1. What is the role of the 'else' block in a try-except statement? Provide an example

scenario where it would be useful.

A: it is executed if no exception occurs, for example:

try:

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

except ValueError:

print("Invalid input. Please enter a valid integer.")

else:

if age >= 18:

print("You are eligible to vote.")

else:

print("You are not eligible to vote yet.")

Output: Enter your age: 21

You are eligible to vote.

2. Can a try-except block be nested inside another try-except block? Explain with an

example.

A: yes, it can be nested and below is the example:

try:

num1 = int(input("Enter a number: "))

num2 = int(input("Enter a number: "))

try:

result = num1 / num2

print("Result:", result)

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

except ValueError:

print("Error: Invalid input. Please enter a valid number.")

3. How can you create a custom exception class in Python? Provide an example that

demonstrates its usage.

A: try:

n = int(input('Please enter your age: '))

if n<0:

raise exception("Your age cannt be less than 0") #custom exception

except ValueError:

print("Invalid input. Please enter a valid integer.")

else:

if n<18:

print('You cannot vote')

else:

print('you can vote')

4. What are some common exceptions that are built-in to Python?

A: ValueError, ZeroDivisionError, IndexError,SyntaxError, KeyError, FileNotFoundError

5. What is logging in Python, and why is it important in software development?

A: The logging module allows us to create log records with different levels of severity, such as debug, info, warning, error, and critical.

6. Explain the purpose of log levels in Python logging and provide examples of when

each log level would be appropriate.

A: the purpose of log records is to distinguish the type of event which are supposed to recorded and every log level has different severity. These log levels are:

* DEBUG: this is a lowest log level, and its purpose is to provide details during debugging.
* INFO: it gives general information. It provides certain information like you have successfully logged in.
* WARNING: It is used to show a warning message to the user. For example, ATM machine, if you enter incorrect PIN then machine will give a warning message like ‘2 tries left’.
* ERROR: It gives you an error message like, ‘You have entered password’.
* CRITICAL: this is the highest level of logging for which upon ignoring, system may damage

7. What are log formatters in Python logging, and how can you customize the log

message format using formatters?

A: Log formatters are used to set a format for log messages. We can customize the log messages by using these formatters.

import logging

logging.basicConfig(format='%(process)d-%(levelname)s-%(message)s')

logging.warning('This is a Warning')

Output:

46353-WARNING-This is a Warning

8. How can you set up logging to capture log messages from multiple modules or

classes in a Python application?

A: by importing logging package and using below statement.

Logging.basicConfig(level=logging.warning)

9. What is the difference between the logging and print statements in Python? When

should you use logging over print statements in a real-world application?

A: print statement gives you a output message and its sole purpose is to display the message written within the paranthesis and also does not give any severity. On the other hand logging is used to display a warning, error and information about the application.

For example: when you login to facebook, if you enter incorrect password, an error message will pop up that you have entered incorrect password, that’s logging.

10. Write a Python program that logs a message to a file named "app.log" with the

following requirements:

● The log message should be "Hello, World!"

● The log level should be set to "INFO."

● The log file should append new log entries without overwriting previous ones.

A: import logging

logging.basicConfig(level=logging.INFO,filename="app.log",filemode="a",

format="%(asctime)s - %(levelname)s - %(message)s")

logging.info("Hello, World!")

11. Create a Python program that logs an error message to the console and a file named

"errors.log" if an exception occurs during the program's execution. The error

message should include the exception type and a timestamp.

A: import logging

logging.basicConfig(

level=logging.ERROR,

format="%(levelname)s - %(message)s",

filename="errors.log",

filemode="a"

)

try:

name = "Navtej"

age = int(name)

except Exception as e:

error\_message = f"Exception of type {type(e).\_\_name\_\_} occurred at {time.strftime('%Y-%m-%d %H:%M:%S')}"

logging.error(error\_message, exc\_info=True)