Computation in Sage

Jeremy Martin
University of Kansas

GRWC 2016, Laramie, WY

This mini-presentation also at
http://www.math.ku.edu/∼jmartin/talks/Sage.pdf

Getting Started

Sage [www.sagemath.org] is a free, open-source mathematics software system based on the Python programming language. You can start using it in several ways:

- Sage Math Cloud: create collaborative projects
- Sage Cell Server: simple web interface a la WolframAlpha
- Local installation: can take several hours, so don't do it now

Getting Help

- A Tour of Sage — start from scratch
- Sage Constructions — how to do mathematics in Sage
- Thematic Tutorials — tutorials on specific topics (everything from Python data structures to numerical computation)
- Reference Manual — comprehensive reference on everything. (Warning: much of the documentation is written for developers rather than end users, so can be daunting. . .)
- Discussion Groups

- Derrick Stolee's Sage Page (used for previous GRWC Sage workshops; lots of good stuff here)

Object Orientation

Sage is an **object-oriented** language.

- Every object is known by Sage to be of some class (number, polynomial, matrix, graph, . . .)
- Every class has methods that can be used on its members.

E.g., if `G` is of class `Graph` then you can tell Sage things like

`G.vertices()`

`G.degree_sequence()`

`G.neighbors(v)`   (if `v` is a vertex of `G`)

To find out what methods are available for an object `foo`, type

`foo.`⟨*tab*⟩

Indentation

Other languages define the scope of loops and conditionals using explicit delimiters (C/C++: { }) or keywords (Maple: `do/od`, `if/then/fi`).

Sage uses colons and **indentation.**

Sage

```
for x in srange(100):
  if x.is_prime():
    print x
```

Maple

```
for x from 0 to 99 do
  if isprime(x) then
    print(x)
 fi od:
```

Lists

A basic data structure in Python/Sage is the list.

```
L = [3,1,4,1,5,9,2,6]
```

How many elements?	`len(L)`
Extract element in ith position	`L[i]`
First element	`L[0]`
Last element	`L[-1]`
Append an element	`L.append(5)`
Find first instance	`L.index(9)`
Sort in-place	`L.sort()`
Sorted copy (don't change L)	`sorted(L)`

List Comprehensions

You may be used to building lists manually by means of a `for` loop.

```
Squares = []
for i in srange(10):
    Squares.append(i^2)
```

The Pythonic way to do this is a list comprehension:

```
Squares = [i^2 for i in srange(10)]
```

Think of this as set-builder notation: $S = \{i^2 \mid 0 \leq i \leq 9\}$.

You can include conditionals too:

```
Primes = [i for i in srange(100) if i.is_prime()]
```