DATA TYPES

List:-

1. What is the difference between append() and extend() in a list?

- append() adds **one element** to the end of the list.
- extend() adds **multiple elements** from another list (or iterable).

Example:

```
a = [1, 2]
a.append([3, 4])  # [1, 2, [3, 4]]
a.extend([5, 6])  # [1, 2, [3, 4], 5, 6]
```

---

2. How is insert() different from append()?

- append() adds to the **end** of the list.
- insert(index, value) adds an item at a **specific position**.

Example:

```
a = [10, 20]
a.insert(1, 15)  # [10, 15, 20]
```

---

3. What does pop() do in a list, and what happens if no index is passed?

- pop() removes and returns the element at a given index.
- If no index is passed, it removes the **last element**.

Example:

```
a = [1, 2, 3]
a.pop()    # returns 3, list becomes [1, 2]
a.pop(0)   # returns 1, list becomes [2]
```

---

4. How do you remove all elements from a list?

Use the clear() method.

Example:

```
a = [1, 2, 3]
```

```
a.clear()  # []
```

---

5. What is the difference between shallow copy and deep copy for lists?

- A **shallow copy** creates a new list but does **not copy inner objects** (they are shared).
- A **deep copy** creates a new list and **copies all inner objects too**.

Example:

```
import copy
original = [[1], [2]]
shallow = original.copy()
deep = copy.deepcopy(original)
```

---

6. Can a list contain duplicate values? How does Python handle them?

Yes, lists can contain duplicates. Python stores them as-is and maintains the **insertion order**.

Example:

```
a = [1, 2, 2, 3]
# duplicates are allowed
```

---

7. How does list slicing work, and can you explain `[::2]`?

Slicing: `list[start:stop:step]`

- `start` → where to begin (default is 0)
- `stop` → where to stop (not inclusive)
- `step` → how many items to skip

`[::2]` returns every second element from the list.

Example:

```
a = [0, 1, 2, 3, 4]
print(a[::2])  # [0, 2, 4]
```

String:-

8. Are strings mutable or immutable in Python? Why is this important?
Strings are **immutable**, meaning they **cannot be changed after creation**.
This is important for memory efficiency and security.

Example:

```
s = "hello"
# s[0] = 'H'  ❌ (not allowed)
```

---

9. What is the difference between strip(), rstrip(), and lstrip()?

- strip() → removes spaces from **both ends**
- rstrip() → removes spaces from **right end only**
- lstrip() → removes spaces from **left end only**

Example:

```
s = "  hello  "
s.strip()   # 'hello'
s.rstrip()  # '  hello'
s.lstrip()  # 'hello  '
```

---

10. How does string slicing work in Python?
Same as list slicing: string[start:stop:step]
You can extract substrings using this method.

Example:

```
s = "Python"
print(s[1:4])  # "yth"
```

---

11. What is the difference between find() and index() in strings?
Both return the position of a substring.

- find() → returns -1 if not found
- index() → raises an error if not found

Example:

```
s = "apple"
s.find('p')    # 1
s.index('x')   # Error
```

---

12. How do you check if a substring exists in a string?
Use the `in` operator.

Example:

```
"py" in "python"  # True
```

---

13. Can strings be looped through using `for` loop? What does it return?
Yes, you can loop through strings. It returns **each character one by one**.

Example:

```
for c in "abc":
    print(c)
# Output: a b c
```

---

14. What does `"Python".replace('P', 'R')` return?
It replaces `'P'` with `'J'`.

Output:

```
"Rython"
```

Dictionary:-

15. How do dictionaries store data in Python?
Dictionaries store data as **key-value pairs** using `{}`.

Example:

```
person = {'name': 'John', 'age': 30}
```

16. What happens if you try to access a key that doesn't exist in a dictionary?
Using dict['key'] will raise a KeyError.

Example:

```
d = {'a': 1}
print(d['b'])  # KeyError
```

---

17. What is the difference between dict.get('key') and dict['key']?

- dict['key'] → raises error if key doesn't exist
- dict.get('key') → returns None (or a default value)

Example:

```
d = {'x': 10}
print(d.get('y'))     # None
print(d['y'])         # Error
```

---

18. How can you add a new key-value pair to a dictionary?
Assign the value directly using square brackets.

Example:

```
d = {}
d['name'] = 'Alice'
```

---

19. Can a dictionary have duplicate keys?
No. If you assign the same key again, the **last value** will overwrite the previous one.

Example:

```
d = {'a': 1, 'a': 2}  # {'a': 2}
```

---

20. What does the items() method return in a dictionary?
Returns a list of tuples – each tuple is a (key, value) pair.

Example:

```
d = {'x': 1, 'y': 2}
print(d.items())  # dict_items([('x', 1), ('y', 2)])
```

Python Tricky Questions

List Based Questions

1.
```
a = [1, 2, 3]
a.append([4, 5])
print(a)                    # Output: [1, 2, 3, [4, 5]]
```

2.
```
a = [1, 2, 3]
a.extend([4, 5])
print(a)                    # Output: [1, 2, 3, 4, 5]
```

3.
```
a = [1, 2, 3]
print(a.pop())
print(a)                    # Output: 3 [1, 2]
```

4.
```
a = [1, 2, 3]
b = a
b.append(4)
print(a)                    # Output: [1, 2, 3, 4]
```

5.
```
a = [1, 2, 3, 4, 5]
print(a[::-1])              # Output: [5, 4, 3, 2, 1]
```

6.
```
a = [1, 2, 3, 4]
print(a[::2])               # Output: [1, 3]
```

7.
```
a = [1, 2]
b = a[:]
b.append(3)
print(a, b)                 # Output: [1, 2] [1, 2, 3]
```

String based Questions

8.
```
s = "python"
print(s[0:3])                          # Output: pyt
```
9.
```
s = " python "
print(s.strip())                       # Output: python
```
10.
```
s = "Python Programming"
print(s.lower().count('p'))            # Output: 2
```
11.
```
s = "Hello World"
print(s.replace("World", "Python"))    # Output: Hello Python
```
12.
```
s = "Hello"
print("e" in s)                                # Output: True
```
13.
```
s = "Python"
for char in s:
print(char, end="-")                    # Output: P-y-t-h-o-n
```

14.
```
s = "abcabc"
print(s.find("b"))
print(s.rfind("b"))                            # Output: 1 4
```

Dictionary based Questions

15.
```
d = {"a": 1, "b": 2}
print(d["a"])                                  # Output: 1
```
16.
```
d = {"x": 5}
print(d.get("y"))                              # Output: None
```
17.
```
d = {"a": 1}
d["b"] = 2
print(d)                                       # Output: {'a': 1, 'b': 2}
```

18.
```
d = {"a": 1, "b": 2}
for k, v in d.items():
 print(k, v)                                  # Output: a 1 b 2
```

19.
```
 d = {"a": 1, "b": 2}
print("c" in d)                          # Output: False
```

20.
```
d = {}
d["list"] = [1, 2, 3]
print(d["list"][1])                                  # Output: 2
```