

Exercises: Week 1

Computation in Algebra and Arithmetic

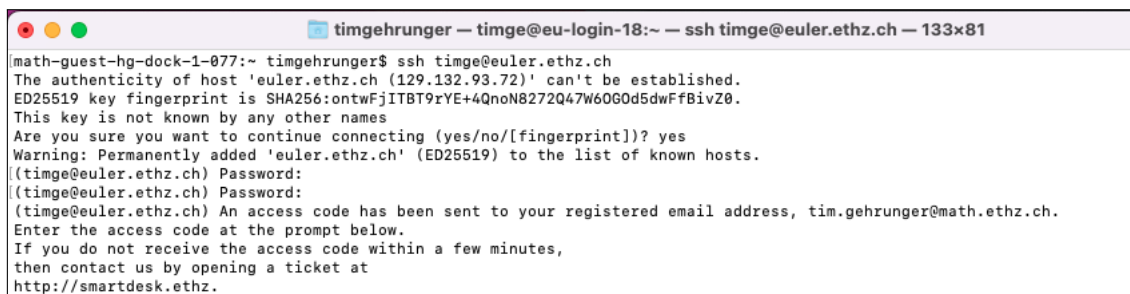
David Loeffler

25/2/2022

1 Starting Sage

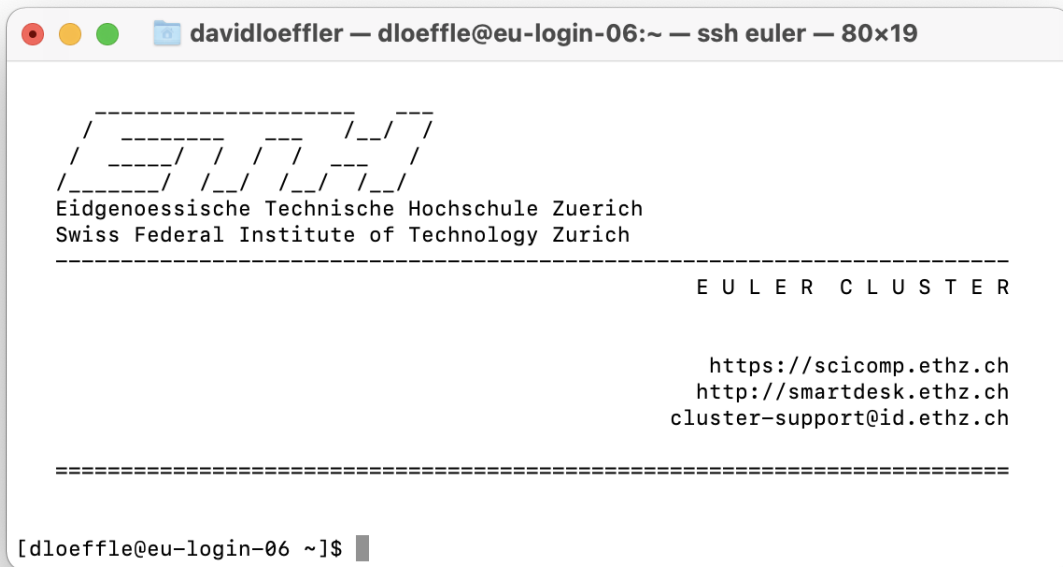
Method 1: Using the ETH Euler cluster

- Open an SSH connection to `euler.ethz.ch`, following the instructions at https://scicomp.ethz.ch/wiki/Getting_started_with_clusters#SSH. If you are using the Euler cluster the first time and are using MacOS or Linux, it will look like this:



```
timgehrunger — timge@eu-login-18:~ — ssh timge@euler.ethz.ch — 133x81
math-guest-hg-dock-1-077:~ timgehrunger$ ssh timge@euler.ethz.ch
The authenticity of host 'euler.ethz.ch (129.132.93.72)' can't be established.
ED25519 key fingerprint is SHA256:ontwFjITBT9rYE+4QnoN8272Q47W6OGod5dwFfBivZ0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'euler.ethz.ch' (ED25519) to the list of known hosts.
(timge@euler.ethz.ch) Password:
(timge@euler.ethz.ch) Password:
(timge@euler.ethz.ch) An access code has been sent to your registered email address, tim.gehrunger@math.ethz.ch.
Enter the access code at the prompt below.
If you do not receive the access code within a few minutes,
then contact us by opening a ticket at
http://smartdesk.ethz.
```

- It'll send an activation code to your ETH email. Type it in at the prompt. Afterwards, you need to type 'yes' in order to agree to the rules and policies of the Euler cluster.
- This should get you to the terminal prompt, which looks like this:

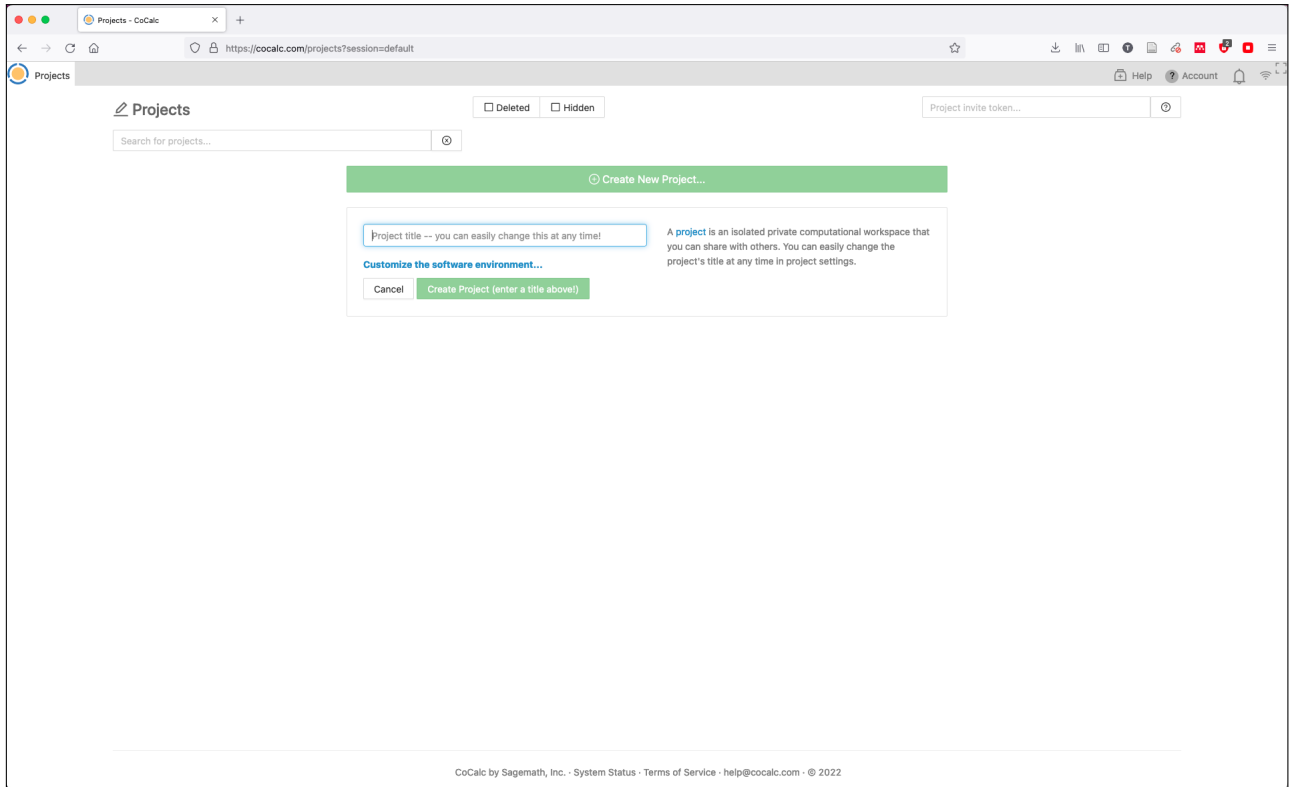


- Type the following arcane runes:

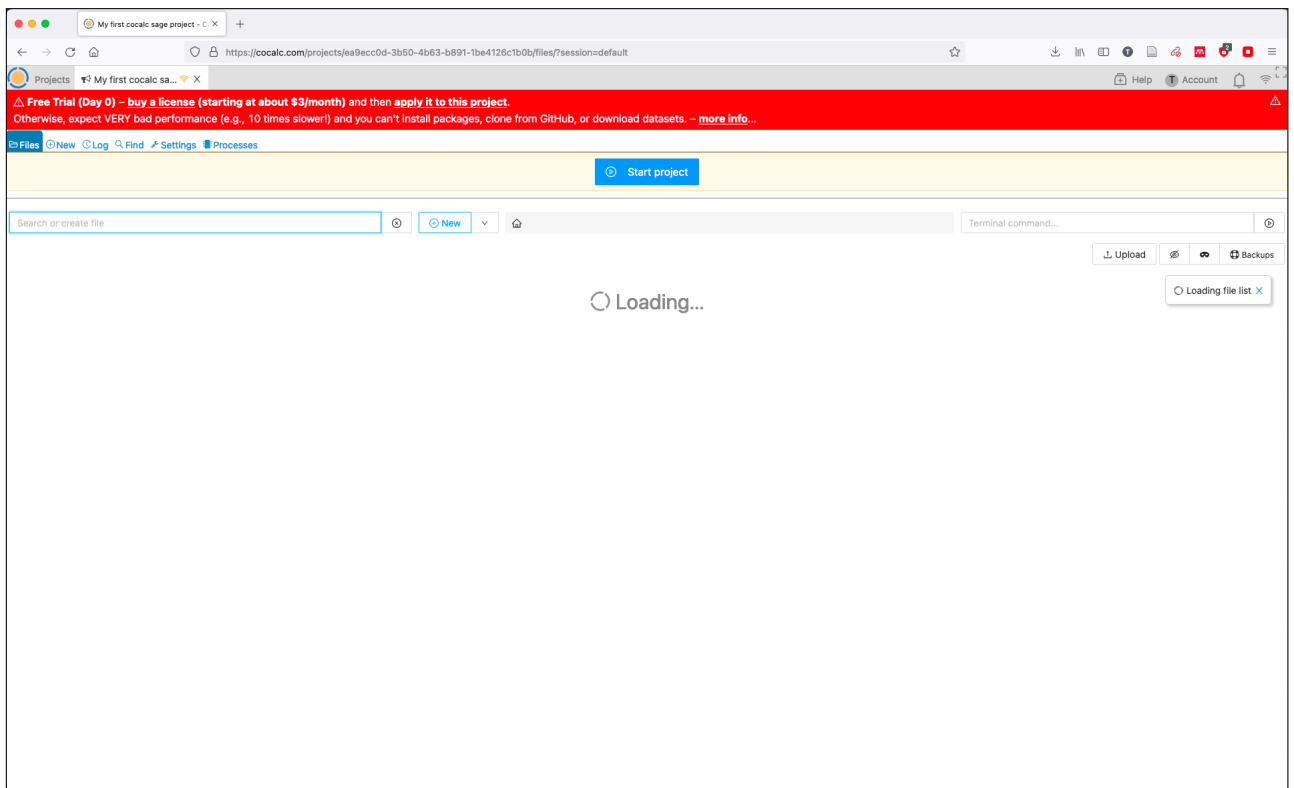
```
module load gcc/6.3.0 sage
sage
```

Method 2: Using CoCalc

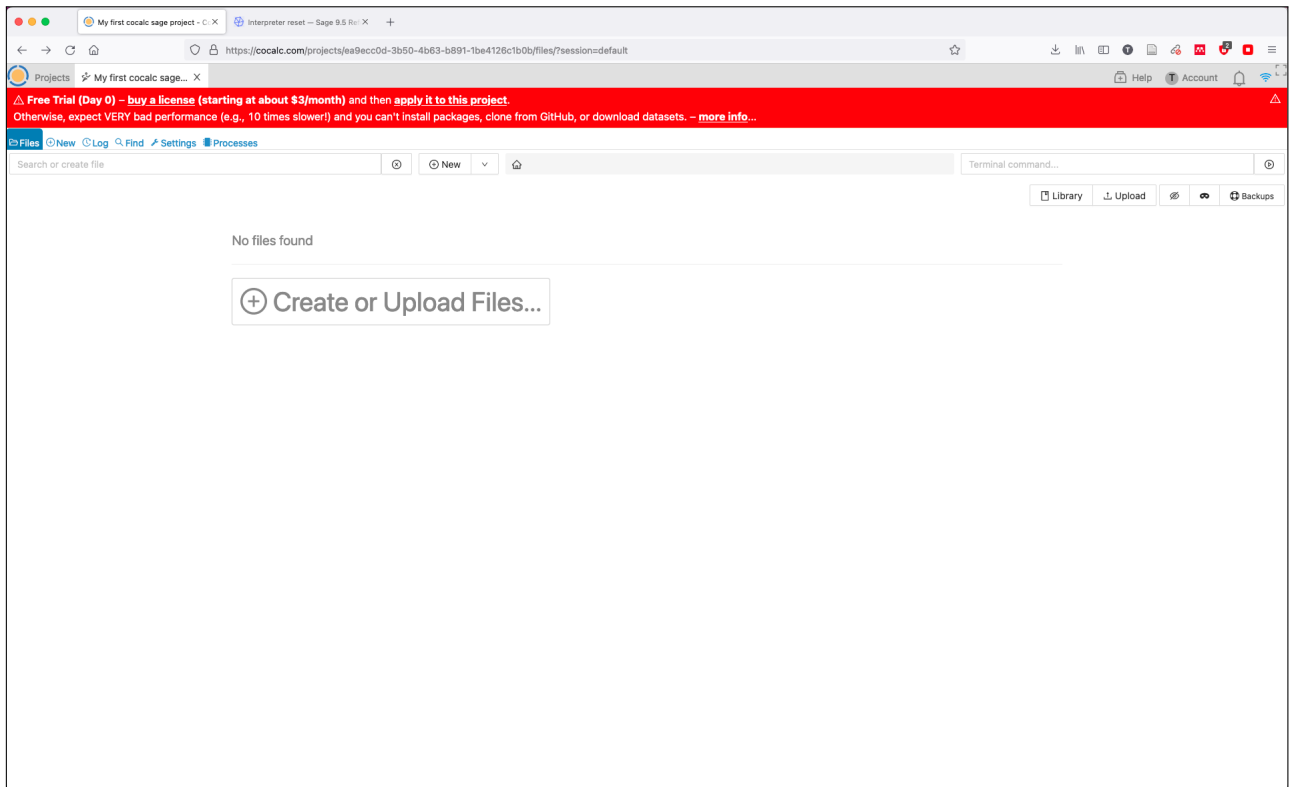
- Open a web browser and go to <https://cocalc.com>.
- Click "Sign Up" to create a user account.
- Once you've made an account and signed in, you are greeted with the following screen:



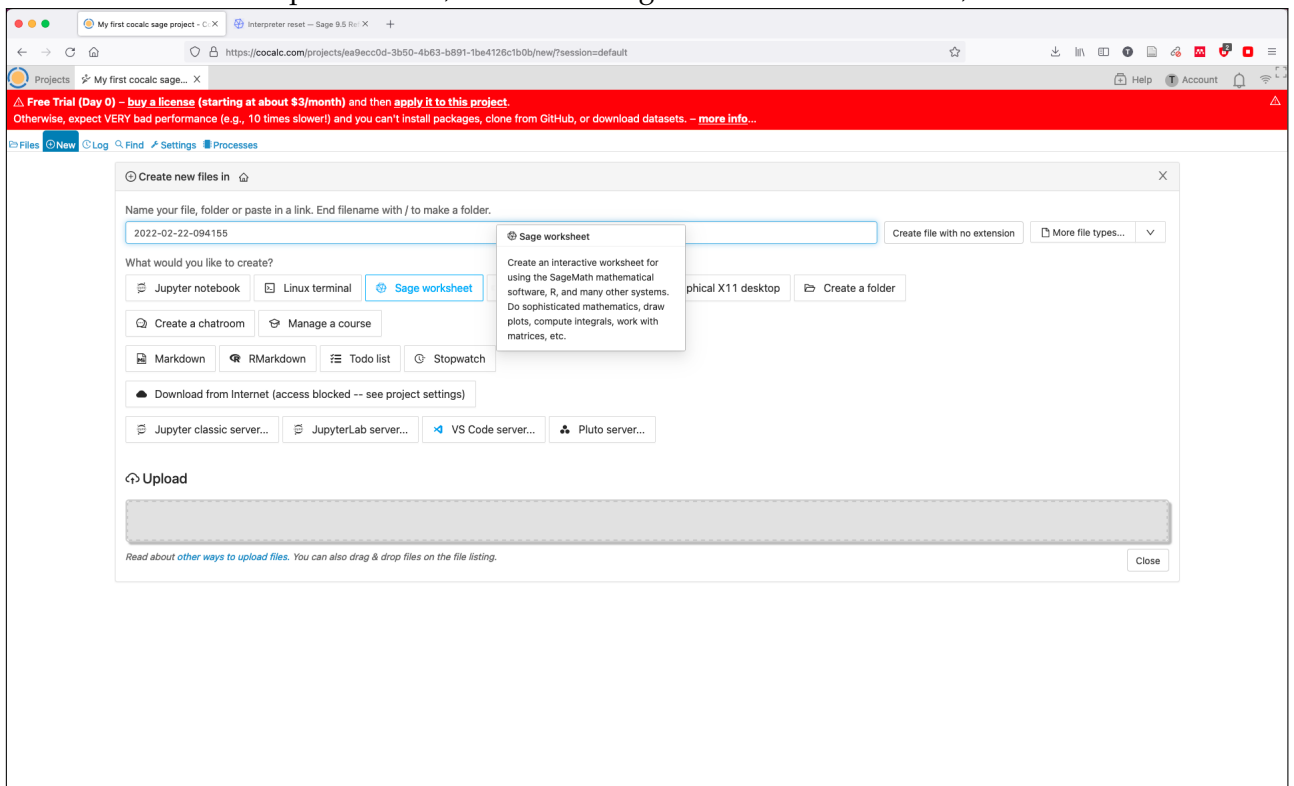
- Click “Create New Project” (a “project” is a space to put your files in), and give your project a name, maybe “My Project”. Your screen should now look like this:



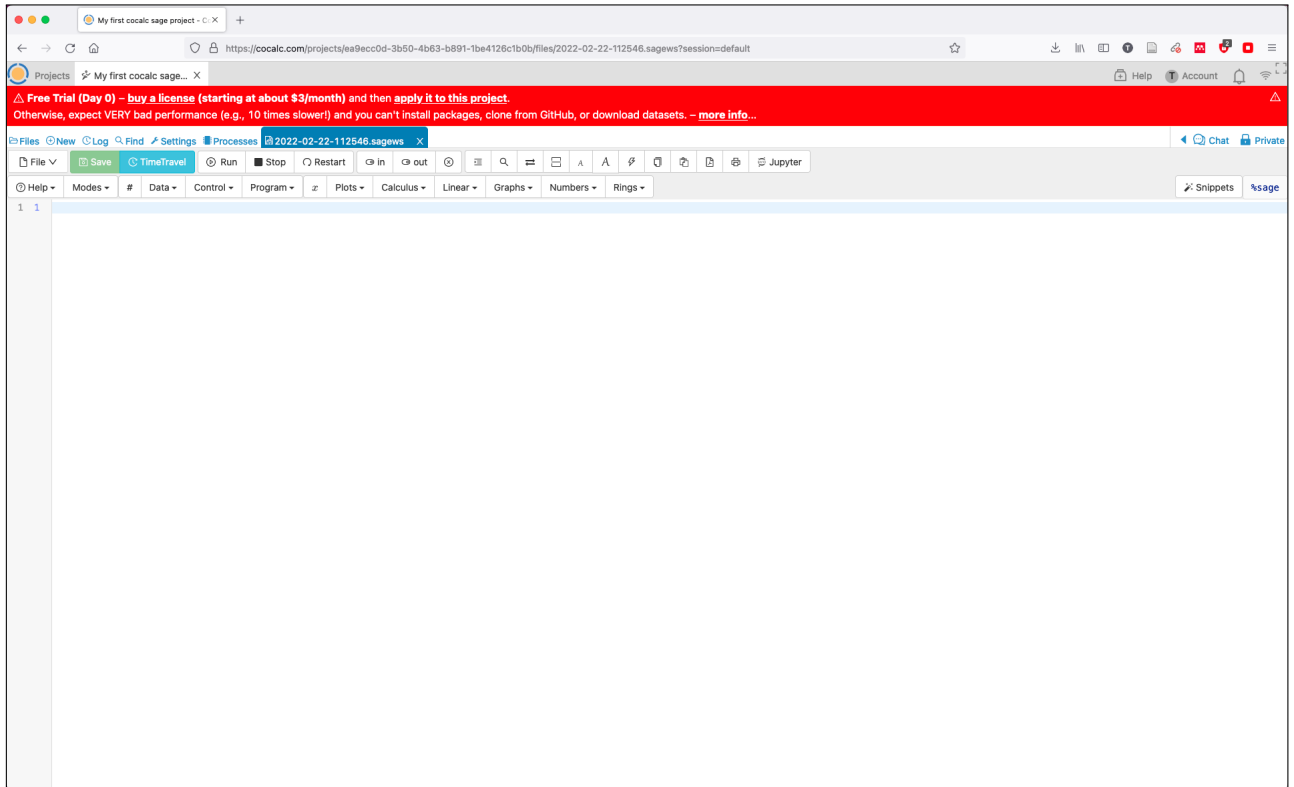
- Click on “Start project”. After some time, your screen will look like this:



- Click on “Create Or Upload Files”, and select “Sage Worksheet” in the box, see:



- Now wait a bit until Sage starts up (free CoCalc accounts are very slow to start, although OK once they get going). Your screen should now look like this:




2 Basic Arithmetic

- Type:
 $2 + 2$
 then hit the return key (if you're using Euler) or shift+return (on CoCalc). It should, in a moment, respond with 4.
- The underscore sign “_” allows you to get at the result of the last computation. So
 $_ + 3$
 should give you 7.
- Play around a bit with integer arithmetic.
- Give a name to a variable:
 $a = 7$
 $2*a + 3$
- Try some arithmetic with fractions:
 $3/5 + 5/3$
 You'll notice it returns an exact fraction (not a decimal approximation). If you want a decimal, type $n(_)$ or even $n(_, digits=100)$.

- Very big numbers:
factorial(1000)
 - Primality testing:
n = 2⁶⁰⁷ - 1; n (the last bit means “show the value of n”; it’s pretty big!)
is_prime(n)
time is_prime(n) (“time” tells it to show how long the computation took)
- Compare this with time is_pseudoprime(n), which tells you the same thing, but more quickly and less reliably.

3 Polynomials

- Type the following command:
R.<x> = QQ[]
This means “R is the polynomial ring over the rationals in a variable called x”. So now you can do things like (x + 1)¹⁷ and you’ll get a sensible answer.
- It can do basic arithmetic with polynomials:
(x³ + 3*x² + 3*x + 1) / (x + 1)
- If f is a polynomial and n is an integer, then f[n] is the coefficient of xⁿ:
f = (x + 1)¹⁰; f[5] (returns 252 = $\binom{10}{5}$)
- Now try S.<y> = Zmod(3) []
(y + 1)¹⁷
This tells Sage that y is a polynomial variable *over* $\mathbf{Z}/3\mathbf{Z}$, so it reduces all the coefficients mod 3.
- You’ll notice that typing x + y, after the above two declarations, will give you an error (as it should, since the sum makes no sense).

 If you just type (x + 1)¹⁷ without making any R.<x> declaration first, it’ll work, but it defines x with a subtly different data-type (as a “symbolic expression”, not a polynomial), which means it won’t automatically expand out. Symbolic expressions are a bit troublesome for various reasons, and also it only works for x, but not any other variable name. It’s best to declare your variables!

4 Tab-completion and help

Try typing a and then the Tab key to see a list of all commands beginning with a, which you can navigate through with the arrow keys:

```
sage: a
      a                absolute_igusa_invariants_wamelen
      abs              abstract_method
      abs_symbolic     acos
      absolute_igusa_invariants_kohel  acosh
```

If you choose one, and then put a question-mark after it, then it'll tell you what the function does. (If you really want, you can even put *two* question-marks and Sage will peek inside itself and show you the Python code for that function – probably only useful for experts!)

That's not all. Sage has two kinds of commands, *functions* and *methods*. The difference is that methods belong to a specific object and allow you to ask that object to do something. So for example `is_prime(3)` is calling a function, but `3.divides(17)` is calling a method belonging to the integer 3: you ask 3, "do you divide 17?" and it says "False", because it doesn't.

Many basic commands are implemented in both ways – as functions and as methods – but more advanced ones tend to be one or the other. Methods of objects *don't* show up in the initial tab-completion list (which only includes functions); but if you type the name of an object and a dot, then hit Tab, you'll see a list of its methods:

```
R.<x> = QQ[]; f = x^2 - 2
f.      [hit Tab to see a list of methods]
f.resultant?  [hit Enter to see the help for the "resultant" method]
f.resultant(x^2 - 5)    (call the command)
```

(Remark: For f, g monic polynomials over a field K , the **resultant** $\text{Res}(f, g)$ is defined as follows: if we write $f(x) = \prod_i (x - \lambda_i)$, $g(x) = \prod_j (x - \mu_j)$ over \bar{K} , then $\text{Res}(f, g) = \prod_{i,j} (\lambda_i - \mu_j)$. One can show (exercise!)¹ that $\text{Res}(f, g) \in K$. So the above is telling us that

$$(\sqrt{2} - \sqrt{5})(-\sqrt{2} - \sqrt{5})(\sqrt{2} + \sqrt{5})(\sqrt{2} - \sqrt{5}) = 9. \quad)$$

5 Pretty pictures

(This will only work on CoCalc, not on euler, unless you do a bit of fiddly extra setup.)

- Try the following:

```
R.<x, y> = QQ[]
T = 144*(x^4 + y^4) - 225*(x^2 + y^2) + 350*x^2*y^2 + 81
implicit_plot(T, xrange=[-1.5, 1.5], yrange=[-1.5, 1.5],
  title="Trott's quartic curve")
```

(NB: You don't have to type it all, you can copy + paste it from these notes.)

- It even works in 3D:

```
R.<x, y, z> = QQ[]
lemon = (x^2 + z^2 - 3*(y^3*(1-y)^3))
implicit_plot3d(lemon, xrange = [-1, 1], yrange=[0,1],
  zrange=[-1,1], color="yellow")
```

¹This follows from Galois theory if K has characteristic 0, or more generally if K is perfect (i.e. $\bar{K} = K^{\text{sep}}$). The case of non-perfect fields is a little more delicate.

6 Polynomial root isolation

For a polynomial over the rationals, the command `real_root_intervals` returns rational intervals containing the roots:

```
sage: R.<x> = QQ[]; f = x^2 - 2
sage: f.real_root_intervals()
[((-3/2, -1), 1), ((1, 3/2), 1)]
```

The output needs a bit of unpacking. It means “a root in $(-\frac{3}{2}, -1)$ with multiplicity 1, a root in $(1, \frac{3}{2})$ with multiplicity 1, and no others”. It’s returned as a list² structure, whose entries are themselves lists. You can get the n -th entry in the list with `[n]` (but note Sage starts counting at 0, not 1):

```
sage: L = f.real_root_intervals()
sage: L[0]
((-3/2, -1), 1)
sage: L[0][0]
(-3/2, -1)
sage: L[0][0][1]
-1
```

7 Algebraic numbers

Here’s an example illustrating `QQbar`, Sage’s implementation of (complex) algebraic numbers.

(NB: In the examples, anything after a hashtag “#” is a comment – Sage ignores it, I’ve put it there to explain to you what’s going on.)

```
sage: R.<x> = QQ[]; f = x^3 + x + 1
sage: a,b,c = f.roots(QQbar, multiplicities=False)
sage: a
-0.6823278038280193?
sage: a.parent() # what is 'a' an element of?
Algebraic Field
sage: b
0.3411639019140097? - 1.161541399997252?*I
sage: c
0.3411639019140097? + 1.161541399997252?*I
sage: a + b + c == 0 # checks (rigorously!) if a + b + c is zero
True
sage: g = (a - b).minpoly(); g # minimal polynomial satisfied by a - b over QQ
```

²Python, the language Sage uses, has lists (with square brackets) and tuples (with round brackets). In practice the difference is rarely relevant.


```

x^6 + 6*x^4 + 9*x^2 + 31
sage: g == (x^2 - (a-c)^2)*(x^2 - (a-b)^2)*(x^2 - (b-c)^2)
True

```

8 Some additional exercises

- (1) Show that $2022!$ has an odd number of digits (in base 10), and compute the middle digit.
- (2) Compute the coefficient of x^{40} in $(1 + x + x^2)^{40}$.
- (3) Prove (directly, without using Sage's root-isolation commands) that the polynomial

$$f(x) = x^5 + 2x^4 - x^3 - 1$$

has a unique root in the interval $[\frac{4}{5}, 1]$. Does it have any roots in $[1, \infty)$?

- (4) Let f, g be two monic polynomials over \mathbf{Q} (with distinct roots), and let $S(f), S(g)$ be the set of roots of f, g respectively. Suppose $\alpha \in S(f), \beta \in S(g)$, and $u, v \in \mathbf{Q}$. If we have

$$|u - \alpha| \leq |u - \alpha'| \quad \forall \alpha' \in S(f)$$

and similarly

$$|v - \beta| \leq |v - \beta'| \quad \forall \beta' \in S(g),$$

does it follow that

$$|uv - \alpha\beta| \leq |uv - \alpha'\beta'| \quad \forall (\alpha', \beta') \in S(f) \times S(g)?$$

Give a proof or counterexample as appropriate.