

# Lecture 3: For Loops

---

CS 51P

September 11, 2023

# Review: if-statements

- syntax **condition**

```
if x == 13:  
    print("that's my favorite number too!")  
    print("what a coincidence!")  
else:  
    print("mine is 13")
```

 print("what a coincidence!")  
else:  
 print("mine is 13")"/>

- condition must be an expression that evaluates to True or False (type bool)

# for loops

- When you want some set of statements to execute repeatedly

```
for <var> in <sequence>:
```



Code



whitespace  
matters

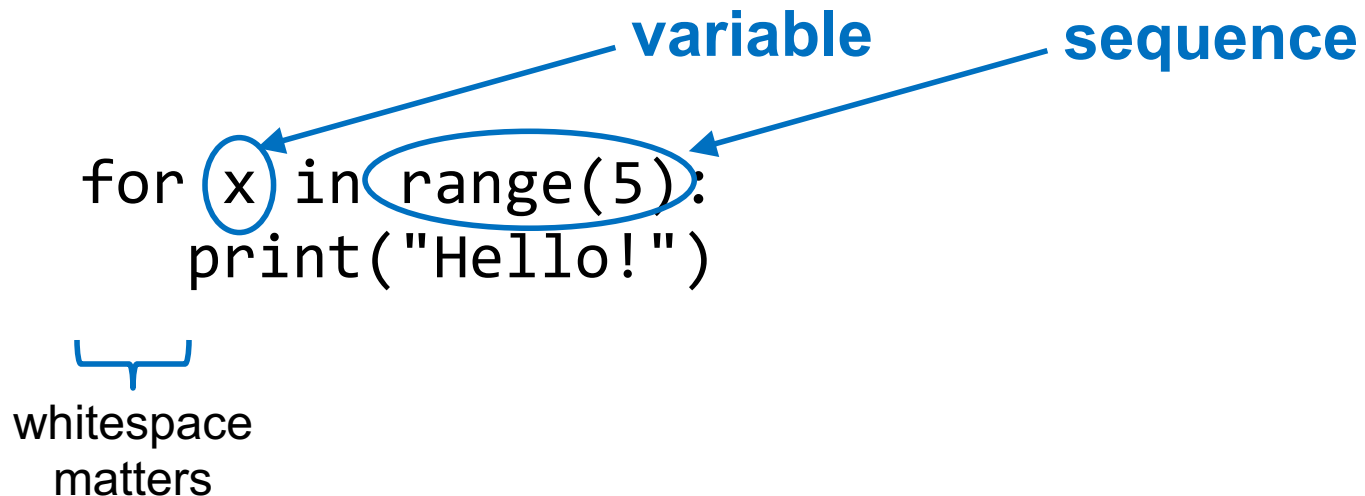
# Example: for loops

- When you want some set of statements to execute repeatedly

**variable**      **sequence**

```
for x in range(5):  
    print("Hello!")
```

whitespace  
matters



The diagram illustrates the components of a Python for loop. The code snippet is: `for x in range(5):` followed by an indented `print("Hello!")`. A blue circle highlights the variable `x`, with an arrow pointing to the label **variable**. Another blue circle highlights the `range(5)` expression, with an arrow pointing to the label **sequence**. A blue bracket is placed under the indentation of the `print` statement, with the text **whitespace matters** below it, indicating that the indentation is crucial for the loop's execution.

# Exercise 1: for loops

Using a for loop, write a program that asks the user for word and a number, then prints the word followed by ! that many times, each on its own line. For example:

Enter a word:

happy

Enter a pos int:

3

happy!

happy!

happy!

# Updates in loops

- You can update variables inside a loop even if they are initially defined outside the loop

```
counter = 13
for x in range(5):
    counter = counter + 1
    print(counter)

print("Final: " + counter)
```

# Exercise 2: loops and accumulators

Using a for loop and **no string duplication**, write a program that asks the user for a positive integer  $n$  and then prints a string containing  $n$  '\*'s.

Enter a pos int:

7

\*\*\*\*\*

# range

- `range ( stop )` generates a sequence of numbers  $0, \dots, \text{stop}-1$
- the number of elements in the sequence determines the number of times the program goes through the loop
- each time through the loop, the index variable gets assigned one of the values in the sequence
- to see the elements, call the function `list`



# Index variables

- Each time a program iterates through the loop, the index variable gets one of the values in the sequence

```
for num in range(5):  
    print(num)
```

# Exercise 3: index variables

Using a for loop, write a program that asks the user for a positive integer and then prints the sum of the odd values between 1 and n.

For example, if the user enters 5, it would print 9 (since  $1 + 3 + 5 == 9$ )

# range

- `range([start,] stop [, step])`
- generates a sequence of numbers
- to see the elements, call the function `list`

`range(5)`

`range(1,10)`

`range(1,15,2)`

`range(1,15,-1)`

`range(10,-5,-3)`

# Exercise 4: ranges

- `range(3)`
- `range(5, 10)`
- `range(5, 0, -1)`
- `range(0, 10, 2)`
- `range(10, 0, 2)`

# Exercise 3: index variables

Using a for loop, write a program that asks the user for a positive integer and then prints the sum of the odd values between 1 and n.

For example, if the user enters 5, it would print 9 (since  $1 + 3 + 5 == 9$ )

How could you modify this code to make the program work without an if statement?