Sage1-Vorlesung

```
3*5
```

```
15
```

```
5*3
```

```
15
```

```
2-4
```

```
-2
```

```
a=5*4
a
```

```
20
```

```
a=5.0/2.0
a
```

```
2.50000000000000
```

```
a="hello"
print a
```

```
hello
```

```
3*2
```

```
6
```

Wir können Ergebnisse auch als LaTeX-Befehle ausgeben:

$1/10$ wird wie in LaTeX ausgegeben als $\frac{1}{10}$:

```
show(1/10)
latex(1/10)
```

$$\frac{1}{10}$$

```
\frac{1}{10}
```

Später wollen wir $\int_0^1 \cos(x)\,dx$ berechnen

```
17 % 5
```

```
2
```

```
5^3
```

```
125
```

```
11
```

```
11
```

```
11 // 2
```

```
5
```

```
5**3
```

```
125
```

```
a=5*4
a *=5
```

```
a
```

```
'hello'
```

```
a="Hello "
b="World!"
a+b
```

```
'Hello World!'
```

```
l1=[2,3,4,6,8,[2.0,"string"]]
l1.append(10)
print l1
l1[1:4]
l1[2:]
```

```
[2, 3, 4, 6, 8, [2.00000000000000, 'string'], 10]
[4, 6, 8, [2.00000000000000, 'string'], 10]
```

```
l2 =[]
```

```
2^1000
```

```
107150860718626732094842504906000181056140481170553360744375038837
510511249361224931983788156958581275946729175531468251871452856923
043598457757469857480393456777482423098542107460506237114187795418
530464749835819412673987675591655439460770629145711964776865421676
429831652624386837205668069376
```

```
2.5
```

```
2.50000000000000
```

```
R=RealField(1000)
genau=R(5.0)/R(3.0)
ungenau1=R(5.0/3.0)
ungenau2=R(5.0)/3.0
genau, ungenau1, ungenau2
```

```
(1.666666666666666666666666666666666666666666666666666666666666666
666666666666666666666666666666666666666666666666666666666666666666
666666666666666666666666666666666666666666666666666666666666666666
666666666666666666666666666666666666666666666666666666666666666666
666666666666666666666666666667,
 1.666666666666666740681534975010436028242111206054687500000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000,
 1.66666666666667)
```

```
5.0/3.0
```

```
1.66666666666667
```

```
a=1234.2+2^100
print type(a)
print parent(a)
```

```
<type 'sage.rings.real_mpfr.RealNumber'>
Real Field with 53 bits of precision
```

```
x=6/3
y=6.0/3.0
z=5/3
w=sqrt(4)
w1=sqrt(5)
x in ZZ, y in ZZ, z in ZZ, w in ZZ, z in QQ, w1 in QQ
```

```
(True, True, False, True, True, False)
```

```
a= 12234556
print factor(a)
print is_prime(a)
print next_prime(a)
print gcd(a,1234)
list(factor(a))
```

```
2^2 * 19 * 160981
False
```

```
12234559
2
[(2, 2), (19, 1), (160981, 1)]
```

```
print sum(a.digits())
```

```
Traceback (click to the left of this block for traceback)
...
NameError: name 'a' is not defined
```

```
gcd?
```

File: /Applications/SageMath-8.0.app/Contents/Resources/sage/local/lib/python2.7/site-packages/sage/arith/misc.py

Type: <type ‘function’>

Definition: gcd(a, b=None, **kwargs)

Docstring:

The greatest common divisor of a and b, or if a is a list and b is omitted the greatest common divisor of all elements of a.

INPUT:

- `a,b` - two elements of a ring with gcd or
- `a` - a list or tuple of elements of a ring with gcd

Additional keyword arguments are passed to the respectively called methods.

OUTPUT:

The given elements are first coerced into a common parent. Then, their greatest common divisor *in that common parent* is returned.

EXAMPLES:

```
sage: GCD(97,100)
1
sage: GCD(97*10^15, 19^20*97^2)
97
sage: GCD(2/3, 4/5)
2/15
sage: GCD([2,4,6,8])
2
sage: GCD(srange(0,10000,10))  # fast  !!
10
```

Note that to take the gcd of n elements for $n \neq 2$ you must put the elements into a list by enclosing them in `[..]`. Before trac ticket #4988 the following wrongly returned 3 since the third parameter was just ignored:

```
sage: gcd(3,6,2)
Traceback (click to the left of this block for traceback)
...
```

```
Z7=Integers(7)
a=Z7(4)
3*a, a^(-1)
```

```
(5, 2)
```

```
list(Z7)
```

```
[0, 1, 2, 3, 4, 5, 6]
```

```
c=sin(pi/3)
c.n()
n(c)
```

```
0.866025403784439
```

```
z1=4+3*i
z2=1-I
z1+z2
real(z1)
imag(z2)
```

```
-1
```

```
5 in ZZ and not sqrt(8) in QQ
```

```
True
```

```
bool(sin(pi/3) ==sqrt(3)/2)
```

```
True
```

```
3*7 == 20+1
```

```
True
```

```
-5 < 2 <6
```

```
True
```

```
x=5
if x>4:
    print x, " ist größer als 4"
    if x % 2 ==0:
    print x, " ist gerade"
    else:
        print x, "ist ungerade"
elif x>3:
    print x, " ist zwischen 3 und 4"
else:
    print x, " ist kleiner als 3"
print "wird immer ausgegeben"
```

```
Traceback (click to the left of this block for traceback)
...
IndentationError: expected an indented block
```

```
x=11
while x>3:
    x=x-3
print x
```

```
2
```

```
for i in [3..6]:
    print i^2
```

```
9
16
25
36
```

```
[0,5..20]
```

```
[0, 5, 10, 15, 20]
```

```
def plus(a,b=3):
    return a+b
```

```
plus(8,-5)
```

```
3
```

```
plus(6)
```

```
9
```

```
plus(6,7)
```

```
13
```

```
def diff(a,b):
    return a-b
```

```
diff(10,4)
```

```
6
```

```
diff(b=4,a=10)
```

```
6
```