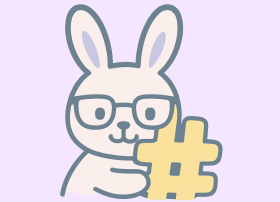
Python For-Loop with List



Today’s goals

- Compute sum (accumulator).
- Find max (current best).
- Filter evens (create new list).
- Reverse list (loop backward).

Review: List Traversal

Traverse means to visit each item, one by one.

Example:

```
numbers = [3, 1, 4]

for x in numbers:
    print(x)
```

- `x` takes on each item in the list, one by one.
- The code inside the loop must be **indented** (4 spaces).

Q1 👣 List Traversal

What is the output?

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

for c in colors:
    print(c)
```

Q1 👣 List Traversal

What is the output?

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

for c in colors:
    print(c)
```

Output:

```
red
blue
green
```

Sum of an Integer List

We use an **accumulator**:

- Start with `total = 0` (the running sum).
- For each number `x`, do `total = total + x`.
- At the end, `total` is the sum.

Sum of an Integer List

We use an **accumulator**:

- Start with `total = 0` (the running sum).
- For each number `x`, do `total = total + x`.
- At the end, `total` is the sum.

Python code:

```
numbers = [2, 5, 1, 4]

total = 0
for x in numbers:
    total = total + x

print(total)
```

Q2 🧮 Compute the Sum

Complete the code and predict the output.

```
numbers = [3, 6, 2]

total = ____
for x in _______:
    total = total + ____

print(total)
```

Output:

```

```

Q2 🧮 Compute the Sum

Complete the code and predict the output.

```
numbers = [3, 6, 2]

total = 0
for x in numbers:
    total = total + x

print(total)
```

Output:

```
11
```

Elephant & Corn 

Imagine an elephant walking across a corn field.

- Each corn has a **length**.
- The elephant can carry **only one corn** with his trunk.
- He wants to take home the **longest corn**.

Elephant & Corn 🐘🌽

Imagine an elephant walking across a corn field.

- Each corn has a **length**.
- The elephant can carry **only one corn** with his trunk.
- He wants to take home the **longest corn**.

How does he make sure he keeps the longest corn?

- Pick the first corn as his current best.
- Compare his carried corn with the new corn in the field.
- If the new corn is **longer**, he **drops** the old corn and **picks up** the new corn.

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
corn:                  [7      2      9      4]
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
              (start)

corn:                  [7      2      9      4]

current_max:      7
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                      x
                      ↓
corn:                [7      2      9      4]

current_max:      7
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                      x
                      ↓
corn:                [7      2      9      4]

current_max:      7      7
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                          x
                          ↓
corn:                [7   2    9    4]

current_max:      7      7
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                          x
                          ↓
corn:                [7   2    9    4]

current_max:      7    7    7
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                                x
                                ↓
corn:                  [7      2      9      4]

current_max:      7      7      7
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                                x
                                ↓
corn:                  [7       2       9       4]

current_max:       7       7       7       9
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                                    x
                                    ↓
corn:                  [7      2      9      4]

current_max:      7      7      7      9
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

```
                                    x
                                    ↓
corn:                  [7      2      9      4]

current_max:      7      7      7      9      9
```

Elephant & Corn 🐘🌽

Python code:

```
corns = [7, 2, 9, 4]       # lengths of the corns in the field

current_max = corns[0]     # elephant picks the first corn

for x in corns:            # elephant walks across the field; x is a new corn
    if x > current_max:    # if the new corn x is longer
        current_max = x    # pick up the new corn x

print(current_max)
```

Output:

```
9
```

Max of an Integer List

Input: A list such as `numbers = [7, 2, 9, 4]`

Goal: Find the **max** number, which is `9` in this example.

The same as the **elephant & corn** story.

- 🌽 `numbers` records the corn lengths.
- 🐘 The elephant wants to carry home **the longest corn** (**max number**).

Max of an Integer List

Idea:

- Use `m` to record the max number.
- Start with the **first item**: `m = numbers[0]`.
- Traverse the list.
- When an item is greater than `m`, we update `m`.

Max of an Integer List

Python code:

```
numbers = [7, 2, 9, 4]

m = numbers[0]
for x in numbers:
    if x > m:
        m = x

print(m)
```

Max of an Integer List

Python code:

```
numbers = [7, 2, 9, 4]

m = numbers[0]
for x in numbers:
    if x > m:
        m = x

print(m)
```

Output:

```
9
```

Q3 🏆 Find the Max Number

Complete the code and predict the output.

```
numbers = [5, 1, 8, 3]

m = numbers[0]
for x in numbers:
    if x ____ m:
        m = ____

print(m)
```

Output:

```

```

Q3 🏆 Find the Max Number

Complete the code and predict the output.

```
numbers = [5, 1, 8, 3]

m = numbers[0]
for x in numbers:
    if x > m:
        m = x

print(m)
```

Output:

```
8
```

Select Even Numbers into a New List

Input: A list such as `numbers = [1, 2, 3, 4, 5, 6]`.

Goal: Create a new list `evens = [2, 4, 6]`.

Select Even Numbers into a New List

Input: A list such as `numbers = [1, 2, 3, 4, 5, 6]`.

Goal: Create a new list `evens = [2, 4, 6]`.

Idea:

- We create an empty list `evens = []`.
- Traverse `numbers` and decide if each item `x` is even.
- Recall: `x` is even if `x % 2 == 0`.
- If `x` is even, append it to the new list `evens`.

Select Even Numbers into a New List

Python code:

```
numbers = [1, 2, 3, 4, 5, 6]

evens = []
for x in numbers:
    if x % 2 == 0:
        evens.append(x)

print(evens)
```

Select Even Numbers into a New List

Python code:

```
numbers = [1, 2, 3, 4, 5, 6]

evens = []
for x in numbers:
    if x % 2 == 0:
        evens.append(x)

print(evens)
```

Output:

```
[2, 4, 6]
```

Q4 🎯 Pick Odds

What is the output?

```
numbers = [10, 11, 12, 13]

odds = []
for x in numbers:
    if x % 2 == 1:
        odds.append(x)

print(odds)
```

Q4 🎯 Pick Odds

What is the output?

```
numbers = [10, 11, 12, 13]

odds = []
for x in numbers:
    if x % 2 == 1:
        odds.append(x)

print(odds)
```

Output:

```
[11, 13]
```

Print a List in Reverse Order

Input: A list such as `colors = ["red", "green", "blue"]`.

Goal:

- Print items from back to front.
- The output will be `blue`, `green`, `red`.

Print a List in Reverse Order

Input: A list such as `colors = ["red", "green", "blue"]`.

Goal:

- Print items from back to front.
- The output will be `blue`, `green`, `red`.

Idea:

- Loop from the last index down to 0.
- `n = len(colors)` gives the length (here, 3).
- `range(n - 1, -1, -1)` gives `2, 1, 0`.
- `print(colors[i])` prints the item at index `i`.

Print a List in Reverse Order

Python code:

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

n = len(colors)  # 3

for i in range(n - 1, -1, -1):  # 2, 1, 0
    print(colors[i])
```

Print a List in Reverse Order

Python code:

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

n = len(colors)  # 3

for i in range(n - 1, -1, -1):  # 2, 1, 0
    print(colors[i])
```

Output:

```
blue
green
red
```

Reverse a List to Make a New List

Create a new list `rev` to store the items in reverse order.

```
students = ["Alice", "Bob", "Chelsea", "David"]

rev = []
n = len(students)  # 4

for i in range(n - 1, -1, -1):  # 3, 2, 1, 0
    rev.append(students[i])

print(rev)
```

Reverse a List to Make a New List

Create a new list `rev` to store the items in reverse order.

```
students = ["Alice", "Bob", "Chelsea", "David"]

rev = []
n = len(students)  # 4

for i in range(n - 1, -1, -1):  # 3, 2, 1, 0
    rev.append(students[i])

print(rev)
```

Output:

```
['David', 'Chelsea', 'Bob', 'Alice']
```

Summary

- **Traverse**: `for item in list: ...`.
- **Sum** with an accumulator: `total = total + x`.
- **Max** with a “current best” value: `if x > m: m = x`.
- **Filter** even numbers into a new list using `append`.
- **Reverse** by looping indexes from the last index down to 0.