

MAT 305: Mathematical Computing

Introduction to Sage

John Perry

University of Southern Mississippi

Fall 2011

Outline

- ➊ Introduction to Sage
- ➋ Using computer memory
- ➌ Summary

Outline

- 1 Introduction to Sage
- 2 Using computer memory
- 3 Summary

How to get Sage

- Download, install to your computer
 - latest version at www.sagemath.org
 - Windows? need LiveCD or VirtualBox player:
www.virtualbox.org/wiki/Downloads
 - ask nicely, & I might give you a DVD with Sage for Windows, Mac, Linux; VirtualBox; jEdit, and other necessary items
- Available in lab (SH 318)
- Access online
 - two options
 - <https://sage.st.usm.edu:8000/>
 - <https://pax.st.usm.edu:8004/>
 - create account
 - can share worksheets with me
 - too many people online simultaneously and it drags...

First steps in Sage

- Start Sage
- Using web interface? create account, login
 - Don't forget password!

Initial state

- Variable x defined
- Define more? define function or use `var()`
 - `var('y')` defines y
 - functions later
- Try to use an undefined variable?

```
sage: x+y+z
```

```
...
```

```
NameError: name 'z' is not defined
```

Arithmetic

operation	sage equivalent
add x, y	$x + y$
subtract y from x	$x - y$
multiply x, y	$x * y$
divide x by y	x / y
raise x to the y th power	$x ** y$ or $x ^ 4$

- Multiply coefficients to variables:
for $2x$ use $2*x$, *not* $2x$
- Prefer $**$ to $^$ for various sordid reasons

Example

- Sage simplifies (of course)

```
sage: 5 + 3
```

```
8
```

```
sage: (x + 3*x**2) - (2*x - x**2)
```

```
4x^2 - x
```


Transcendental numbers, functions

number	sage equivalent
e	e
π	pi

operation	sage equivalent
e^x	e**x
$\ln x$	ln(x)
$\sin x, \cos x, \text{etc.}$	sin(x), cos(x), etc.

- Don't forget to use parentheses when necessary
e**(2*x) and e**2*x are not the same

Some useful operations

operation	sage equivalent
factor $expr$	<code>factor($expr$)</code>
simplify $expr$	<code>simplify($expr$)</code>
expand $expr$	<code>expand($expr$)</code>
round $expr$ to n decimal places	<code>round($expr$, n)</code>

Examples

- Some algebraic expressions simplify automatically; others need a hint

```
sage: (x**2 - 1) / (x - 1)
(x^2 - 1)/(x - 1)
sage: (factor(x**2 - 1)) / (x - 1)
x + 1
```

- Expand the product $(x-1)(x^3+x^2+x+1)$

```
sage: expand((x-1)*(x**3+x**2+x+1))
x^4 - 1
```

- Round e to 5 decimal places

```
sage: round(e,5)
2.71828
```

- L^AT_EX?
 - system for laying out mathematical documents, created by Leslie Lamport
 - built on T_EX, system for typesetting mathematics, originally by Donald Knuth
- Sage is L^AT_EX-friendly!

```
sage: latex(factor(x^3-1))  
{\left(x - 1\right)} {\left(x^{\{2\}} + x +  
1\right)}
```

Getting help

- Online Sage documentation (tutorial, manual, etc.) at <http://www.sagemath.org/doc/>
- These notes: www.math.usm.edu/perry/mat305fa11/
- In-Sage help: command, question mark, <Enter>

```
sage: round?  
[output omitted]
```
- Email: john.perry@usm.edu

Outline

- 1 Introduction to Sage
- 2 Using computer memory
- 3 Summary

Expressions

- Use a computer's memory by defining *expressions* with the *assignment symbol* =

```
sage: f = x**2 - 1
```

Sage does not answer when you define an expression

- Expressions are remembered until you terminate Sage

```
sage: f  
x^2 - 1
```

Valid names

Names for expressions (“*identifiers*”) can

- contain letters (A–Z), digits (0–9), or the underscore (`_`) *but*
- must begin with a letter or the underscore *and*
- may not contain other character (space, tab, `!@#$%^`, etc.)

Using expressions

- Manipulate expression in the same way as the mathematical object it represents

```
sage: factor(f)
(x - 1)*(x + 1)
sage: f - 3
x^2 - 4
```

- Avoid repeating computations: substitute!

```
sage: f(x=3)
8
sage: f(x=-1)
0
sage: f(x=4)
15
```

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage: f = x**2 + y**2
```

```
sage: f(x=3)
```

```
9 + y^2
```

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage: f = x**2 + y**2
```

```
sage: f(x=3)
```

```
9 + y^2
```

```
sage: z = x
```

```
sage: f(z=3)
```

```
x^2 + y^2
```

Here we let z stand in place of x
We want to replace x by 3, but...

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage: f = x**2 + y**2
```

```
sage: f(x=3)
```

```
9 + y^2
```

```
sage: z = x
```

```
sage: f(z=3)
```

```
x^2 + y^2
```

```
sage: f({x:3})
```

```
9 + y^2
```

Here we let z stand in place of x
We want to replace x by 3, but...

This also means replace x by 3 in f

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage: f = x**2 + y**2
```

```
sage: f(x=3)
```

```
9 + y^2
```

```
sage: z = x
```

```
sage: f(z=3)
```

```
x^2 + y^2
```

```
sage: f({x:3})
```

```
9 + y^2
```

```
sage: f({z:3})
```

```
9 + y^2
```

Here we let z stand in place of x
We want to replace x by 3, but...

This also means replace x by 3 in f

This works where $f(z=3)$ did not

Expressions as functions

Define function using natural notation

```
sage: f(x) = x**2
```

```
sage: f(2)
```

```
4
```

Expressions as functions

Define function using natural notation

```
sage: f(x) = x**2
```

```
sage: f(2)
```

```
4
```

Automatically defines variables!

```
sage: f(w,z) = 4*w**2-4*z**2
```

```
sage: f(3,2)
```

```
20
```

```
sage: f(1,z)/z
```

```
-4*(z**2 - 1)/z
```

```
sage: f(3,2)/z
```

```
20/z
```

Expressions as functions

Define function using natural notation

```
sage: f(x) = x**2
```

```
sage: f(2)
```

```
4
```

Functions really expressions

```
sage: factor(f)
```

```
4*(w - z)*(w + z)
```

```
sage: type(f)
```

```
<type 'sage.symbolic.expression.Expression'>
```


Outline

- 1 Introduction to Sage
- 2 Using computer memory
- 3 Summary

Summary

- Basic, intuitive facilities for arithmetic
- Create variables to your heart's content
- Define expressions to avoid repeating computations