Indexing

PYTHON KA CHILL

Indexing in Python refers to accessing individual elements from a **sequence** (such as lists, strings, or tuples) using their position (index) within the sequence.

- Python uses zero-based indexing, meaning that the first element in a sequence is at index 0, the second element is at index 1, and so on.

Indexing

PYTHON KA

Types of Sequences that Support Indexing:

- **Strings:** A sequence of characters.
- **Lists:** A sequence of items that can be of any data type.
- **Tuples:** A sequence of immutable items.

Indexing

PYTHON KA

Types of Sequences that Support Indexing:

- **Strings:** A sequence of characters.

Example:

```python
text = "Python"

# Accessing individual characters
print(text[0])  # Output: P
print(text[3])  # Output: h
print(text[-1])  # Output: n (negative index starts from the end)
```

Explanation:

- `text[0]` accesses the first character `'P'`.
- `text[3]` accesses the fourth character `'h'`.
- `text[-1]` accesses the last character `'n'` (negative indexing allows you to start counting from the end).

Types of Sequences that Support Indexing:

- **Lists:** A sequence of items that can be of any data type.

Example:

```python
fruits = ["apple", "banana", "cherry", "date"]

# Accessing elements using positive indexing
print(fruits[0])  # Output: apple
print(fruits[2])  # Output: cherry

# Accessing elements using negative indexing
print(fruits[-1])  # Output: date
print(fruits[-3])  # Output: banana
```

Explanation:

- `fruits[0]` accesses the first item `'apple'`.
- `fruits[2]` accesses the third item `'cherry'`.
- `fruits[-1]` accesses the last item `'date'`.

Types of Sequences that Support Indexing:

- **Tuples:** A sequence of immutable items.

Example:

```python
coordinates = (10, 20, 30)

# Accessing tuple elements
print(coordinates[0])  # Output: 10
print(coordinates[2])  # Output: 30
```

Explanation:

- `coordinates[0]` accesses the first element `10`.
- `coordinates[2]` accesses the third element `30`.

Indexing

PYTHON KA CH

Negative Indexing

Python allows negative indexing, where `-1` refers to the last element, `-2` refers to the second last element, and so on.

Example with Negative Indexing:

```python
animals = ["cat", "dog", "elephant", "tiger"]

# Accessing elements using negative indexing
print(animals[-1])  # Output: tiger
print(animals[-3])  # Output: dog
```

In this example:

- `animals[-1]` accesses the last element `'tiger'`.
- `animals[-3]` accesses the second element from the start (`'dog'`).

Indexing

PYTHON KA CH

Accessing Elements in Nested Structures

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \text{matrix} \end{bmatrix}$$

Indexing

PYTHON KA CHIL

Accessing Elements in Nested Structures

Example:

python Copy code

```python
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Accessing elements in a nested list
print(matrix[0][2])  # Output: 3 (first row, third column)
print(matrix[2][1])  # Output: 8 (third row, second column)
```

In this example:

- `matrix[0][2]` accesses the third element in the first row, which is `3`.
- `matrix[2][1]` accesses the second element in the third row, which is `8`.

Indexing

PYTH

Errors

Example:

```python
fruits = ["apple", "banana", "cherry"]

# Trying to access an index out of range
print(fruits[3])  # This will raise an IndexError
```

Output:

```sql
IndexError: list index out of range
```

Indexing

PYTHON KA CH

Summary

- **Zero-based indexing:** The first element has an index of 0.
- **Positive indexing:** Counts from the start (left to right).
- **Negative indexing:** Counts from the end (right to left).
- **Indexing with nested structures:** Use multiple indices to access elements within nested sequences.
- **IndexError:** Occurs when trying to access an index that doesn't exist.