Lecture 6:

MATHEMATICAL OPERATORS AND FORMATTING

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Announcements 1/2

MassMutual

Data Science Development Program

Announcements 2/2

FROM CLASSROOM TO CONFERENCE ROOM:

THE NUTS AND BOLTS OF A DATA SCIENCE CAREER

Emily Dodwell (Smith `11) a Senior Inventive Scientist in the Data Science and AI Research organization at AT&T Labs, will provide suggestions for pursuing a career in data science, informed by lessons she learned along the way. She will highlight steps that undergraduate students can take to build a professional network, tips to evaluate and apply for data science jobs, and considerations to translate coursework to an industry environment. She will also introduce a current project at AT&T Labs to demonstrate a data science workflow and the importance of ongoing professional development in the field.

Monday September 24, 12:10-1:10 pm Ford Hall 240.

Pizza lunch will be served. Sponsored by Smith College SDS program

About the assignment rubric

- Assignments are generally worth 10 points in this course
- For assignment 2, these points fall into 3 categories:

Style points (3)

The submission:

- includes a header including both partners' names, the filename, and a description of the program
- uses appropriate, informative variable names
- runs without syntax errors

About the assignment rubric

- Assignments are generally worth 10 points in this course
- For assignment 2, these points fall into 3 categories:

First level (5 points)

The calculator correctly performs the following mathematical operations:

- addition
- subtraction
- multiplication
- division
- exponentiation

About the assignment rubric

- Assignments are generally worth 10 points in this course
- For assignment 2, these points fall into 3 categories:

Second level (2 points)

The calculator correctly correctly parses the user's input, including:

- successful interpretation of the two numeric values
- correctly parsing the selected mathematical operation, differentiating between integers and floats when appropriate

About the assignment rubric

- Assignments are generally worth 10 points in this course
- For assignment 2, these points fall into 3 categories
- Recommendation: **go in order**
  - Start with first level and style points
  - *Then* go on to second level points
  - And *then* attempt bonus points (if applicable)

Overview of the week

- ✓More about variables
- ✓Numeric values and basic operations
- More mathematical operators
  - Revisiting ints and floats
  - The `math` module
  - Formatting
- Lab: Cash Machine
- Conditional (“if”) statements

(RECAP) Core concept 2: numeric values

- Two kinds of **numbers** in CS:
  - integers (“whole numbers”)
  - floats (“decimals” or “floating point numbers”)
- Basic **operators**:
  - addition: `+`
  - subtraction: `-`
  - multiplication: `*`
  - division: `/`
  - integer division: `//`
  - exponentiation: `**` (power)
  - modular arithmetic: `%` (modulo)

(RECAP) Core concept 2: numeric values

- Two kinds of **numbers** in CS:
  - integers ("whole numbers")
  - floats ("decimals" or "floating point numbers")
- Basic **operators**:
  - addition: `+`
  - subtraction: `-`
  - multiplication: `*`
  - division: `/`
  - integer division: `//`
  - exponentiation: `**` (power)
  - modular arithmetic: `%` (modulo)

Reviewing integer operators: `//` and `%`

What is the result of the following operations?

```
21 // 5       # 4
21 % 5        # 1
9 // 3        # 3
9 % 3         # 0
13 // 5       # 2
13 % 5        # 3
139 // 20     # 6
139 % 20      # 19
```

Built-in functions that work on numbers

- `abs(x)` # return the absolute value of x
- `float(x)` # return x parsed as a float
- `int(x)` # return x parsed as an int
- `max(…)` # return the largest of a list of numbers
- `min(…)` # return the smallest of a list of numbers
- `round(x[, n])` # return *x* rounded to *n* digits after the
  # decimal point. If *n* is omitted, it
  # returns the nearest integer value
- `sum(…)` # return the sum of a list of numbers

Aside: what does **parsed** mean?

- `abs(x)` # return the absolute value of x
- `float(x)` # return x parsed as a float
- `int(x)` # return x parsed as an int
- `max(…)` # return the largest of a list of numbers
- `min(…)` # return the smallest of a list of numbers
- `round(x[, n])` # return *x* rounded to *n* digits after the
  # decimal point. If *n* is omitted, it
  # returns the nearest integer value
- `sum(…)` # return the sum of a list of numbers

Aside: what does **`return`** mean?

- `abs(x)` # return the absolute value of x
- `float(x)` # return x parsed as a float
- `int(x)` # return x parsed as an int
- `max(…)` # return the largest of a list of numbers
- `min(…)` # return the smallest of a list of numbers
- `round(x[, n])` # return *x* rounded to *n* digits after the
  # decimal point. If *n* is omitted, it
  # returns the nearest integer value
- `sum(…)` # return the sum of a list of numbers

RECAP: Keywords

- Some words in Python* are reserved as keywords, and cannot be used as a variable name:

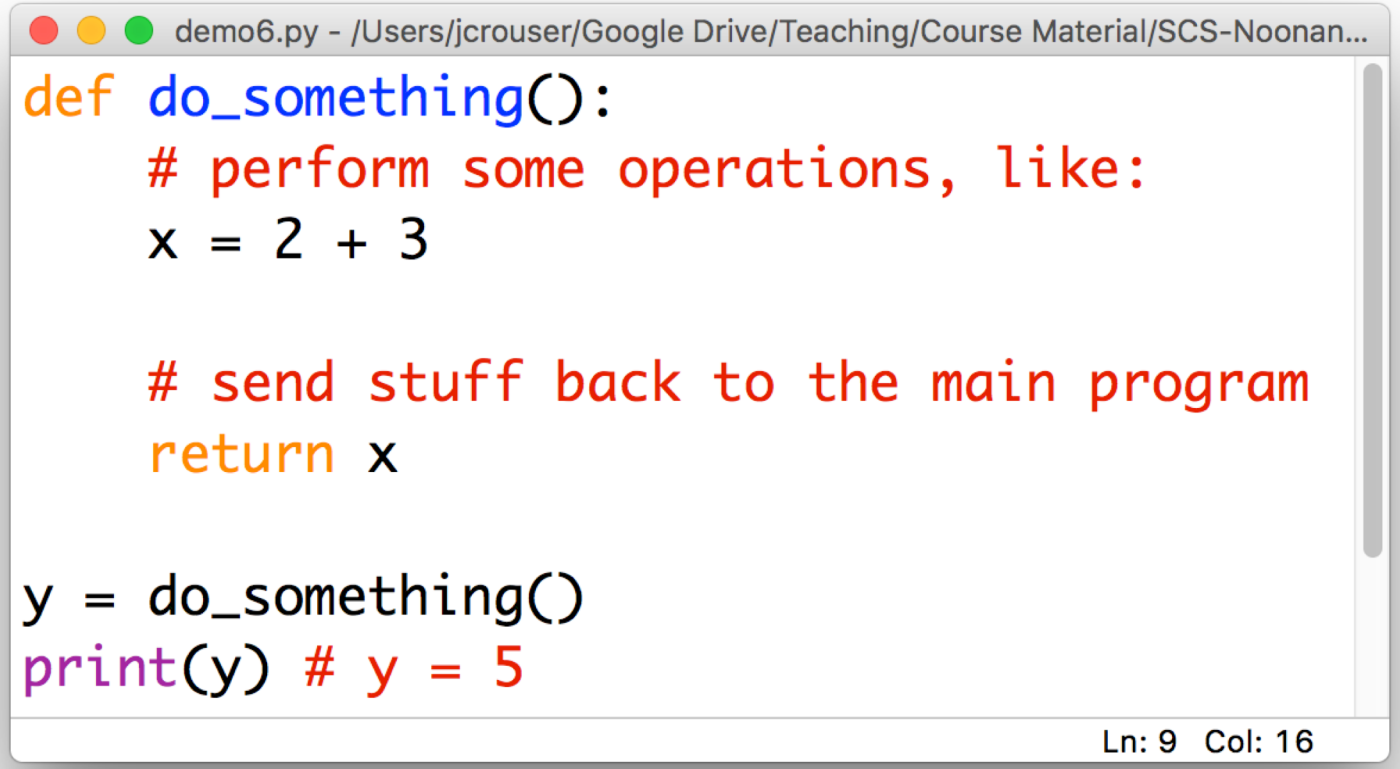
```
and as assert break class continue def del elif else
except exec finally for from global if import in is lambda
not or pass raise return try while with yield
```

a reserved keyword

- We saw a few in lab yesterday:

* other languages have their own set of reserved words

Foreshadowing: “functions”



The `math` module

- Lots of other things we might want to do with numerical values are available as functions in the **`math`** module

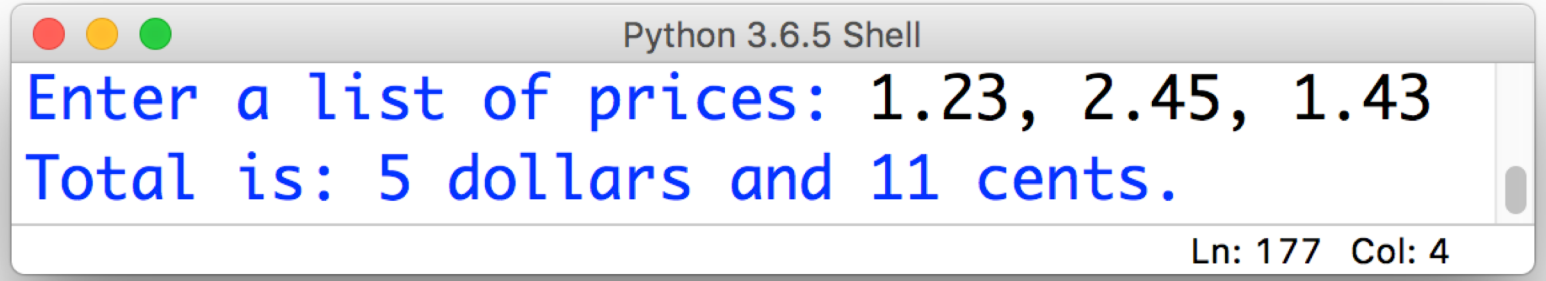
  - In Python, modules are just files containing Python definitions and statements (ex. ***`name.py`***)
  - These can be imported using **`import`** ***`name`***
  - To access **`name`**'s functions, type ***`name.function()`***

- `import math`
  - `math.floor(f)` # round float f down
  - `math.ceil(f)` # round float f up
  - `math.sqrt(x)` # take the square root of x

And more! Check out: https://docs.python.org/2/library/math.html

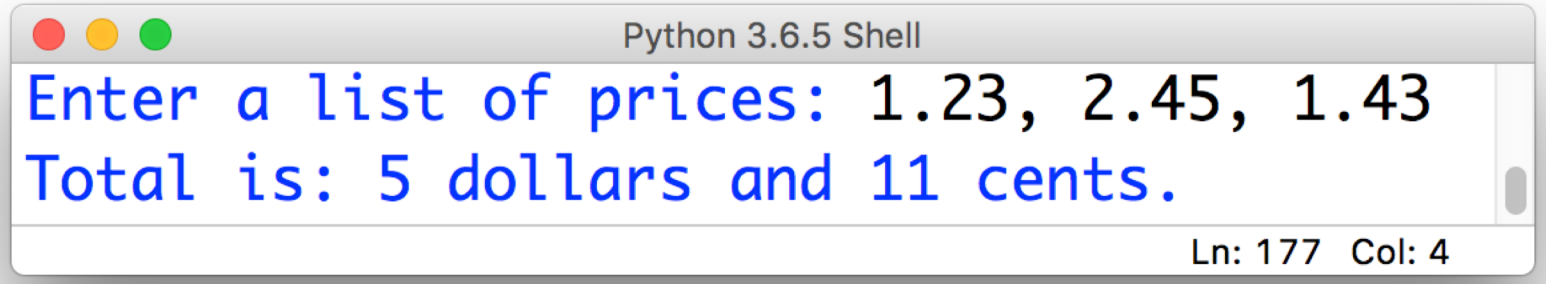
15-minute exercise: dollars and cents

Use **built-in functions** and functions from the **`math` module** to take a list of prices, calculate their sum, and output their total formatted like this:

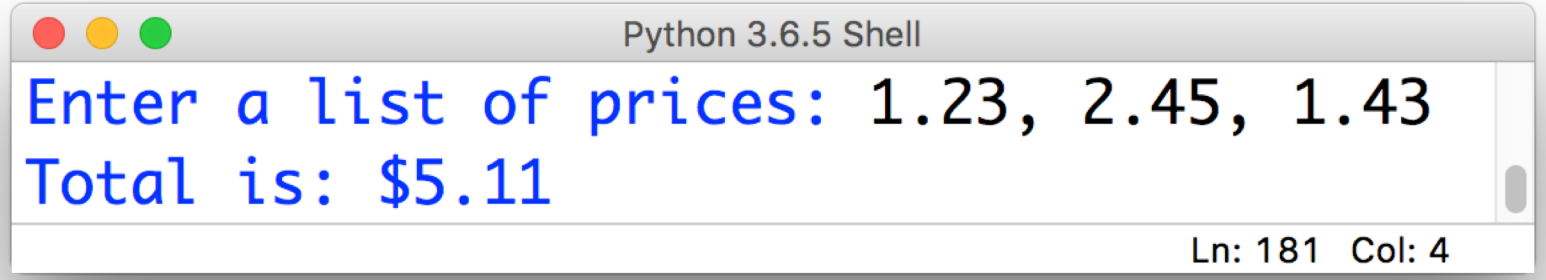


Finishing touches…

- What we have now:



- What we probably want:



Discussion

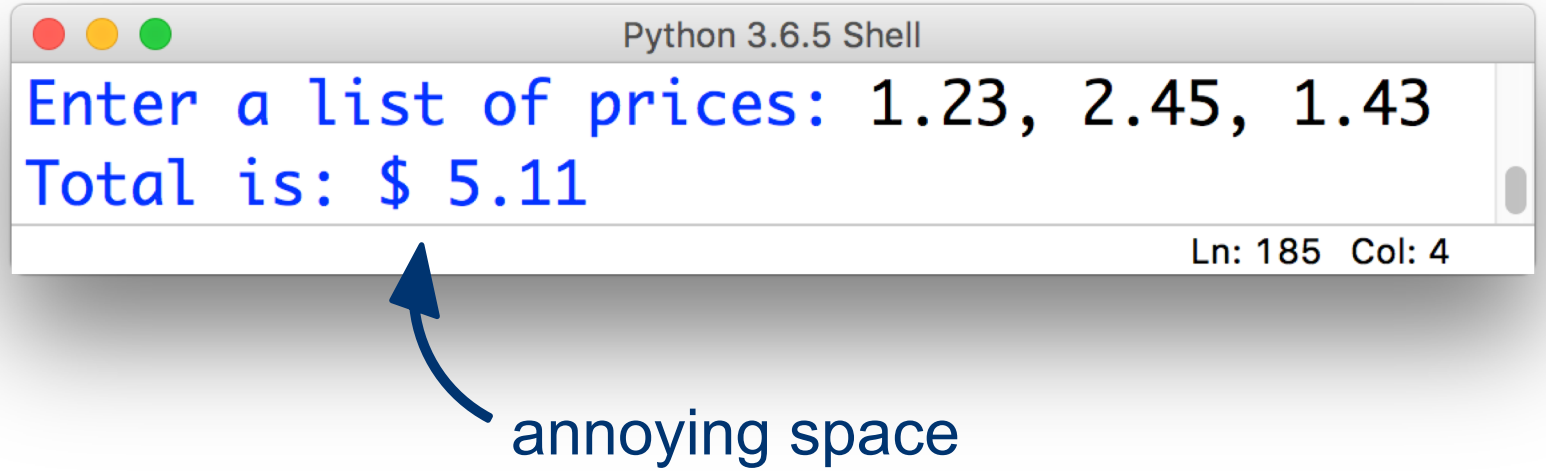
Ideas?

What tools do you have?

Close, but not quite:

- Just using concatenation:

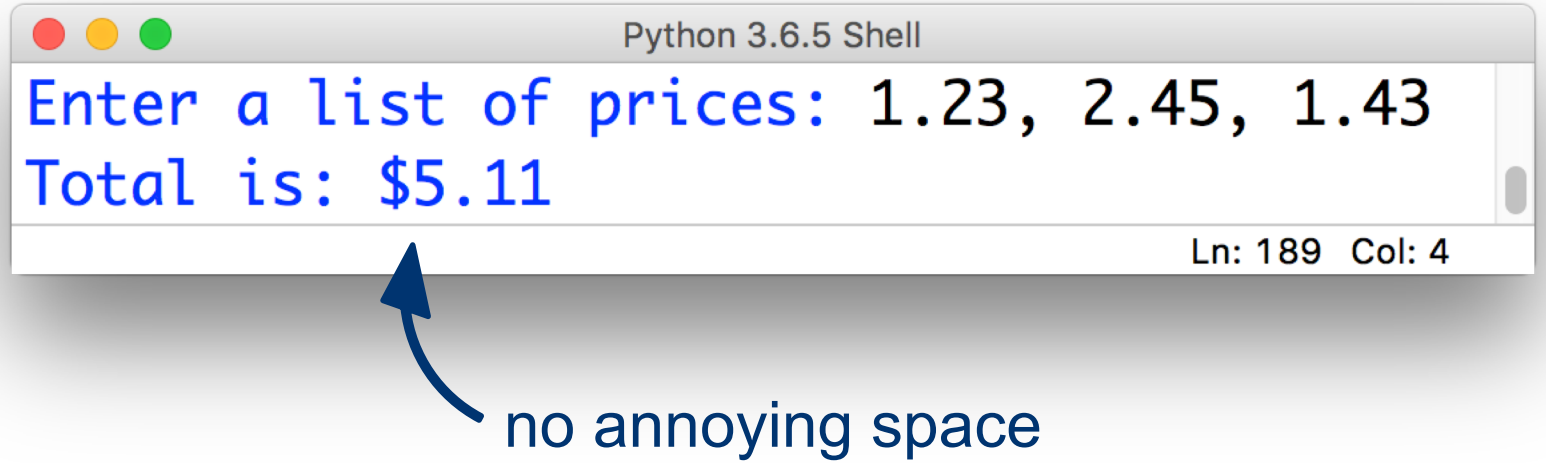
```
print("Total is: $", sum(x)))
```



Closer, but unsatisfying (and fragile)

- Using concatenation and casting to string:

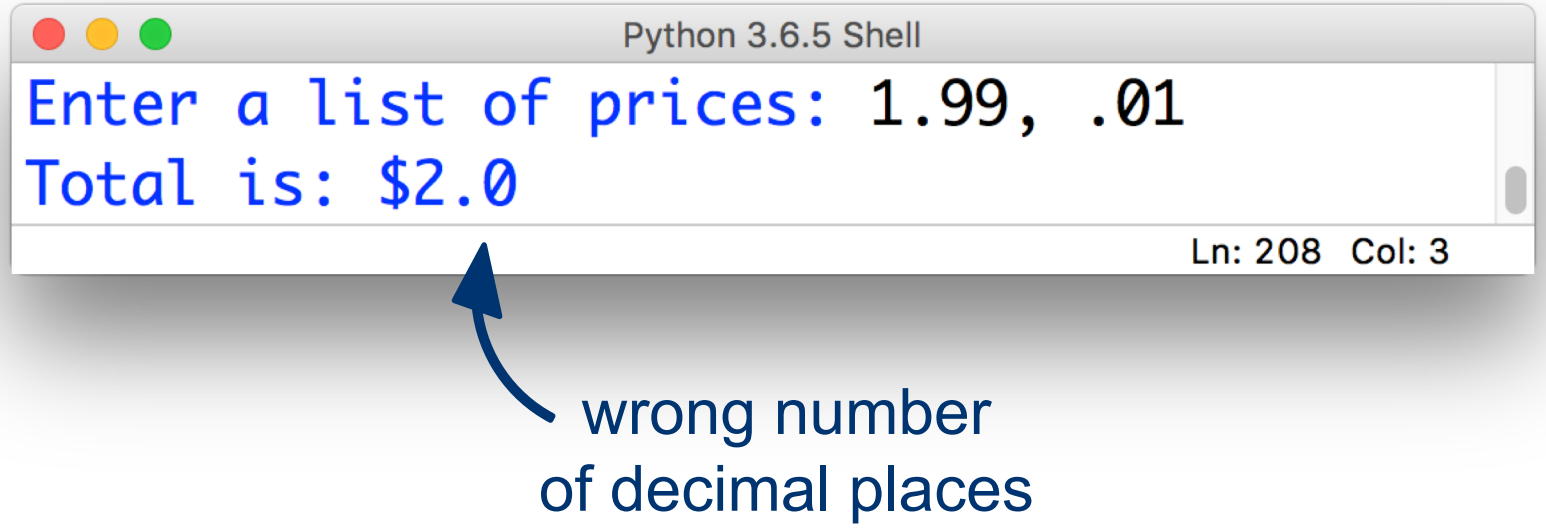
```
print("Total is: $", str(sum(x))))
```



Closer, but unsatisfying (and fragile)

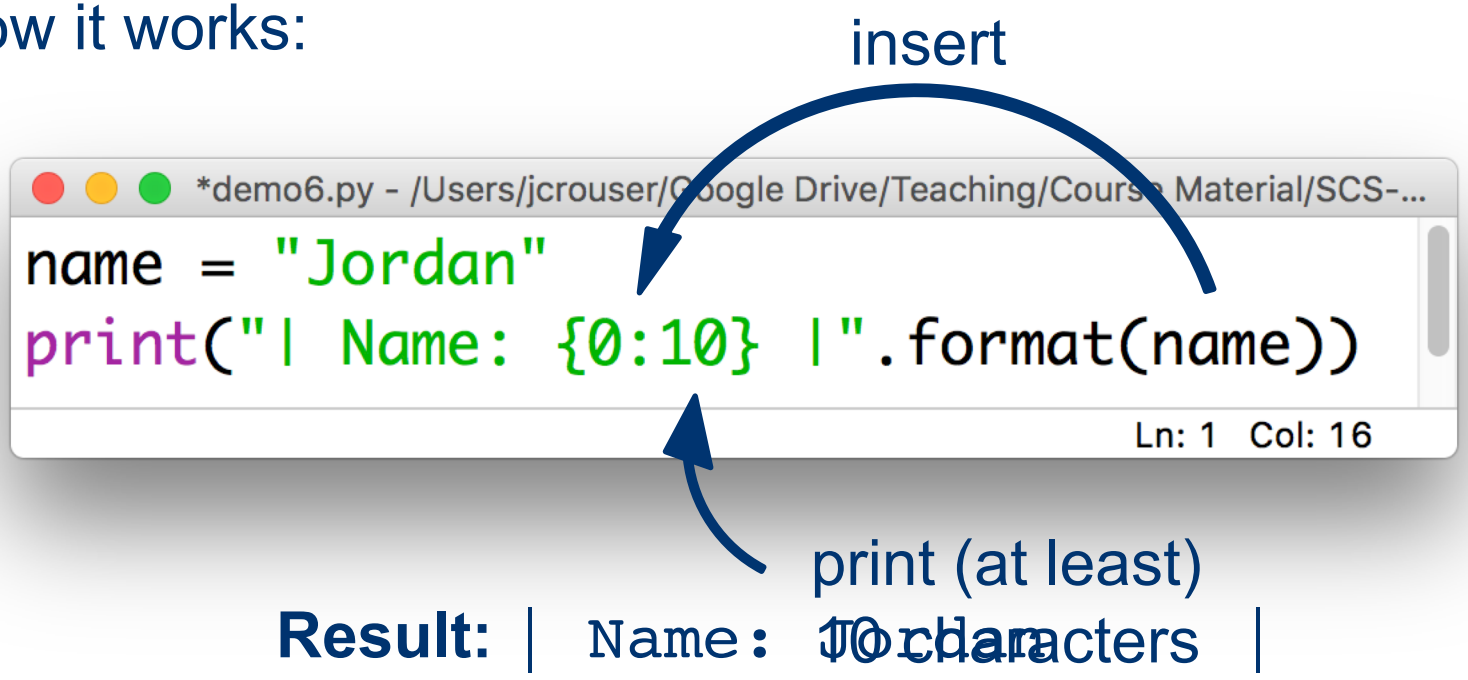
- Using concatenation and casting to string:

```
print("Total is: $", str(sum(x))))
```



Solution: formatting with `.format()`

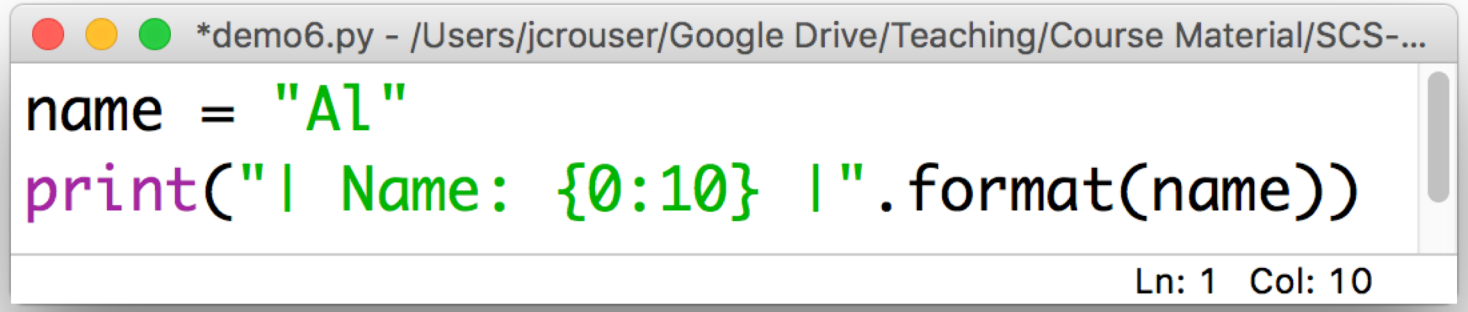
- The `.format()` method (which gets called on a `string`) might be helpful here!
- How it works:



Result: | Name: Jordan |

Solution: formatting with `.format()`

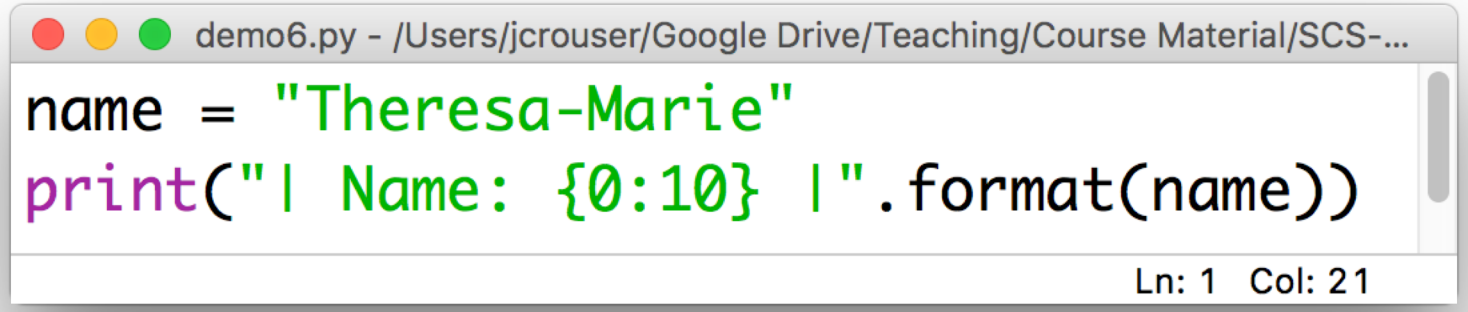
- The `.format()` method (which gets called on a `string`) might be helpful here!
- How it works:



Result: `| Name: Al |`

Solution: formatting with `.format()`

- The `.format()` method (which gets called on a `string`) might be helpful here!
- How it works:



Result: `| Name: Theresa-Marie |`

doesn't truncate

Calling `.format()` with multiple inputs

- Can also handle multiple inputs, e.g.

demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan-CSC/labs/demo6.py (3.6.5)

```
first = "Jordan"
last = "Crouser"
print("| Name: {0:10} {1:10} |".format(first, last))
```

Ln: 3 Col: 24

0th 1st

put the 1st thing here

put the 0th thing here

Result: `| Name: Jordan Crouser |`

Right-justification with `.format()`

- To align the format to the right instead of to the left, use **>**

demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan-CSC/labs/demo6.py (3.6.5)

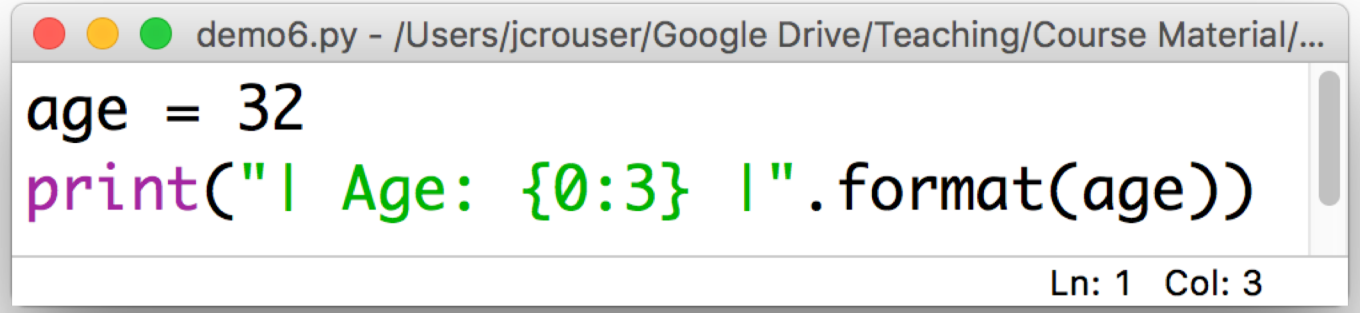
```
first = "Jordan"
last = "Crouser"
print("| Name: {0:>10} {1:10} |".format(first, last))
```

Ln: 3 Col: 19

Result: `| Name:      Jordan Crouser     |`

`.format()` on integers

- Calling `.format()` on an integer works just like with strings, but they're automatically right-aligned



Result: `| Age:  32 |`

`.format()` on integers

- Use < to left-align:

demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/...

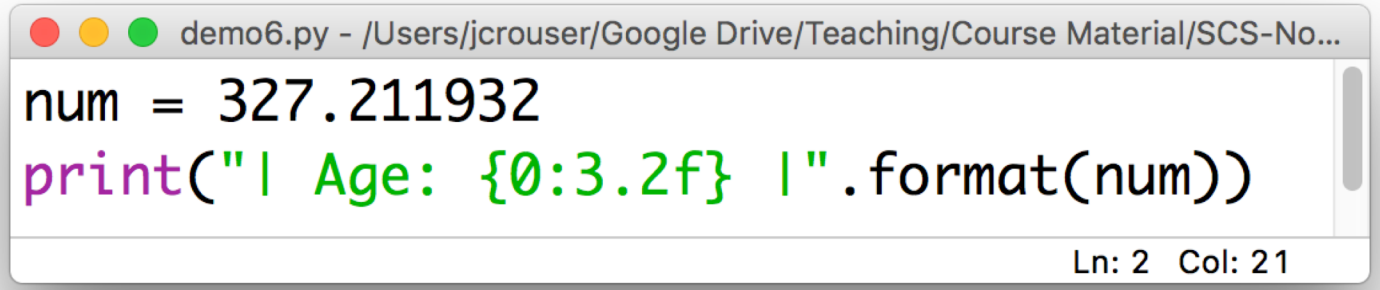
```
age = 32
print("| Age: {0:<3} |".format(age))
```

Ln: 2 Col: 18

Result: `| Age: 32  |`

`.format()` on floats

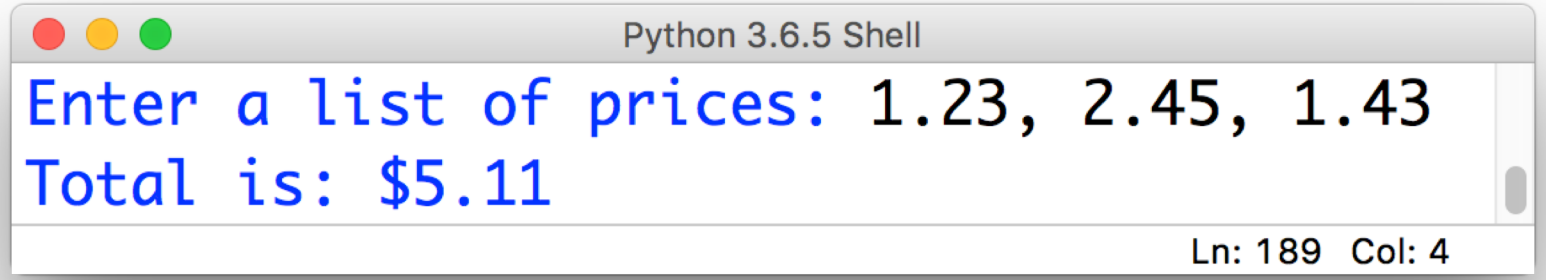
- We need to specify a number of digits **before** and **after** the decimal point:



Result: `| Age: 327.21 |`

Revisiting: dollars and cents

Modify your previous code to use the **`.format()`** method so that your output looks like this:



Overview of the week

- ✓More about variables
- ✓Numeric values and basic operations
- ✓More mathematical operators
  - ✓Revisiting ints and floats
  - ✓The `math` module
  - ✓Formatting
- Lab: Cash Machine
- Conditional (“if”) statements

Lab 2: Cash Machine