

Looping Through a List

Name		Date	
Class / Sec		Roll no.	

1 · AIM

To write a Python program that uses a **for loop** to work through a list — printing each item, numbering them with enumerate, testing each against a condition, and building a running total.

2 · SOFTWARE AND LIBRARIES USED

TOOL	WHAT IT IS FOR
Python 3	The programming language the program is written in
Jupyter Notebook	The environment used to write and run the code (IDLE or Colab also acceptable)
No extra library	The for loop and enumerate are built into Python — nothing to import

3 · THEORY

A **for loop** repeats the same instructions once for every item in a list. It saves writing the same line six times, and it works whatever the list length — the same code handles 6 marks or 600.

PATTERN	GIVES YOU	USE IT WHEN
for m in marks:	Each value in turn	You only need the value itself
for i, m in enumerate(marks):	The index and the value	You need to number the items too
for m in marks: if ...	Values matching a test	Only some items should be acted on
total = total + m	A running accumulator	Building a sum or a count as you go

Indentation is not decoration: Everything **indented** under the for line runs once per item. Anything not indented runs only once, after the loop finishes. In Python the spacing **is** the instruction — get it wrong and the program behaves differently.

4 · PROGRAM

```
marks = [78, 65, 88, 54, 91, 72]

# 1 - Print every item in the list
print("All marks:")
for m in marks:
    print(" ", m)

# 2 - Number the items using enumerate
print("\nNumbered:")
for i, m in enumerate(marks, start=1):
    print(" Student", i, "scored", m)

# 3 - Test each item against a condition
print("\nPass or fail (pass mark 70):")
passed = 0
for m in marks:
    if m >= 70:
        print(" ", m, "- PASS")
```

```

        passed = passed + 1
    else:
        print(" ", m, "- NEEDS WORK")

# 4 - Build a running total inside the loop
total = 0
for m in marks:
    total = total + m

# These lines are OUTSIDE the loop - they run once
print("\nPassed :", passed, "out of", len(marks))
print("Total   :", total)
print("Average :", round(total / len(marks), 2))

```

Watch the last three lines: They are **not indented**, so they run once after the loop ends. If they were indented, you would see a total printed six times — once per item — which is a very common mistake.

5 • EXPECTED OUTPUT

```

All marks:
 78
 65
 88
 54
 91
 72

Numbered:
Student 1 scored 78
Student 2 scored 65
Student 3 scored 88
Student 4 scored 54
Student 5 scored 91
Student 6 scored 72

Pass or fail (pass mark 70):
78 - PASS
65 - NEEDS WORK
88 - PASS
54 - NEEDS WORK
91 - PASS
72 - PASS

Passed   : 4 out of 6
Total    : 448
Average  : 74.67

```

[PASTE SCREENSHOT HERE]

Paste a screenshot of YOUR program running, with the code and output visible together

6 • TRACING THE ACCUMULATOR

The running total is the part students find hardest. Trace it by hand, one pass at a time.

PASS	m	total BEFORE	total AFTER
start	—	—	0
1	78	0	0 + 78 = 78
2	65	78	78 + 65 = 143

3	88	143	$143 + 88 = \mathbf{231}$
4	54	231	$231 + 54 = \mathbf{285}$
5	91	285	$285 + 91 = \mathbf{376}$
6	72	376	$376 + 72 = \mathbf{448}$

Check it against sum(): The loop total of **448** is exactly what **sum(marks)** returns. Writing the loop yourself shows you understand what sum() is doing — and $448 \div 6$ gives the average of 74.67.

7 · OBSERVATION AND RESULT

Two or three sentences in your own words. Say what the loop did and why indentation mattered.

Example observation: A single for loop repeated the same instructions once for every item in the list, so six marks were handled by three lines of code. Adding a condition inside the loop let each mark be tested separately, giving 4 passes out of 6. The running total built up to 448 across the six passes, and dividing by the length gave an average of 74.67 — the same result as sum(marks) divided by len(marks).

Worth knowing: The same loop works on a list of **six** marks or **six hundred** without changing a line. That is the real reason loops exist — not to save typing, but so the code does not depend on how much data there is.

8 · VIVA QUESTIONS FOR THIS PROGRAM

LIKELY QUESTION	SHAPE OF A STRONG ANSWER
How many times does your loop run?	Six — once for every item in the list. len(marks) decides it, not a number I wrote
What does enumerate() give you?	Both the index and the value. Without it you only get the value
Why is total set to 0 before the loop?	It must exist before you add to it. Inside the loop it would reset to 0 every pass
What if the last three prints were indented?	They would run six times instead of once — a total printed after every mark
Your total is 448. How can you check it?	Compare with sum(marks) , or add the six numbers by hand. Both give 448

9 · HOW THIS ENTRY IS MARKED

FULL MARKS

All four loop patterns shown · correct indentation throughout · commented code · a screenshot of **your own** output · the accumulator **traced by hand** · you can say how many times the loop runs and why

LOSES MARKS

Total printed **inside** the loop, so it appears six times · total reset to 0 inside the loop · no screenshot, or someone else's · an observation that just says a loop was used

Before you submit: Use **your own** marks, and **trace the total by hand** in your file the way the table above does. That trace is what proves you followed the loop rather than copied its output.