MAT 305: Mathematical Computing

Lecture 1: Introduction to Mathematical Computing

John Perry

University of Southern Mississippi

October 6, 2009

Outline

Outline

Description

- **Online:** Introduction to a computer algebra system using calculus-based projects. Students will solve mathematical problems in the MAPLE environment which require an understanding of calculus concepts.
- **Syllabus:** Introduction to a computer algebra system using calculus-based projects. Students will solve mathematical problems in the Sage environment which require an understanding of calculus concepts.

Problem solving

- This class about problem-solving, like mathematics
- Some problems best attacked with a computer
  - Repetitive/tedious
  - Long
- Computers require instructions, called **programs**
- We study *some* programming, but class not about programming

Sage?

- Software for Algebra and Geometry Exploration
- Computer Algebra System "started" by William Stein

- Depends on other CASs
  - Maxima for Calculus
  - Singular for Commutative Algebra
  - GAP for group theory
  - etc.

Why Sage?

- "Free" software
  - "Free as in beer": no cost to you
    - Downloading free
    - Installing free
    - Copying free
    - Bug fixes free
    - Future versions free
  - "Free as in speech":
    - Open-source software
    - No hidden algorithms
    - Can study implementation
    - Can correct, improve, contribute to Sage

Analogy

Free mathematics

Theorem

Every odd integer m has the form $m = 2q + 1$, where n is an integer.

Proof.

Let m be an odd integer. By the Division Theorem, there exist integers q, r such that $m = 2q + r$ and $0 \leq r < 2$. This leaves two possible values of r, 0 or 1. If $r = 0$, then $m = 2q$ is a multiple of 2. Thus m is even by definition of the word. This contradicts the choice of m as an odd integer! Hence $r = 1$, and $m = 2q + 1$ as desired. $\square$

Analogy

Proprietary mathematics

Theorem

Dr. Perry is the world's best math professor.

Proof.

Trust me: I get paid to write theorems. $\square$

But I prefer Maple!

- Fine, buy your own copy
    - Student discount available
    - I will tell you the Maple equivalents for everything we do in Sage
    - You can submit homework as Maple worksheet
- Be warned:
    - Future versions not free
    - Bug fixes not free
    - I used to use Maple and switched to Sage
    - Recent versions disappointed me
    - After you graduate, pay full price
    - Not always backwards compatible (neither is Sage, but Sage is free)

Outline

Why program?

- Programming bridges gap between humans, computers
    - Computers don't understand human languages
        - Humans intuitive, poetic; computers literal, direct
        - Computers understand electric states: **on** or **off**
    - (Most) humans don't understand a computer's native language
        - Mathematics literal and precise, but (most) humans don't understand it, either!
        - Even the humans that do, prefer not to talk to the computer in that language
- Firmer control over computer
- Deeper understanding of computer technology

Kinds of computer languages

- Compiled
  - C/C++
  - Fortran
- Interpreted or scripting
  - Python
  - BASIC
- Mixed (“bytecode”)
  - Java

Python

- Sage built primarily in Python
- Not all *components* of Sage built in Python:
    - Maxima in LISP
    - Singular in C/C++
- Python also interface between Sage and user

Advantages of Python

- Modern language
    - Facilities for object-oriented, functional programming
- Wide distribution and usage
- Flexible
- Many good packages enhance it
- Many employers use it

Python $\neq$ Sage

- Some Python commands don't work in Sage's worksheet mode
    - `input()`
- Sage commands do not work in plain Python

Outline

How to get Sage

- Download, install to your computer
  - latest version at `www.Sagemath.org`
  - Windows users must also download VM-ware player at `www.vmware.com/products/player`
  - ask nicely, & I might give you a DVD with Sage for Windows, Mac, Linux; VM-ware; jEdit, and other necessary items
- Available in lab (SH 318)
  - Very old version, may be out of date
- Access online at `https://sage.st.usm.edu:8000/`
  - Create account
  - Can share worksheets with me
  - Too many people online simultaneously and it drags…

First steps in Sage

- Start Sage
- If not using web interface, create account and login
    - Don't forget your password

Initial state

- Variable `x` is defined
- To define more variables, use the `var()` command
    - `var('y')` defines y
- Try to use an undefined variable?

```
sage: x+y+z
...
NameError: name 'z' is not defined
```

Arithmetic

operation	sage equivalent
add x and y	`x + y`
subtract y from x	`x - y`
multiply x and y	`x * y`
divide x by y	`x / y`
raise x to the yth power	`x ** y` or `x ^ 4`

- Do not forget to multiply coefficients to variables: represent $2x$ by `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 and functions

number	sage equivalent
e	`e`
π	`pi`

operation	sage equivalent
e^x	`e**x`
$\ln x$	`ln(x)`
$\sin x$, $\cos x$, 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*	`factor(`*expr*`)`
simplify *expr*	`simplify(`*expr*`)`
expand *expr*	`expand(`*expr*`)`
round *expr* to n decimal places	`round(`*expr*`, ` n`)`

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
```

Getting help

- Online Sage documentation (tutorial, manual, etc.) at `http://www.sagemath.org/doc/`
- Command-line help: type command, followed by question mark, and press Enter

```
sage: round?
```

[output omitted]

- Email: `john.perry@usm.edu`

Outline

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

Outline

Summary

- Sage can help solve math problems
- Basic, intuitive facilities for arithmetic
- Create variables to your heart's content
- Define expressions to avoid repeating computations