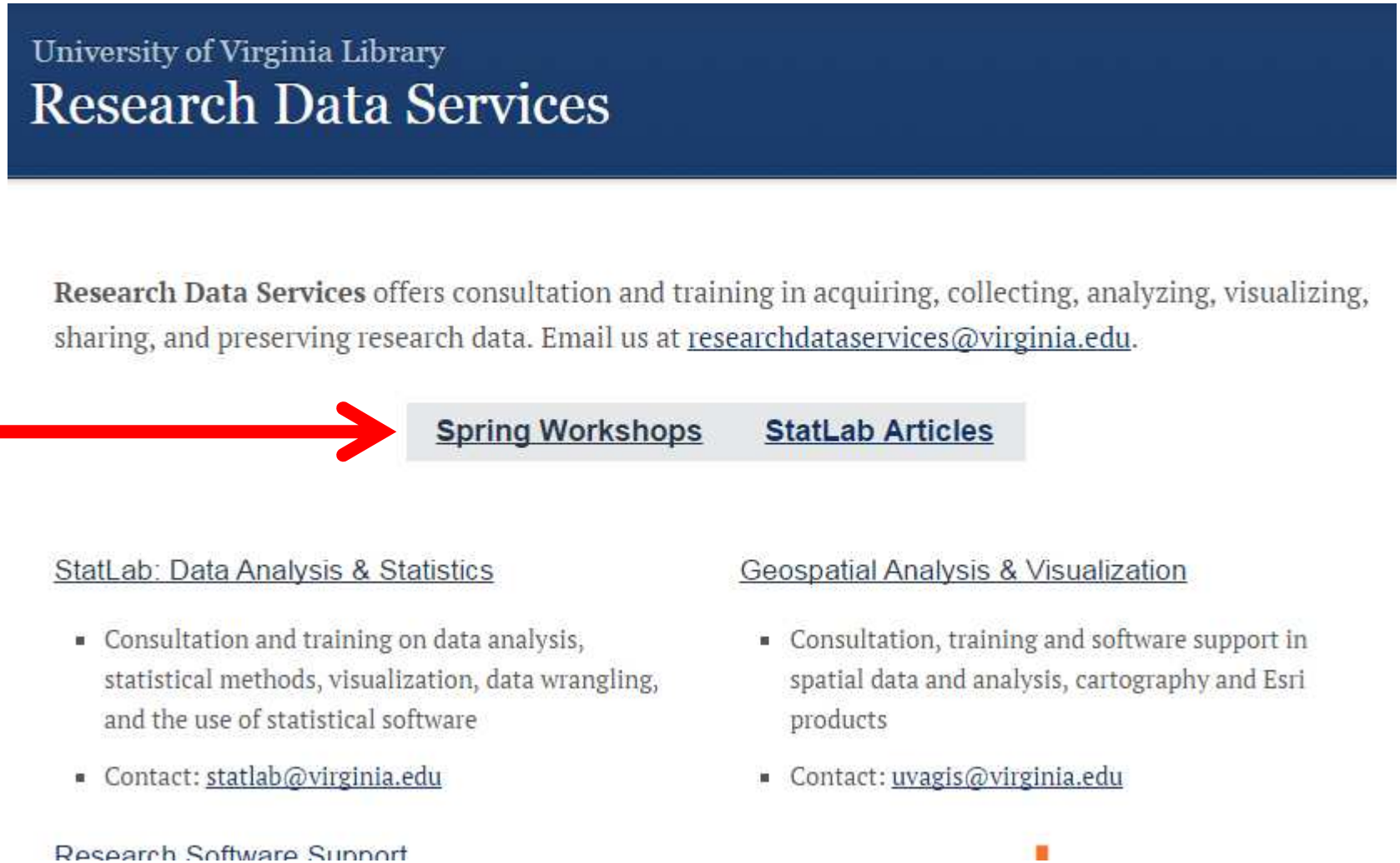
An Intro to Python

Pete Alonzi
UVa Library; Research Data Services
2016-02-16

Check us out

data.library.virginia.edu



What we're gonna do today

- 3:30-5:00
- Brief history of Python - https://www.python.org/
- Why python
- Installation of Python (Python 2 vs Python 3)
- Orientation to the interface
  - Terminal/Interpreter, Scripts (.py),Spyder,Notebook
- The 'Input, process, return' model
- 'Just in time' compiler
- Syntax and errors # and commenting
- Types: int, float, string, dict, list
- Programming, the usual suspects: if, while, for
  - Semicolon and Indentation is king
- Functions and classes and imports (a little physics)
- File input/output (I/O)

Brief history of Python - https://www.python.org/

- **Designed by** Guido van Rossum
- 1991
- successor to the ABC language
- Python 2.0 was released on 16 October 2000
- Python 3.0 was released on 3 December 2008

python™

Why Python

- Popular (TIOBE top 10 since 2003)
- Google, Yahoo!, CERN, NASA use it (told you it was popular)
- scripting languages >> conventional languages
  - string manipulation
  - Searching a dictionary.
- Python can serve as a scripting language for web applications
- Libraries like NumPy, SciPy and Matplotlib allow the use of Python in scientific computing

Installation of Python

- Two Notes
  - Language (python) vs language distribution (anaconda)
  - Python 2 vs Python 3
- https://www.continuum.io/downloads
  1. Scroll to your operating system
  2. Click on the blue box under Python 2.7
  3. Follow the instructions

Say Hello to Python

- Windows: Start->All Programs->Anaconda
  - Anaconda Command Prompt aka Terminal
  - IPython Console
  - IPython Notebook
  - Spyder
- Python files: *.py

The Python Interpreter

- Input, process, return
- Just in time compiler
- “Built-in”

Syntax and errors # and commenting

- Reserved words (more on next slide)
- Caps matters
- Indentation matters (more to come)
- Quotes: ‘ or “ as long as open=close
- Comments start with “#”

- Read errors backwards
  - `Print ‘hello’  # it should be print ‘hello’`
  - `a.lower()  # a is not a string`
  - `b.lower()  # b does not exist`

Reserved Words

And	exec	Not
Assert	finally	or
Break	for	pass
Class	from	print
Continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Types: int, float, string, dict, list

- Python is a typed language
- There is a helper function: `type(…)`
  - Boolean: `True, False,1==1,True and False`
  - `int, float, long, complex`
  - `string`
  - `list, tuple`
  - `dict`

Just gotta memorize

- Functions and tuples use parenthesis: ( )
- Lists use brackets: [ ]
- Dictionaries use braces: { }

Switch from terminal to spyder

- Spyder is what's called an IDE
- IDE = Integrated Development Environment
  - Text editor
  - Runtime environment
  - State monitor
  - autocompletion

Programming, the usual suspects: if, while, for

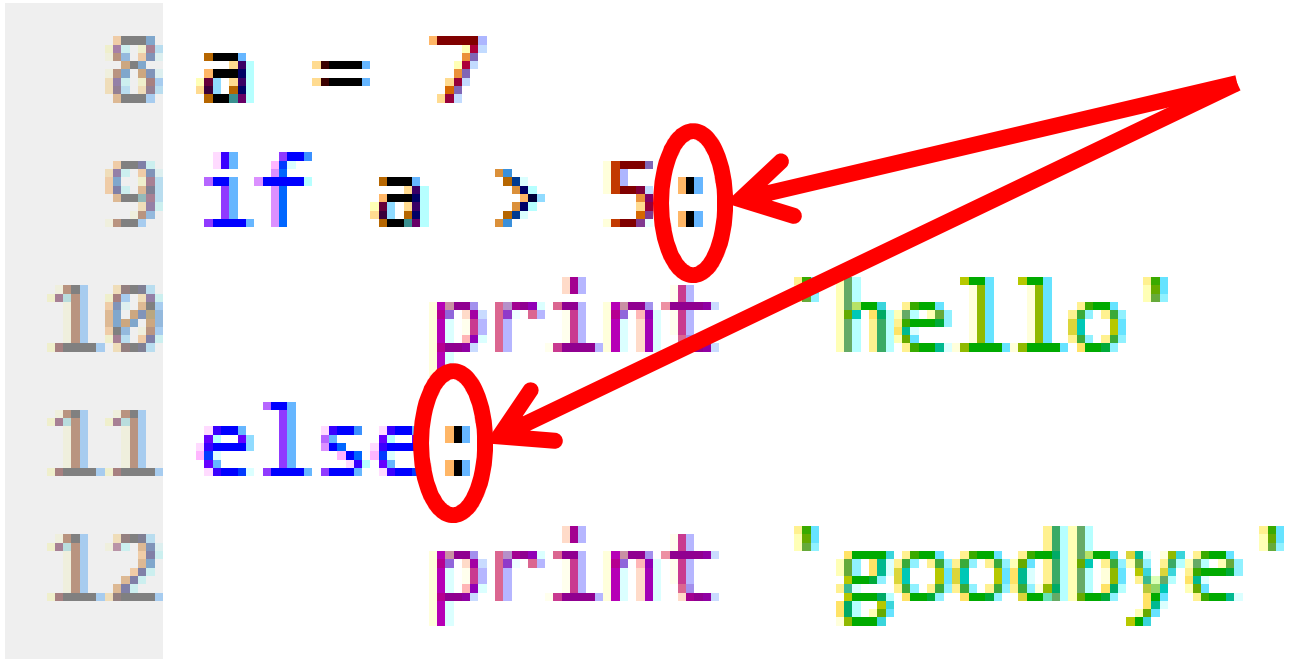
- If statements

```
 8 a = 7
 9 if a > 5:
10     print 'hello'
11 else:
12     print 'goodbye'
```

Notice the coloring of words. My editor put that in.

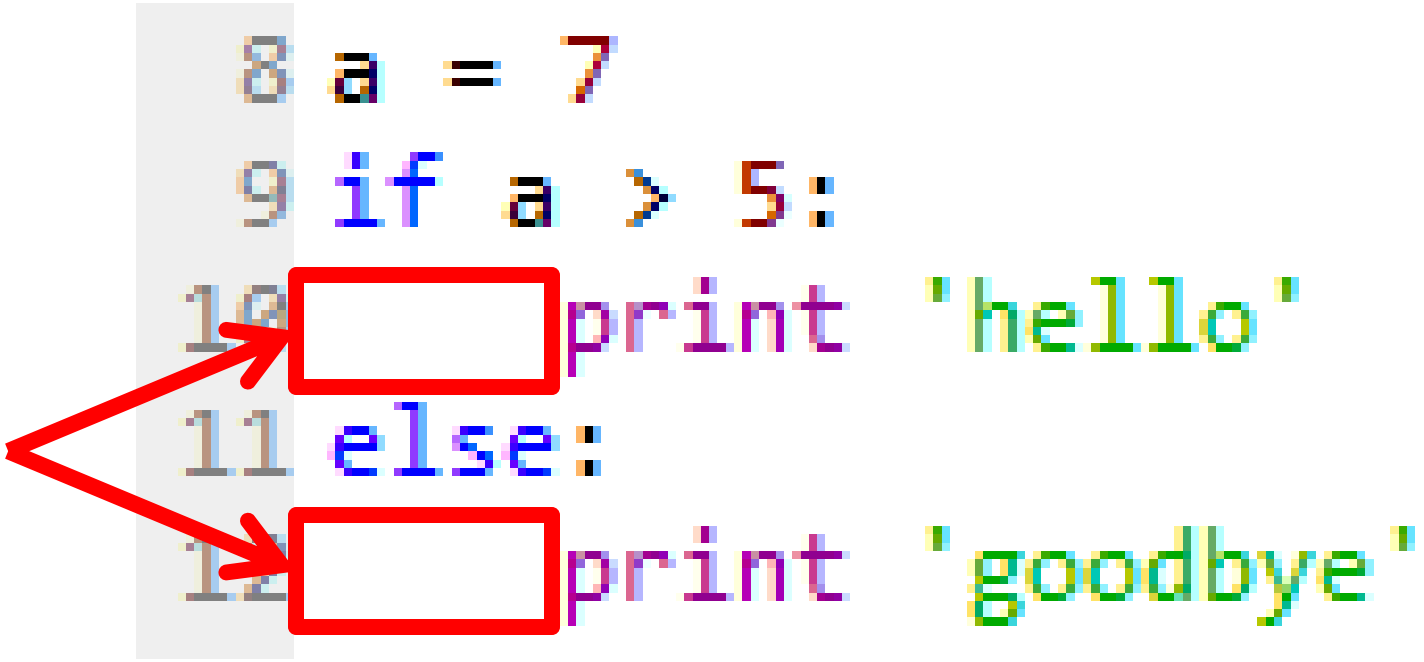
Programming, the usual suspects: if, while, for

- If statements



Programming, the usual suspects: if, while, for

- If statements



Programming, the usual suspects: if, while, for

- For loops

```
14 for i in range(5):
15     print i
```

We created the variable `i` on the fly.

Programming, the usual suspects: if, while, for

- For loops

```
17 fruits = ['apple','banana','cherry','kiwi']
18 for fruit in fruits:
19     print fruit
```

Programming, the usual suspects: if, while, for

- While loops

```
21 i = 0
22 while i<5:
23     print i
24     i+=1
```

Functions

- Extremely useful
- Used for code replication and compartmentalization
- Functions are called and they take arguments
- Functions return values

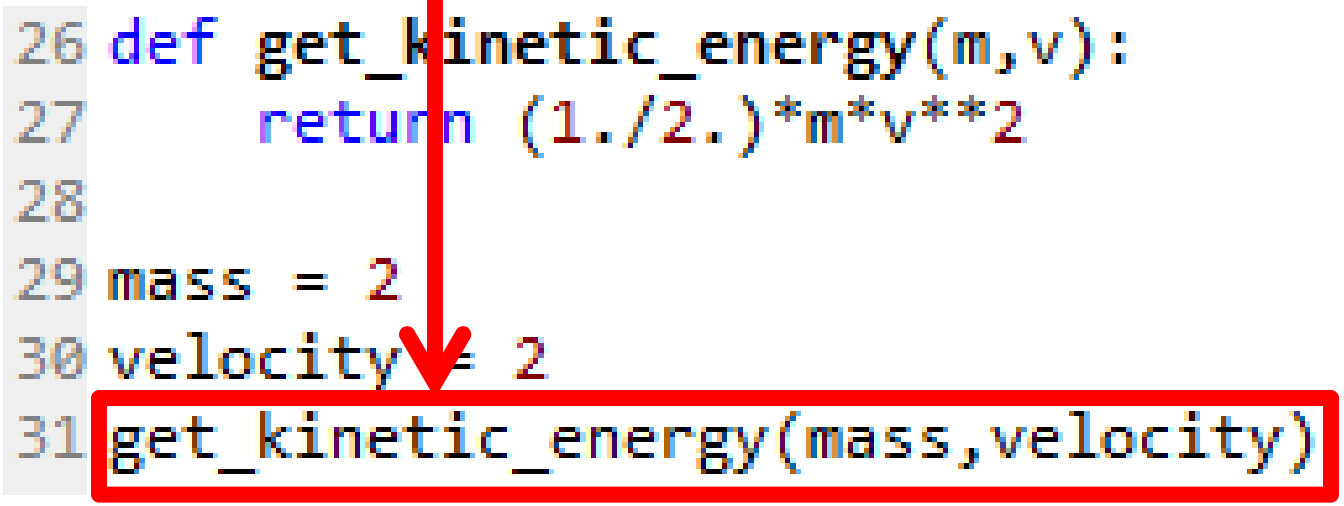
Functions

- Extremely useful
- Used for code replication and compartmentalization
- Functions are called and take arguments
- Functions return values

```
26 def get_kinetic_energy(m,v):
27     return (1./2.)*m*v**2
28
29 mass = 2
30 velocity = 2
31 get_kinetic_energy(mass,velocity)
```

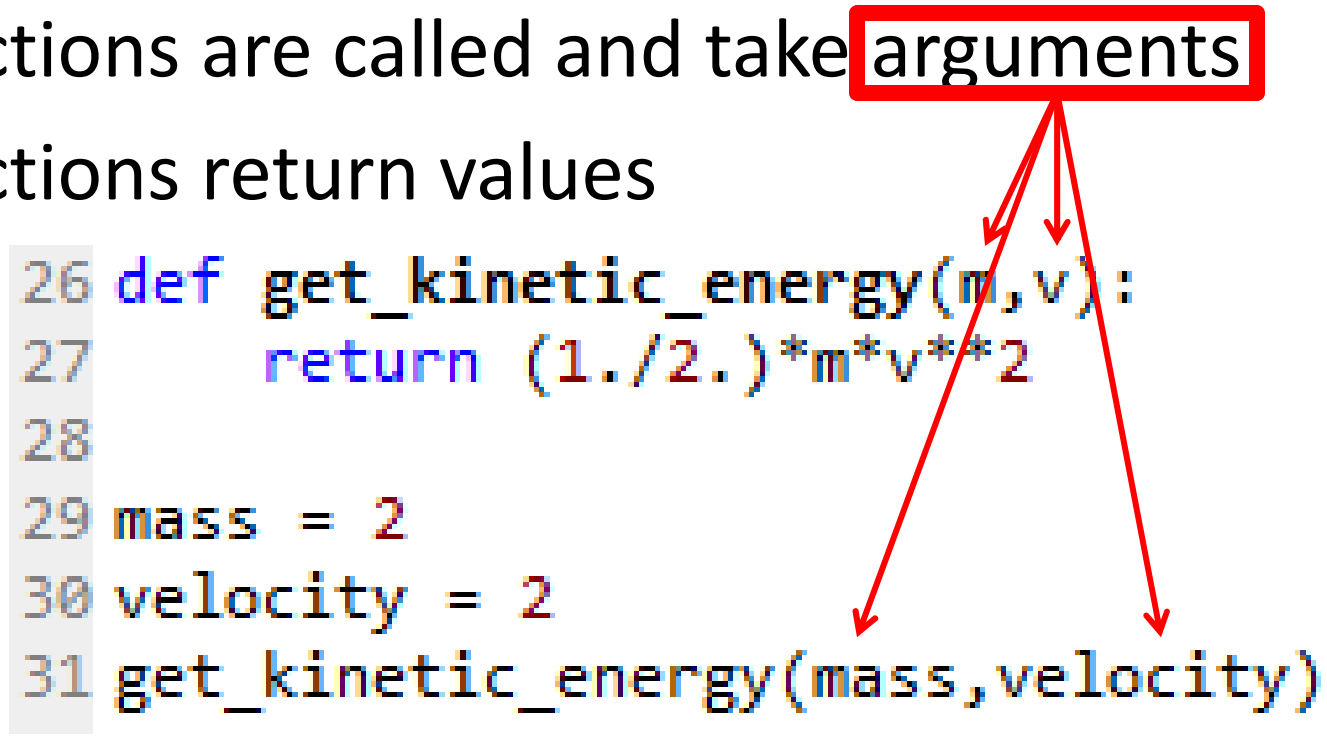
Functions

- Extremely useful
- Used for code replication and compartmentalization
- Functions are called and take arguments
- Functions return values



Functions

- Extremely useful
- Used for code replication and compartmentalization
- Functions are called and take arguments
- Functions return values



Functions

- Extremely useful
- Used for code replication and compartmentalization
- Functions are called and take arguments
- Functions return values

```
26 def get_kinetic_energy(m,v):
27     return (1./2.)*m*v**2
28
29 mass = 2
30 velocity = 2
31 get_kinetic_energy(mass,velocity)
```

Classes

- Buzzword alert: Object Oriented Programming

Class: Jedi

lightsabre color: blue
master: Yoda
alignment: light side

Classes

- Buzzword alert: Object Oriented Programming

Class: Jedi

lightsabre color: green
master: ???
alignment: light side

Classes

- Buzzword alert: Object Oriented Programming

Class: Jedi

lightsabre color: red
master: Obi-Wan
alignment: dark side

Classes

- Recurring set of attributes
- Different values for the attributes

Now we can look at some code

Import

- Python is modular
- Use `import` to load a module
- We'll look at one example `numpy`

```
57 import numpy
```

There are too many modules to go into here but suffice it to say … if you want to do something complicated, there's a module for it.

File input/output (I/O)

- Reading from a file

```
62 f = open('lyrics.txt','r')
63 for line in f:
64     print line
65 f.close()
```

File input/output (I/O)

- Writing to a file

```
67 f = open('numbers.txt','w')
68 for i in range(5):
69     f.write(str(i))
70 f.close()
```

What we did today!!
Questions?

- 3:30-5:00
- Brief history of Python - https://www.python.org/
- Why python
- Installation of Python (Python 2 vs Python 3)
- Orientation to modes of access
  - Terminal/Interpreter, Scripts (.py),Spyder,Notebook
- The 'Input, process, return' model
- 'Just in time' compiler
- Syntax and errors # and commenting
- Types: int, float, string, dict, list
- Programming, the usual suspects: if, while, for
  - Semicolon and Indentation is king
- Functions and classes and imports
- File input/output (I/O)