Random Magma knowledge

This is a compilation of some useful Magma tricks that I have learned through the years. Thank you to all the wonderful people who have shared their knowledge with me: Eran Assaf, Edgar Costa, Sachi Hashimoto, Avi Kulkarni, Alex McCleary, Sam Schiavone, Pim Spelier, Allan Steel, Drew Sutherland, John Voight, David Zureick-Brown.

- Useful links: Documentation and general examples.
- You can write the first letter of a function and tab complete twice to get a list of possible functions.
- To kill a process, use `control` + `c`. To kill Magma, do this twice.
- To ignore `>`, use `SetIgnorePrompt(true);`. This is very helpful when you are copying code from the terminal.
- `$1` denotes the last printed result (you can also call `$2` and `$3`).
- The rational numbers are not a number field in Magma. You can instead call `RationalsAsNumberField();`. Careful: linear algebra is slower here.
- When you are debugging functions, you can `SetDebugOnError(true);`. This will give you access to the "inside" of your function, up to where Magma got stuck. You print things by `p whateverYouWant`. Use `q` to quit back to the Magma terminal. You can see all the functions that were used in your computation using `bt` (shows you the "frames"). Go to a specific frame by writing `f theNumberYouWant`. Warning: do not use `;`.
- In your home directory you can create a `file.Magmarc`, if does not exist, to have certain commands to run on start. To find the file, you can go to your home directory and press Command + Shift + . (period).
- Use `control` + `e` to get to the last character of a line in the terminal. Use `control` + `a` for the first one. Do `control` + `k` to delete the line. You can also find other combinations here.
- Do `%p` to print all the Magma session.
- You can search only for signatures without inheritance: `ListSignatures(ModFrmHilElt : Isa := false);`. Moreover, you can also just look for functions where your type is an argument or a return values (very useful when you try to find a function producing the type that another function needs…): `ListSignatures(ModFrmHilElt : Search := "ReturnValues", Isa := false);`.
- Magma complains about types. Elements need to be coerced to have the type you want. For example, to coerce and integer or type Rational to type Integer: `Integers()!r`.

- The function `Discriminant(K)` returns the discriminant of the polynomial used to define, not the discriminant of K. Use `Discriminant(RingOfIntegers(K))` to get the discriminant of K.
- Use for verbose: `vprintf VerboseLevel : "whatever you want to print";`
- To run a magma Jupyter kernel, go to this GitHub repository, and then you can run `sage -n jupyter` .
- To print in a Magma-readable format, do: `Sprint(whateverYouWant, "Magma");`.
- To coerce a list into the same set: `[Integers() | 4/1, 5/1]` .
- To iterate over a list, you can use `& + (operation)`. For example, to add all the prime numbers ≤ 20, do:

  ```
  &+[p : p in [1..20] | IsPrime(p)];
  ```

  Pro-tip: Magma does not like iterating on empty lists unless you coerce it somewhere

  ```
  &+[Integers()| p : p in [0,1] | IsPrime(p)];
  ```

- In a for loop, you can grab the index! For example, `for i -> s in ["a","b"] do`.
- To load a Magma file line by line do `iload "NameOfFile.m";`.
- To run computations in a server and be able to go back even when you disconnect, you can use a screen. For commands see this link.
- To set up ssh without a password, use something like this link.
- Careful with the order of matrix multiplication! `Eigenspace(M,3);` computes the eigenspace of the action on the right. Consider taking transposes :).
- Magma requires elements of sequences `[ ]` to have the same universe. If you want a list that takes anything, you can try `[* *]`.
- You can coherce a one variable polynomial living in a polynomial ring with more variables by using `UnivariatePolynomial()`.
- To see the last 20 lines: `%P`.
- To keep magma from breaking up lines (and in particular, from adding "extra characters" when breaking up lines, which is annoying if you write output to file).

  ```
  SetColumns(0); SetAutoColumns(false);
  ```

- If you want to upgrade form lists, you can use `AssociativeArray(indexUniverse)`. To assign a value: `yourArray[index] := whatever`. To see which indexes have something assigned, try `Keys(yourArray)`.

- To delete a variable that you have defined: `delete nameOfVariable;`. This may be useless, but I like it!

Do you have more random knowledge? Please email it to me and I will add it to this list!