Lists and Tuples

A list holds many values in order and can change; a tuple holds many values in order and cannot — learn to create, read, change and loop over both.

Introduction

So far, every variable held one value: one name, one age, one price. But real life is full of groups: a list of students, a list of marks, a shopping list. Python handles groups with **lists** and **tuples**.

A **list** is an ordered group of values inside square brackets, like `[85, 90, 78]`. You can add to it, remove from it, and change its items. A **tuple** looks similar but uses round brackets, like `(85, 90, 78)`, and once it is made, you cannot change it. That fixed quality is useful when a group should never change by accident — for example, the days of the week, or a fixed set of coordinates.

In this lesson you will create both, read items by position, add items to a list, loop over a group, and learn when to pick each one.

(Roman Urdu: List aik tartib wali group hoti hai jisay badla ja sakta hai, jab ke tuple bhi group hai magar isay bananay ke baad badla nahi ja sakta.)

Lists feel easy until something goes wrong silently. The classic trap is doing an operation and not noticing what it returned: `sort()` changes the list in place and returns nothing, while `sorted()` returns a new sorted list and leaves the original alone. Knowing which kind an operation is saves hours of confusion.

Key points

- **A list uses square brackets.** Example: `marks = [85, 90, 78, 92]`.
- **A tuple uses round brackets.** Example: `days = ("Mon", "Tue", "Wed")`. A one-item tuple needs a comma: `("Mon",)`.
- **Both keep order and allow indexing.** `marks[0]` is the first item; `marks[-1]` is the last item.
- **Lists can change; tuples cannot.** You can `append()` to a list, change `marks[0]`, or remove an item. Trying the same on a tuple gives an error.
- **`.append()` adds one item at the end of a list.** `.remove()` removes the first matching value. `len()` counts the items.
- **The `in` keyword checks membership.** `"Ali" in names` is `True` if Ali is in the list.
- **Looping visits every item, one by one.** `for mark in marks:` runs the indented block once per item.
- **Slicing works on both.** `marks[1:3]` gives a new list or tuple with the middle items.
- **Choose a list for changing data, a tuple for fixed data.** Marks that grow during the term suit a list; a fixed pair like a screen size suits a tuple.

- **`.sort()` sorts in place and returns `None`.** `sorted(marks)` returns a new sorted list and leaves the original unchanged. Pick the one you actually want.
- **`.insert()` puts an item at a position; `.pop()` takes one out.** `.pop()` with no argument removes the last item; `.pop(0)` removes the first.

Examples

Example 1 — Creating and reading a list

```
marks = [85, 90, 78, 92]
print(marks[0])
print(marks[-1])
print(len(marks))
print(marks[1:3])
```

What you will get:

```
85
92
4
[90, 78]
```

Just like strings, list indexing starts at 0, and slicing stops before the end index.

Example 2 — Changing a list with append and assignment

```
fruits = ["apple", "banana"]
fruits.append("mango")
print(fruits)
fruits[0] = "orange"
print(fruits)
fruits.remove("banana")
print(fruits)
```

What you will get:

```
['apple', 'banana', 'mango']
['orange', 'banana', 'mango']
['orange', 'mango']
```

`append()` always adds at the end. Assignment with an index, like `fruits[0] = "orange"`, replaces that one item only.

Example 3 — A tuple stays fixed, and a loop visits every item

```
subjects = ("Maths", "English", "Science")
print(subjects[0])

for subject in subjects:
    print(f"I study {subject}")

marks = [85, 90, 78]
total = 0
for mark in marks:
    total = total + mark
print(f"Total: {total}")
```

What you will get:

```
Maths
I study Maths
I study English
I study Science
Total: 253
```

The loop body must be indented. If you tried `subjects[0] = "Urdu"`, Python would stop with an error, because tuples cannot be changed.

Example 4 — Biggest, smallest, total, and position

A list of marks is more useful once you summarise it. Four built-in tools do that in one line each:

```
marks = [78, 92, 65, 88]
print(max(marks), min(marks), sum(marks))
print(marks.index(92))
marks.sort()
print(marks)
```

What you will get:

```
92 65 323
1
[65, 78, 88, 92]
```

`max()` and `min()` find the ends, `sum()` adds everything (78 + 92 + 65 + 88 = 323), and `.index(92)` says where 92 sits — index 1, because counting starts at 0. After `.sort()`, the list itself is changed.

Example 5 — Inserting and removing at exact positions

`append()` only adds at the end. When order matters, `insert()` and `pop()` give you control over the middle and the front:

```
todo = ["homework", "reading"]
todo.insert(1, "prayer")
print(todo)
todo.pop()
print(todo)
todo.pop(0)
print(todo)
```

What you will get:

```
['homework', 'prayer', 'reading']
['homework', 'prayer']
['prayer']
```

`insert(1, "prayer")` slides "prayer" into position 1. Plain `pop()` removes the last item ("reading"); `pop(0)` removes the first ("homework"). Note that `pop()` also returns the removed item, so `removed = todo.pop()` would keep it for later use.

Example 6 — Real-world case: a shop's bill from a price list

A shopkeeper keeps the day's item prices in a list. The bill, the most expensive item, and the new total after one more sale are all one line each:

```
prices = [120, 250, 80, 300]
print("Items:", len(prices))
print("Total bill:", sum(prices))
print("Most expensive:", max(prices))
prices.append(150)
print("New total:", sum(prices))
```

What you will get:

```
Items: 4
Total bill: 750
Most expensive: 300
New total: 900
```

This is why shops, schools and offices store numbers in lists instead of separate variables: one list, and every summary — total, biggest, count — is a single simple call.

Quick recap

- Lists use `[ ]` and can change; tuples use `( )` and cannot.
- Both are ordered, indexed from 0, and support slicing and `len()`.
- `.append()` adds to a list; `.remove()` takes a value out.
- `for item in group:` visits every item in order.
- Use a list for growing data, a tuple for fixed data.
- `sorted()` returns a new list; `.sort()` changes the list in place.
- `.insert()` adds at a position; `.pop()` removes and returns an item.

Common mistakes

1. **Forgetting the comma in a one-item tuple** — `("Mon")` is just a string in brackets, not a tuple. Fix: write `("Mon",)` with a comma.
2. **Trying to change a tuple** — `subjects[0] = "Urdu"` gives an error. Fix: use a list if the data must change, or build a new tuple.
3. **Using an index that does not exist** — with 3 items, index 3 is out of range. Fix: remember the last valid index is `len(group) - 1`.
4. **Forgetting the indent inside a loop** — the loop body must be indented, or Python will not know what belongs to the loop. Fix: indent every line that should repeat.
5. **Confusing append with adding numbers** — `marks.append(10)` adds the item 10 to the list; it does not add 10 to every mark. Fix: to change values, loop and update each item.
1. **Removing a value that is not there** — `fruits.remove("mango")` when mango is not in the list gives a ValueError. Fix: check first with `if "mango" in fruits:`.
2. **Storing the result of `.sort()`** — `sorted_list = marks.sort()` gives `None`, because `.sort()` changes the list instead of returning one. Fix: use `sorted_list = sorted(marks)`.

Practice

1. Make a list of your five favourite foods. Print the first and last items, then append one more food and print the whole list.
2. Make a tuple of the four seasons. Try to change one season and read the error message. Write down, in one line, what the error means.
3. Make a list of four marks. Loop over it and print each mark with the message "Mark: [value]". Then loop again and find the total.
4. Check membership: make a list of three friends' names and test two names with `in` — one that is present and one that is not. Print both results.
5. Make a list of six marks. Print the highest, lowest, total and average (total divided by `len()`). Then print the list sorted with `sorted()`, and confirm the original list is unchanged.

6. Make a shopping list of three items. Insert one item at position 1, remove the last item with `pop()`, and print the list after each step.
7. Make a tuple of three city coordinates, like `("Lahore", "Karachi", "Multan")`, and unpack it into three variables in one line: `a, b, c = cities`. Print each variable.

Practice quiz

1. How do you add an item to the end of a list?

- add()
- append()
- push()
- insert() with no position

Answer: append()

2. What is the difference between a list and a tuple?

- Lists hold numbers, tuples hold text
- Lists can be changed after creation, tuples cannot
- Tuples are faster to type
- There is no difference

Answer: Lists can be changed after creation, tuples cannot

3. In fruits = ["apple", "banana", "mango"], what is fruits[0]?

- mango
- banana
- apple
- An error

Answer: apple

Watch the video lesson: https://youtu.be/BE-0mvJcZvQ

Read online: https://scopetreasure.com/notes/py-lists-tuples · More free tools and notes at https://scopetreasure.com