**Exception Handling**

**Q1. How do you handle exceptions in Python?**  
Using try, except, finally, else.

**Q2. Difference between Error and Exception in Python?**  
Unlike Java, everything is an Exception. Names ending with Error are still subclasses of Exception.

**Q3. Can we create custom exceptions?**  
Yes, by subclassing Exception.

**Q4. What happens if an exception is not handled?**

Program terminates and prints traceback.

**Q5. What is the use of finally block?**  
 Executes regardless of whether an exception occurred. Useful for cleanup.

**Q6. What is an exception?**  
Runtime error that disrupts normal program flow.

**Q7. Difference between syntax error and runtime error?**  
Syntax error detected before execution; runtime error occurs during execution.

**Q8. What happens if you don’t handle an exception?**  
Program stops and traceback is printed.

**Q9. How do you catch multiple exceptions?**

try:

...

except (ValueError, TypeError) as e:

...

**Q10. What is the role of else in try-except?**  
Executes only if no exception occurs.

**Q11. Difference between assert and raising exceptions?**  
assert is for debugging, raise is for explicit exception handling.

**Q12. What is raise keyword used for?**  
To manually raise exceptions.

**Q13. Can Python exceptions be user-defined?**  
Yes, subclass Exception.

**Q14. How do you re-raise the last exception?**  
raise without arguments inside except block.

**Q15. What is the base class for all exceptions?**  
BaseException.

**Q16. Which exceptions inherit directly from BaseException?**  
SystemExit, KeyboardInterrupt, GeneratorExit.