MACAULAY2 - RTG SEMINAR

DECEMBER 4TH, 2023

We'll work on documentation today. You may use `TempPackage2.m2` from the website, or your group's own package. If you have at least two functions, one of which has options your own package should work fine for today's purpose.

If you want to use essentially a blank package, grab this one.

`https://www.math.utah.edu/~schwede/M2RTG/TempPackage2.m2`

Warning: Macaulay2 documention is super finnicky about how many spaces you are tabbing in. I recommend you use spaces, and not tabs (google how to do this for your editor) and also display spaces/whitespace characters (again, google how to do this for your editor).

Step 1 (Installing documentation)

One of the more important things to do is to write documentation for your users. In the sample file, `firstFunction` is documented.

I generally recommend running

```
loadPackage("TempPackage2", Reload=>true)
```

or

```
needsPackage("TempPackage2")
```

before installing the documentation.

You can install the documentation (and also check for missing documentation) by running the command

```
installPackage "TempPackage2"
```

After installing it, you may want to run `uninstallPackage "TempPackage2"`.

Installing the package will run all the examples, store the output, and make nicer linked documentation pages. You may need to restart Macaulay2 before it works. (For example, before/after uninstalling the package and restarting M2, run `help firstFunction` or `viewHelp firstFunction`).

If you are not using `TempPackage2`, go modify your package so it has an entry for the package itself and at least one function (remove the dots signifying an ellipsis).

```
doc ///
    Key
        TempPackage2
    ...
///

doc ///
    Key
        firstFunction
    ...
///
```

2 (Documenting options)

Your functions will have options, so you will need to document them. Frequently the option help key, `[secondFunction, Strategy]`, is simply inserted below the below `secondFunction` key in the same doc (and the various ways to call `secondFunction`). This is what `TempPackage2` does. You can also create a whole separate help page for the options via something like:

```
doc ///
    Key
        [secondFunction, Strategy]
    ...
```

Regardless though, you want to to show people how to use them. For example, since I combined my option documentation with the main key doc:

```
    Usage
        l = secondFunction(r, Strategy=>m)
    Inputs
        r:QQ
        a rational number
    Strategy=>Thing
        a valid value for Strategy
```

Of course, you also need to write more detailed text lower down in the `Text` describing the option more completely.

Exercise: Document another option in your package. Add some text and even an `Example` (see `TempPackage2` for examples of `Example`s).

3 (Hyperlinks)

There are various hyperlinks you might want.

- Inserting a link to your own documentation.

  ```
  @TO firstFunction@
  ```

  if inserted in a `Text` block in your documentation (replace `firstFunction` with your own function name if applicable).
- Inserting a link to Macaulay2's documentation.

  ```
  @TO2 {"Macaulay2Doc :: primaryDecomposition", "primaryDecomposition link text"}@
  @TO2 {"Macaulay2Doc :: primaryDecomposition(Ideal)", "link text"}@
  ```

- Inserting a link to the web.

  ```
  @HREF "https://www.github.com"@
  ```

- Inserting a link to wikipedia.

  ```
  @wikipedia "normal ring"@
  ```

Exercise: Take the documentation you are working with. Add various hyperlinks in the `Text`. Install the documentation and make sure they all work!

4 (Formatting text) To format text, try stuff like:

```
@TT "hello"@
@BOLD "world"@
@EM "are you there?"@
```

Notice you have various headings like **Acknowledgements** and **See Also** in various help files? You can make your own via the command:

```
Text
    @SUBSECTION "A new section name"@

    some relevant text
```

The spacing and indenting are important here (`Text` is normally indented one from `Description`).

Exercise. Make your help files more readable by playing around with some of these features.

5 (Lists)

Ok, but what if you want lists in your text. Into some `Text` block, insert something like:

```
Description
    Text
        some text
    Tree
        :Heading name
        "firstFunction"
            :an silly function
        "secondFunction"
            :a better function
            :still under development
```

Remember, tabbing is important. Compare the output to this one:

```
Description
    Text
        Some text
    Tree
        :Heading name
            "firstFunction"
                :an silly function
            "secondFunction"
                :a better function
```

or this one:

```
Description
    Text
        Some text
    Tree
        :a heading
            :a subheading describing things
        :a second heading
```

Exercise: Go add some lists to your documentation!