Python

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

Be Boulder.

Python

- Daniel Trahan
- *Email: Daniel.Trahan@Colorado.edu*
- *RC Homepage: https://www.colorado.edu/rc*

Sign in! http://tinyurl.com/curc-names

- Slides available for download at:
  https://github.com/ResearchComputing/SWE_Fall_2021

Outline

- Why Python
- Installing Python
- ‘Hello World’
- Variables
- Functions
- Lists and Iterations

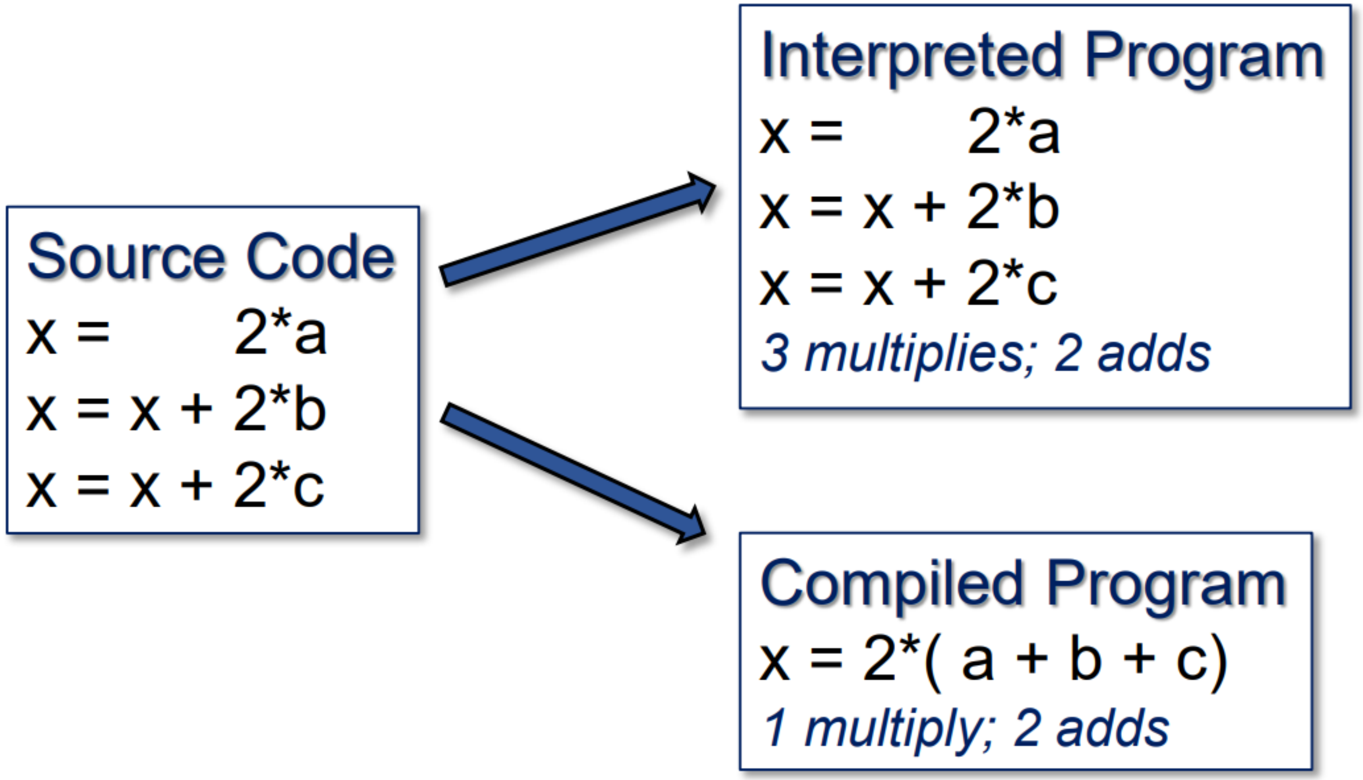
Why Python

- Python is an interpreted programming language that is relatively simple to get started with.
  - Syntax is very forgiving
  - Dynamic Memory Allocation
  - Dynamic Typing
  - No Compiling!
- Great as a learning tool, powerful as a
- Used in a lot of Data Science
- Machine learning Utility

Python, an Interpreted Language

- Python is an interpreted language
  - What does this mean?
- Separate program (the interpreter) runs Python code.
- Interpreters execute code “naively.” (line by line)
- Compilers take holistic approach. Interpreters do not.
- Efficiency losses when compared to compiled code.

An Interpreted Language…



Source Code

```
x =       2*a
x = x + 2*b
x = x + 2*c
```

Interpreted Program

```
x =       2*a
x = x + 2*b
x = x + 2*c
```

3 multiplies; 2 adds

Compiled Program

```
x = 2*( a + b + c)
```

1 multiply; 2 adds

Executing Python

- Python code is very versatile and can be executed in various ways…
- Command Line:

```
$ python <your-python-script>
```

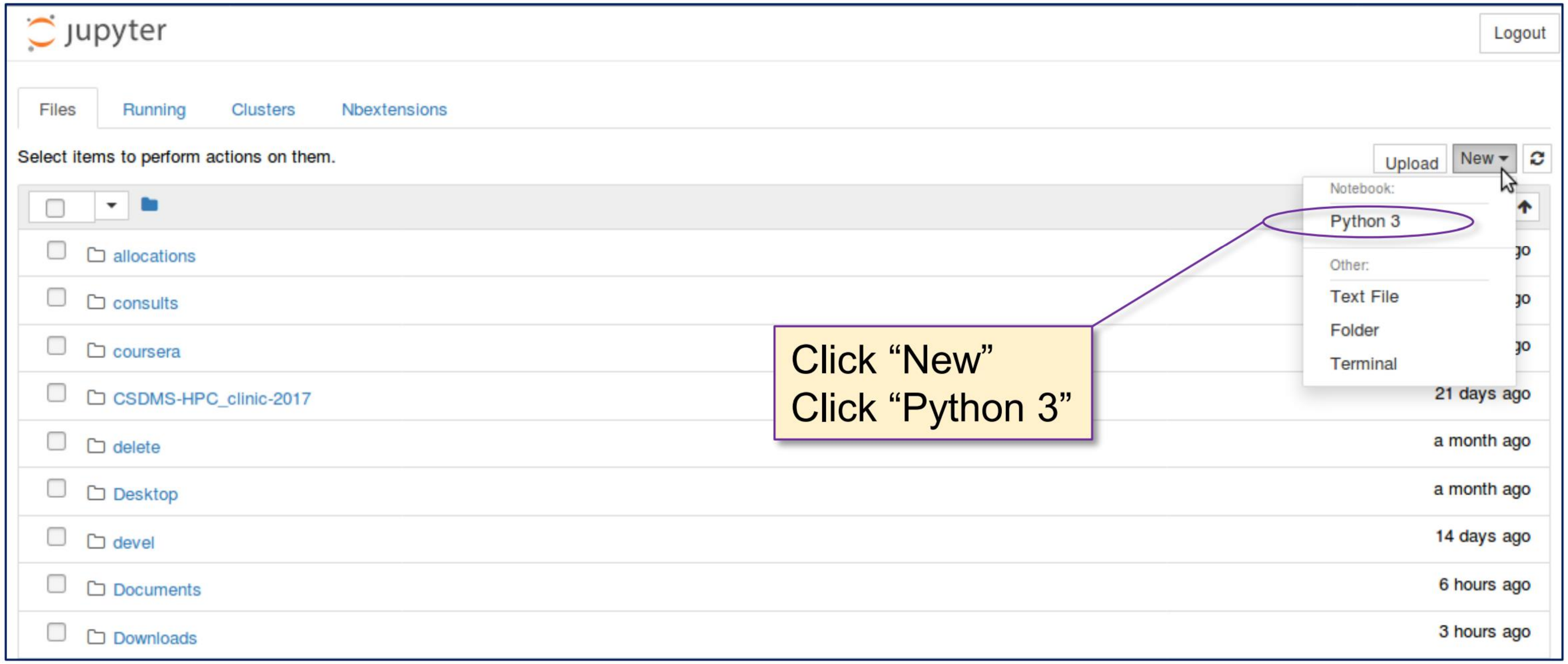
- Interactive Interpreter:

```
$ python
```

Jupyter Notebooks

- The way we will be utilizing python is through Jupyter notebook
- **Jupyter Notebook** is a Python IDE that runs through your browser.
  - Opens a file browser where you can create or open new **Notebooks**
  - Notebooks are simply interactive sessions of this IDE
- To Open Jupyter, simply open Anaconda Navigator
- Click on the Jupyter Notebook Box in the Home tab

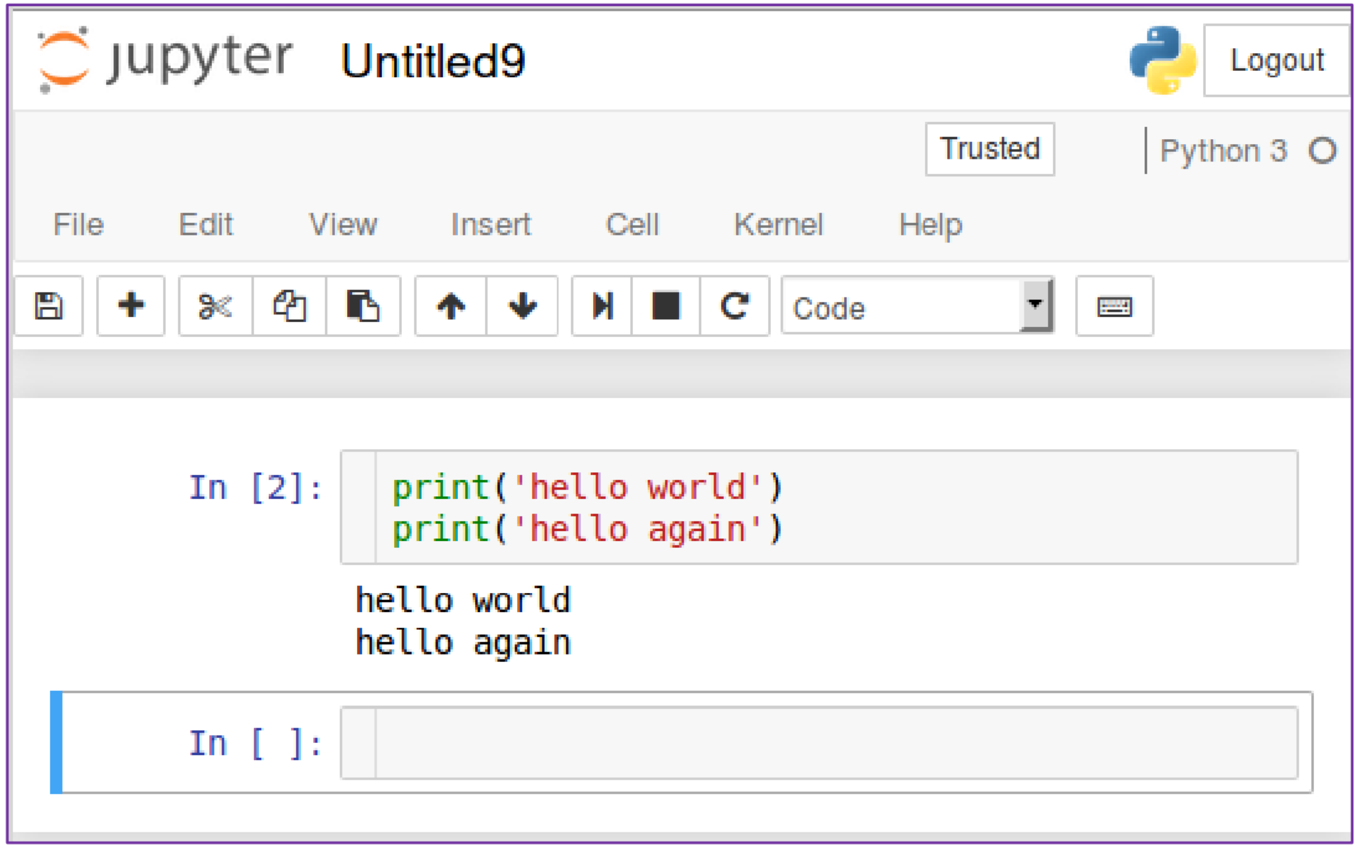
Jupyter Notebooks



Select items to perform actions on them.

allocations	
consults	
coursera	
CSDMS-HPC_clinic-2017	21 days ago
delete	a month ago
Desktop	a month ago
devel	14 days ago
Documents	6 hours ago
Downloads	3 hours ago

The Jupyter Interface



```
In [2]: print('hello world')
        print('hello again')

        hello world
        hello again

In [ ]:
```

Hello World

- In your Jupyter Notebook click on the first cell and type:

```
print(“hello, world!”)
```

- Press Shift + Enter
- This is a complete Python program!
  - …no semicolons, brackets
  - …no “begin program,” or “end program”
  - Outside of Jupyter the script is saved as a “.py” file

Python Print Statement

```
print(item1, item2, item3, ..., sep=‘ ’, end=‘\n’)
```

- item1, item2, item3
- Comma-separated list of variables whose values you wish to display
- sep:
  - optional keyword parameter
  - separation string inserted between displayed values (defaults to whitespace)
- end:
  - optional keyword parameter
  - string appended to end of printed values (defaults to newline)

Python Variables

- Variables are not declared but implicitly typed
- Created at assignment time
- Examples

• z = 2	int
• y = 3.0	float
• Z = “hello”	str
• z = True	bool

- NOTE: Python is CASE SENSITIVE (z is not Z)
- Can Check type using the type function:

```
print(“z is: ”, type(z))
```

Arithmetic in Python

- Arithmetic in Python follows order of operations

Addition:	+
Subtractrion:	-
Mulitplication	*
Division	/
Floor Division	//
Modulo (Remainder)	%
Exponentiation	**

- Some operators can work with strings!
  - X = ‘hello’ + ‘world’
  - print(X) -> displays ‘Hello, World’

Type Casting in Python

- Variables can be recast using type conversion functions
  - x = int (43.4) -> x = 43
  - y = float (x) -> y = 43.0
  - z = str (x) -> z = “43”
  - n = bool (0) -> n = False
  - m = bool (x) -> m = True

Basic User Input

- The input function can be used to grab user input:

```
num_str = input( “Enter a number: “ )
cat_name = input ( “What is your cat’s name?” )
```

- Accepts one string argument that contains the prompt seen by the user.

- Note that it ALWAYS returns a string.
- Recast as int or float to do math…

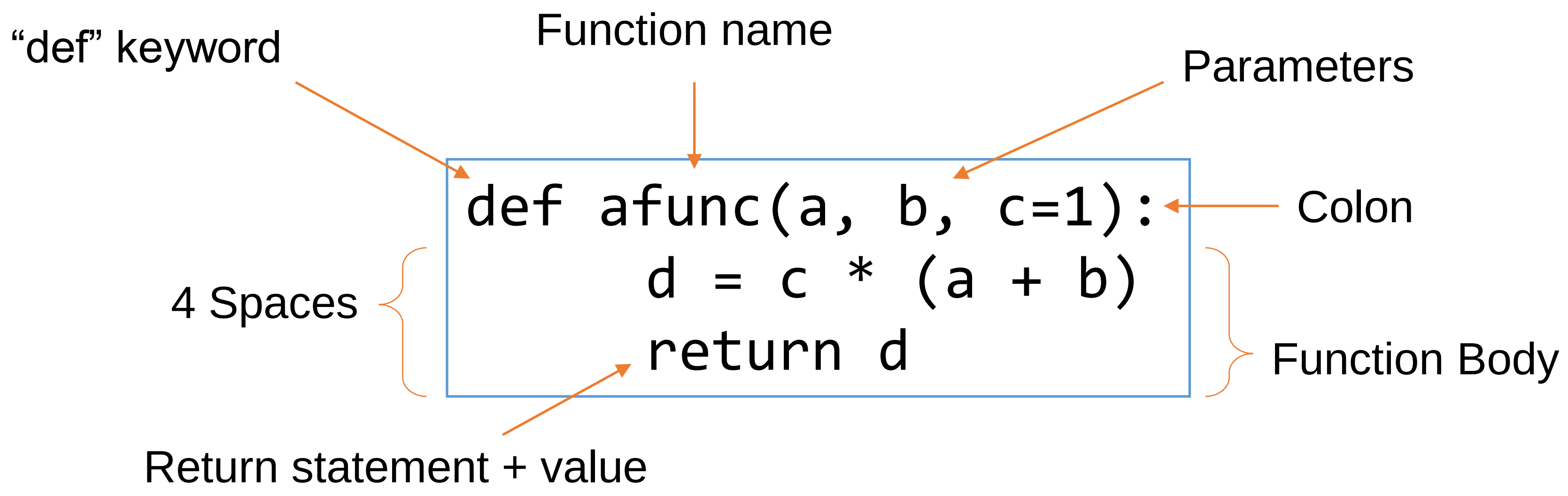
Functions in Python

- Functions are lines of code that can be executed repeatedly throughout an application.
- In Python, must be defined before called
- Example:

```
def afunc(a, b, c=1):
     d = c * (a + b)
     return d
```

- Lets break this down!

Functions



```python
def afunc(a, b, c=1):
    d = c * (a + b)
    return d
```

Calling Functions

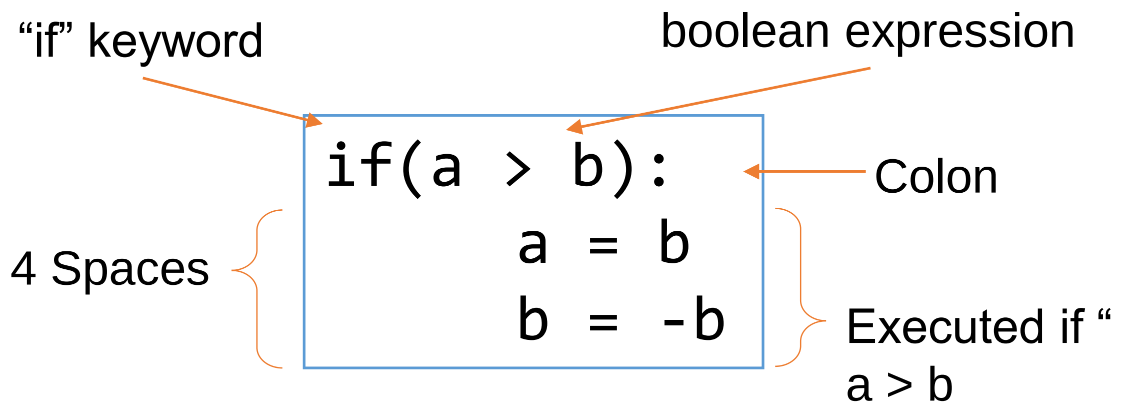
```
def afunc(a, b, c=1):
      d = c * (a + b)
      return d

myval = afunc(1, 2, 4)
```

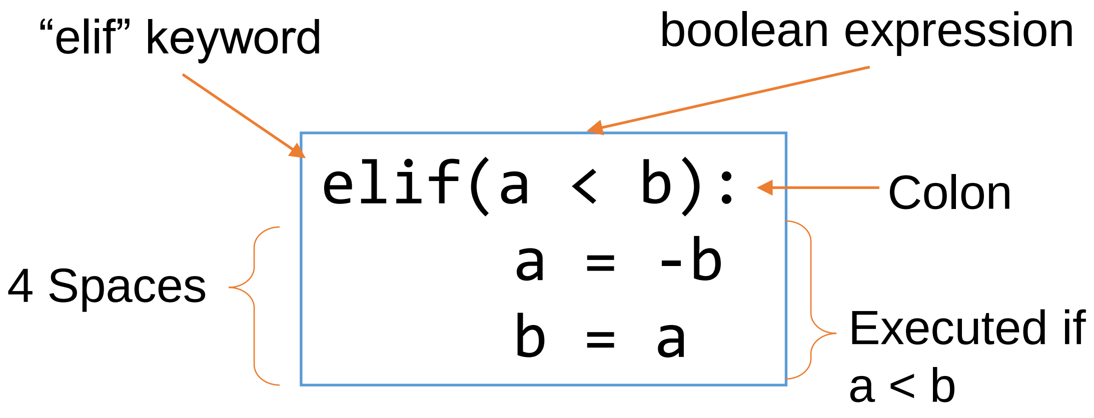
- Functions may be called once defined
- Value of ***d*** assigned to ***myval*** via return statement

Conditionals

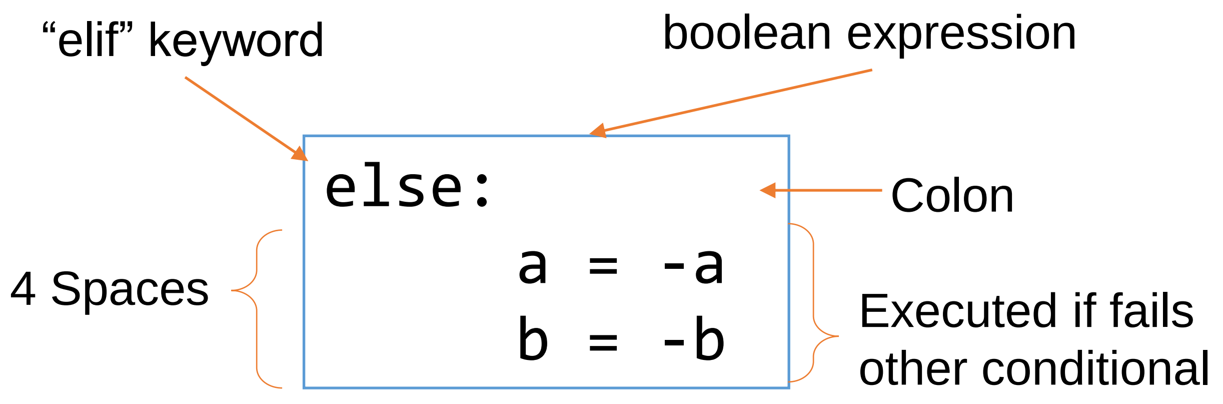
- 3 Conditionals exist within Python
  - Execute on satisfaction of the expression
  - if, else, elif
- Follow syntax like function definitions:



```
if(a > b):
    a = b
    b = -b
```



```
elif(a < b):
    a = -b
    b = a
```



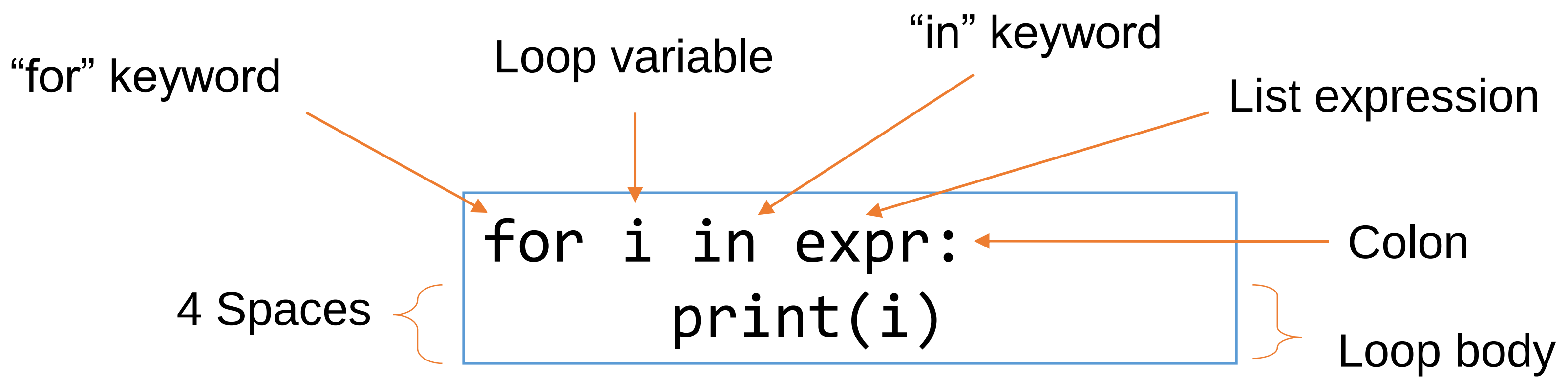
```
else:
    a = -a
    b = -b
```

Lists

- Lists are collections of data grouped together
- Enclosed with brackets
- Can be different types
- Index starts at 0

```
mylist = [41, ‘apple’, 1.1, True]
```

Iteration: For Loop



```
for i in expr:
    print(i)
```

For each element in expr:

- Assign its value to 'i'
- Execute statements in loop body

Questions?

Thank you!

- Please fill out the survey: http://tinyurl.com/curc-survey18
- Sign in! http://tinyurl.com/curc-names
- Contact information: rc-help@Colorado.edu
- Slides: https://github.com/ResearchComputing/SWE_Fall_2021