FREE EDITION · NOTES + 3 SAMPLE MCQS

CUET · COMPUTER SCIENCE · CLASS XI · CODE 308

Strings

CUET unit: Strings

By **UniDrill** · NCERT-grounded study material

WWW.UNIDRILL.IN

📌 Snapshot

- A Python string is an immutable sequence data type built from UNICODE characters — the foundation for all text-based programming tasks tested in CUET.
- String creation, indexing (positive and negative), slicing with step size, and traversal using `for` and `while` loops are all directly testable through code-output questions.
- Built-in string methods (`len` , `upper` , `lower` , `find` , `replace` , `split` , `join` , `partition` , etc.) are high-frequency MCQ sources; keep the reference table handy.
- The Handling Strings section (§8.6) has five programs (palindrome check, vowel replacement, reverse string) that form the basis of case/example-based NTA questions.
- String manipulation underpins data processing, and every concept here has a clear, unambiguous correct answer — ideal for MCQ format, so CUET tests it heavily.

📖 Detailed Notes

2.1 Core concepts

- **What is a string?** A string is a sequence of one or more UNICODE characters — letters, digits, whitespace, or any other symbol. It can be created by enclosing characters in single (`'...'`), double (`"..."`), or triple quotes (`"""..."""` or `'''...'''`). Triple-quoted strings can span multiple lines. (NCERT §8.2, p. 175)
- **Accessing characters — indexing:** Each character is accessed via its integer index written in square brackets. Positive indices start at 0 (first character) and go up to n-1 (last character). Negative indices start at -1 (last character) and go down to -n (first character). Using an out-of-range index raises an `IndexError` ; using a non-integer index raises a `TypeError` . (NCERT §8.2.1, p. 176)
- **Positive and negative index table:** For `str1 = 'Hello World!'` (length 12), positive indices are 0–11 and negative indices are -12 to -1. `str1[0]` = `'H'` , `str1[-1]` = `'!'` , `str1[-12]` = `'H'` . (NCERT §8.2.1, Table 8.1, p. 177)

- `len()` **function:** The built-in function `len()` returns the number of characters in a string. For `str1 = 'Hello World!'`, `len(str1)` returns 12. `str1[n-1]` gives the last character, `str1[-n]` gives the first character. (NCERT §8.2.1, p. 177)
- **String immutability:** Strings are immutable — their contents cannot be changed after creation. Attempting `str1[1] = 'a'` raises `TypeError: 'str' object does not support item assignment`. (NCERT §8.2.2, p. 177)
- **Concatenation (`+`):** Two strings can be joined using the `+` operator. `'Hello' + 'World!'` produces `'HelloWorld!'`. The original strings remain unchanged after the operation. (NCERT §8.3.1, p. 178)
- **Repetition (`*`):** A string can be repeated using the `*` operator with an integer. `'Hello' * 2` produces `'HelloHello'`. The original string remains unchanged. (NCERT §8.3.2, p. 178)
- **Membership (`in`, `not in`):** The `in` operator returns `True` if the first string appears as a substring in the second string, otherwise `False`. `'not in'` does the reverse. Example: `'W' in 'Hello World!'` → `True`; `'My' in 'Hello World!'` → `False`. (NCERT §8.3.3, p. 178)
- **Slicing (`str1[n:m]`):** Slicing extracts a substring from index n (inclusive) to m (exclusive). The number of characters in the result always equals m-n. If the first index is omitted, slicing starts from 0; if the second is omitted, slicing goes to the end. A too-large second index is silently truncated to the string's length. If the first index is greater than the second, an empty string is returned. (NCERT §8.3.4, p. 179)
- **Slicing with step (`str1[n:m:k]`):** A third index specifies step size — every k-th character is extracted. Default step is 1. `str1[0:10:2]` on `'Hello World!'` gives `'HloWr'`. Using `str1[::-1]` reverses the string. (NCERT §8.3.4, p. 179–180)
- **Traversal using `for` loop:** A `for` loop iterates over each character automatically from the first to the last. (NCERT §8.4, p. 180)
- **Traversal using `while` loop:** A `while` loop with an integer index variable runs while `index < len(str)`, incrementing the index each iteration. (NCERT §8.4, p. 180)
- **Built-in string methods (Table 8.2):** Python provides a rich set of methods. Key ones: `len()`, `title()`, `lower()`, `upper()`, `count(str, start, end)`, `find(str, start, end)`, `index(str, start, end)`, `endswith()`, `startswith()`, `isalnum()`, `islower()`, `isupper()`, `isspace()`, `istitle()`, `lstrip()`, `rstrip()`, `strip()`, `replace(oldstr, newstr)`, `join()`, `partition()`, `split()`. (NCERT §8.5, Table 8.2, pp. 180–184)
- `find()` **vs** `index()`**:** Both return the position of the first occurrence of a substring. The difference: `find()` returns -1 if the substring is not found, while `index()` raises a `ValueError`. (NCERT §8.5, Table 8.2, p. 181)
- `partition()`**:** Splits the string at the first occurrence of the separator into a 3-tuple: (before separator, separator, after separator). If separator is not found, returns (whole string, '', ''). (NCERT §8.5, Table 8.2, p. 184)

- `split()`: Returns a list of words delimited by the specified substring. If no delimiter is given, words are separated by spaces. (NCERT §8.5, Table 8.2, p. 184)
- `join()`: Returns a string in which all characters of a given string have been joined by a separator. Example: `'-'.join('HelloWorld!')` → `'H-e-l-l-o-W-o-r-l-d-!'`. (NCERT §8.5, Table 8.2, p. 183)
- **Handling Strings — user-defined programs (§8.6):** Five programs demonstrate applying the above concepts: (1) count character occurrences, (2) replace vowels with `*`, (3) print string in reverse using negative indexing in a `for` loop, (4) reverse string into a new variable using a user-defined function, (5) palindrome check using a `while` loop comparing characters from both ends. (NCERT §8.6, pp. 184–186)
- **UNICODE underlies all Python strings (NCERT §8.2, p. 175).** Python 3 strings store sequences of Unicode code points; this is why Hindi, Bengali or emoji characters all behave identically with the same indexing and slicing rules. Each code point is counted as a single character regardless of the underlying UTF-8 byte length.
- **Concatenation and repetition are O(n) operations.** The `+` operator on strings returns a new string by copying both operands; the `*` operator copies the source n times. For very large strings, repeated concatenation in loops becomes inefficient — `''.join(...)` is the idiomatic alternative.
- `endswith()`, `startswith()` **and case-checking methods (NCERT Table 8.2, pp. 181-182).** `s.startswith(prefix)` returns True if `s` begins with `prefix`; `s.endswith(suffix)` returns True if `s` ends with `suffix`. `isalnum()` returns True if all characters are alphanumeric; `islower()` / `isupper()` / `istitle()` test casing patterns; `isspace()` checks pure whitespace.
- `title()` **capitalises each word (NCERT Table 8.2, p. 180).** `"hello world".title()` → `"Hello World"`. NTA often pits `title()` against `capitalize()` (which is not in NCERT scope); stick to the textbook list.
- **Comparison of strings (NCERT §5.8.2, cross-ref).** Strings are compared lexicographically using Unicode code points; `'apple' < 'banana'` is True because 'a' < 'b'. CUET sometimes mixes a string-comparison MCQ into the strings unit.
- `isspace()` **and whitespace-only strings (NCERT Table 8.2, p. 182).** `' '.isspace()` is True; `''.isspace()` is False because the empty string has no characters at all. Beware the empty-string edge case.
- **Negative indices walk from the end (NCERT §8.2.1, p. 176).** For `s = "Python"`, `s[-1] = 'n'`, `s[-2] = 'o'`, `s[-6] = 'P'`. Going below `-len(s)` raises `IndexError`.
- **Strings can be used as iterables (NCERT §8.4, p. 180).** `for ch in s:` traverses character-by-character, which underlies the palindrome and frequency-count programs.

- **String repetition with zero or negative count (NCERT §8.3.2, p. 178).** `"ab" * 0` returns `""`; `"ab" * -1` also returns `""` — Python clamps non-positive multipliers to 0.
- `replace()` **returns a NEW string (NCERT Table 8.2, p. 183).** Since strings are immutable, `s.replace(old, new)` does not modify `s` — you must reassign: `s = s.replace(old, new)`.
- **Implicit concatenation of adjacent string literals (Python feature, not stressed by NCERT).** Avoid relying on `"foo" "bar"` style — use explicit `+` for clarity. NTA does not test this directly but exam discipline matters.

2.2 Definitions to memorise

Term	Definition	Page
String	A sequence of one or more UNICODE characters enclosed in single, double, or triple quotes	175
Indexing	A technique to access individual characters of a string using an integer position written in square brackets	176
Positive index	Index starting from 0 (first character) to n-1 (last character)	176
Negative index	Index starting from -1 (last character) to -n (first character)	176
IndexError	Error raised when an index value is out of the valid range of a string	176
TypeError	Error raised when a non-integer value is used as a string index	176
Immutable	A data type whose contents cannot be changed after creation; strings in Python are immutable	177
Concatenation	Joining two strings using the `+` operator	178
Repetition	Repeating a string a specified number of times using the `*` operator	178
Membership operator	`in` / `not in` — checks whether a substring exists within a string, returns `True` or `False`	178
Slicing	Extracting a substring by specifying an index range `str[n:m]` (n inclusive, m exclusive)	179
Step size	Third parameter in slicing `str[n:m:k]` — specifies the interval between characters extracted	179
Traversal	Accessing each character of a string one by one using a loop	180
Palindrome	A string that reads the same forwards and backwards (e.g., "kanak")	186
`find()`	Returns lowest index of a substring; returns -1 if not found	181
`index()`	Returns lowest index of a substring; raises ValueError if not found	181

Term	Definition	Page
`count()`	Returns number of non-overlapping occurrences of a substring	181
`upper()`	Returns a copy of the string with all characters uppercase	181
`lower()`	Returns a copy of the string with all characters lowercase	181
`title()`	Returns a copy with the first character of each word capitalised	180
`strip()`	Returns string with leading and trailing whitespace removed	182
`replace(old, new)`	Returns a copy where every occurrence of `old` is replaced with `new`	183
`split(sep)`	Returns a list of substrings split by the delimiter `sep`	184
`join(iter)`	Joins items of an iterable into a string using the calling string as separator	183
`partition(sep)`	Returns a 3-tuple (before, sep, after) splitting at the first occurrence	184

2.3 Diagrams / processes to remember

- **Table 8.1 — Indexing of characters in string 'Hello World!':** A two-row table showing positive indices 0–11 and negative indices -12 to -1 mapped to each character. Must-memorise layout for output-based questions. (NCERT p. 177)
- **Table 8.2 — Built-in functions for string manipulations:** A multi-row table listing method name, description, and example for all major string methods. Covers `len`, `title`, `lower`, `upper`, `count`, `find`, `index`, `endswith`, `startswith`, `isalnum`, `islower`, `isupper`, `isspace`, `istitle`, `lstrip`, `rstrip`, `strip`, `replace`, `join`, `partition`, `split`. (NCERT pp. 180–184)
- **Program 8-5 — Palindrome check logic:** Uses two pointers `i = 0` and `j = len(st) - 1`, a `while(i <= j)` loop, compares `st[i]` with `st[j]`, returns `False` on first mismatch, returns `True` after loop ends. (NCERT p. 186)
- **Reverse string using slice:** `str1[::-1]` reverses the string in one expression — a common NTA output question. (NCERT §8.3.4, p. 180)

2.4 Common confusions / NTA trap points

- **Slicing end index is exclusive:** `str1[1:5]` on `'Hello World!'` gives `'ello'` (indices 1, 2, 3, 4 — NOT 5). Students often include the end index character. NTA frequently asks the output of a slice like `str1[0:5]` expecting students to count correctly.
- `find()` **returns -1 vs** `index()` **raises ValueError:** A favourite NTA trap — students mix up which returns -1 on failure. `find('Hee')` on `'Hello World!'` returns `-1`; `index('Hee')` raises `ValueError: substring not found`.
- `islower()` **with non-alphabet characters:** `islower()` returns `True` even if the string contains digits or special symbols, as long as there is at least one lowercase

alphabet and no uppercase alphabets. `'hello 1234'.islower()` → `True`. Students incorrectly assume digits make it return `False`.

- **Immutability confusion:** Students assume `str1 * 2` or `str1 + str2` modifies `str1`. The NCERT explicitly states the original string remains the same after concatenation and repetition operations.
- **Negative step reversal (NCERT §8.3.4, p. 180).** `str1[::-1]` is a common output question. Students confuse the empty start/end with index 0 when step is negative — with step -1 and no bounds, Python starts from the last character and goes to the first.
- **`'in'` checks substring, not character (NCERT §8.3.3, p. 178).** `'Hello' in 'Hello World!'` is True; it checks substring containment for strings. NTA distractor may claim `in` only works on single characters.
- **`upper()` does NOT mutate (NCERT §8.5).** `s.upper()` returns a new string; `s` remains unchanged because strings are immutable. NTA trap: assumes in-place mutation.
- **`count()` is non-overlapping (NCERT Table 8.2, p. 181).** `'aaaa'.count('aa')` = 2, not 3 — Python finds non-overlapping occurrences.
- **Triple-quoted strings allow newlines (NCERT §8.2, p. 175).** They preserve internal line breaks, unlike single/double-quoted strings.
- **Slicing never raises IndexError (NCERT §8.3.4, p. 179).** Out-of-range slice indices are silently truncated. `s[0:1000]` works even if `len(s)` is 5.
- **`split()` with no argument splits on any whitespace (NCERT Table 8.2, p. 184).** Multiple spaces/tabs/newlines are collapsed; `"a b".split()` returns `['a','b']` not `['a','','b']`.

🎯 Practice MCQs

Q1. What will be the output of the following Python statement? ```python str1 = 'Hello World!' print(str1[2:9:2]) ```

A. ``'elloWo'``

B. ``'loWr'``

C. ``'lloWor'``

D. ``'loWrl'``

Q2. Which of the following statements about Python strings is/are correct? 1. A string can be created using single, double, or triple quotes. 2. The index of the last character of a string of length n is n. 3. Strings are immutable data types. 4. Negative indexing is not supported for strings in Python.

A. 1 and 3 only

B. 1, 3 and 4

C. 2 and 3 only

D. 1, 2 and 3

Q3. Consider `str1 = 'Hello World! Hello Hello'`. What does `str1.find('Hello', 15, 25)` return?

A. 0

B. 13

C. 19

D. -1

📊 PYQ Alignment

Chapter 8 (Strings) is one of the most frequently tested chapters in CUET Computer Science, appearing consistently across 2023–2025 papers with questions focused on string indexing/slicing output prediction, identification of the correct string method for a given task, and error identification in string operations (immutability, index out of range). Case-based questions using a given string and asking outputs of 3–4 operations in sequence are particularly common. See PYQ archive for Computer Science.