Objectives

- Review: importing modules
- Intro to design patterns
- Definite loops

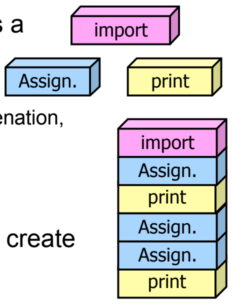
Review: Python Libraries

- Python has a rich library of functions and definitions available for your use
  - The library is broken into **modules**
  - A **module** is a file containing Python definitions and statements
- To use a module's definitions, use an **`import`** statement
  - Goes at the top of your program (after the first comments)
- **Benefits** of functions/definitions in modules
  - Don't need to rewrite someone else's code
  - If it's in a module, it is very efficient (in terms of computation speed and memory usage)

assign_constant.py

Programming Building Blocks

- Each type of statement is a building block
  - Initialization/Assignment
    - Arithmetic, string concatenation, functions
  - Print
  - Import
- We can combine them to create more complex programs
  - Solutions to problems

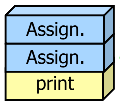


Design Patterns

- General, repeatable solution to a commonly occurring problem in software design
  - Template for solution

Design Patterns

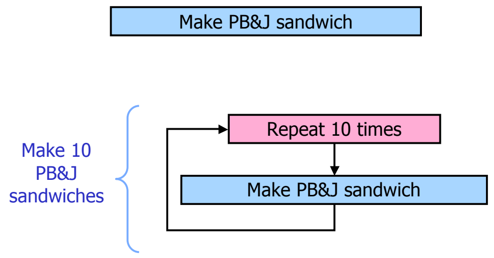
- General, repeatable solution to a commonly occurring problem in software design
  - Template for solution
- Example (**Standard Algorithm**)
  - Get input from user
  - Do some computation
  - Display output



```
x = input("…")
ans = …
print ans
```

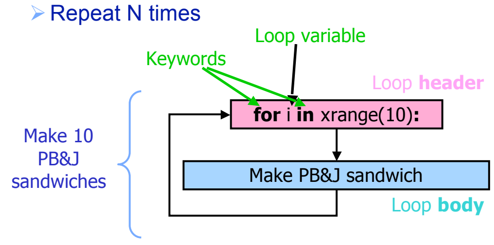
- **Today**: learn new building block, new design pattern

Looping/Repetition



The **for** Loop

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



Using the **For** Loop

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

Times to repeat

```
for i in xrange(10):
    statement_1
    statement_2
    …
    statement_n
```

- "Body" of for loop
- Gets repeated
- Note indentation

Using the **For** Loop

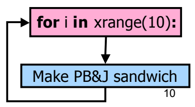
- If only ***one*** statement to repeat

```
for i in xrange(5): print "Hello!"
```

simple_for.py

What Goes in the Loop Body?

- Make PB&J Sandwich
  - Gather materials (bread, PB, J, knives, plate)
  - Open bread
  - Put 2 pieces of bread on plate
  - Spread PB on one side of one slice
  - Spread Jelly on one side of one slice
  - Place PB-side facedown on Jelly-side of bread
  - Close bread
  - Clean knife
  - Put away materials



What Goes in the Loop Body?

- Make PB&J Sandwich
  - Gather materials (bread, PB, J, knives, plate)
  - Open bread

Initialization

Loop Body

  - Put 2 pieces of bread on plate
  - Spread PB on one side of one slice
  - Spread Jelly on one side of one slice
  - Place PB-side facedown on Jelly-side of bread

  - Close bread
  - Clean knife
  - Put away materials

Finalization

Using the **For** Loop

- Good for when know how many times loop will execute
  - Repeat N times

Times to repeat

```
for i in xrange(10):
    statement_1
    statement_2
    …
    statement_n
```

- "Body" of for loop
- Gets repeated
- Note indentation

Analyzing `xrange()`

- `xrange` is a built-in function
- What does `xrange` do, exactly?
  - Simulate on paper

xrange_analysis.py

xrange([start,] stop[, step])

- What does the above signature mean?

xrange([start,] stop[, step])

- 1 argument: xrange(stop)
- 2 arguments: xrange(start, stop)
- 3 arguments: xrange(start, stop, step)

using_xrange.py

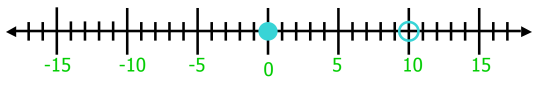
xrange([start,] stop[, step])

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

using_xrange.py

`xrange()`

- `xrange` is a built-in function
  - 1 argument: xrange(stop)
  - 2 arguments: xrange(start, stop)
  - 3 arguments: xrange(start, stop, step)

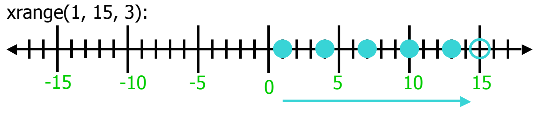


xrange(10)
xrange(0,10)

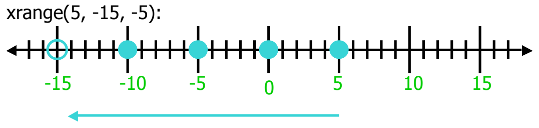
[start, stop)

`xrange()`

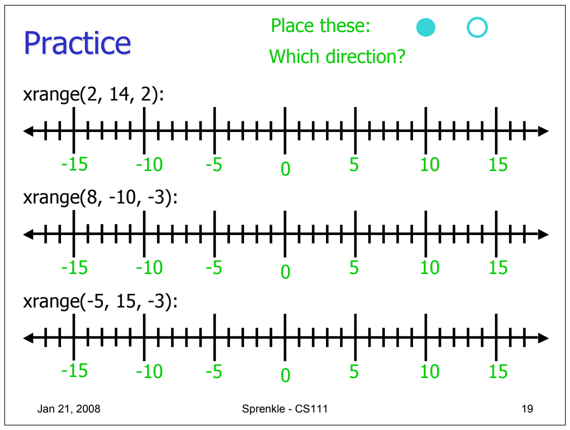
xrange(1, 15, 3):



xrange(5, -15, -5):



new_for.py



Practice

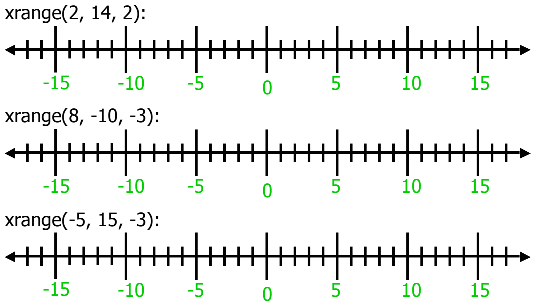
Place these: ● ○

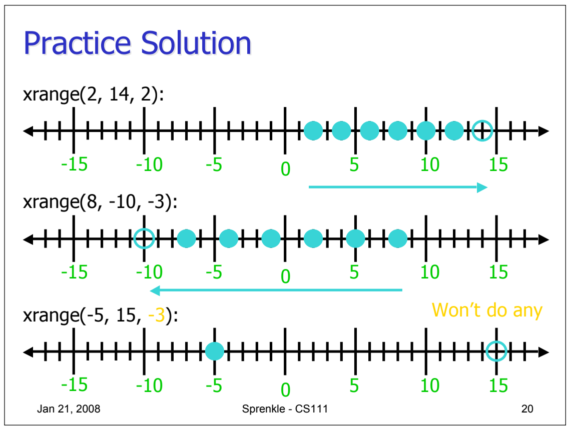
Which direction?

xrange(2, 14, 2):

xrange(8, -10, -3):

xrange(-5, 15, -3):



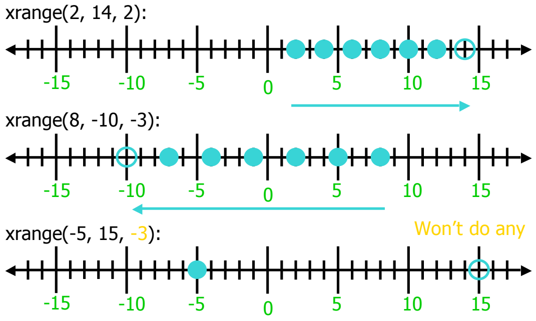


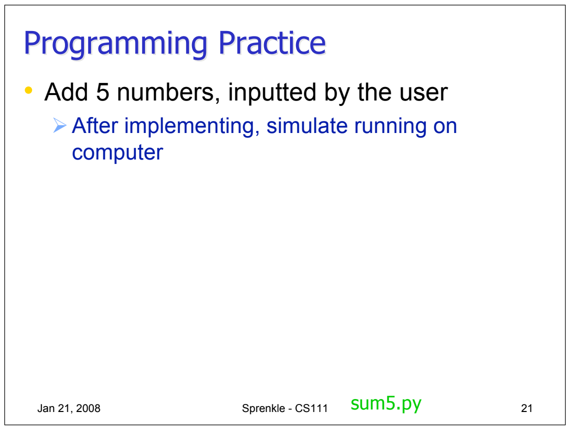
Practice Solution

xrange(2, 14, 2):

xrange(8, -10, -3):

xrange(-5, 15, -3):

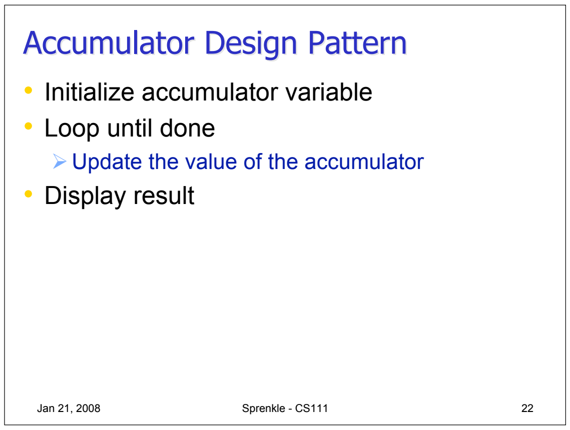




Programming Practice

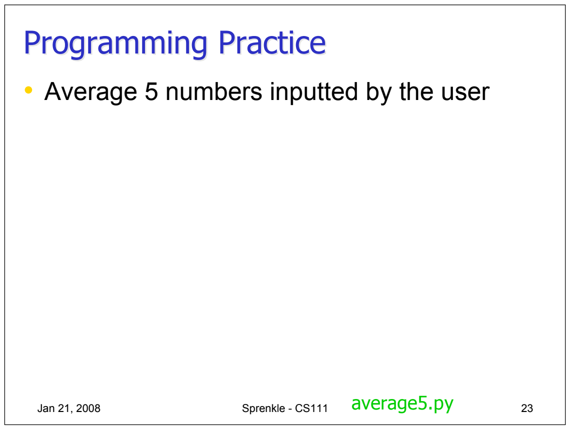
- Add 5 numbers, inputted by the user
  - After implementing, simulate running on computer

sum5.py



Accumulator Design Pattern

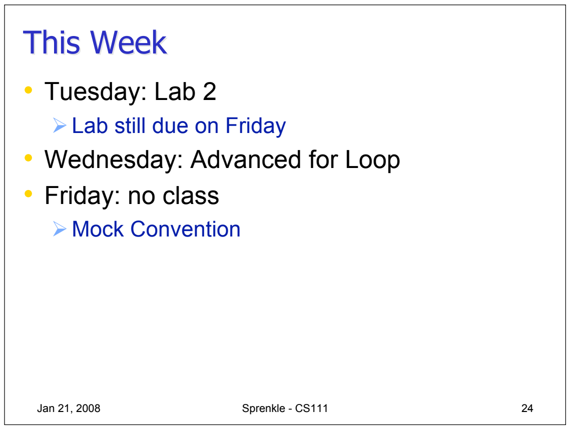
- Initialize accumulator variable
- Loop until done
  - Update the value of the accumulator
- Display result



Programming Practice

- Average 5 numbers inputted by the user

average5.py



This Week

- Tuesday: Lab 2
  - Lab still due on Friday
- Wednesday: Advanced for Loop
- Friday: no class
  - Mock Convention