Objective

- More for loop
- Designing for Change
- Using Functions

Lab Review

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

Task	Objective
Creating a Text object	Confirming that you know how to use the API, using a class that you hadn’t used previously.
Making a picture	Allow you to show creativity

Recommendations

- Review the slides, example programs, and/or text book 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"

Review

- Which lab are we working on?
  - How many have you completed?
- What statement do we use to repeat something?
- What are the possible ways to use the `range` function?
  - What do they mean?

Practicing for Loops

What is getting repeated?
How many times?

- A) 1
  2
  3
  4
  Tell me that you
  love me more
- 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
- C) 10
  9
  8
  7
  …
  1
  Blast off!

Programming Practice

- Add 5 numbers, inputted by the user
  - After implementing, simulate running on computer
- How would have implemented this last week?
  - How can we improve that based on our new knowledge?

`sum5.py`

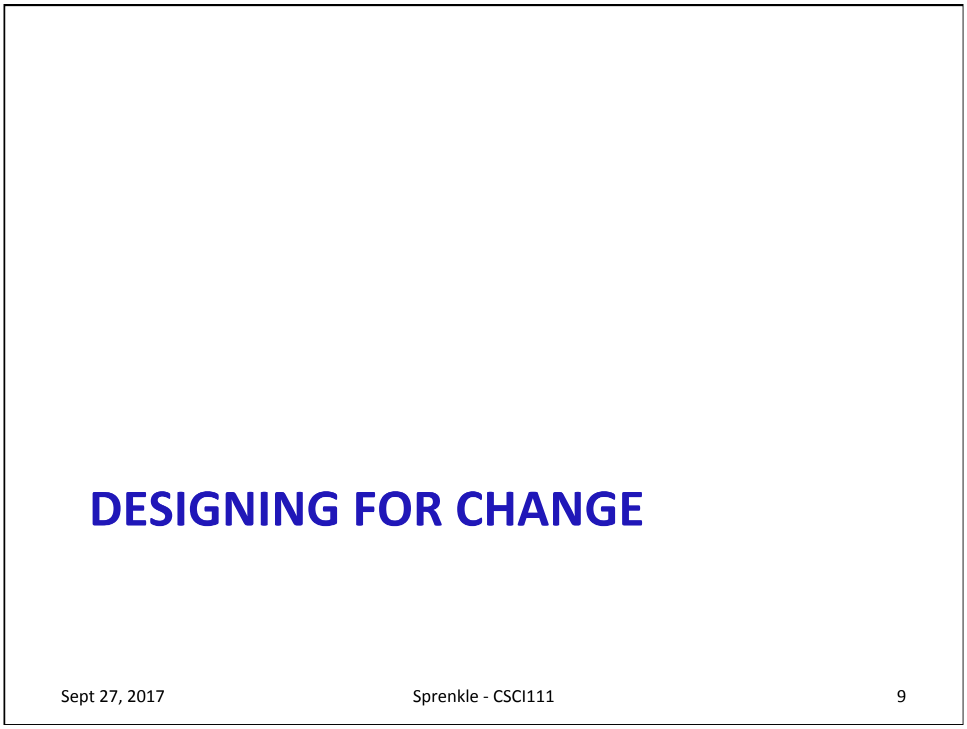
Generalizing Solution: Accumulator Design Pattern

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

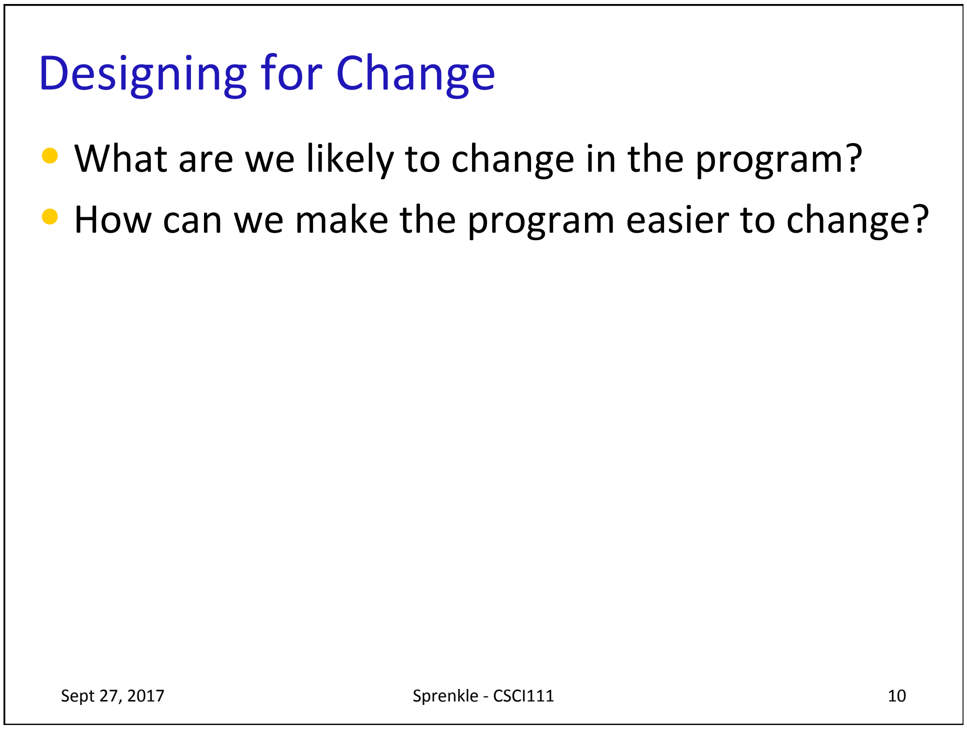
Programming Practice at Home

- Average 5 numbers inputted by the user
- Good example of how to build up to a solution
  - Break down into smaller pieces

average5.py

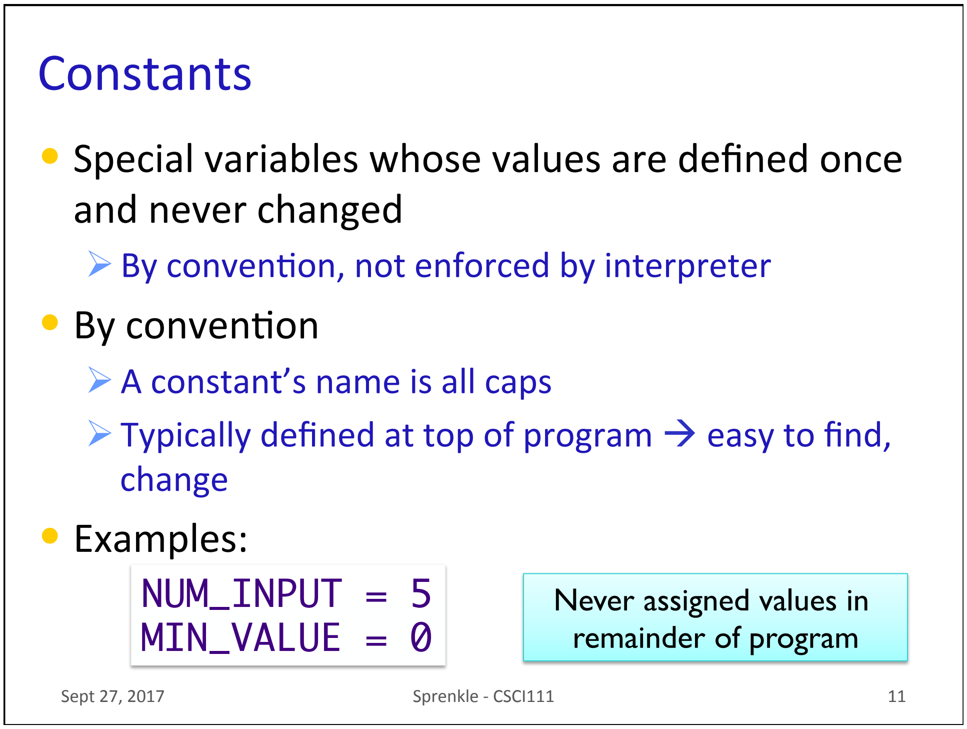


DESIGNING FOR CHANGE



Designing for Change

- What are we likely to change in the program?
- How can we make the program easier to change?

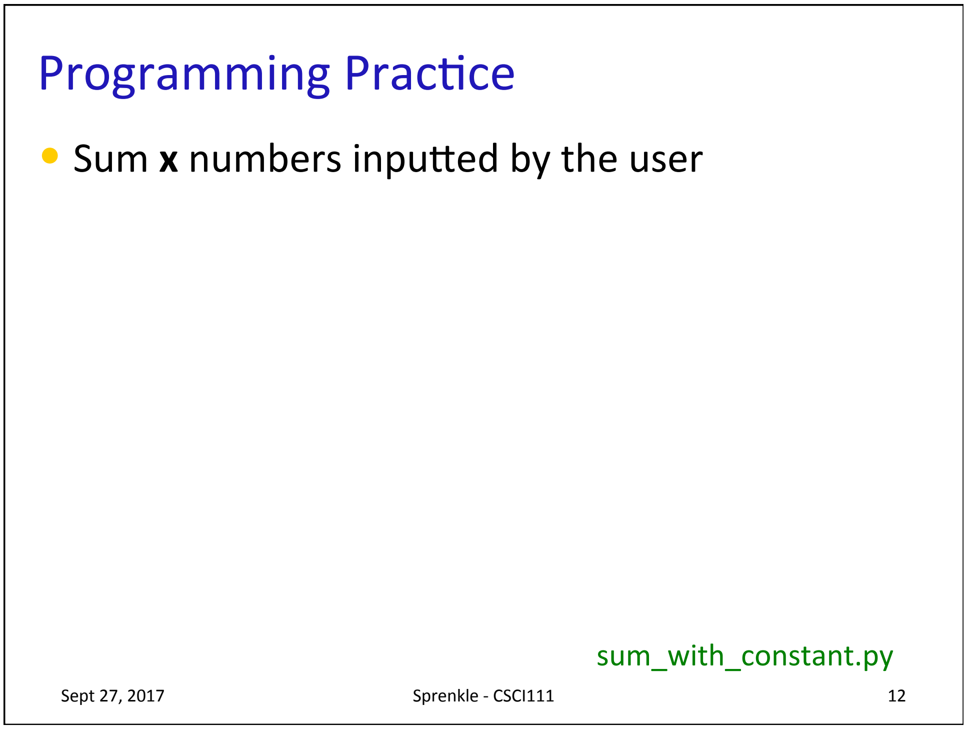


Constants

- Special variables whose values are defined once and never changed
  - By convention, not enforced by interpreter
- By convention
  - A constant’s name is all caps
  - Typically defined at top of program → easy to find, change
- Examples:

```
NUM_INPUT = 5
MIN_VALUE = 0
```

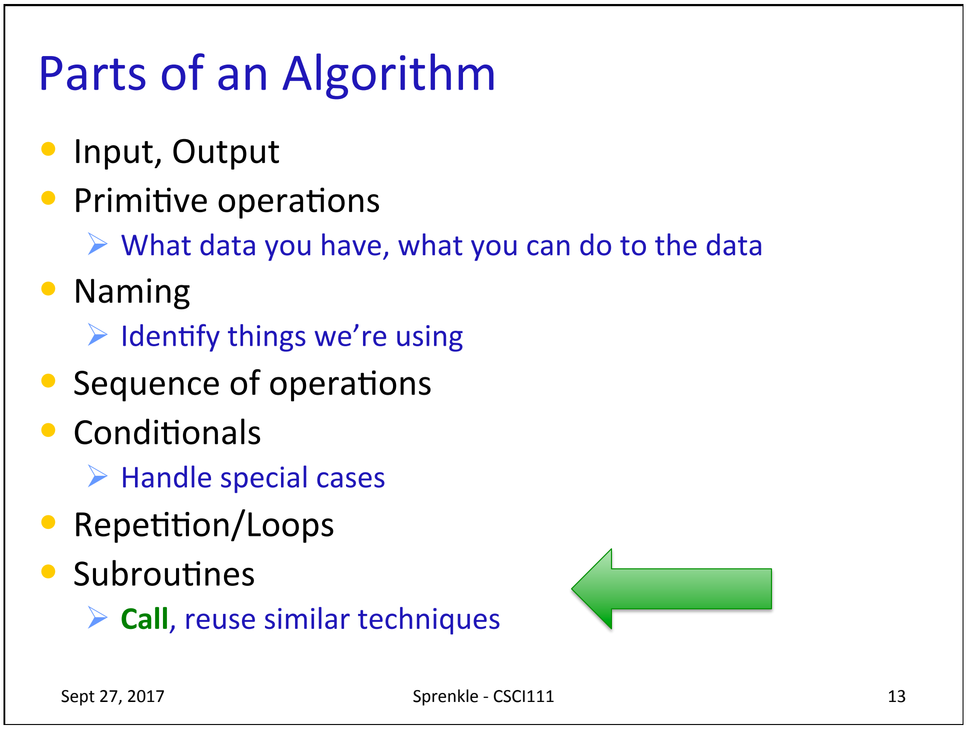
Never assigned values in remainder of program



Programming Practice

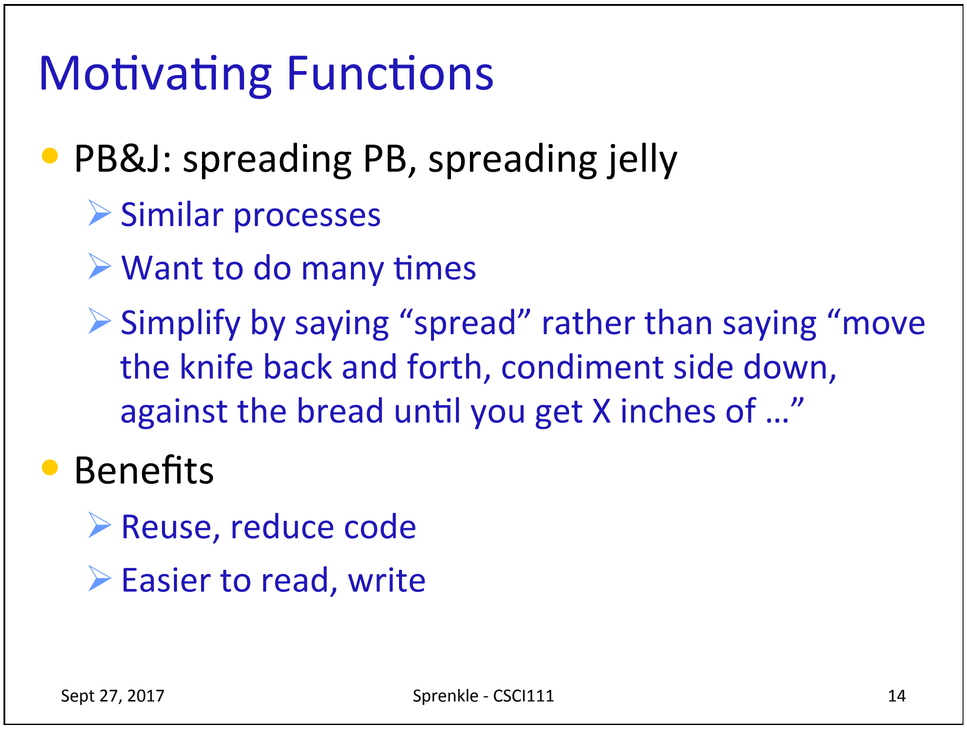
- Sum **x** numbers inputted by the user

sum_with_constant.py



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

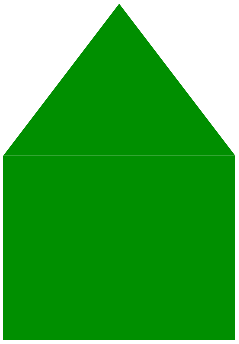


Motivating Functions

- PB&J: spreading PB, spreading jelly
  - Similar processes
  - Want to do many times
  - Simplify by saying "spread" rather than saying "move the knife back and forth, condiment side down, against the bread until you get X inches of …"
- Benefits
  - Reuse, reduce code
  - Easier to read, write

Example

- How would you find the area of this shape?

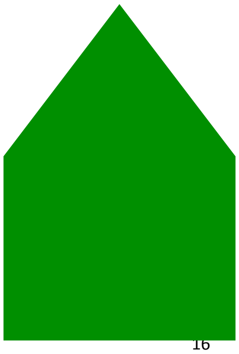


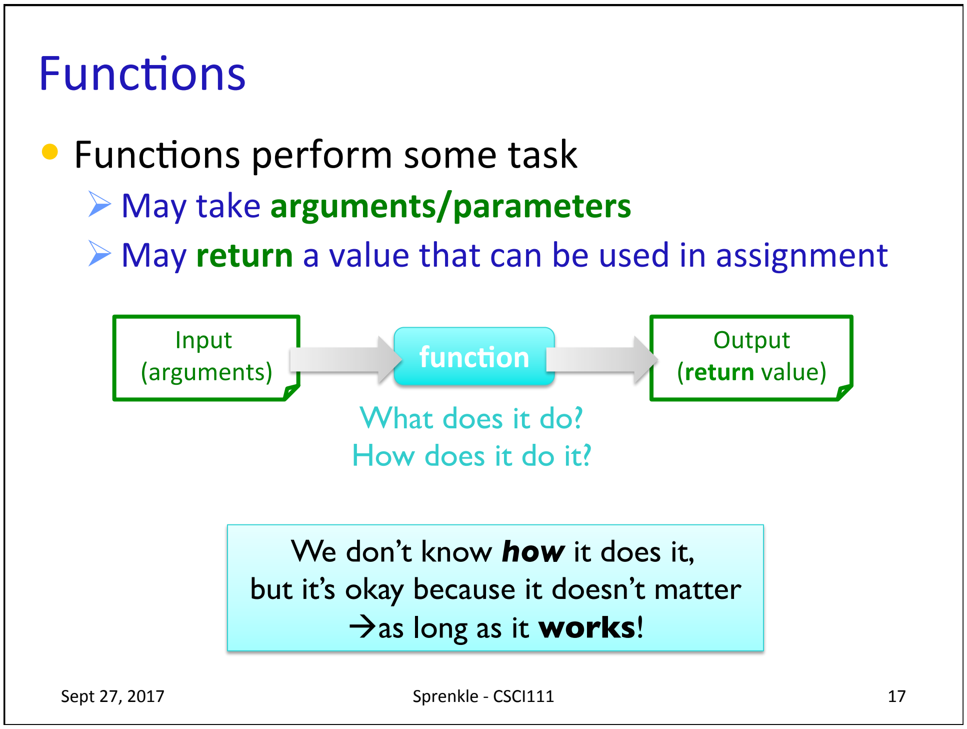
Example

- How would you find the area of this shape?
- Algorithm Possibilities:
  - Total Area = $\frac{1}{2} b_t h_t + w_r * h_r$
  - Total Area = Area of triangle + Area of rectangle

Which algorithm is easier to understand?

For (most) humans,
words and abstraction of ideas
are easier to understand





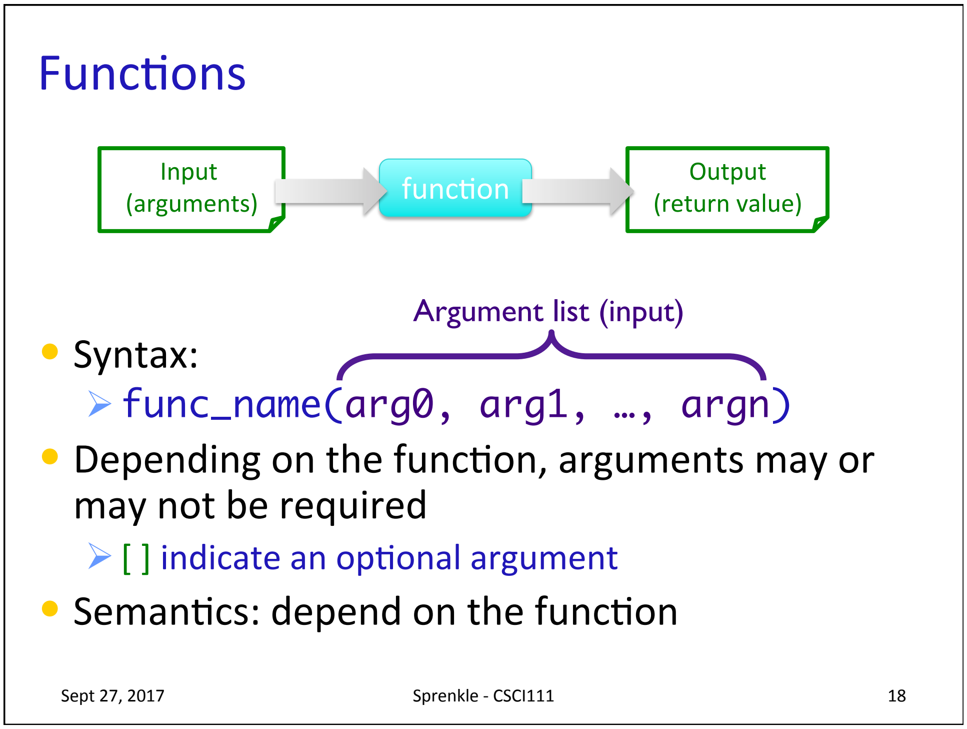
Functions

- Functions perform some task
  - May take **arguments/parameters**
  - May **return** a value that can be used in assignment

Input (arguments) → function → Output (**return** value)

What does it do?
How does it do it?

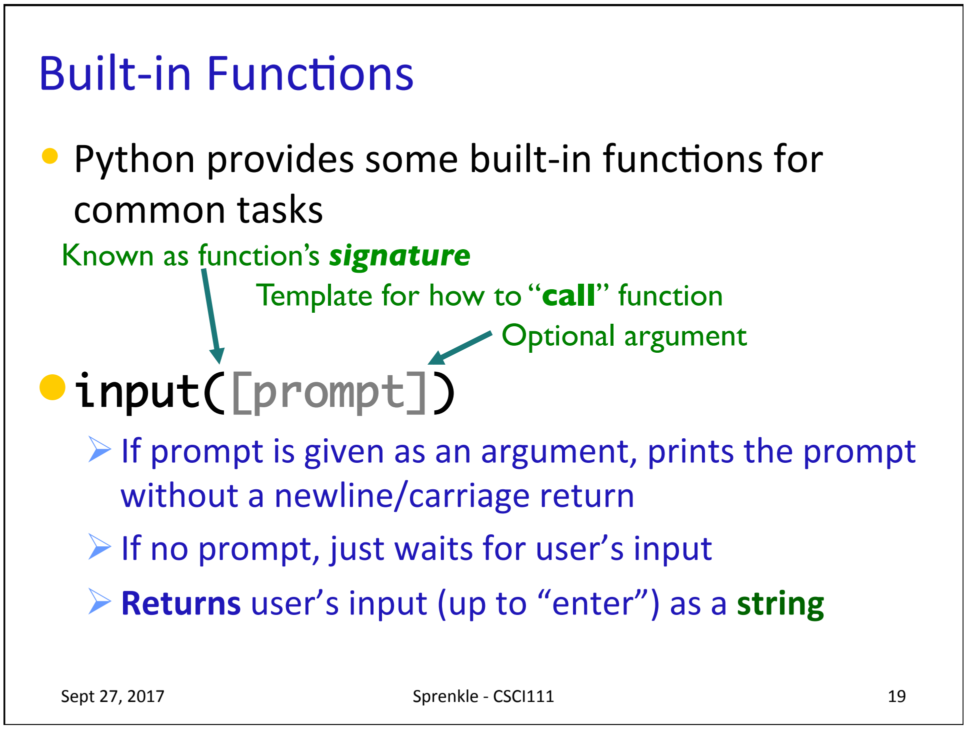
We don't know ***how*** it does it,
but it's okay because it doesn't matter
→as long as it **works**!



Functions

Input (arguments) → function → Output (return value)

- Syntax:
  - $\underbrace{\texttt{func\_name(arg0, arg1, …, argn)}}$ Argument list (input)
- Depending on the function, arguments may or may not be required
  - [ ] indicate an optional argument
- Semantics: depend on the function



Built-in Functions

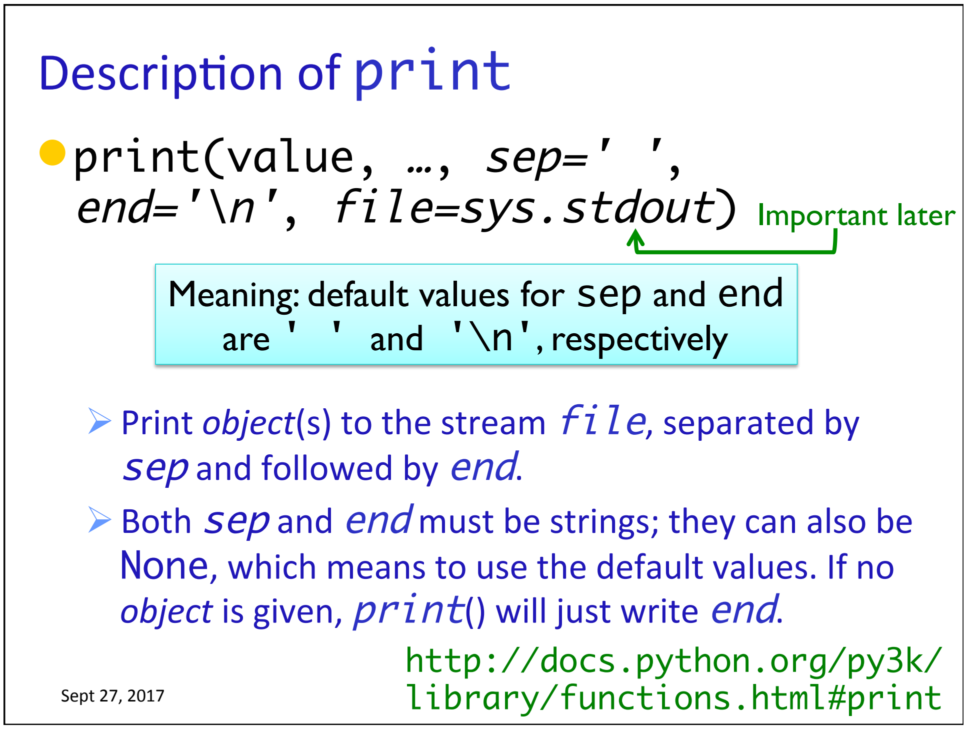
- Python provides some built-in functions for common tasks

Known as function's ***signature***

Template for how to "**call**" function

Optional argument

- `input([prompt])`
  - If prompt is given as an argument, prints the prompt without a newline/carriage return
  - If no prompt, just waits for user's input
  - **Returns** user's input (up to "enter") as a **string**



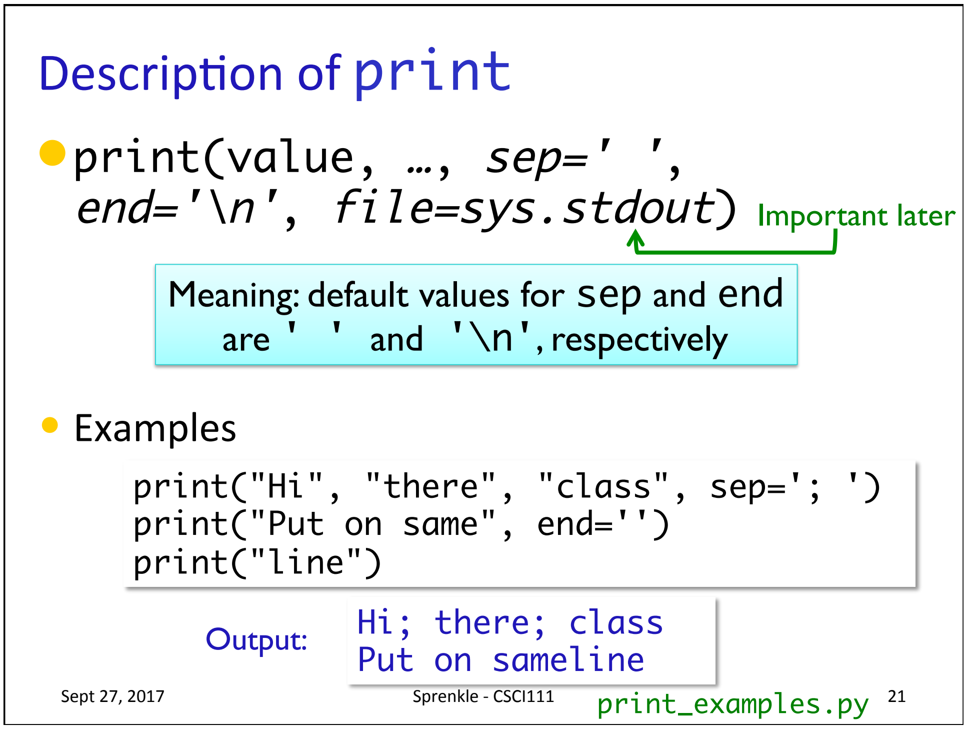
Description of `print`

- `print(value, …, sep=' ', end='\n', file=sys.stdout)` Important later

Meaning: default values for `sep` and `end` are `' '` and `'\n'`, respectively

  - Print *object*(s) to the stream *`file`*, separated by *`sep`* and followed by *`end`*.
  - Both *`sep`* and *`end`* must be strings; they can also be `None`, which means to use the default values. If no *object* is given, *`print`*() will just write *`end`*.

`http://docs.python.org/py3k/library/functions.html#print`



Description of `print`

- `print(value, …, sep=' ', end='\n', file=sys.stdout)` Important later

Meaning: default values for `sep` and `end` are `' '` and `'\n'`, respectively

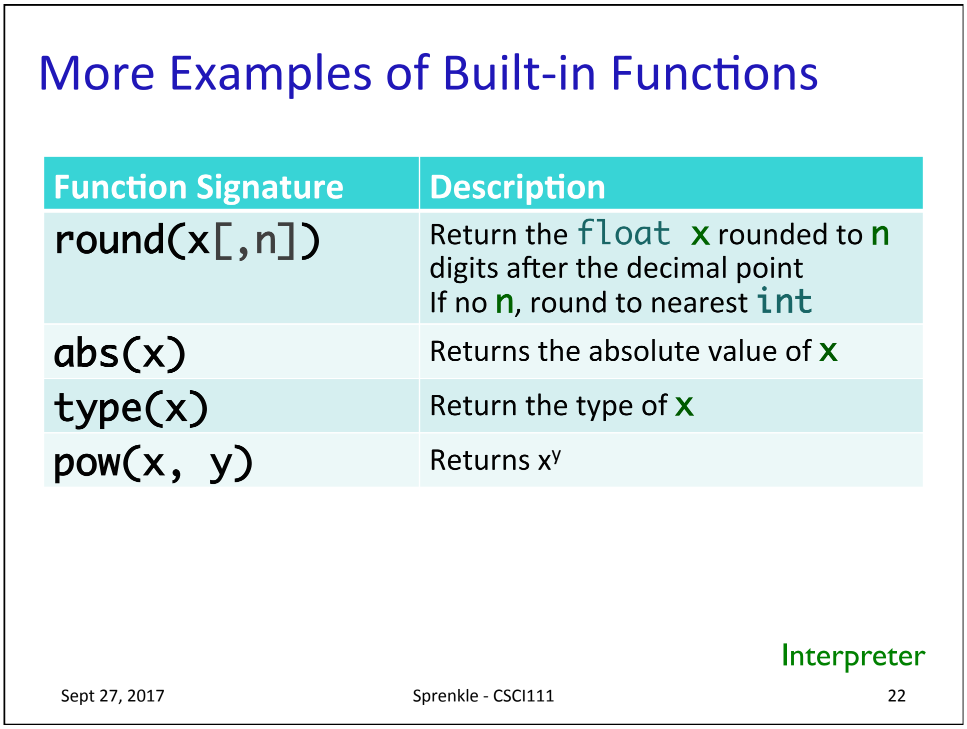
- Examples

```
print("Hi", "there", "class", sep='; ')
print("Put on same", end='')
print("line")
```

Output:

```
Hi; there; class
Put on sameline
```

print_examples.py



More Examples of Built-in Functions

Function Signature	Description
`round(x[,n])`	Return the `float` `x` rounded to `n` digits after the decimal point If no `n`, round to nearest `int`
`abs(x)`	Returns the absolute value of `x`
`type(x)`	Return the type of `x`
`pow(x, y)`	Returns x^y

Interpreter

Using Functions

- Example use: Alternative to exponentiation
  - Objective: compute -3^2
  - Python alternatives:
    - pow(-3, 2)
    - (-3) ** 2
- We often use functions in assignment statements
  - Function does something
  - Save the output of function (what is ***returned*** in a variable

```
roundedX = round(x)
```

function_example.py

Looking Ahead

- Lab 2 due Friday
- BI: Facebook issues due Friday