Python Basics

1. What are mutable and immutable types in Python? Give examples.

2. Explain the difference between `is` and `==`.

3. Whats the difference between `a = 5` and `a == 5`?

4. How does Python handle dynamic typing?

5. Whats the output of `bool("False")` and why?

6. What are Pythons truthy and falsy values?

7. Write a Python program to find if a number is prime.

8. Explain `pass`, `continue`, and `break`.

9. What is the scope of a variable?

10. How do you take input from the user and convert it to float?

11. Explain indentation and its role in Python.

12. What are list comprehensions? Give an example.

13. What are ternary operators in Python?

14. What does `if __name__ == "__main__"` mean?

15. Whats the difference between `del`, `remove`, and `pop()`?

Data Structures

1. How is a list different from a tuple?

2. What are the time complexities for accessing, inserting, deleting in lists?

3. How do you remove duplicates from a list?

4. Write code to flatten a nested list.

5. What is the use of `collections.Counter`?

6. Compare sets and dictionaries in terms of use and performance.

7. How do you sort a dictionary by its values?

8. How do you merge two dictionaries in Python 3.9+?

9. When should you use a tuple instead of a list?

10. Explain list slicing with an example.

11. How to implement a queue and stack using `deque`?

12. Write a function to find the most common element in a list.

13. How can you invert a dictionary?

14. How do you implement a priority queue?

15. What are frozensets and when are they used?

Functions and Modules

1. Difference between `*args` and `**kwargs`?

2. How does Python handle default arguments?

3. What are lambda functions and where should you use them?

4. What is the use of `functools.lru_cache`?

5. Can a function return multiple values?

6. What are decorators and how do they work?

7. Write a recursive function for factorial.

8. Explain the scope of a variable inside a function.

9. What is a closure? How is it used?

10. What is the difference between `global` and `nonlocal`?

11. How do you import a module from another directory?

12. Explain function annotations.

13. What are the built-in functions that help with introspection?

14. How does Python implement first-class functions?

15. How do you dynamically import a module?

Object-Oriented Programming (OOP)

1. Explain the difference between a class and an object.

2. What are `__init__`, `__str__`, and `__repr__`?

3. What is method overloading and does Python support it?

4. Explain inheritance with an example.

5. Whats the difference between classmethod and staticmethod?

6. How is encapsulation implemented in Python?

7. What is polymorphism in Python?

8. What is the MRO (Method Resolution Order)?

9. What is multiple inheritance? Show an example.

10. What is the purpose of `super()`?

11. How do you make a class iterable?

12. How is operator overloading done?

13. What are abstract classes and how are they defined?

14. Difference between `isinstance()` and `issubclass()`?

15. What are metaclasses in Python?

Error Handling and Debugging

1. What is the difference between `Exception` and `BaseException`?

2. How do you catch multiple exceptions?

3. What is the finally block used for?

4. Write a custom exception class.

5. What does the `raise` keyword do?

6. How to suppress exceptions in Python?

7. What is a traceback?

8. What is `assert` and when is it used?

9. How to log exceptions using the `logging` module?

10. What is the difference between syntax error and runtime error?

11. How does `pdb` work for debugging?

12. What are the best practices for exception handling in production ML code?

13. Whats the purpose of `try-except-else-finally`?

14. How can you log an exception with stack trace?

15. How do you handle errors in external libraries (like Pandas or NumPy)?

NumPy

1. Difference between Python list and NumPy array?

2. What is broadcasting in NumPy?

3. How to generate a 3x3 identity matrix?

4. What is the difference between `np.array()` and `np.asarray()`?

5. How do you index multi-dimensional arrays?

6. How to compute column-wise mean of a 2D array?

7. How to reshape a 1D array into 2D?

8. What is the use of `np.where()`?

9. How can you find NaN values in an array?

10. How to create a random array with a fixed seed?

11. What are the differences between `np.sum()` and Pythons `sum()`?

12. How to normalize a NumPy array?

13. How to concatenate two arrays?

14. Whats the difference between `np.copy()` and shallow copy?

15. Whats the use of `np.dot()` vs `np.matmul()`?

Pandas

1. How do you read a CSV into a DataFrame?

2. Difference between `loc[]` and `iloc[]`?

3. How do you handle missing values in Pandas?

4. How to filter rows based on multiple conditions?

5. How to group data and calculate aggregates?

6. What is `pivot_table()` used for?

7. How to merge DataFrames like SQL joins?

8. How do you handle datetime features in Pandas?

9. What is the difference between `apply()`, `map()`, and `applymap()`?

10. How do you sort a DataFrame by multiple columns?

11. How do you check for duplicates and drop them?

12. How to get summary statistics of a DataFrame?

13. How to rename columns and index?

14. How do you handle categorical variables using Pandas?

15. How to export a DataFrame to an Excel file?

Iterators and Generators

1. Whats the difference between iterable and iterator?

2. What is the purpose of `__iter__()` and `__next__()`?

3. Write a generator to yield Fibonacci numbers.

4. What is the benefit of using generators in ML pipelines?

5. Difference between generator expressions and list comprehensions?

6. How to manually iterate over a generator?

7. How does `StopIteration` work?

8. Whats the use of `yield from`?

9. How do you create an infinite generator?

10. Can you return values from a generator?

11. How does a generator maintain its state?

12. What is lazy evaluation and how do generators help?

13. How to create a custom iterable class?

14. What is the `send()` method used for in generators?

15. Compare memory usage between a list and a generator.