A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Computing with formulas

Foundation of programming (CK0030)

Francesco Corona

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

FdP

- ☹ **Intro to variables, objects, modules, and text formatting**
- ☹ Programming with WHILE- and FOR-loops, and lists
- ☹ Functions and IF-ELSE tests

- ☹ Data reading and writing
- ☹ Error handling
- ☹ Making modules

- ☹ Arrays and array computing
- ☹ Plotting curves and surfaces

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

A formula
Computing with formulas

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

A formula

Example

Consider the vertical motion of a ball thrown up in the air

From Newton's second law of motion one can set up a mathematical model for the motion of the ball and find that the vertical position of the ball, called y, varies with time t according to the simple formula

$$y(t) = v_0 t - \frac{1}{2} g t^2 \tag{1}$$

- v_0 is the initial velocity of the ball
- g is the acceleration of gravity
- t is time

The y axis is chosen such that the ball starts from $y = 0$ at $t = 0$

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

A formula (cont.)

To get an overview of the time it takes for the ball to move upwards and return to the ground again, we look for solutions to equation $y(t) = 0$

$$v_0 t - \frac{1}{2} g t^2 = t(v_0 - \frac{1}{2} g t) = 0; \quad \Longrightarrow \quad t = 0, \text{ or } t = 2\frac{v_0}{g} \qquad (2)$$

The ball returns in $2v_0/g$ seconds, we are interested in $t \in [0, 2v_0/g]$

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

A formula (cont.)

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

We evaluate the formula for some values of v_0 and g

- $v_0 = 5 \text{ ms}^{-1}$
- $g = 9.81 \text{ ms}^{-2}$

We want to compute the ball's height for $t = 0.6$ s

Computing with formulas

UFC/DC
FdP - 2017.1

A formula

Programs and programming

Variables, reserved words

Comments, text formats and numbers

Another formula

Integer division

Objects in Python

Avoiding integer division

Arithmetic operators

Mathematical functions

Examples

Rounding errors

Interactive computing

The shell

Type conversion

IPython

Complex numbers

Complex arithmetics

Complex functions

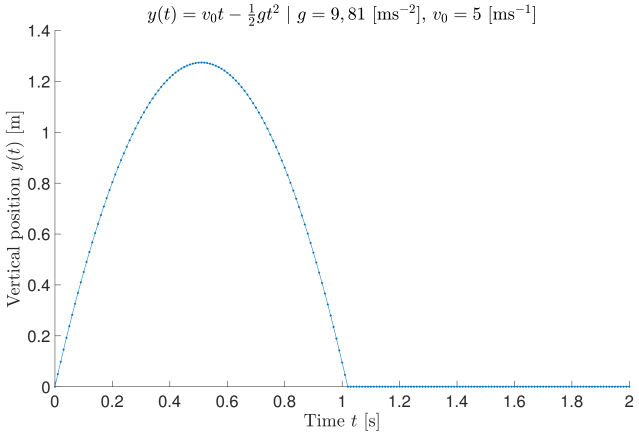
Symbolic computing

Differentiation/integration

Equation solving

Taylor series and more

A formula (cont.)



A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

A formula (cont.)

$$y = \underbrace{5}_{v_0} \cdot \underbrace{0.6}_{t} - \frac{1}{2} \cdot \underbrace{9.81}_{g} \cdot \underbrace{0.6^2}_{t^2} \tag{3}$$

```python
print 5*0.6 - 0.5*9.81*0.6**2
```

Remark

The four **standard arithmetic operators** are written as `+`, `-`, `*` and `/`

- The exponentiation employs a double asterisk `**` notation

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

A formula (cont.)

The arithmetic expression is easily evaluated and printed

- A one-line Python program

The ball comes back after some time $t = 2v_0/g \approx 1$ [s]

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Programs and programming
A formula

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Programs and programming

Our task is to create programs/code and run it

There are three main types of tools for writing Python code

- A plain text editor
- An integrated development environment (IDE) with a text editor
- An IPython notebook

Remark

What you choose depends on how you access Python

- There are various possibilities to install it on your own computer, access a pre-installed environment, or access it in cloud services

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Programs and programming (cont.)

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

```
print 5*0.6 - 0.5*9.81*0.6**2
```

A simple yet **complete Python program** for evaluating the formula

- *Save the line to a* **text file** *with name* `ball1.py`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Programs and programming (cont.)

The action required to run this program depends on the chosen tool

- Terminal window, IPython, Spyder, IPython notebook, ...

Example

```
Terminal> python ball1.py

1.2342
```

After execution of `ball1.py`, the output (`1.2342`) is printed to screen

[*Run me in a terminal, with/without ipython, in spyder, a notebook ...*]

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Programs and programming (cont.)

Suppose you now want to evaluate the formula for $v_0 = 1$ and $t = 0.1$

1. One must first edit the program text
2. Then, program must be re-executed

Example

$$y = \underbrace{1}_{v_0} \cdot \underbrace{0.1}_{t} - \frac{1}{2} \cdot \underbrace{9.81}_{g} \cdot \underbrace{0.1^2}_{t^2}$$

```
print 1*0.1 - 0.5*9.81*0.1**2
```

```
Terminal> python ball1.py

0.05095
```

The calculation has changed and the output is now 0.05095

A formula

Programs and programming

Variables, reserved words

Comments, text formats and numbers

Another formula

Integer division

Objects in Python

Avoiding integer division

Arithmetic operators

Mathematical functions

Examples

Rounding errors

Interactive computing

The shell

Type conversion

IPython

Complex numbers

Complex arithmetics

Complex functions

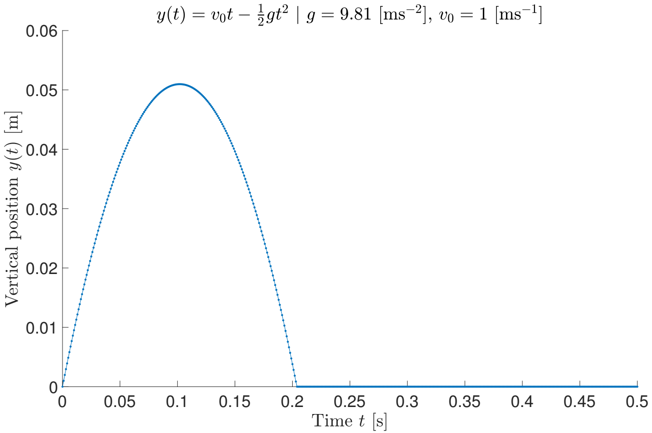
Symbolic computing

Differentiation/integration

Equation solving

Taylor series and more

A formula (cont.)



A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Programs and programming (cont.)

Whenever we would want to evaluate $y(t)$ for many values of t, we would have to modify the value of t at two places in our program

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

Such modifications would be simpler to perform if we could express formulas in terms of **variables** (symbols) rather than numerical values

- Most programming languages, Python included, can use variables

A formula

Programs and programming

Variables, reserved words

Comments, text formats and numbers

Another formula

Integer division

Objects in Python

Avoiding integer division

Arithmetic operators

Mathematical functions

Examples

Rounding errors

Interactive computing

The shell

Type conversion

IPython

Complex numbers

Complex arithmetics

Complex functions

Symbolic computing

Differentiation/integration

Equation solving

Taylor series and more

Programs and programming (cont.)

Definition

Variables are defined by setting a name (here `v0`, `g`, `t`, or `y`) equal to a numerical value or an expression involving already defined variables

Example

```
v0 = 5
g = 9.81
t = 0.6

y = v0*t - 0.5*g*t**2

print y
```

This second program is much easier to read because closer to the mathematical notation used in the original formula

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Programs and programming (cont.)

- *Store the program text in a file* `ball2.py`
- Running the program outputs `1.2342`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Variables and reserved words
A formula

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Variables and reserved words

Variable names can contain any lower or upper case letter, numbers from 0 to 9, and underscore, but first character cannot be a number

Remark

- Python distinguishes between upper and lower case letters
- `X` is different from `x`, `Xx` from `xX`, ...

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Variables and reserved words (cont.)

Example

```
initial_velocity = 5
acceleration_of_gravity = 9.81
TIME = 0.6

VerticalPositionOfBall = initial_velocity*TIME - \
                         0.5*acceleration_of_gravity*TIME**2

print VerticalPositionOfBall
```

With long variables names, the code for evaluating the formula got long

- We broke it into two lines (the backslash at the end of the line)
- Make sure there are no blanks after the backslash

Long names explain well what they represent, but checking correctness of the formula for `y` is harder than in the program using `v0`, `g`, `t`, and `y0`

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Variables and reserved words (cont.)

A standard convention is to have variable names with lower case letters

- then, words are separated by an underscore

Example

Whenever the variable represents a mathematical symbol, we use it

- y in mathematics becomes `y` in the program
- v_0 in mathematics becomes `v0` in the program

Resemblance between mathematical symbols and variables names is important for easy reading of the code and for detecting errors

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Variables and reserved words (cont.)

Certain words are reserved in Python: Utilised to build the language

These **reserved words** cannot be used as variable names

- `and`, `as`, `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `False`, `finally`, `for`, `from`, `global`, `if`, `import`, `in`, `is`, `lambda`, `None`, `nonlocal`, `not`, `or`, `pass`, `raise`, `return`, `True`, `try`, `with`, `while`, and `yield`

Remark

To use a reserved word as variable name, add an underscore at the end

- For some quantity λ, use `lambda_`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Variables and reserved words (cont.)

Program files can have a freely chosen name, but it is good practice to avoid names that coincide with **keywords** or **module** names in Python

- `math.py`, `time.py`, `random.py`, `os.py`, `sys.py`, `while.py`, `for.py`, `if.py`, `class.py`, or `def.py`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers

A formula

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Comments, text formats and numbers

Along with code statements, it is informative to provide **comments**

- In a natural language, to explain the idea behind statements

Definition

Comments in Python start with the # character, everything after this character on a line is ignored when the program is executed

Example

```python
# Compute the height of a ball in vertical motion

v0 = 5                    # initial velocity
g = 9.81                  # acceleration of gravity
t = 0.6                   # time

y = v0*t - 0.5*g*t**2 # vertical position

print y
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

Remark

If you use **non-English characters** in comments, Python will complain

```
SyntaxError: Non-ASCII character '\xc3' in file ...
but no encoding declared; see
http://www.python.org/peps/pep-0263.html for details
```

Non-English characters are enabled by using a line in the code beginning

```
# -*- coding: utf-8 -*-
```

- This is a comment that is not ignored by Python

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Comments, text formats and numbers (cont.)

Instead of printing a numerical value of y, we may want to write a more informative text: `At t=0.6 s, the height of the ball is 1.23 m`

Printf syntax: Output from a `print` statement plus number formatting

- The oldest and most widely used such technique is known as printf formatting (from function `printf` in the C programming language)
- The syntax of printf formatting may look awkward, but it is easy to learn and very convenient and flexible to work with
- The printf syntax is used in a lot of other programming languages

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Comments, text formats and numbers (cont.)

```
print 'At t=%g s, the height of the ball is %.2f m.' % (t, y)
```

- The `print` statement prints a **string**: everything enclosed in quotes (either single, `'`, or double `"`) denotes a string

The string above is formatted using printf syntax

- The string has '**slots**', starting with a percentage sign, in which variables in the program can be inserted
- In the example: `%g` and `%.2f`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

```
print 'At t=%g s, the height of the ball is %.2f m.' % (t, y)
```

We have two ‘slots’, thus two variables must be put into the slots

The relevant syntax is to list the variables inside parentheses after the string, `% (t, y)`, separated from it by a percentage symbol

- The first variable, `t`, goes into first ‘slot’ with format specification `%g`, where the percentage sign **marks the slot** and the following character, `g`, is the chosen **format specification**
- The `g` format instructs the real number to be compactly written

- The next variable, `y`, goes into second ‘slot’ with format `.2f`, which means the real number written with two decimal digits
- The `f` in the `.2f` format stands for floating-point number

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Comments, text formats and numbers (cont.)

Example

```
v0 = 5
g = 9.81
t = 0.6

y = v0*t - 0.5*g*t**2

print 'At t=%g s, the height of the ball is %.2f m.' % (t, y)
```

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Comments, text formats and numbers (cont.)

There are many ways to specify formats

- `e` writes a number in **scientific notation**: A number between 1 and 10 followed by a power of 10 ($1.2432 \cdot 10^{-3}$, as `1.2432e-03`)
- Capital `E` in the exponent is possible: Replace `e` by `E` (`1.2432E-03`)

- For **decimal notation** we use letter `f`, as in `%f`, and the output number then appears with digits before and/or after a comma (`0.0012432` instead of `1.2432E-03`)

- With the `g` format, the output is in scientific notation for large or small numbers and decimal notation otherwise (**compact output**)
- A lower case `g` leads to lower case `e` in scientific notation, while upper case `G` implies `E` instead of `e` in the exponent

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

Remark

One can also specify the format in some very sophisticated manner

Example

- `10.4f`
- `14.6E`

- In the first case a float is written in decimal notation with 4 decimals in a field of width equal to 10 characters
- In the second case a float is written in scientific notation with 6 decimals in a field of 14 characters

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Comments, text formats and numbers (cont.)

Format	Explaination
`%s`	A string
`%d`	An integer
`%0xd`	An integer in a `x`-width field, padded with leading zeros
`%f`	Decimal notation with six decimals
`%e`	Compact scientific notation, `e` in the exponent
`%E`	Compact scientific notation, `E` in the exponent
`%g`	Compact decimal or scientific notation, with `e`
`%G`	Compact decimal or scientific notation, with `E`
`%xz`	Format `z` right-adjusted in a `x`-width field
`%-xz`	Format `z` left-adjusted in a `x`-width field
`%.yz`	Format `z` with `y` decimals
`%x.yz`	Format `z` with `y` decimals in a `x`-width field
`%%`	The percentage sign

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

Example

```
i = 62
r = 189876545.7654675432

# Print out numbers with quotes "" to see width of field

print '"%d"' % i          # minimum field
print '"%5d"' % i         # field of width 5 characters
print '"%05d"' % i        # pad with zeros

print '"%g"' % r          # r is big number, scientific notation
print '"%G"' % r          # E in the exponent
print '"%e"' % r          # compact scientific notation
print '"%E"' % r          # compact scientific notation
print '"%20.2E"' % r      # 2 decimals, field of width 20
print '"%30g"' % r        # field of width 30 (right-adjusted)
print '"%-30g"' % r       # left-adjust number
print '"%-30.4g"' % r     # 3 decimals

print '%s' % i            # convert i to string automatically
print '%s' % r

# Use %% to print the percentage sign
print '%g %% of %.2f Euro is %.2f Euro' % \
      (5.1, 346, 5.1/100*346)
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

Example

```
v0 = 5
g = 9.81
t = 0.6

y = v0*t - 0.5*g*t**2

print """
At t=%f s, a ball with
initial velocity v0=%.3E m/s
is located at the height %.2f m.
""" % (t, v0, y)
```

A **triple-quoted string**, started and ended by three single/double quotes

Triple-quoted strings are used for text that spans several lines

- `t` is printed in the `f` format (by default six decimals);
- `v0` is written in the `.3E` format (three decimals and the number spans as narrow field as possible);
- `y` is two decimals in narrow decimal notation, `.2f`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

```
Terminal> python ball_print2.py

At t=0.600000 s, a ball with
initial velocity v0=5.000E+00 m/s
is located at the height 1.23 m.
```

- `t` is printed in the `f` format (by default six decimals);
- `v0` is written in the `.3E` format (three decimals and the number spans as narrow field as possible);
- `y` is two decimals in narrow decimal notation, `.2f`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

Format string syntax: Offers all the functionality of the printf format

- and much more, through a different syntax

Example

We illustrate this syntax on the one-line output that was used earlier

```
print 'At t={t:g} s, the height of the ball is {y:.2f} m.' \
      .format(t=t, y=y)
```

- Slots are denoted by curly braces (rather than a percentage sign)
- Variable are listed with an optional colon and format specifier
- Variables and their values are listed at the end of the statement
- Slots have names (the sequence of variables is not important)

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

At times, we want to write out text that spans several lines

```
print """
At t={t:f} s, a ball with
initial velocity v0={v0:.3E} m/s
is located at the height {y:.2f} m.
""".format(t=t, v0=v0, y=y)
```

We can obtain such an output by using triple-quoted strings

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Comments, text formats and numbers (cont.)

The **newline character**: We can also use ordinary single-quoted strings and a special character for indicating where line breaks should occur

- The special character is `\n` (a backslash followed by the letter `n`)

Example

```
print """y(t) is
the position of
our ball."""

print 'y(t) is\nthe position of\nour ball'
```

The two `print` statements have identical output

```
y(t) is
the position of
our ball.
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Another formula
Computing with formulas

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Another formula

Example

Consider the expression for converting a temperature measurement in degrees Celsius (C) to corresponding value in degrees Fahrenheit (F)

$$F = \frac{9}{5}C + 32 \tag{4}$$

Given the formula above and a value of C, our goal is to compute F

A first attempt at implementing the formulamay be

```
C = 21
F = (9/5)*C + 32
print F
```

Obviously, the parentheses are not strictly needed

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Another formula (cont.)

```
C = 21
F = (9/5)*C + 32
print F
```

When run under Python version 2.x, the program prints the value 53

Testing correctness is easy, we evaluate the formula on a calculator

$$\frac{9}{5} \cdot 21 + 32 = 69.8 \neq 53$$

What is wrong? The formula typed in the program looks correct!

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Integer division
Another formula

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Integer division

The error is one of the most common errors in math coding

- For a newcomer to programming, it is not at all obvious

In many computer languages, there are two types of divisions

- **Float division** and **integer division**

Definition

Float division is what you expect from standard arithmetics

- $9/5$ becomes 1.8 in decimal notation

Integer division a/b with integers a and b is an integer c

- It is the largest integer c such that $bc \leq a$

- $9/5$ is 1, as $1 \cdot 5 = 5 \leq 9$ and $2 \cdot 5 = 10 > 9$
- $1/5$ is 0, as $0 \cdot 5 \leq 1$ and $1 \cdot 5 > 1$

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Integer division (cont.)

Remark

Many computer languages (..., Fortran, C, C++, Java, and Python 2.x) interpret a/b as integer division, if both operands a and b are integers

- If either a or b are real (floating-point) numbers, then a/b implies the standard mathematical float division

Other languages (..., MATLAB and Python 3.x) interpret a/b as float division even if both operands are integers, or complex division if one of the operands is a complex number

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Integer division (cont.)

The issue with the program is the coding of the formula `(9/5)*C + 32`

```
C = 21
F = (9/5)*C + 32
print F
```

- First, `9/5` is calculated: Python interprets `9` and `5` as integers
- `9/5` is thus interpreted as a division between two integers
- Python chooses by default integer division, giving `1`
- Then, `1` is (normally) multiplied by `C`, giving `21`
- `21` and `32` are added, with `53` as (wrong) result

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Objects in Python

Another formula

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Objects in Python

In an assignment statement like `C = 21` Python interprets number 21 as an integer and creates an **int** (for integer) **object** holding the value 21

- The variable `C` acts as **variable name** for this **int object**

Similarly, in `C = 21.0`, Python recognises `21.0` as a real number and it creates a **float** (for floating-point) **object** holding the value 21.0

- The variable `C` is the **variable name** of this **float object**

Remark

Any assignment statement has the form of a variable name, on the left-hand side, and an object, on the right-hand side

- It is not needed to know now what an object exactly is

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Objects in Python (cont.)

As initial simplification, one can think of an `int object` as a collection

- It is like a storage box, with some information about an integer
- The information is stored somewhere within the computer's memory and the name `C` is used to access this information

Remark

The key issue is that 21 and 21.0 are identical numbers in mathematics

- In Python, `21` gives an `int object` while `21.0` a `float object`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Objects in Python (cont.)

There are various object types, some are pre-built some are user-defined

- Objects may contain a lot of data, not just integer/real numbers

Example

```
print 'A text with an integer %d and a float %f' % (2, 2.0)
```

A **str** (for string) **object**, without a name, is first created from 'the text between quotes' and then the `str object` is printed

We can alternatively do this in two, sequential, steps:

```
s = 'A text with an integer %d and a float %f' % (2, 2.0)
print s
```

A formula

Programs and programming

Variables, reserved words

Comments, text formats and numbers

Another formula

Integer division

Objects in Python

Avoiding integer division

Arithmetic operators

Mathematical functions

Examples

Rounding errors

Interactive computing

The shell

Type conversion

IPython

Complex numbers

Complex arithmetics

Complex functions

Symbolic computing

Differentiation/integration

Equation solving

Taylor series and more

Avoiding integer division
Another formula

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Avoiding integer division

Careful to avoid integer division when coding mathematical formulas

- When a program uses integer division, a double forward-slash `//` should be used as division operator
- This is Python's way of explicitly indicating integer division

Remark

Python 3.x has no problem with unintended integer division

- Only with Python 2.x (and other languages)

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Avoiding integer division

There are several ways to avoid integer division with the plain / operator

The simplest remedy in Python version 2 is to write

```
from __future__ import division
```

This import statement must be present in the beginning of EVERY single file where the / operator ALWAYS shall imply float division

Alternatively, one can run any Python program `someprogram.py` from the command line with the argument `-Qnew` for the Python interpreter

```
Terminal> python -Qnew someprogram.py
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Avoiding integer division (cont.)

A more widely used method, common also to other programming languages, is to force one of the operands to be a `float object`

Example

```
F = (9.0/5)*C + 32
F = (9/5.0)*C + 32

F = float(C)*9/5 + 32
```

In the first two lines, one of the operands is written as a decimal number, implying a `float object` and therefore float division

In the last line, `float(C)*9` means (`float` times `int`), which results in a `float object`, and thus float division is implicitly guaranteed

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Avoiding integer division (cont.)

Example

```

F = C*float(9/5) + 32          # !! does not work correctly !!
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Avoiding integer division (cont.)

To locate *potential* integer divisions, Python programs can be executed with a `-Qwarnall` argument, which will display a warning every time an integer division expression is found in Python 2.x

```
Terminal> python -Qwarnall someprogram.py
```

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Avoiding integer division (cont.)

Remark

We could have run into problems if instead of writing formula $\frac{1}{2}gt^2$ as `0.5*g*t**2` had we written it as `(1/2)*g*t**2`

- The term `(1/2)` would always be zero

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Arithmetic operators
Another formula

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Arithmetic operators

In Python, formulas are evaluated as they are mathematically

- Given an expression, from left to right, term by term
- Terms are separated by plus (+) or minus (-)

Within terms, power operations (a^b, `a**b`)
has precedence over multiplication/division

Parentheses dictate how a formula is evaluated

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Arithmetic operators (cont.)

Example

```
5/9 + 2*a**4/2
```

$5/9$ (`5/9`) is evaluated (as integer division, with `0` as result), a^4 (`a**4`) is evaluated, 2 and a^4 are multiplied (`2*a**4`) and the result is divided by 2 (`2*a**4/2`), the result is added to the result of $5/9$ (`5/9 + 2*a**4`)

- The answer is therefore `a**4`

Example

```
5/(9+2)*a**(4/2)
```

$\frac{5}{9+2}$ (`5/(9+2)`) is evaluated (integer division, yielding `0`), $4/2$ (`4/2`) is computed (integer division, yielding `2`), `a**2` (`a**(4/2)`) is calculated, the result is multiplied by the result of `5/(9+2)` (`5/(9+2)*a**(4/2)`)

- The answer is thus always `0`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Arithmetic operators (cont.)

It is easy to unintentionally get integer division in formulas

Integer division can be turned off in Python, it is nonetheless important to be aware of the concept and develop programming habits to avoid it

Remark

The concept of integer division appears in many programming languages

- It is better to learn as early as possible how to deal with it, rather than using Python-specific (or else) features to remove the problem

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Mathematical functions
Computing with formulas

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Evaluating mathematical functions

Standard mathematical formulas frequently involve common functions

- sin, cos, tan, sinh, cosh, exp, log, ...

On a pocket calculator you have special buttons for such functions

- Similarly, in a language you have ready-made functionalities

Remark

In principle, one could write his/her own program for evaluating (e.g., the $\sin(x)$ function), but how to do it efficiently is a non-trivial task

- Experts have worked on such problem for decades and implemented their best recipes in pieces of software, that should be re-used

We discuss how to reach sin, cos, and similar functions within Python

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples
Mathematical functions

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Examples, `sqrt` and `sinh`

Example

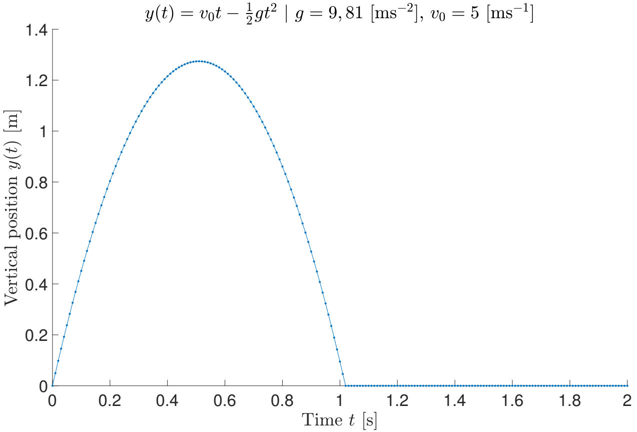
The height y of a ball in vertical motion, with initial upward velocity v_0

$$y = v_0 t - \frac{1}{2} g t^2$$

In the formula, we are using g for the gravity acceleration and t for time

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)



How long time does it take for the ball to reach the height y_c?

- There are two solutions (t_1 and t_2), one when the ball reaches y_c on its way up (t_1) and one when it reaches on its way down (t_2)

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

- When $y = y_c$, we have $y_c = v_0 t - \frac{1}{2} g t^2$ and the equation

$$\frac{1}{2} g t^2 - v_0 t + y_c = 0 \tag{5}$$

- A quadratic form[1] that must be solved with respect to t

$$t_1 = \frac{v_0 - \sqrt{v_0^2 - 2 g y_c}}{g}$$
$$t_2 = \frac{v_0 + \sqrt{v_0^2 - 2 g y_c}}{g}$$

[1] $ax^2 + bx + c = 0$, $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

For the expressions of t_1 and t_2, we need (want) the square root ($\sqrt{(\cdot)}$)

Remark

In Python, the square root function and other mathematical functions, such as sin, cos, sinh, exp, log, ... are available in a **module** called `math`

We must first **import a module** to make its functions available

- We can write `import math` in our program

To take the square root of variable a, $\sqrt{a}$, write `math.sqrt(a)`

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Example

$$t_{(1|2)} = \frac{v_0 \mp \sqrt{v_0^2 - 2gy_c}}{g}$$

```
v0 = 5
g = 9.81
yc = 0.2

import math
t1 = (v0 - math.sqrt(v0**2 - 2*g*yc))/g
t2 = (v0 + math.sqrt(v0**2 - 2*g*yc))/g

print 'At t=%g s and %g s, the height is %g m.' % (t1, t2, yc)
```

The output from this program is

```
At t=0.0417064 s and 0.977662 s, the height is 0.2 m.
```

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Definition

The standard way to import a module, say `math`, is `import math` and then one accesses individual functions using the module name as prefix

- `module_name.function_name`

Example

```python
import math
x = math.sqrt(y)
```

Clearly, the use of `math.sqrt(y)` is less pleasing than a plain `sqrt(y)`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Definition

An alternative import syntax allows us to skip the module name prefix

- `from module_name import function_name`

A specific example is `from math import sqrt exp log sin`, which allows to work with `sqrt` (or else) directly, without the `math.` prefix

Example

```
v0 = 5
g = 9.81
yc = 0.2

from math import sqrt                         # WAS: import math
t1 = (v0 - sqrt(v0**2 - 2*g*yc))/g
t2 = (v0 + sqrt(v0**2 - 2*g*yc))/g
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Definition

All functions in a module can also be imported all at once

```
from math import *
```

- This includes `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `sinh`, `cosh`, `tanh`, `exp`, `log` (base e), `log10` (base 10), `sqrt`, and numbers (`e`, `pi`, ...)
- Importing all (`*`) functions from a module is convenient
- Not recommended to import more functions than needed
- The convenience of a compact import syntax often wins

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Definition

Modules and functions can be given new names in the import statement

Example

```
import math as m          # m is now the name of the math module

v = m.sin(m.pi)

from math import log as ln
v = ln(5)

from math import sin as s, cos as c, log as ln
v = s(5)*c(5) + ln(5)
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Remark

Since in Python everything is an object, variables refer to objects and new variables may refer to modules, functions, numbers and strings

```
m = math
ln = m.log
s = m.sin
c = m.cos
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

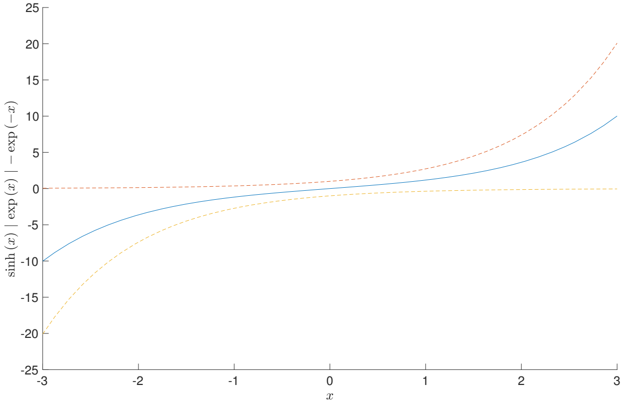
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Example

Consider the definition of the hyperbolic function $\sinh(x)$

$$\sinh(x) = \frac{1}{2}\Big(e^{x} - e^{-x}\Big) \tag{6}$$



A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

$$\sinh(x) = \frac{1}{2}\left(e^{x} - e^{-x}\right)$$

We can evaluate $\sinh(x)$ in three different ways

- By calling `math.sinh`, directly
- By computing the RHS using `math.exp`
- By computing the RHS using `e` and power expressions `math.e**x` and `math.e**(-x)`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Examples, `sqrt` and `sinh` (cont.)

Example

```
from math import sinh, exp, e, pi

x = 2*pi

r1 = sinh(x)
r2 = 0.5*(exp(x) - exp(-x))
r3 = 0.5*(e**x - e**(-x))

print r1, r2, r3
```

All three computations are mathematically equivalent

- Output from `print` displays identical results

```
267.744894041 267.744894041 267.744894041
```

..., SQN

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Rounding errors
Mathematical functions

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Rounding errors

Example

A print out of `r1`, `r2`, `r3` that displays 16 decimals

```
print '%.16f %.16f %.16f' % (r1,r2,r3)
```

```
267.7448940410164369
267.7448940410164369
267.7448940410163232
```

shows how `r1`, `r2`, `r3` are different, but why is it so?

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Rounding errors (cont.)

Remark

A program computes with *wannabe* real numbers[2], and true real numbers (Dedekind) may require an infinite number of decimals

- Because of finite storage, the sequence of decimals is truncated
- On computers, it is standard to keep 17 digits in a real number

Remark

Real numbers on a computer often have a small error, only few can be represented exactly, the rest are approximations

- Most arithmetic operations on a computer necessarily involve inaccurate real numbers
- This results in inaccurate calculations

[2] Let $x \in \mathbb{R}$ and let $\text{fl}(x)$ its (rounded) representation in a computer. We have that $x \neq \text{fl}(x)$ with $\dfrac{|x - \text{fl}(x)|}{|x|} \leq \dfrac{1}{2}\varepsilon_M$ in which the quantity ε_M is called *machine precision*.

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Rounding errors (cont.)

Example

Think of $\frac{1}{49}49 = 1$ and $\frac{1}{51}51 = 1$ when performed in Python

```
print '%.16f %.16f' % (1/49.0*49, 1/51.0*51)
```

```
0.9999999999999999 1.0000000000000000
```

- $1/49$ is not correctly represented in the computer
- $1/51$ also has an inexact representation,
  but error does not show too much (:/)

Errors in floating-point numbers may propagate through computations and the results are approximations to the exact mathematical values

- Such errors are commonly called **rounding errors**

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Rounding errors (cont.)

Remark

Python has module `decimal` and `SymPy` package has module `mpmath` that allows for real numbers to be represented with adjustable accuracy

- Rounding errors can be made as small as desired

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Interactive computing
Computing with formulas

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Interactive computing

Python can execute statements and evaluate expressions interactively

The environments where one works interactively are **Python shells**

- The simplest Python shell is invoked by `python` in terminal

```
Terminal> python
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more
    information.

>>>
```

Some Python messages are displayed together with a **prompt** > > >

- After that, you can start issuing commands

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Interactive computing (cont.)

Example

The interactive shell as calculator

Type `3*4.5-0.5` and press `Return`

```
Terminal> python
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more
    information.

>>> 3*4.5-0.5
    13.0
```

The text after the > > > prompt is the **shell input** and the text without the > > > prompt is the result that Python calculates (**shell output**)

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Interactive computing (cont.)

Remark

- The shell makes it easy to recover previous input and edit the text
- This helps experimenting with statements and expressions

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

The shell
Interactive computing

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

The shell

Example

The program for the vertical position of the ball

```
v0 = 5
g = 9.81
t = 0.6
y = v0*t - 0.5*g*t**2
print y
```

It can be re-typed line-by-line in the Python shell

```
>>> v0 = 5
>>> g = 9.81
>>> t = 0.6
>>> y = v0*t - 0.5*g*t**2
>>> print y
    1.2342
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

The shell

We can easily calculate the `y` value corresponding to another `v0` value

- Hit the arrow-up key (⇑), to recover previous statements
- Repeat pressing ⇑, until the `v0 = 5` statement shows up
- You can then edit the relative line

```
>>> v0 = 6                                    # It was: v0 = 5
```

- Press `Reurn`, to execute this statement
- To check the new value of `v0` either type `v0` or `print v0`

```
>>> v0
    6

>>> print v0
    6

>>> print y                         # Old, needs be re-computed
    1.2342
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

The shell (cont.)

The next step is to re-compute `y`, with the new `v0` value

- Hit the arrow-up key ($\Uparrow$) multiple times to recover the statement where `y` is assigned
- Press `Return`
- Write `y` or `print y` to see the result

```
>>> y = v0*t - 0.5*g*t**2
>>> y
    1.8341999999999996

>>> print y
    1.8342
```

The reason why we get two slightly different results is that typing `y` prints out all the decimals that are stored in the computer (16)

- `print y` prints out `y` with fewer decimals, standard format

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

The shell (cont.)

Remark

Computations on a computer often suffer from rounding errors

- The present calculation is no exception

The correct answer is 1.8342, rounding errors lead to a number that is incorrect, in the 16th decimal

- The error is $4 \cdot 10^{-16}$

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Type conversion
Interactive computing

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Type conversion

In Python it is often possible to work without bothering too much about the **type of objects** variables refer to

- Yet, we encountered a serious problem with integer division
- Important to be careful about the involved types of objects

The interactive shell is useful for exploring types (the `type` function)

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Type conversion (cont.)

Example

Let us create some `int object` `C` and check its type with `type(C)`

```
>>> C = 21
>>> type(C)
    <type 'int'>

>>> C
    21
```

We convert the `int object` `C` to a corresponding `float object`

```
>>> C = float(C)                                          # type conversion
>>> type(C)
    <type 'float'>

>>> C
    21.0
```

Statement `C = float(C)` creates a new object, `C`, from the original one, also referred to by the name `C`, and binds it to the same name `C`

- After the statement, `C` refers to a different object
- The original `int object`, value `21`, is unreachable

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Type conversion (cont.)

Example

We can also convert a `float object` to a corresponding `int object`

```
>>> C = 20.9
>>> type(C)
    <type 'float'>

>>> D = int(C)                                  # type conversion
>>> type(D)
    <type 'int'>

>>> D
    20                                      # decimals are truncated
```

Converting a `float` to an `int` implied stripping off the decimals

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Type conversion (cont.)

Example

Conversion according to rounding rules is achieved by `round` function

```
>>> round(20.9)
    21.0

>>> int(round(20.9))
    21
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython
Interactive computing

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython

There are several improvements of the standard Python shell

- **IPython** is the common interactive shell
- You need to have `ipython` installed

Typing `ipython` in a terminal window starts the shell

```
Terminal>  ipython
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
Type "copyright", "credits" or "license" for more information.

IPython 2.3.0 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra
              details.

In [1]:
```

Definition

The (default) prompt in `ipython` is not `> > >` but `In [X]:`

- `X` is the number of the present input command

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

iPython (cont.)

Example

Running programs

```
Terminal> ipython
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
Type "copyright", "credits" or "license" for more information.

IPython 2.3.0 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra
              details.

In [1]: run ball2.py
        1.2342
```

The command requires that you have `cd`'ed to the folder with `ball2.py`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython (cont.)

On Windows you may, as alternative to starting IPython from a DOS or PowerShell window, double click on the IPython icon or use Start menu

- You must move to the folder where your program is located
- If `ball2.py` is in the folder `div` under `My Stuff` of user `me`, this is done by the `os.chdir` (change directory) command

```
In [1]: import os
In [2]: os.chdir(r'C:\Documents and Settings\me\My Stuff\div')
In [3]: run ball2.py
```

- Note the `r` before the quote in the string
- It is required to let a backslash (\\) really mean the backslash character

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython (cont.)

Remark

If you frequently have to type the `os.chdir` command in `ipython`, note that this and other commands can be suitably placed in a **startup file**

- A file that is automatically executed when you launch `ipython`
- To create one from Terminal, `ipython create profile`

Inside `ipython` you can invoke any operating system command

This allows us to navigate the filesystem by using Unix or Windows commands (`cd`) instead of Python's (`os.chdir`)

```
In [1]: cd C:\Documents and Settings\me\My Stuff\div
In [3]: run ball2.py
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython (cont.)

Remark

If Python variables are defined with the same name as an OS command (`date=30`), then, within Python, the corresponding OS command must be called with an exclamation mark (`!`) in front of it (`!date`)

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython (cont.)

It is recommended to run all Python programs from inside `ipython`

- When something goes wrong, `ipython` can help examine the state of variables and locate bugs

Remark

- To execute a program in `ipython`, type `run` before program name
- To run a program in a Terminal, `python` prior to program name

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython (cont.)

Definition

Output from Python statements or expressions in `ipython` are preceded by `Out [X]`, where `X` is the command number of last `In [X]` prompt

- When programs are executed, as with the `run` command, or when OS commands are run, the output is from the OS itself
- In this case, the output is not preceded by any `Out[X]` label

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

IPython (cont.)

Definition

Output recovery

Outputs (`Out [X]`) from previous statements in `ipython` are available in variables of the form `_iX` (underscore `_`, `i`, and a number `X`), `X` is `1` for the last statement, `2` for the second last statement, and so forth

- Short forms are `_` (for `_i1`), `__` (for `_i2`), and `___` (for `_i3`)

Example

The output from input `In [1]` was `1.2342`, we can now refer to it by an underscore and perform operations on it, say multiply it by `10`

```
In  [2]:  _*10
Out[2]:  12.341999999999999
```

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

IPython (cont.)

Definition

Command recovery

The command history is navigated by typing `Ctrl+p` or $\Uparrow$ for going backward or `Ctrl+n` or $\Downarrow$ for going forward

- Any command you hit can be edited and re-executed
- Also commands from previous sessions are in history

Definition

The **command history** from previous `ipython` sessions is available

- This feature makes it easy to modify work from a previous session by hitting arrow-up to recall commands and edit them as needed

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython (cont.)

Definition

Tab completion

Pressing the TAB key completes incompletely typed variable names

- It can save some typing

Definition

OS commands

```
In [3]: date
        Thu Nov 18 11:06:16 CET 2010

In [4]: ls
        myfile.py yourprog.py

In [5]: mkdir mytestdir

In [6]: cd mytestdir
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

IPython (cont.)

Remark

Notebooks

Alternative to interactive shells: It allows to record/replay interactive sessions with a mix of text, mathematics, Python code, and graphics

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex numbers
Computing with formulas

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex numbers

Let $x^2 = 2$ and $x = \pm\sqrt{2}$ be the solution of the equation

- What if $x^2 = -2$? We need to define complex numbers

Definition

A complex number is a pair of real numbers a and b jointly written in the form $a + ib$ or $a + ib$, i is the imaginary unit acting as label

- $i = \sqrt{-1}$

Intuitively, one of the *features* of complex numbers is the possibility to take square roots also of negative numbers

- $\sqrt{-2} = \sqrt{2}i = \sqrt{2}\sqrt{-1}$
- $\sqrt{-2} = \pm\sqrt{2}i$

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Remark

Let $u = a + bi$ and $v = c + di$

$$\begin{aligned}
u &= v, \quad \rightarrow a = c, b = d \\
-u &= -a - bi \\
u* &= a - bi \text{ (complex conjugate)} \\
u + v &= (a + c) + (b + d)i \\
u - v &= (a - c) + (b - d)i \\
uv &= (ac - bd) + (bc + ad)i \\
u/v &= \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2}i \\
|u| &= \sqrt{a^2 + b^2} \\
e^{iq} &= \cos(q) + i\sin(q)
\end{aligned}$$

Addition/subtraction/multiplication/division of complex numbers

- Raising a complex number $z = a + ib$ to a real power
- Transcendental functions of complex numbers $\sin(z)$, $\cos(z)$, $\tan(z)$, e^z, $\ln(z)$, $\sinh(z)$, $\cosh(z)$, $\tanh(z)$

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex arithmetics
Complex numbers

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Complex arithmetics

Python supports computation with complex numbers

In Python the imaginary unit is `j` (the i in mathematics)

- A complex number $2 - 3i$ is thus expressed as `(2-3j)`
- Number i is written as `1j`, not just `j`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex arithmetics

The definition of complex numbers and some simple arithmetics

Example

```
>>> u = 2.5 + 3j        # create a complex number
>>> v = 2               # this is an integer
>>> w = u+v             # complex + integer
>>> w
    (4.5+3j)

>>> a = -2              #
>>> b = 0.5             #
>>> s1 = a+b*1j         # complex number from two floats
>>> s1
    (-2+0.5j)

>>> s2 = complex(a,b) # alternative way
>>> s2
    (-2+0.5j)

>>> s*w                 # complex*complex
    (-10.5-3.75j)

>>> s/w                 # complex/complex
    (-0.25641025641025639+0.28205128205128205j)
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex arithmetics (cont.)

A `complex object` (`s`) is equipped with functionalities for extracting real and imaginary parts, as well as computing the complex conjugate

Example

```
>>> s = -2+0.5j
>>> s.real
    -2.0

>>> s.imag
    0.5

>>> s.conjugate()
    (-2-0.5j)
```

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex functions
Complex numbers

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex functions

Plainly calculating the sine of a complex number is not permitted

Example

```
>>> from math import sin
>>> r = sin(w)
    Traceback (most recent call last):
     File "<input>", line 1, in ?
    TypeError: can't convert complex to float; use abs(z)
```

- The reason is that the `sin` function from the `math module` only works with real (float) arguments, and not with complex ones
- A similar module, `cmath`, defines functions that take a complex number as argument and return a complex number as result

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex functions (cont.)

Example

To illustrate the `cmath module`, we can show that $\sin(ai) = i \sinh(a)$

```
>>> from cmath import sin, sinh

>>> r1 = sin(8j)
>>> r1
    1490.4788257895502j

>>> r2 = 1j*sinh(8)
>>> r2
    1490.4788257895502j
```

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Complex functions (cont.)

Example

Another relation that can be verified $e^{iq} = \cos(q) + i\sin(q)$

```
>>> q = 8

>>> exp(1j*q)
    (-0.14550003380861354+0.98935824662338179j)

>>> cos(q) + 1j*sin(q)
    (-0.14550003380861354+0.98935824662338179j)
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex and real functions

Complex numbers

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Complex and real functions

The functions in the `cmath module` return complex numbers (objects)

Functions that return a `float object` if the result is a real number and a `complex object` if the result is a complex number would be useful

- `NumPy` has such versions of mathematical functions
- In the `math module` and in the `cmath module`

Computing with formulas

UFC/DC
FdP - 2017.1

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex and real functions (cont.)

To access/import these, more flexible, versions of such functions

```
from numpy.lib.scimath import *
```

or

```
from scipy import *
from scitools.std import *
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex and real functions (cont.)

Example

We start by taking a square root using sqrt in the math module

```
>>> from math import sqrt
>>> sqrt(4)                                        # float
    2.0

>>> sqrt(-1)                                       # illegal
    Traceback (most recent call last):
     File "<input>", line 1, in ?
    ValueError: math domain error
```

We then import sqrt from the cmath module and repeat

```
>>> from cmath import sqrt
>>> sqrt(4)                                        # complex
    (2+0j)

>>> sqrt(-1)                                       # complex
    1j
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex and real functions (cont.)

The `sqrt` function from `math` is *overwritten* by the `sqrt` from `cmath`

- More precisely, the name `sqrt` was previously bound to a function `sqrt` from the `math module`, but is now bound to another function `sqrt` from the `cmath module`

As a result, taking a square root now results in a `complex object`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex and real functions (cont.)

We can import, among other things, yet another `sqrt` function

```
>>> from numpy.lib.scimath import *
```

This function is slower than the versions from `math` and `cmath`, but it is more flexible as it returns a `float object` if possible, or a `complex one`

Example

```
>>> sqrt(4)                                              # float
    2.0

>>> sqrt(-1)                                           # complex
    1j
```

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Complex and real functions (cont.)

Example

To illustrate the flexible treatment of both complex and real numbers, we can code the roots of the quadratic function $f(x) = ax^2 + bx + c$

```
>>> a = 1; b = 2; c = 100           # polynomial coefficients

>>> from numpy.lib.scimath import sqrt

>>> r1 = (-b + sqrt(b**2 - 4*a*c))/(2*a)
>>> r2 = (-b - sqrt(b**2 - 4*a*c))/(2*a)

>>> r1
    (-1+9.94987437107j)

>>> r2
    (-1-9.94987437107j)
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Complex and real functions (cont.)

Using arrow-up (⇑), we can go back to the definitions of the coefficients

- We change them to be different numbers
- Then, we recompute r1 and r2

```
>>> a = 1; b = 4; c = 1              # polynomial coefficients
    ...
    ...

>>> r1
    -0.267949192431

>>> r2
    -3.73205080757
```

In this case, the results are float objects

Remark

Had we applied sqrt from cmath, r1 and r2 would always be complex objects, while sqrt from math would not handle the complex case

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

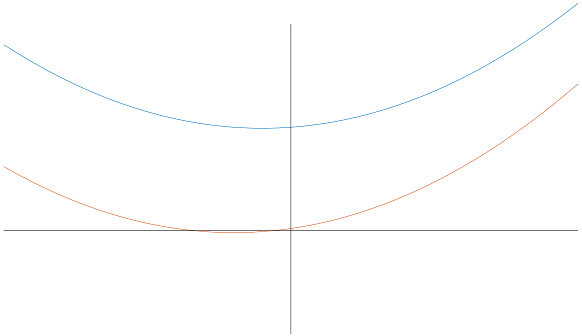
Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Taylor series and more (cont.)

Example

$$f_1(x) = x^2 + 2x + 100 \mid f_2(x) = x^2 + 4x + 1$$



A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Symbolic computing
Computing with formulas

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Symbolic computing

Python has a package `SymPy` for doing symbolic computing

- Symbolic integration and differentiation
- Equation solving, Taylor series expansion, etc

For interactive work with `SymPy` either use `ipython` or the special interactive shell `isympy`, which is installed along with `SymPy` itself

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Symbolic computing (cont.)

It is a good practice to explicitly import each symbol we need from `SymPy`, to emphasise that those symbols come from that package

Example

- It is important to know whether `sin` means the sine function from the `math module`, aimed at real numbers, or the special sine function from `SymPy`, aimed at symbolic expressions

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Differentiation and integration

Symbolic computing

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Differentiation and integration

How to differentiate functions (or expressions like $v_0 t - \frac{1}{2} g t^2$) with respect to an independent variable (in the example t)

- Then, we integrate the answer back

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Differentiation and integration (cont.)

Example

```
>>> from sympy import (
 ... symbols,      # define symbols for symbolic math
 ... diff,         # differentiate expressions
 ... integrate,    # integrate expressions
 ... Rational,     # define rational numbers
 ... lambdify,     # turn symbolic expr. into Python functions
 ... )

>>> t, v0, g = symbols('t v0 g')
>>> y = v0*t - Rational(1,2)*g*t**2
>>> dydt = diff(y,t)
>>> dydt
    -g*t + v0

>>> # 2nd derivative acceleration: -g
>>> print 'acceleration:', diff(y,t,t)
>>> y2 = integrate(dydt, t)
>>> y2
    -g*t**2/2 + t*v0
```

Note that `t` is a `symbolic variable` (not a `float`, as in numerical computing), and `y` (`y2`) is a `symbolic expression` (not a `float`)

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Differentiation and integration (cont.)

In SymPy, symbolic expressions can be turned into ordinary numerical functions, by using the lambdify command

Example

Take the dydt expression and turn it into a Python function v(t,v0,g)

- From symbolic to numerical

```
>>> # arguments in v and symbolic expression
>>> v = lambdify([t,v0,g],dydt)
>>> v(t=0,v0=5,g=9.81)
    5

>>> v(2,5,9.81)
    -14.62

>>> # control the previous calculation
>>> 5 - 9.81*2
     -14.62
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Equation solving
Symbolic computing

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Equation solving

A linear equation defined through an expression f (`e`) that is zero can be solved using `solve(f,t)` if t (`t`) is the unknown in the equation

Example

We want to find the roots of the expression $y = v_0 t - \frac{1}{2} g t^2$

```
>>> from sympy import solve

>>> t, v0, g = symbols('t v0 g')
>>> y = v0*t - Rational(1,2)*g*t**2

>>> roots = solve(y,t)
>>> roots
    [0, 2*v0/g]
```

A formula
- Programs and programming
- Variables, reserved words
- Comments, text formats and numbers

Another formula
- Integer division
- Objects in Python
- Avoiding integer division
- Arithmetic operators

Mathematical functions
- Examples
- Rounding errors

Interactive computing
- The shell
- Type conversion
- IPython

Complex numbers
- Complex arithmetics
- Complex functions

Symbolic computing
- Differentiation/integration
- Equation solving
- Taylor series and more

Equation solving (cont.)

We can easily check the answer, by inserting the roots in y

Example

```
>>> y.subs(t, roots[0])
    0

>>> y.subs(t, roots[1])
    0
```

Remark

- Inserting expressions `f2` for `f1` in expression `f` can done by using `f.subs(f1,f2)`

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Taylor series and more
Symbolic computing

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Taylor series and more

Calculate a Taylor polynomial of order n (`n`) of some function f (`f`) in the independent variable t (`t`) around an arbitrary point t_0 (`t0`)

Remark

The Taylor series of a real- (complex-) valued function $f(x)$ infinitely differentiable at a real (complex) argument $x = a$ is the power-series

$$= f(a) + f^{(1)}(a)(x-a) + \frac{f^{(2)}(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \cdots$$
$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

$f^{(n)}(a)$ is the n-th derivative of $f(x)$ evaluated at point $x = a$

$n!$ is the factorial of n: $n! = \prod_{k=1}^{n} k$, with $n! = 1$, for $n = 0$

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Taylor series and more (cont.)

- The Taylor's polynomial is computed by `e.series(t,t0,n)`

Example

Consider functions e^t and $e^{\sin(t)}$

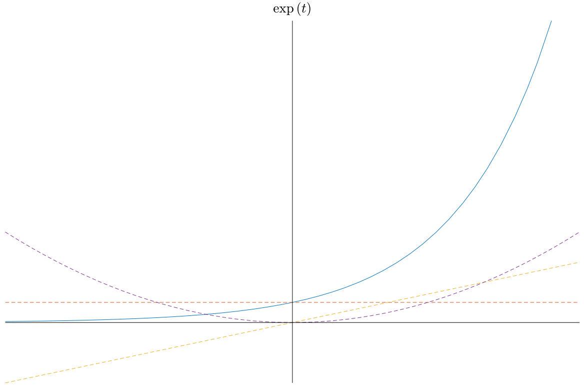
```
>>> from sympy import exp, sin, cos

>>> f = exp(t)
>>> f.series(t,0,3)
    1+t+t**2/2+O(t**3)

>>> f = exp(sin(t))
>>> f.series(t,0,8)
    1 + t + t**2/2 - t**4/8 - t**5/15 -
    t**6/240 + t**7/90 + O(t**8)
```

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Taylor series and more (cont.)

Example

$$f(t) = \exp(t) \approx 1 + t + \frac{t^2}{2} + \mathcal{O}(t^3)$$

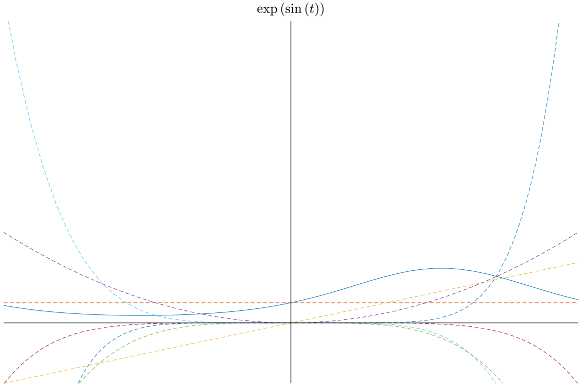


A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers
Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators
Mathematical functions
Examples
Rounding errors
Interactive computing
The shell
Type conversion
IPython
Complex numbers
Complex arithmetics
Complex functions
Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Taylor series and more (cont.)

Example

$$f(t) = \exp(\sin(t)) \approx 1 + t + \frac{t^2}{2} - \frac{t^4}{8} - \frac{t^5}{15} - \frac{t^6}{240} + \frac{t^7}{90} + \mathcal{O}(t^8)$$



A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Taylor series and more (cont.)

Remark

The output math expressions can displayed using the syntax of LaTeX

```
>>> from sympy import latex

>>> print latex(f.series(t,0,7))
    '1 + t + \frac{t^{2}}{2} - \frac{t^{4}}{8} -
    \frac{t^{5}}{15} - \frac{t^{6}}{240} +
    \mathcal{O}\left(t^{7}\right)'
```

$$1 + t + \frac{t^{2}}{2} - \frac{t^{4}}{8} - \frac{t^{5}}{15} - \frac{t^{6}}{240} + \mathcal{O}\left(t^{7}\right)$$

A formula
Programs and programming
Variables, reserved words
Comments, text formats and numbers

Another formula
Integer division
Objects in Python
Avoiding integer division
Arithmetic operators

Mathematical functions
Examples
Rounding errors

Interactive computing
The shell
Type conversion
IPython

Complex numbers
Complex arithmetics
Complex functions

Symbolic computing
Differentiation/integration
Equation solving
Taylor series and more

Taylor series and more (cont.)

There are also tools to expand and simplify mathematical expressions

Example

Consider the angle sum/difference identities used in trigonometry:

$$\cos(x \pm y) = \cos(x)\cos(y) \mp \sin(x)\sin(y)$$
$$\sin(x \pm y) = \sin(x)\cos(y) \pm \sin(x)\cos(y)$$

```
>>> from sympy import simplify, expand

>>> x, y = symbols('x y')

>>> f = -sin(x)*sin(y) + cos(x)*cos(y)
>>> simplify(f)
    cos(x + y)                  # Known trigonometric identity

>>> expand(sin(x + y), trig=True)   # Needs trigonometric hint
    sin(x)*cos(y) + sin(y)*cos(x)
```