

# Python List Operations



# Everyday Examples

In real life, we use lists all the time:

- Shopping list: milk, eggs, bread.
- To-do list: homework, piano practice, video games.
- Score list: 95, 88, 100.

In Python, a **list** is a way to store **many items in one variable**.

# Create a List

We use **square brackets** [ ] to make a list.

```
# A list of numbers
scores = [95, 88, 100]

# A list of strings
fruits = ["apple", "banana", "cherry"]

# A list can be empty
items = []
```

- The items are separated by **commas**.
- The **order** matters: first, second, third, ...

# Access an Item by Index

- Each item in a list has a **position number** called an **index**.
- We can use **square brackets** `[index]` to access **one item**.

# Access an Item by Index

- Each item in a list has a **position number** called an **index**.
- We can use **square brackets** `[index]` to access **one item**.

Example:

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

print(fruits[0]) # apple
print(fruits[1]) # banana
print(fruits[2]) # cherry
```

Remember: Python starts counting from **0**, not **1**.

# Q1 🍓 Which Fruit?

What is the output?

```
fruits = ["apple", "banana", "cherry", "mango"]  
  
print(fruits[1])  
print(fruits[3])
```

Remember: Python starts counting from **0**, not **1**.

# Q1 🍓 Which Fruit?

What is the output?

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

print(fruits[1])
print(fruits[3])
```

Remember: Python starts counting from **0**, not **1**.

Output:

```
banana
mango
```

# Change an Item

We can **replace** a value in a list using **assignment**.

Example:

```
scores = [90, 85, 100]

# replace 85 with 95
scores[1] = 95

print(scores)
```

# Change an Item

We can **replace** a value in a list using **assignment**.

Example:

```
scores = [90, 85, 100]

# replace 85 with 95
scores[1] = 95

print(scores)
```

Output:

```
[90, 95, 100]
```

# Add an Item with `append`

`append` adds one new item **at the end** of the list.

Example:

```
numbers = [1, 2, 3]  
  
numbers.append(4)  
numbers.append(5)  
  
print(numbers)
```

# Add an Item with `append`

`append` adds one new item **at the end** of the list.

Example:

```
numbers = [1, 2, 3]  
  
numbers.append(4)  
numbers.append(5)  
  
print(numbers)
```

Output:

```
[1, 2, 3, 4, 5]
```

## Q2 🌟 Append and Change

What is the final value of numbers ?

```
numbers = [10, 20]  
  
numbers.append(30)    # [10, 20, 30]  
numbers[0] = 5        # [5, 20, 30]  
numbers.append(40)    # [5, 20, 30, 40]  
  
print(numbers)
```

## Q2 🌟 Append and Change

What is the final value of `numbers` ?

```
numbers = [10, 20]
numbers.append(30)      # [10, 20, 30]
numbers[0] = 5          # [5, 20, 30]
numbers.append(40)      # [5, 20, 30, 40]
print(numbers)
```

Output:

```
[5, 20, 30, 40]
```

# Remove an Item with `pop`

`pop()` removes the last item from the list and returns it.

Example:

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

last = fruits.pop()      # removes "cherry"
print(last)              # cherry
print(fruits)            # ['apple', 'banana']
```

# Remove an Item with **pop**

**pop()** removes the last item from the list and returns it.

Example:

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

last = fruits.pop()      # removes "cherry"
print(last)              # cherry
print(fruits)            # ['apple', 'banana']
```

Output:

```
cherry
['apple', 'banana']
```

# Q3 🏃 Using pop

What is the output?

```
numbers = [1, 2, 3, 4]  
  
x = numbers.pop()  
  
print(x)  
print(numbers)
```

# Q3 🯚 Using `pop`

What is the output?

```
numbers = [1, 2, 3, 4]  
  
x = numbers.pop()  
  
print(x)  
print(numbers)
```

Output:

```
4  
[1, 2, 3]
```

# Traverse a List with **for**

- Traverse: **visit every item** in a list.
- Syntax: **for item in list:**

Example:

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

for f in fruits:
    print(f)
```

Important: the **print** line must be **indented** (4 spaces).

# Traverse a List with **for**

- Traverse: **visit every item** in a list.
- Syntax: **for item in list:**

Example:

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

for f in fruits:
    print(f)
```

Output:

```
apple
banana
cherry
```

# Q4 🏀 Print All Players

Fill in the blanks to print each player's name.

```
players = ["Alice", "Bob", "Charlie"]  
for ____ in ____:  
    print(____)
```

Output:

# Q4 🏀 Print All Players

Fill in the blanks to print each player's name.

```
players = ["Alice", "Bob", "Charlie"]

for name in players:
    print(name)
```

Output:

```
Alice
Bob
Charlie
```

# Q5 Mix of Operations

What is the output?

```
numbers = [2, 4, 6]
numbers.pop()      # [2, 4]
numbers[1] = 5    # [2, 5]
numbers.append(8) # [2, 5, 8]

print(numbers[1])
print(numbers[2])
```

# Q5 Mix of Operations

What is the output?

```
numbers = [2, 4, 6]
numbers.pop()      # [2, 4]
numbers[1] = 5    # [2, 5]
numbers.append(8) # [2, 5, 8]

print(numbers[1])
print(numbers[2])
```

Output:

```
5
8
```

# Summary

- A **list** stores many items in one variable.
- Items are in **order** and have **indexes** starting from 0.
- Common operations:
  - **Access:** `print(numbers[2])`
  - **Change:** `numbers[2] = 67`
  - **Append:** `numbers.append(88)`
  - **Pop:** `numbers.pop()` removes the last item
  - **Traversal:** `for item in numbers: ...`



ALGORITHM  
WITH CHELSEA