### \*\*1. Strings\*\*

#### \*\*Q1:\*\* How to reverse a string without using built-in functions?

\*\*Answer:\*\*

```python

s = "hello"

reversed\_str = s[::-1] # Slicing trick

print(reversed\_str) # Output: "olleh"

```

#### \*\*Q2:\*\* How to count the occurrences of each character in a string?

\*\*Answer:\*\*

```python

s = "programming"

char\_count = {}

for char in s:

char\_count[char] = char\_count.get(char, 0) + 1

print(char\_count) # Output: {'p': 1, 'r': 2, 'o': 1, ...}

```

#### \*\*Q3:\*\* Check if a string is a palindrome (case-insensitive).

\*\*Answer:\*\*

```python

def is\_palindrome(s):

s = s.lower().replace(" ", "")

return s == s[::-1]

print(is\_palindrome("Madam")) # Output: True

```

---

#### \*\*Q6:\*\* Merge two lists into a list of tuples (e.g., `[1, 2]` and `["a", "b"]` → `[(1, "a"), (2, "b")]`).

\*\*Answer:\*\*

```python

list1 = [1, 2]

list2 = ["a", "b"]

merged = list(zip(list1, list2))

print(merged) # Output: [(1, 'a'), (2, 'b')]

```

---

### \*\*3. Tuples\*\*

#### \*\*Q7:\*\* Why are tuples faster than lists? Demonstrate with an example.

\*\*Answer:\*\* Tuples are immutable, so Python optimizes their storage.

```python

import timeit

print(timeit.timeit('(1,2,3,4)', number=1000000)) # Faster

print(timeit.timeit('[1,2,3,4]', number=1000000)) # Slower

```

#### \*\*Q8:\*\* Swap two tuple elements (e.g., `(10, 20)` → `(20, 10)`).

\*\*Answer:\*\*

```python

t = (10, 20)

swapped = t[::-1] # Slicing

print(swapped) # Output: (20, 10)

```

---

### \*\*4. Dictionaries\*\*

#### \*\*Q9:\*\* Combine two dictionaries (`dict1` and `dict2`) into one.

\*\*Answer:\*\*

```python

dict1 = {"a": 1, "b": 2}

dict2 = {"c": 3, "d": 4}

combined = {\*\*dict1, \*\*dict2} # Python 3.5+

print(combined) # Output: {'a': 1, 'b': 2, 'c': 3, 'd': 4}

```

#### \*\*Q10:\*\* Get the key with the maximum value in a dictionary.

\*\*Answer:\*\*

```python

scores = {"Alice": 90, "Bob": 85, "Charlie": 95}

max\_key = max(scores, key=scores.get)

print(max\_key) # Output: "Charlie"

```

---

### \*\*5. Mixed Problems\*\*

#### \*\*Q12:\*\* Check if all items in a list are unique.

\*\*Answer:\*\*

```python

def all\_unique(lst):

return len(lst) == len(set(lst))

print(all\_unique([1, 2, 2, 3])) # Output: False

```

#### \*\*Q14:\*\* Convert a list of tuples to a dictionary (e.g., `[("a", 1), ("b", 2)]` → `{"a": 1, "b": 2}`).

\*\*Answer:\*\*

```python

tuple\_list = [("a", 1), ("b", 2)]

dict\_result = dict(tuple\_list)

print(dict\_result) # Output: {'a': 1, 'b': 2}

```

#### \*\*Q19:\*\* What is the output of `bool("False")` and why?

\*\*Answer:\*\* `True` (non-empty strings are `True` in Python).

```python

print(bool("False")) # Output: True

```

#### \*\*Q20:\*\* Explain the difference between `list.sort()` and `sorted(list)`.

\*\*Answer:\*\*

- `list.sort()` modifies the original list and returns `None`.

- `sorted(list)` returns a new sorted list without modifying the original.

```python

lst = [3, 1, 2]

lst.sort()

print(lst) # Output: [1, 2, 3] (original modified)

lst = [3, 1, 2]

new\_lst = sorted(lst)

print(lst) # Output: [3, 1, 2] (original unchanged)

print(new\_lst) # Output: [1, 2, 3]

```

---

Here are detailed answers and code solutions for all the questions:

---

### \*\*1. Introduction to Python\*\*

#### \*\*Q1:\*\* Python is a high-level, interpreted programming language known for its readability and versatility. It's called a general-purpose language because it can be used for:

- Web development (Django, Flask)

- Data science (Pandas, NumPy)

- AI/ML (TensorFlow, PyTorch)

- Automation, scripting, and more.

#### \*\*Q2:\*\* 5 Key Features of Python:

1. \*\*Easy to Learn\*\*: Simple syntax (closer to English).

2. \*\*Interpreted\*\*: Executes line-by-line (no compilation needed).

3. \*\*Cross-Platform\*\*: Works on Windows, Linux, Mac.

4. \*\*Dynamically Typed\*\*: No need to declare variable types.

5. \*\*Large Standard Library\*\*: Built-in modules for files, OS, etc.

#### \*\*Q3:\*\* Limitations of Python:

- \*\*Slower\*\* than C/C++ (interpreted nature).

- \*\*Not ideal for mobile development\*\* (weak in Android/iOS).

- \*\*Memory consumption\*\* can be high.

---

### \*\*2. Python Installation and IDEs\*\*

#### \*\*Q4:\*\* \*\*Anaconda Navigator\*\* is a GUI for managing Python environments and packages, especially for data science (comes with pre-installed libraries like NumPy, Pandas).

#### \*\*Q5:\*\* Python IDEs:

1. \*\*Spyder\*\*: Great for scientific computing (built-in variable explorer).

2. \*\*Jupyter\*\*: Interactive notebooks (good for data visualization).

3. \*\*PyCharm\*\*: Full-featured IDE with debugging support.

#### \*\*Q6:\*\* \*\*Interactive Mode\*\* (REPL):

- Execute one line at a time (e.g., `>>> 2+2`).

- \*\*Script Mode\*\*:

- Write code in a `.py` file and run it entirely.

---

### \*\*3. Python Basics\*\*

#### \*\*Q7:\*\* Valid identifiers:

- Valid: `myVar`, `\_hello`

- Invalid: `2ndVar` (starts with number), `break` (keyword), `$money` (special char), `for` (keyword).

#### \*\*Q8:\*\* 5 Python Keywords:

1. `if/else`: Conditional logic.

2. `for`: Looping.

3. `def`: Function definition.

4. `import`: Load modules.

5. `try/except`: Error handling.

#### \*\*Q9:\*\* Data Type Categorization:

| Data Type | Ordered | Sequential |

|-----------|---------|------------|

| `list` | Yes | Yes |

| `tuple` | Yes | Yes |

| `dict` | No | No |

| `str` | Yes | Yes |

| `set` | No | No |

---

### \*\*4. Strings\*\*

#### \*\*Q10:\*\* String Operations:

```python

s = "Python Programming"

print(s[7:]) # Output: "Programming"

print(s.upper()) # Output: "PYTHON PROGRAMMING"

print(s.replace("Python", "C++")) # Output: "C++ Programming"

print(s.startswith("Py")) # Output: True

```

#### \*\*Q11:\*\* Output of `"hello".upper().islower()`:

```python

print("hello".upper().islower()) # Output: False

# Explanation: "hello".upper() → "HELLO" → islower() checks if all lowercase.

```

#### \*\*Q12:\*\* Count Vowels:

```python

s = "Hello World"

vowels = "aeiouAEIOU"

count = sum(1 for char in s if char in vowels)

print(count) # Output: 3

```

---

### \*\*5. Lists, Tuples, Dictionaries\*\*

#### \*\*Q13:\*\* List Operations:

```python

my\_list = [1, 2, 3, 4, 5]

my\_list.reverse() # Reverse in-place → [5, 4, 3, 2, 1]

my\_list.append(6) # Add 6 → [5, 4, 3, 2, 1, 6]

my\_list.pop(1) # Remove 2nd element (4) → [5, 3, 2, 1, 6]

print(3 in my\_list) # Output: True

```

#### \*\*Q14:\*\* List vs. Tuple:

- \*\*List\*\*: Mutable (can modify after creation).

```python

fruits = ["apple", "banana"]

fruits[0] = "orange" # Valid

```

- \*\*Tuple\*\*: Immutable (fixed data, faster).

```python

colors = ("red", "green")

colors[0] = "blue" # Error! Use for constants like days of the week.

```

#### \*\*Q15:\*\* Dictionary Operations:

```python

my\_dict = {"name": "Alice", "age": 25}

my\_dict["city"] = "New York" # Add key-value

my\_dict["age"] = 26 # Update value

print(my\_dict.keys()) # Output: dict\_keys(['name', 'age', 'city'])

print(my\_dict.values()) # Output: dict\_values(['Alice', 26, 'New York'])

print("country" in my\_dict) # Output: False

```

---

### \*\*Coding Challenges\*\*

#### \*\*1. String Palindrome:\*\*

```python

def is\_palindrome(s):

s = s.lower().replace(" ", "")

return s == s[::-1]

print(is\_palindrome("madam")) # Output: True

```

#### \*\*2. Merge Lists & Remove Duplicates:\*\*

```python

list1 = [1, 2, 3]

list2 = [3, 4, 5]

merged = list(set(list1 + list2))

print(merged) # Output: [1, 2, 3, 4, 5]

```

#### \*\*3. Dictionary Average Marks:\*\*

```python

students = {"Alice": 90, "Bob": 85, "Charlie": 95}

average = sum(students.values()) / len(students)

print(average) # Output: 90.0

```

---