Python Strings & Unicode — Questions & Answers (Set 9)

Q1. In Python 3.X, what are the names and functions of string object types?

- str: Immutable text (Unicode code points), human-readable text.
- bytes: Immutable binary data (0–255 integers).
- bytearray: Mutable binary data (editable sequence of bytes).
- memoryview: Zero-copy view on a bytes-like object.

Q2. How do the string forms in Python 3.X vary in terms of operations?

- Common sequence operations apply: indexing, slicing, len, +, *, in.
- str has Unicode-aware operations (casing, normalization, regex, encode).
- bytes/bytearray have byte-oriented analogs, no Unicode semantics.
- No implicit mixing: str + bytes is a TypeError, must encode/decode.

Q3. In 3.X, how do you put non-ASCII Unicode characters in a string?

- Direct literal (UTF-8 source): s = 'café'
- Escape sequences: '\u00E9', '\U0001F600', '\N{GREEK SMALL LETTER PI}'
- chr() from code point: chr(0x1F600)
- Decode from bytes: b'\xc3\xa9'.decode('utf-8')

Q4. In Python 3.X, what are the key differences between text-mode and binary-mode files?

- Text mode: reads/writes str, applies encoding/decoding, newline translation.
- Binary mode: reads/writes bytes, no encoding, no newline translation.

Q5. How can you interpret a Unicode text file containing text encoded in a different encoding than your platform's default?

- Explicitly specify encoding when opening:
with open('data.txt', 'r', encoding='iso-8859-1') as f: txt = f.read()
- Or read bytes then decode: data_bytes.decode('utf-16')
- Use errors= argument for fault tolerance (ignore, replace, etc.).

Q6. What is the best way to make a Unicode text file in a particular encoding format?

with open('out.txt', 'w', encoding='utf-8', newline='') as f:
f.write('café\n')
- Prefer UTF-8, unless a specific encoding is required.
- For JSON, use json.dump(..., ensure_ascii=False).

Q7. What qualifies ASCII text as a form of Unicode text?

- ASCII 0x00–0x7F is a subset of Unicode U+0000–U+007F.
- In UTF-8, ASCII encodes as identical single-byte values.
- Therefore all ASCII is valid UTF-8 and valid Unicode.

Q8. How much of an effect does the change in string types in Python 3.X have on your code?

- Major shift: str is Unicode, bytes is binary.
- Must be explicit at boundaries (files, sockets, encodings).
- No implicit mixing of str/bytes; need explicit encode/decode.
- Pure-text logic simpler; I/O and protocol code more explicit and robust.