# Slicing

PYTHON KA CHILLA

You can also access multiple elements from a sequence using **slicing**, which involves specifying a range of indices.

**Syntax for Slicing:**

```python
sequence[start:stop:step]
```

- `start` : Starting index (inclusive).
- `stop` : Ending index (exclusive).
- `step` : Step size (optional, used to skip elements).


```
[17]  list = [10,20,30,40,50,60,870,0,90,100]
      ✓ 0.0s

[18]  list[0:90:2]
      ✓ 0.0s
...   [10, 30, 50, 70, 90]
```

# Slicing

PY1

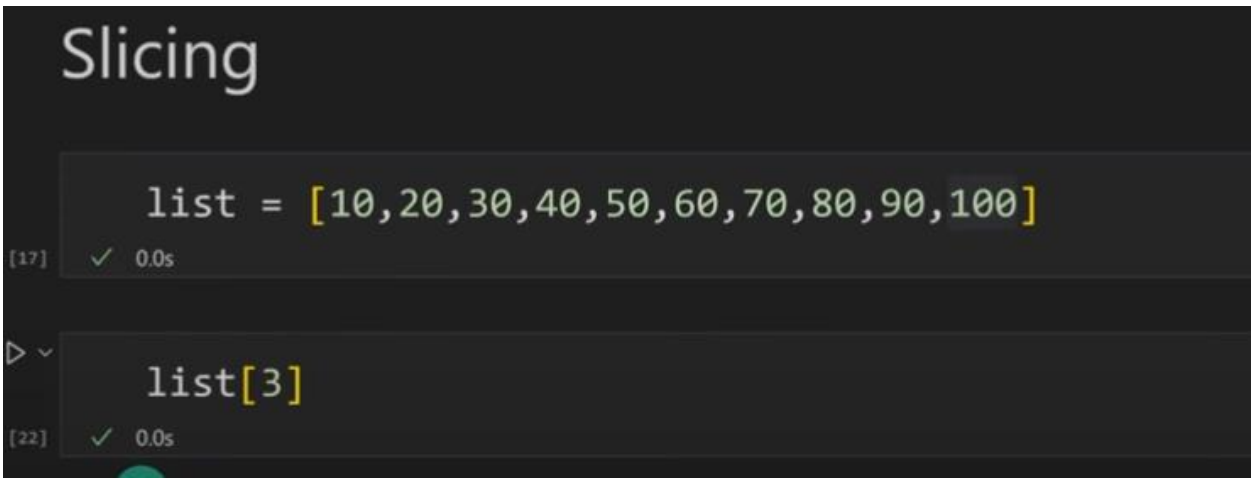
**Example: Slicing in Lists**

python Copy code

```python
numbers = [10, 20, 30, 40, 50, 60]

# Slicing from index 1 to 4
print(numbers[1:4])  # Output: [20, 30, 40]

# Slicing with step
print(numbers[::2])  # Output: [10, 30, 50] (every second element)

# Slicing with negative indices
print(numbers[-4:-1])  # Output: [30, 40, 50]
```

**Explanation:**

- `numbers[1:4]` slices the list from index `1` to `3` (excluding index `4`).
- `numbers[::2]` slices the list from start to end, selecting every second element.
- `numbers[-4:-1]` slices the list using negative indexing.



# Mutable element

## Modifying Lists Using Indexing

PYTHON KA C

Since lists are mutable, you can change elements using indexing

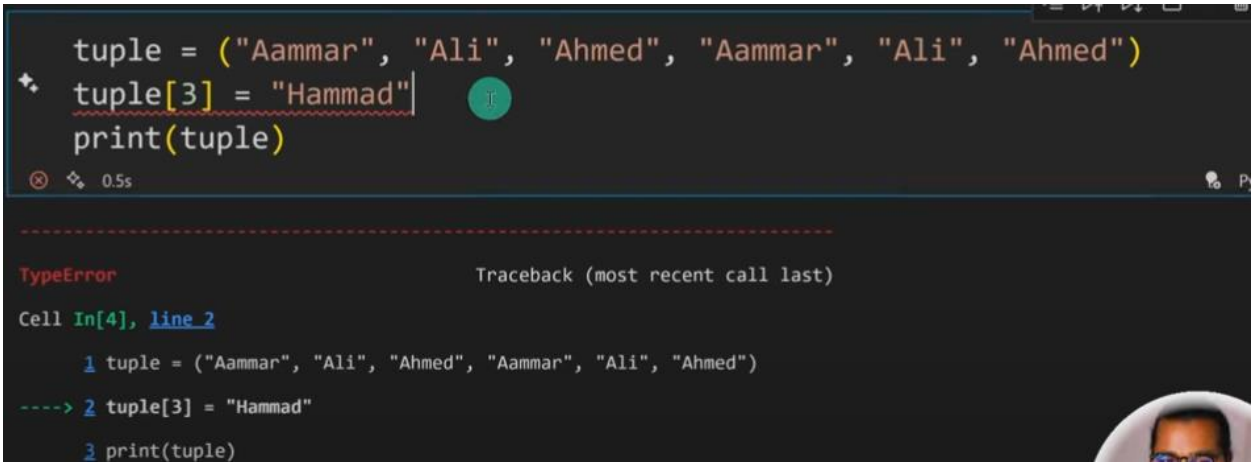
**Example:**

python Copy code

```python
colors = ["red", "green", "blue"]

# Modifying an element
colors[1] = "yellow"
print(colors)  # Output: ['red', 'yellow', 'blue']
```

```
list = ["Aammar", "Ali", "Ahmed", "Aammar", "Ali", "Ahmed"]
✓ 0.0s

list[3] = "Hammad"
✓ 0.0s

print(list)
✓ 0.0s
['Aammar', 'Ali', 'Ahmed', 'Hammad', 'Ali', 'Ahmed']
```

**This is tuple it is immutable**


tuple = ("Aammar", "Ali", "Ahmed", "Aammar", "Ali", "Ahmed")
tuple[3] = "Hammad"
print(tuple)
0.5s
TypeError
Traceback (most recent call last)
Cell In[4], line 2
1 tuple = ("Aammar", "Ali", "Ahmed", "Aammar", "Ali", "Ahmed")
----> 2 tuple[3] = "Hammad"
3 print(tuple)