Sage and Xronos

How sage interacts with Xronos.

How to Write Sage in LaTeX

There are two phases to writing sage code using sagetex. The first phase is writing the actual code, which generates all the content you want to utilize, and the second phase is calling parts of the code from the first phase to display or use.

Phase One

The first phase should almost certainly be written in the "sagesilent" block, which is where you write pure sage code that contains everything from generating the function you want to display, to the answer you want the student to enter.

For example, we would write something like the following:

```
\begin{sagesilent}
p1f1 = 3*x+1
p1ans = 10
\end{sagesilent}
```

This generates the sage variable named `p1f1` and `p1ans`. This may seem like an odd answer scheme but there is a good reason for it (see the "potential pitfalls section below". There is also a "sageblock" environment, but this tries to display the running sage code as if it was being done in a sagecell prompt as you go, which is almost always a bad idea.

Phase Two

The second phase is when you want to display the results of something from the sage code you executed to define all the variables/functions/etc you needed to populate the problem. In the current Xronos system, the only sage command that loads sage content for the student to see, is the `\sage` command. Most importantly, the other typical sage commands, like `\sagestr` and `\sageplot` do not work at all, and should be avoided. So you will want to only use the

Learning outcomes:

`\sage` command to call content, even though other commands may work locally - they won't work on a Xronos page[1].

Example

Consider the following problem:

Problem 1 *Let* $f(x) = 3\,x + 1$. *Then* $f(4) =$ $\boxed{13}$

The above problem is generated with the following code:

```
\begin{sagesilent}
p2f1 = 3*x+1
p2ans = p2f1(x=4)
\end{sagesilent}
\begin{problem}
Let $f(x) = \sage{p2f1}$. Then $f(4) = \answer{\sage{p2ans}}$
\end{problem}
```

The composition bit may look a little odd, but you can see why this is necessary in the "Pitfalls and Problems" section below.

Potential Pitfalls and Problems

Composition is a little weird

When doing composition in sage, it (now) requires you to use "x = ()" style notation. For example, if you use something like:

```
\begin{sagesilent}
## Function for first problem:
p3c1 = 4
p3c2 = 5
p3f1 = const1*x+const2
# Answer is just f(3)
p3ans = p3f1(3)
\end{sagesilent}
```

Sage will error and you will get a "spinning wheel of death" that kills everything after it on the page. Instead, you need to write:

[1] at least currently, but we are trying to fix that for the next iteration of the platform

```
\begin{sagesilent}
## Function for first problem:
p3c1 = 4
p3c2 = 5
p3f1 = const1*x+const2
# Answer is just f(3)
p3ans = p3f1(x=3)# Could also use p3f1(x=(3))
\end{sagesilent}
```

If the substitution is just one variable, you can use "x=var" but if you need something more complex, like substituting in $3x + 1$ you would need to use extra parentheses, like `p3f1( x=(3x+1) )`.

Xronos processes all the Sage code fully before being populated

Unlike when you compile locally (for most editors) all of the sage code is processed independently before it is populated into the page in Xronos. This means that if you accidentally use the same variable for two different problems, the second definition of that variable will overwrite the value for that variable for *both* problems. In other words, if you had something like:

```
\begin{sagesilent}
## Function for first problem:
const1 = 4
const2 = 5
func1 = const1*x+const2

## Function for second problem:
const1 = pi
func2 = sin(x=(x - const1))

\end{sagesilent}
```

Then the first function will come out as $\pi \cdot x + 5$ not $4x + 5$ because the `const1` sage variable was overwritten in the second problem's code and overwrote it for *every* problem where that variable is used - including the first problem. Annoyingly, you may compile it locally, and depending on your editor, it may have shown the first function as $4x + 5$ leading to more confusion.

Strings are tricky to get Xronos to accept...

Since sagetex usually uses the `\sagestr` command to generate text, and since the `\sage` command requires mathemode, it makes strings a little... interesting.

In particular, trying to actually display strings, you get some weird latex code. Consider the following code:

```
\begin{sagesilent}
p4str = 'this is a string'
p4ans = 'yes'
\end{sagesilent}
\begin{problem}
Is the following text?
\[
\sage{p4str}
\]
Type ``yes'' or ``no'' (without the quotes!)
$\answer{\sage{p4ans}}$
\end{problem}
```

You get:

Problem 2 *Is the following text?*

$$\texttt{this is a string}$$

Type "yes" or "no" (without the quotes!) `yes`

The text shows up correctly as a string of text, but it is surrounded by brackets and the LaTeX command. Moreover, if you type in yes (or even 'yes') it won't work. Indeed, if you were to check the code to see what answer it was expecting, you would discover that it is expecting `\textit{yes}` - just like the displayed string.

But, there is a way to get a string to display correctly, both in actual text and in the validator - by using the "LatexExpr(r'string')" command. So if we make some minor modifications to our previous code:

```
\begin{sagesilent}
p5str = LatexExpr(r'\text{this is a string}')
p5ans = LatexExpr(r'yes')
\end{sagesilent}
\begin{problem}
Is the following text?
\[
\sage{p5str}
\]
Type ``yes'' or ``no'' (without the quotes!)
$\answer{\sage{p5ans}}$
\end{problem}
```

Then we get:

Problem 3 *Is the following text?*

this is a string

Type "yes" or "no" (without the quotes!) *yes*

Which both displays the problem, and accepts the answer, correctly. Notice that we need to use the `\text` command when we want to display text correctly to the student - otherwise mathmode will display the provided text as if it were in mathmode (i.e. without any spaces, and as if all letters were variables). Also note that we didn't use "format=string" in the answer command - indeed, doing so makes the validation fail because of how LaTeX handles fonts in mathmode (this is deep LaTeX magic, so you probably don't want details - suffice it to say it isn't something that can be easily fixed or worked around with the current system). This means that the validator is really thinking that the 'y', 'e', and 's' are variables, so any permutation of those variables would be accepted. This doesn't typically impact student answers however, but it might be worth remembering just in case.