

Objective

- for loop

Lab Review

- Follow examples
 - Find solutions to similar problems
 - Understand the solution
 - Adapt the solution to your problem

Task	Objective
Creating snowperson	Using an API to solve a new problem
Making a picture	Allow you to show your creativity!

- Celebrate your successes!

Lab Review: Preferred Scenario

- You: What about this test case? It's similar to a test case I used in an earlier problem, but it doesn't work here.
- Me: Good! You should consider that test case! But you don't need to demo it because we can't handle that issue yet

Recommendations


- Review the slides, example programs, and/or textbook every day to review what we discussed
 - This problem made sense in class... Does it still make sense?
- Practice a problem every day
 - I rarely use problems from the text book so they're good practice
- Ask questions
- “sense of accomplishment after lab”

Benefits of Object-Oriented Programming

- **Abstraction**
 - Hides details of underlying implementation
 - Easier to change implementation
- Collects related data/methods together
 - Easier to reason about data/write code
- Less code in main program
 - Example: Our program code is relatively simple, but `graphics.py` has a ton of code in it

FOR LOOPS

Parts of an Algorithm

- Input, Output
 - Primitive operations
 - What data you have, what you can do to the data
 - Naming
 - Identify things we're using
 - Sequence of operations
 - Conditionals
 - Handle special cases
 - Repetition/Loops
 - Subroutines
 - Call, reuse similar techniques
- Super Power:
Superhuman Speed
- 

Looping/Repetition

We know how to
make a PB&J
Sandwich:

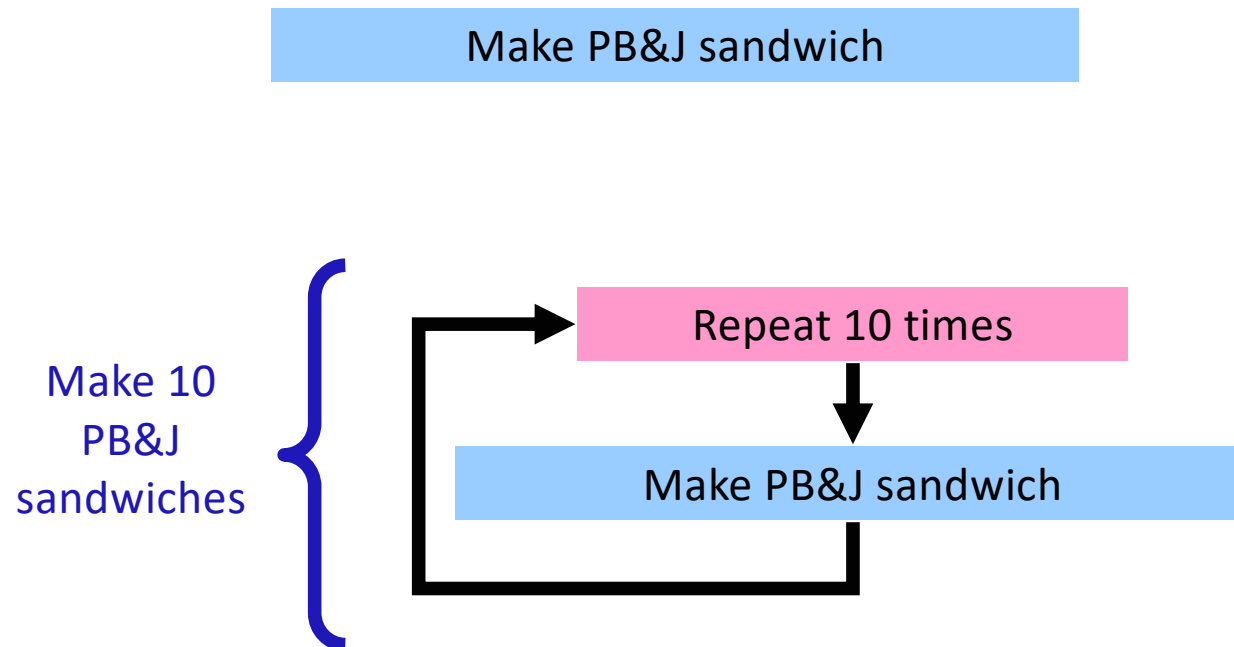
Make PB&J sandwich

Make 10
PB&J
sandwiches

Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich
Make PB&J sandwich

Repetition is common in programming.
Is there some simpler way to say that
we want to repeat something?

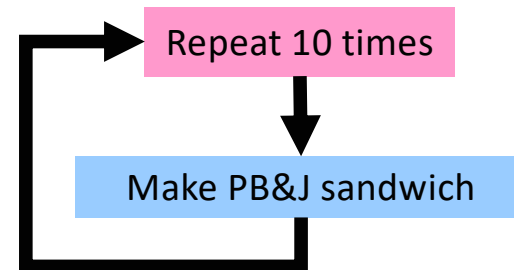
Looping/Repetition



What Goes in the Loop Body?

- Make PB&J Sandwich

1. Gather materials (bread, PB, J, knives, plate)
2. Open bread
3. Put 2 pieces of bread on plate
4. Spread PB on one side of one slice
5. Spread Jelly on one side of other slice
6. Place PB-side facedown on Jelly-side of bread
7. Close bread
8. Clean knife
9. Put away materials



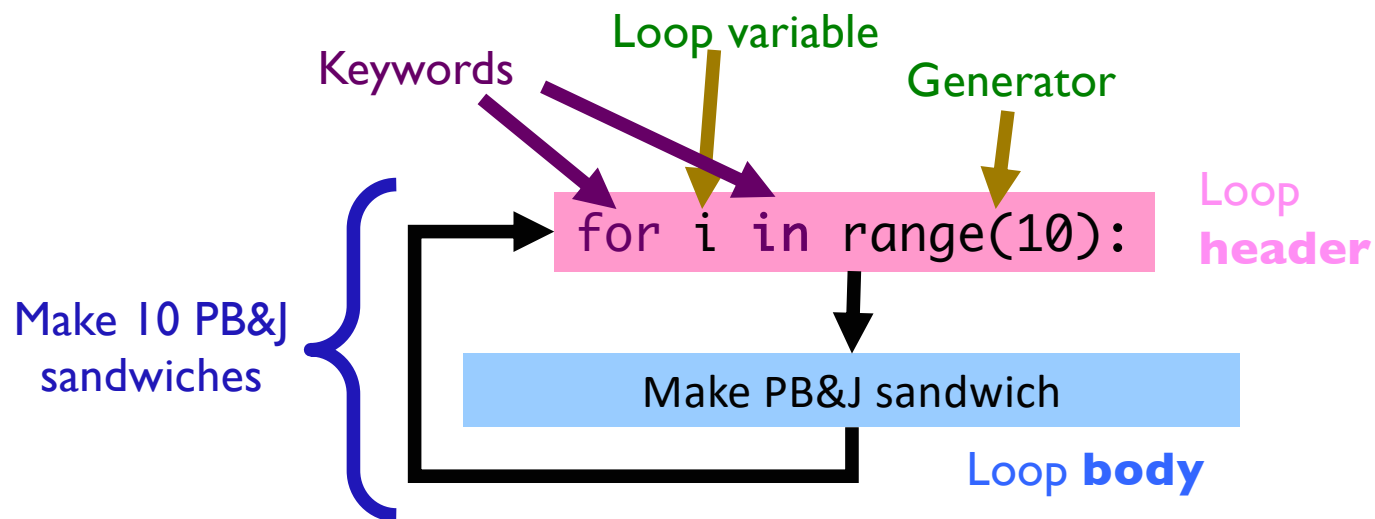
What Goes in the Loop Body?

- Make PB&J Sandwich

- | | |
|---|-----------------------|
| 1. Gather materials (bread, PB, J, knives, plate) | Initialization |
| 2. Open bread | |
| 3. Put 2 pieces of bread on plate | Loop Body |
| 4. Spread PB on one side of one slice | |
| 5. Spread Jelly on one side of other slice | |
| 6. Place PB-side facedown on Jelly-side of bread | |
| 7. Close bread | Finalization |
| 8. Clean knife | |
| 9. Put away materials | |

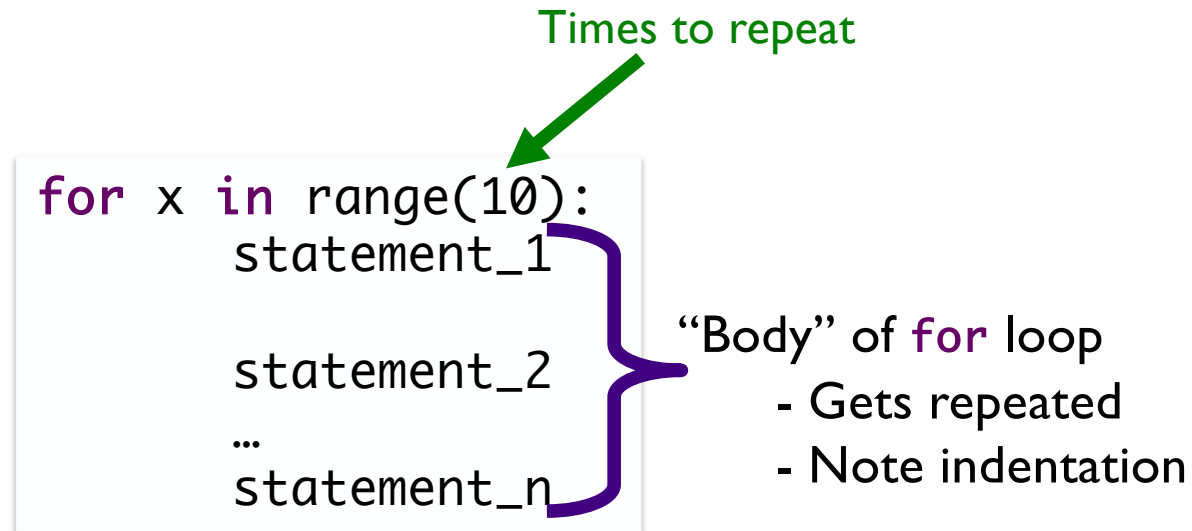
The for Loop

- Use when know how many times loop will execute
 - Repeat N times



for Loop Syntax and Semantics

- Use when know how many times loop will execute
 - Repeat N times



Analyzing `range()`

- `range` is a *generator*
- What does `range` do, exactly, with respect to the loop variable `i`?

```
for i in range(5):  
    print(i)  
    print("Horray!")  
  
print("After the loop:", i)
```

`range_analysis.py`

Analyzing range()

```
    ① Set i to 0
for i in range(5):
    print(i)    ② Display the value of i
    print("Horray!")    ③ Display Horray!

print("After the loop:", i)
```

Analyzing range()

```
    4 Set i to 1
for i in range(5):
    print(i)    5 Display the value of i
    print("Horray!")    6 Display Horray!

print("After the loop:", i)
```


Analyzing range()

```
...  
    13 Set i to 4  
for i in range(5):  
    print(i) 14 Display the value of i  
    print("Horray!") 15 Display Horray!  
  
print("After the loop:", i) 16
```

for loop analysis

```
for i in range(5):  
    # like assigning i the values(0,1,2,3,4)  
    # consecutively, changing each time through loop  
  
    # then, execute the loop body ...
```

- When we have `range(5)`,
 - `i` is set to the values (0, 1, 2, 3, 4)
 - Which means that loop executes 5 times

`range([start,] stop[, step])`

- `[xxx]` means that xxx is optional
- 1 argument: `range(stop)`
- 2 arguments: `range(start, stop)`
- 3 arguments: `range(start, stop, step)`

`range([start,] stop[, step])`

- 1 argument: `range(stop)`
 - Defaults: `start = 0, step = 1`
 - Iterates from 0 to `stop-1` with step size=1
- 2 arguments: `range(start, stop)`
 - Default: `step = 1`
 - Iterates from `start` to `stop-1` with step size=1
- 3 arguments: `range(start, stop, step)`
 - Iterates from `start` to `stop-1` with step size=`step`

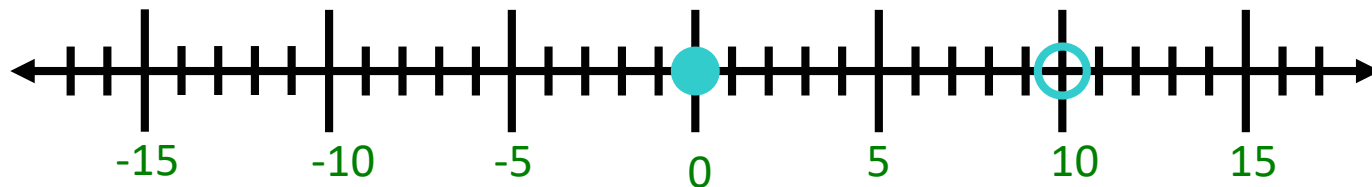
range

- range is a number generator

- 1 argument: `range(stop)`

- 2 arguments: `range(start, stop)`

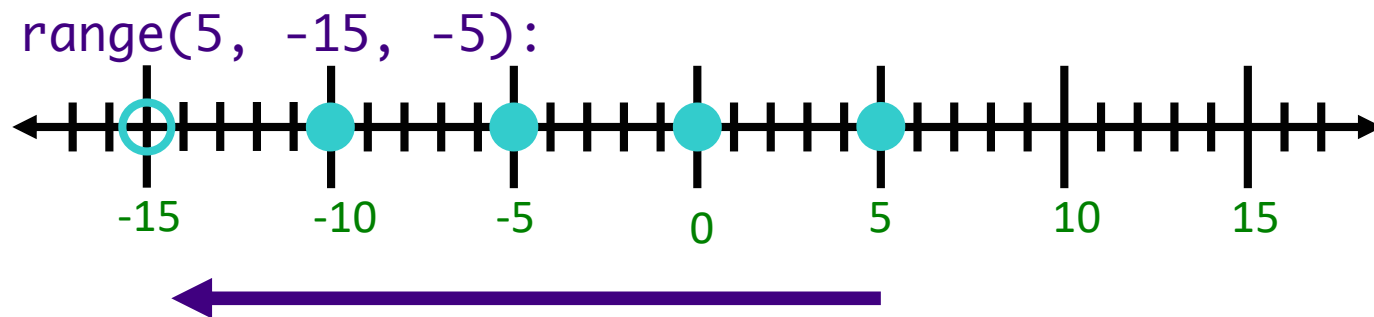
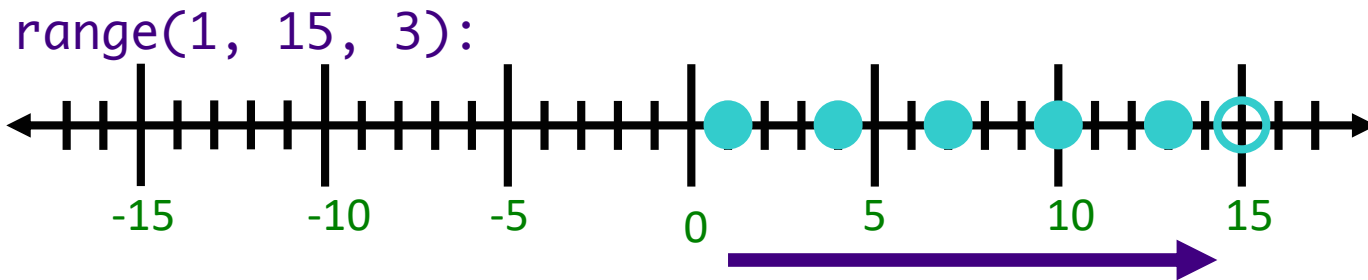
- 3 arguments: `range(start, stop, step)`



[start, stop)

`range(10)`
`range(0, 10)`
`range(0, 10, 1)`

Sequence generated by range



[more_range_examples.py](#)

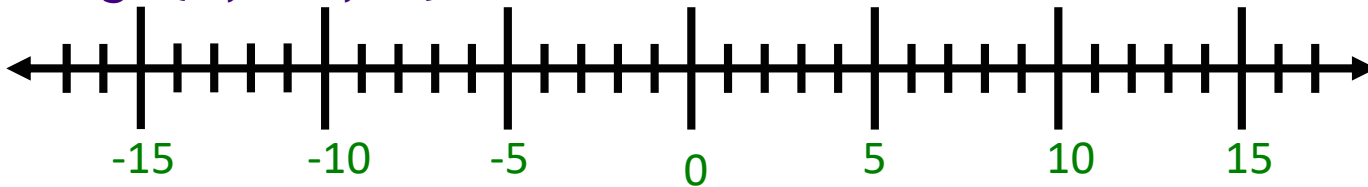
Practice

Place these:

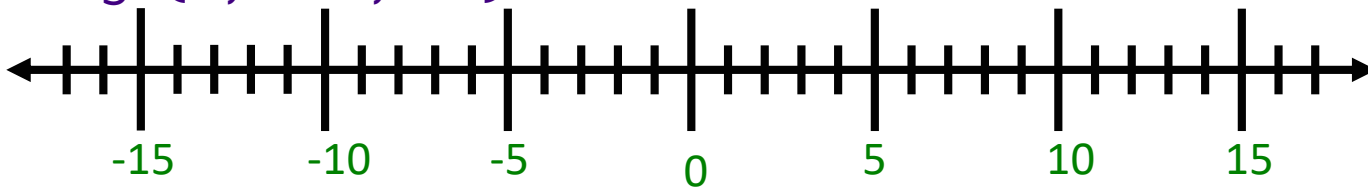


Which direction?

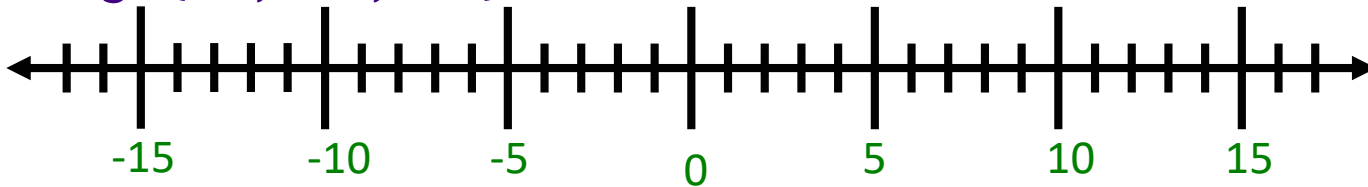
$\text{range}(2, 14, 2):$



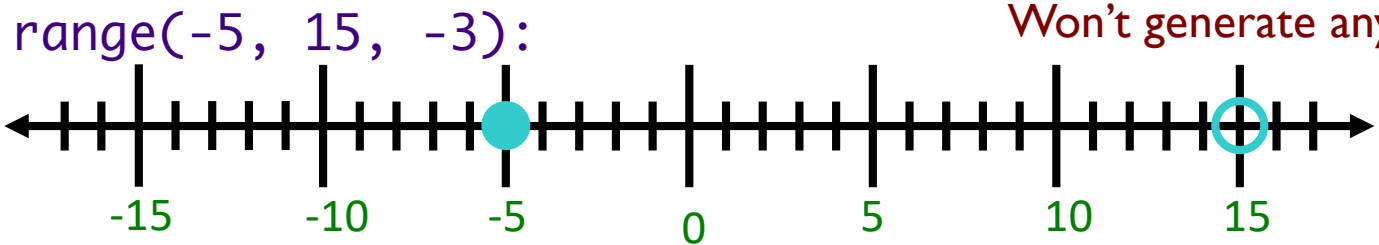
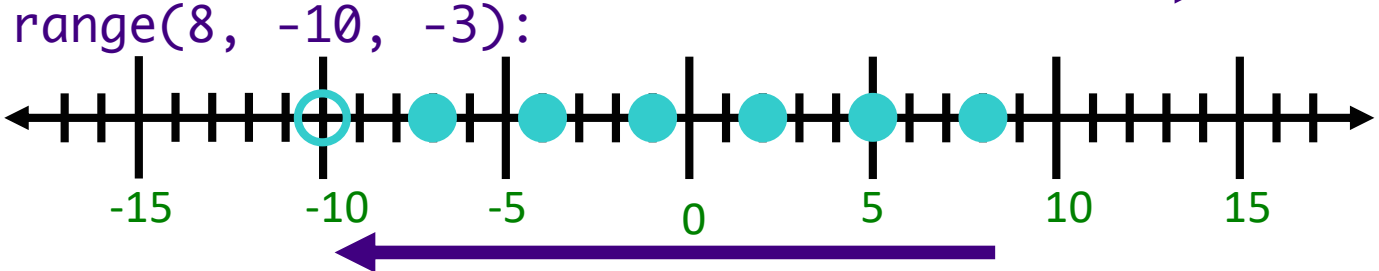
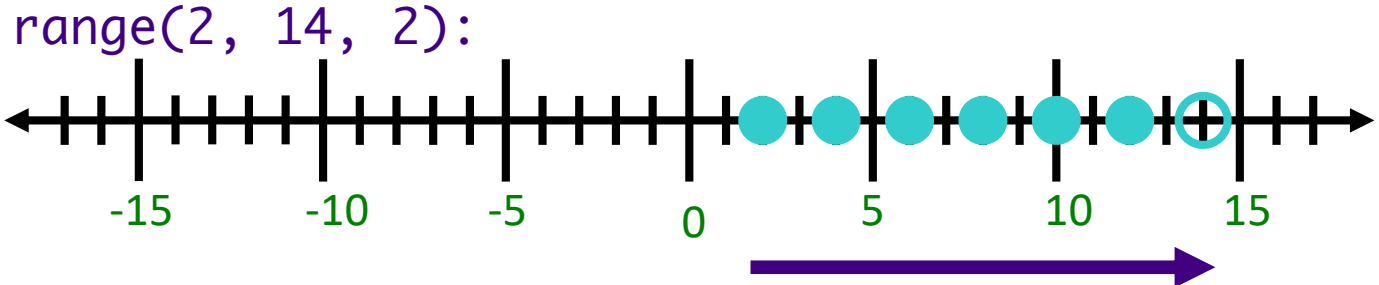
$\text{range}(8, -10, -3):$



$\text{range}(-5, 15, -3):$



Practice Solution



Practicing **for** Loops

- Write the Python code to display the following:

➤ A) 1
2
3
4
5

➤ B) 2
5
8
11

➤ C) ****

Questions to ask:

- What is getting repeated?
- How many times?

How do the answers to those questions inform your solution?

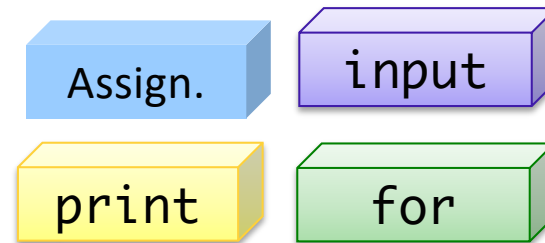
Using **for** loops in Programs

- Use a **for** loop when you want to repeat something
- Process of solving loop problems
 - What is getting repeated?
 - Informs what goes in the *loop body*
 - How many times?
 - Informs what the arguments to range should be

Programming Building Blocks

- Adding to your tool set!
- We can combine them to create more complex programs

➤ Solutions to problems



Practice

- Problem: Add 5 numbers, inputted by the user
 - We could have implemented this program before learning loops, BUT we want to apply what we learned today.
- Consider what program *should* do – example behavior
 - Consider if problem specification changes to adding 3 numbers or 10 numbers
- After implementing solution, simulate running on computer
 - You can pretend to be the computer

Generalizing Solution: Accumulator Design Pattern

1. Initialize accumulator variable
2. Loop until done
 - Update the value of the accumulator
3. Display result

Discussion: Programming Practice

- Problem: Add 5 numbers, inputted by the user
- We could have implemented this program last week
 - 5 separate input statements, add up the numbers
- Consider how much easier this program is to change if we want a different number of numbers added up

Practicing **for** Loops

What is getting repeated?
How many times?

- A) 1
2
3
4
Tell me that you
love me more

- C) 10
9
8
7
...
1
Blast off!

- B) I had the time of my life
And I never felt this way before
And I swear this is true
And I owe it all to you

} 3 times,
followed by Dirty bit

Looking Ahead

- Broader Issue: Algorithm Bias due Thursday at 11:59 p.m.
- Lab 2 due Friday before class