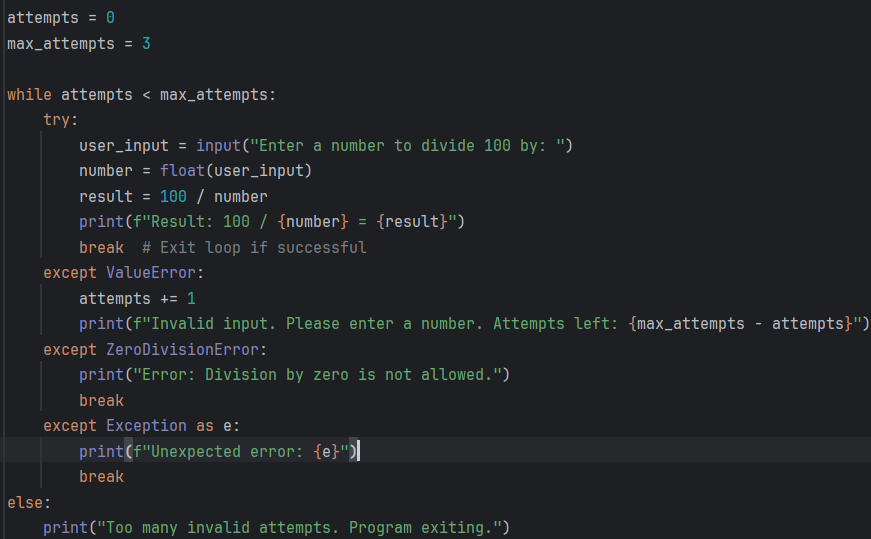

5. What will the following code output?  
 try:  
 print(1 / 0)  
 except ZeroDivisionError:  
 print("Divided by zero")  
 finally:  
 print("Done")

Output will be both the ZeroDivisionError and finally block

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.



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

RAISE: Use raise in production code for robust error handling.  
age = int(input("Enter your age: "))

if age < 18:

raise ValueError("You must be at least 18 years old.")

print("Access granted.")

ASSERT: Use assert fordebugging or testing internal assumptions; avoid it for user input validation.

x = -5

assert x >= 0, "x must be non-negative"

print("x is non-negative")

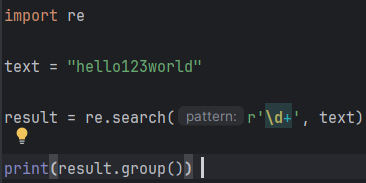
8. Write a regex pattern to match:  
 - At least one uppercase letter  
 - At least one digit  
 - At least one special character from `@#$%&`  
 - Minimum 8 characters

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

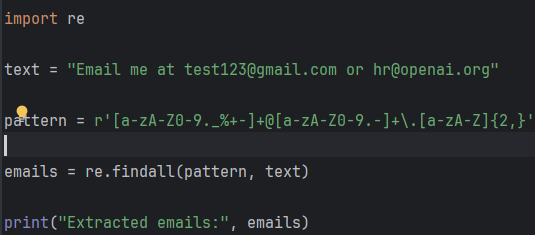
9. Explain the difference between `re.match()` and `re.search()` with code examples.

|  |  |
| --- | --- |
| match() | : Checks for a match only at the beginning of the string |
|  | Output: None |

|  |  |
| --- | --- |
| re.search() | : Checks for a match anywhere in the string |

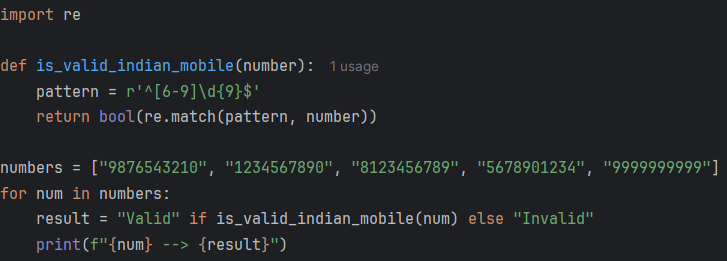
output: 123

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



Output: 

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

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


output:

9876543210 --> Valid

1234567890 --> Invalid

8123456789 --> Valid

5678901234 --> Invalid

9999999999 --> Valid

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

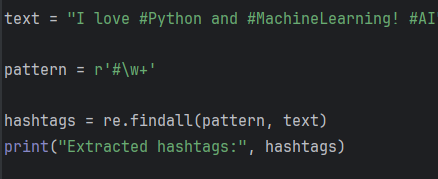
^ → Start of the string

[A-Za-z0-9\_] →   
Allowed characters: Uppercase letters A–Z, Lowercase letters a–z, Digits 0–9, Underscore \_

{3,15} → The string must be 3 characters to 15characters long

$ → End of the string

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



Output: Extracted hashtags: ['#Python', '#MachineLearning', '#AI']

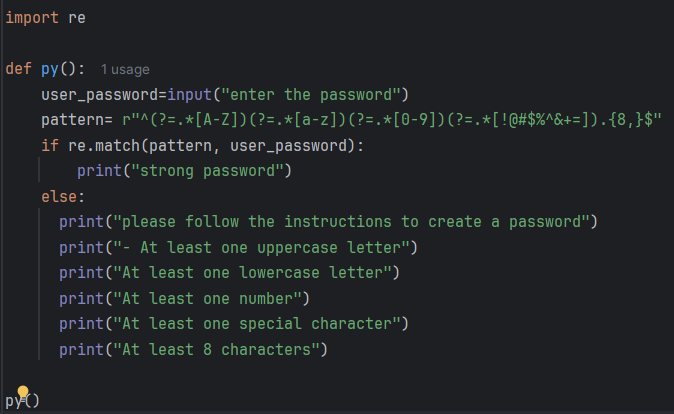
14. What is the purpose of `re.match()`? Show how it improves performance when using the same pattern multiple times.

The re.match() function in Python is used to check for a match only at the beginning of a string using a regular expression pattern.

re.match(pattern, string)

It returns: A match object if the beginning of the string matches else None

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



Output:   
enter the password: Shreya@135

strong password