**Integer & Float Mix**

* Create an integer a and a float b.
* Perform addition, subtraction, multiplication, and division on them.
* Print the results and observe the type of each result with type().

**Large Integers & Type**

* Assign a very large integer to a variable (e.g., in the billions).
* Print it and confirm its type is still int in Python or not.

**Complex Number Basics**

* Create a complex number z = 3 + 4j.
* Print its real part, imaginary part, and confirm its type is complex.
* Perform a basic arithmetic operation with another complex number (e.g., (3 + 4j) + (1 + 2j)).

**Boolean from Comparisons**

* Create two variables, m = 10 and n = 15.
* Define status = (m > n) and print status.
* Confirm type(status) is bool.
* Assign status = (m != n) and print again.

**String Creation & Indexing**

* Create a string text = "HelloWorld".
* Print the first and last characters using positive and negative indexing.
* Comment on the total length of the string.

**String Slicing**

* With lang = "PythonProgramming", print the substring from index 2 to 8.
* Print the substring from the start up to index 5.
* Print the entire string in reverse using slicing.

**String Methods**

* Let phrase = " Hello, Python World! ".
* Demonstrate strip(), upper(), and replace() on this string.
* Print the results and comment on immutability of strings in Python.

**String Formatting**

* Create two variables: name = "Rajesh" and score = 95.
* Print: "Alice scored 95 points." using two methods (concatenation and an f-string or str.format()).

**Boolean Operations in Expressions**

* Write a small expression using and, or, and not with boolean values.
* Example: result = not(True and False) or (5 > 3).
* Print result and explain how Python evaluated the expression.

**List Creation & Access**

* Create a list of 5 different integers.
* Print the first item, middle item, and last item using indexing.

**List Insertion & Deletion**

* Start with a list nums = [10, 20, 30, 40].
* Insert 25 at index 2.
* Remove the last element.
* Print the updated list.

**List Slicing**

* Given letters = ["a", "b", "c", "d", "e"], print the slice that contains only ["b", "c", "d"].
* Print the slice that omits the first and the last element.

**Sorting & Reversing**

* Create a list of random integers.
* Sort the list in ascending order and print it.
* Reverse the sorted list and print again.

**Combining Lists**

* Let list1 = [1, 2, 3] and list2 = [4, 5, 6].
* Combine them into a single list and print.
* Demonstrate two ways: using + and using .extend().

**Aggregating List Values**

* Create a list of floats, e.g., [2.5, 3.6, 1.2, 5.0].
* Print the sum, minimum, and maximum of that list using built-in functions.

**Tuple Creation**

* Create a tuple t = (10, 20, "Hello", True).
* Print the tuple and confirm its type with type(t).

**Tuple Indexing & Slicing**

* Print the first two elements of t using slicing.
* Print the last element of t using negative indexing.

**Tuple Unpacking**

* Suppose t2 = ("Tom", 25, "Engineer").
* Unpack it into three separate variables: name, age, profession.
* Print these variables individually.

**Attempt Tuple Mutation**

* Try to change an element of t (t[0] = 999) and observe the error.
* In comments, explain why the error occurs.

**Set Creation & Membership**

* Create a set my\_set = {1, 3, 5, 7}.
* Check if 5 is in my\_set.
* Check if 2 is not in my\_set.

**Add & Remove Elements**

* Add 9 to my\_set.
* Remove 3 from my\_set.
* Print the updated set.

**Set Operations**

* Create two sets: setA = {1, 2, 3} and setB = {3, 4, 5}.
* Print the union, intersection, and difference (setA - setB).

**Check Unique Values**

* Define a list vals = [1, 2, 2, 3, 3, 3, 4].
* Convert it to a set.
* Print both the list and the set to show how duplicates are removed.

**Frozenset Creation**

* Create a frozenset from a list [2, 4, 4, 6].
* Print it and observe whether duplicates are preserved.

**Immutability Demonstration**

* Attempt to add an element to your frozenset.
* Observe and explain the error in comments.

**Frozenset Use as a Dictionary Key**

* Create a dictionary where the key is your frozenset, e.g., my\_dict = {frozenset([1,2,3]): "value"}.
* Print my\_dict and explain why a normal set cannot be used as a key.

**Dictionary Creation & Access**

* Create a dictionary student = {"name": "Ramesh", "age": 20, "grade": "A"}.
* Print the value of "name" and "age" individually.

**Adding & Updating Keys**

* Add a new key-value pair: "city": "Kathmandu" to student.
* Update "age" to 21.
* Print the dictionary.

**Removing Keys**

* Remove the "grade" key from student.
* Print the resulting dictionary.

**Nested Dictionary Example**

* Create a dictionary record = {"id": 101, "info": {"name": "Bob", "dept": "IT"}}.
* Print the department of Bob.

**Operator Precedence**

* Define a = 4, b = 2, c = 5.
* Print the result of a + b \* c vs. (a + b) \* c.
* Explain in comments how the result differs.

**Modulo & Floor Division**

* Let x = 17 and y = 4.
* Print x % y and x // y.
* Explain the difference between these two operators in comments.

**Power Operator**

* Print the result of 2 \*\* 3.
* Write a line to calculate 3 \*\* 4.
* Print the addition of both.

**String Comparison**

* Compare "apple" and "banana" with <, >, and ==.
* Print the results.

**Mixed Type Comparison**

* Compare 5 and 5.0 with ==.
* Compare 5 and 5.0 with is.
* Discuss the results in comment.

**Chain Comparisons**

* Evaluate the expression 2 < 3 < 5.
* Print the result and explain how Python handles chained comparisons.

**Logical and**

* Define p = True and q = False.
* Print p and q.
* Demonstrate a real-world example: (age > 18) and (has\_ID == True).

**Logical or**

* Using the same p and q, print p or q.

**Logical not**

* Let r = (10 > 5).
* Print r, then print not r.

**Using len()**

* Create a list with 7 elements.
* Use len() to get the total count.
* Print the result.

**Using type()**

* For each of the following: 10, 10.5, "ten", True, 3+2j, print type(...).
* Summarize in comments the data types you observed.

**Using abs()**

* Print abs(10), abs(-10), and abs(-3.5).
* Explain what abs() does in comments.

**Using round()**

* Demonstrate round(3.14159, 2).
* Show how round(2.5) behaves in Python.

**Using sum(), max(), min()**

* Create a list of numeric values.
* Print sum(), max(), and min() of that list.

**Using sorted()**

* Create a list or tuple vals = (3, 1, 4, 2).
* Use sorted(vals) and print the result.
* Show that vals itself is unchanged.

**Using any() / all()**

* Create a list of booleans, for example [True, False, True].
* Print any() on the list, then all() on the list.
* Show the difference in how they evaluate.

**Storing Booleans from Comparisons**

* Compare two values in separate expressions, e.g., a = (10 > 3), b = (5 == 5).
* Combine these booleans with and or or.
* Print the final result.

**Multiline String & Counting**

* Create a multiline string describing your favorite hobby.
* Use the string method .count(substring) to count how many times the letter "a" appears (case-insensitive).
* Print the count and explain any steps taken to handle case sensitivity.

**Advanced String Slicing**

* Take the string "ABCDEFGHIJ" and slice every second character, resulting in "ACEGI".
* Print the sliced string.
* Also slice in reverse step.

**Casefold vs. Lower**

* Create two strings, "Case" and "case".
* Compare them with the regular == operator directly.
* Compare them again after applying .casefold().
* Print results and comment on how .casefold() differs from .lower() in edge cases.

**Formatted Printing**

* Define name = "Ramesh", product = "Notebook", quantity = 2, and price = 12.50.
* Use an f-string (or str.format()) to print:  
  "Ramesh purchased 2 Notebook for a total of $25.0."

**Type Conversion Chain**

* Ask a user for a string that represents a number, e.g., "0".
* Convert it to a float, then to a bool, and print the intermediate and final results.

**String List Sorting**

* Given fruits = ["apple", "banana", "cherry"], sort them in descending alphabetical order.
* Print the sorted list, then use the .reverse() method to flip it. Compare the two results.

**Insert Using Slicing**

* Start with a list [2, 5, 7, 1, 9].
* Insert the value 4 right after the 2 **using slicing** (not insert() or append()).
* Print the modified list.

**Indexing Within a Mixed List**

* Create a list info = ["John", 28, True, 4500.75].
* Print only "John" and 4500.75 using their indices.

**Tuple Concatenation & Replication**

* Create two tuples (1, 2) and (3, 4).
* Concatenate them into (1, 2, 3, 4).
* Replicate (1, 2) 3 times to get (1, 2, 1, 2, 1, 2).

**Single-Element Tuple**

* Demonstrate that (42,) is a tuple whereas (42) is just an integer.

**Intersection & Union**

* Let setA = {1, 2, 3, 4} and setB = {1,2,3}.
* Print their intersection using setA & setB.
* Print their union using setA | setB.

**Subset and Superset**

* Check if setB is a subset of setA using setB.issubset(setA).
* Check if setA is a superset of setB using setA.issuperset(setB).
* Print the results.

**Countries and Capitals**

* Create a dictionary capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}.
* Print just the keys, then just the values, then all key-value pairs.

**Summing Dictionary Values**

* Create a dictionary numbers = {"x": 4, "y": 10, "z": 6}.
* Use sum(numbers.values()) to sum all values.
* Print the sum.