1. What is the role of the 'else' block in a try-except statement? Provide an example scenario where it would be useful.

Ans1. ‘else’ block will execute when there is no exception in the code.

try:

print('This is try block')

except:

print("This is exception block")

else:

print("This is else part")

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

Ans2. Yes, try-except block can be nested inside try-except block.

try:

print('This is try block')

try:

print('One level nested try block')

except:

print('One level nested except block')

except:

print('this is except block')

1. How can you create a custom exception class in Python? Provide an example that demonstrates its usage.

Ans3. We need to inherit Exception class in custom class first.

class custom\_exception(Exception):

def \_\_init\_\_(self,salary,message='Salary not in range between 5000 to 15000'):

self.salary=salary

self.message=message

super().\_\_init\_\_(message)

salary = int(input("Enter salary amount: "))

if not 5000 < salary < 15000:

raise custom\_exception(salary)

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

Ans4. There are few common exceptions mentioned here:

ValueError

IndexError

AttributeError

ZeroDivisionError

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

Ans5. Logging is for tracking events in software development cycle. Logging applied on many levels; debug, info, warning, error, critical.

1. Explain the purpose of log levels in Python logging and provide examples of when each log level would be appropriate.

Ans6.

Debug: This log is used to give detailed information.

Info: These are used to confirm that things are working as expected.

Warning: These are used when something unexpected happened.

Error: This tells that due to a more serious problem.

Critical: This tells serious error, indicating that the program itself may be unable to continue running.

1. What are log formatters in Python logging, and how can you customize the log message format using formatters?

Ans7. Formatter objects to format a log record into a string-based log entry.

formatter = logging.Formatter('%(created)f:%(levelname)s:%(name)s:%(module)s:%(message)s')

1. How can you set up logging to capture log messages from multiple modules or classes in a Python application?

Ans8. Can create a log file separately and method inside it and according to use we can call the same method where log is required. We can inherit the log file.

1. 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?

Ans9.

Logging:

* Record the events and error that occur at the run time
* There are different level logging which help to record the error or events
* Mainly used in production environment

Print:

* Display the events at the run time
* There is no level of print the message on display
* Maily used for debugging