

# For Loops

A loop allows you to execute the same statements multiple times. `for` loops are used to iterate over the items of a sequence, either by value or by index.

Manager:

Recorder:

Presenter:

Reflector:

## Content Learning Objectives

*After completing this activity, students should be able to:*

- Describe the syntax and the purpose of a `for` statement.
- Predict how `range()` works given 1, 2, or 3 arguments.
- Explain the difference of iterating values versus indexes.

## Process Skill Goals

*During the activity, students should make progress toward:*

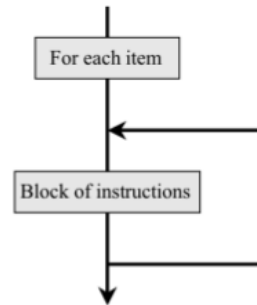
- Describing behavior when running experiments in a shell. (Critical Thinking)



## Model 1 for Each Value

A `for` loop executes the same block of code “for each item in a sequence”. Create a new file named `loops.py`, and enter the following code:

```
print("hello")
for x in [2, 7, 1]:
    print("the number is", x)
print("goodbye")
```



### Questions (15 min)

**Start time:**

1. Run the `loops.py` program. How many times does the indented line of code execute under the `for` loop?
2. How many times does the line of code NOT indented execute after the `for` loop?
3. Identify the value of `x` each time the indented line of code is executed.
  - a) 1st time:
  - b) 2nd time:
  - c) 3rd time:
4. Modify the list `[2, 7, 1]` in the following ways, and rerun the program each time. Indicate how many times the `for` loop executes.
  - a) non-consecutive numbers: `[5, -7, 0]`
  - b) numbers decreasing in value: `[3, 2, 1, 0]`
  - c) all have the same value: `[4, 4]`
  - d) single value in a list: `[8]`

5. In general, what determines the number of times that the loop repeats?

6. What determines the value of the variable `x`? Explain your answer in terms of what is assigned (`x = ...`) each time the loop runs.

7. Modify the program as follows:

a) Write a statement that assigns `[0, 1, 2, 3, 4]` to the variable `numbers`.

b) Rewrite the `for x ...` statement to use the variable `numbers` instead.

c) Does the assignment need to come before or after the `for` statement?

8. Add the following code at the end of your program:

```
for c in "Hi!":  
    print(c)
```

a) What is the output of this `for` statement?

b) What determined how many times `print(c)` was called?

c) Explain what a `for` statement does with strings.

9. (Optional) What other types, besides lists and strings, can a `for` loop handle? Experiment by adding examples to your `loops.py` program. Summarize here what works and what doesn't.

## Model 2 The `range` Type

The `range` type represents a sequence of integers. The `range()` function can be called with one, two, or three arguments.

Python code	Shell output
<code>range(5)</code>	<code>range(0, 5)</code>
<code>list(range(5))</code>	<code>[0, 1, 2, 3, 4]</code>
<code>x = range(3)</code>	
<code>print(x)</code>	<code>range(0, 3)</code>
<code>print(list(x))</code>	<code>[0, 1, 2]</code>
<code>list(range(5, 10))</code>	<code>[5, 6, 7, 8, 9]</code>
<code>list(range(-3, 4))</code>	<code>[-3, -2, -1, 0, 1, 2, 3]</code>
<code>list(range(4, 10, 2))</code>	<code>[4, 6, 8]</code>
<code>for i in range(5):     print(i)</code>	<i>prints 0, 1, 2, 3, 4 (on separate lines)</i>

### Questions (15 min)

**Start time:**

10. Explain the difference in output between the first two lines of code (with and without the `list` function).

11. If the argument of the `range` function specifies a single number ( $x$ ):

- What will be the first number listed?
- What will be the last number listed?
- How many numbers will be in the list?
- Use the range function to generate the sequence 0, 1, 2, 3.

12. If the argument of the `range` function specifies two numbers ( $x, y$ ):

- What will be the first number listed?
- What will be the last number listed?
- How many numbers will be in the list?
- Use the range function to generate the sequence 1, 2, 3, 4.

13. If the argument of the `range` function specifies three numbers  $(x, y, z)$ :
- What will be the first number listed?
  - What does the third argument represent?
  - How many numbers will be in the list?
  - Use the range function to generate the sequence 1, 3, 5, 7.
14. In your Editor, make a copy of the Model 1 code. Then modify the `for` statement so that the number of times the loop executes is determined by a variable named `times`.
- How did you change the `for` statement?
  - How would you cause the loop to print the values 0 to 5?
15. Consider the `for` statement used in Model 1 versus #14.
- If you wanted to execute a loop 100 times, which type of `for` statement would you choose and why?
  - If you wanted to use each item of an existing list inside the loop, which type of `for` statement would you choose and why?
16. (Optional) Write a `for` loop that uses the `range()` function to print the letters A to Z one at a time. *Hint:* In Unicode, 'A' is 65 and 'Z' is 90. Use the `chr()` function inside the loop.

## Model 3 for Each Index

Indexes are needed in order to update the elements of a list.

```
names = ["emma", "liam", "aisha", "mateo", "sofia", "ravi"]

for i in range(len(names)):
    name = names[i]
    # replace element at index i
    names[i] = name.capitalize()
```

### Questions (15 min)

Start time:

17. Based on the above code:

- What is the length of the names list?
- What sequence of values does `range(len(names))` represent?
- What word does the variable `i` stand for?

18. Run the code, and describe what the `for` loop does.

19. Compare the `for` loops in Model 1 and Model 3.

- Why does Model 1 use the variable name `x`?
- Why does Model 3 use the variable name `i`?

20. Can the loop in Model 3 be rewritten, without using `range()`, to look like the loop in Model 1? Explain why or why not.

21. The built-in `enumerate()` function can be used to shorten the code in Model 3. Run the following two lines in a shell, and record the output.

a) `enumerate(names)`

b) `list(enumerate(names))`

22. Describe in your own words what the `enumerate()` function does.

---

*The loop in Model 3 can be rewritten as follows:*

```
for i, name in enumerate(names):  
    # replace element at index i  
    names[i] = name.capitalize()
```

---

23. Write a `for` loop that adds an "!" to each string in the `names` list.

24. Write a `for` loop that increases each value in the `prices` list by 5%. (*Hint: Multiply by 1.05*)  
`prices = [19.99, 6.34, 1.00, 12.79, 2.50]`