InfPALS LaTeX Activity 2

Time to Shine

Code and Mathematics in LaTeX

> I think the bubble sort would be the wrong way to go.
>
> *Barack Obama*

1 Introduction

So far anything you have done you could have done with a regular word processor, just not as pretty as with LaTeX. However, this activity will guide you through two areas in which LaTeX hands down beats the competition: typesetting code and mathematics. Especially the latter can de facto only be done efficiently in LaTeX.

2 Typesetting Mathematics

This topic is so vast that stepping you through all the things you could potentially do would take far too long. Instead, we'll provide a basis and a couple of examples to replicate to give you a bit of practice with a common subset of what mathematics in LaTeX has to offer you.

For any of this to work we need to import the following two packages: `amsmath` and `amssymb`. Not very surprisingly you can't just typeset mathematics in text like this. You need a special environment and because it is so common it is made to be very quick to type:

```
\[
\]
```

Anything you write within the above two lines is considered to be in the so-called *mathmode*. Try inserting `\frac{1}{2}` and compile. Should you ever want to avoid interrupting the text-flow and put in some symbols within a sentence like $\frac{1}{2}$, then put them between two `$`, so for instance `$\frac{1}{2}$`. This is also considered to be in mathmode.

Before you start the next task there's one last treat for you. At one point or another you'll come across the situation where you know what the symbol you want to type looks like, but have no idea how to look for it in order to find out how to type it in LaTeX. Well, you're not the first one and that's exactly why we have Detexify. Simply draw the symbol and it'll tell you exactly how to replicate it in LaTeX.

At the end of this activity you'll get to do some more exploring. With the Wikibooks reference manual try to replicate the following equations:

$$\sum_{n=1}^{\infty} n = -\frac{1}{12} \qquad \int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi} \qquad \lim_{x \to 0} \frac{\sin x}{x} = 0$$

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} \qquad R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \qquad S = -k_B \sum_i p_i \log p_i$$

3 Typesetting Code

We'll start off very lightly with probably the simplest way to include code in your documents, the `verbatim` environment:

```
\begin{verbatim}
\end{verbatim}
```

Anything within these two lines will be put into the document as-is (even stuff like curly braces). To keep you from having to come up with some code you should have been provided with a file called `best_sort.py`. Open it and take some code from there to explore the effects of the above command. In case you want to embed code directly into a sentence like `this`, you may find the following useful:

```
\verb|inline code here|
```

Notice the vertical bars instead of the typical curly braces.

Now, this is helpful in distinguishing code from regular text, but we can do better. Much better. Go ahead and import the module `listings`. Put your code form above within the following two lines:

```
\begin{lstlisting}
\end{lstlisting}
```

If it seems like we didn't quite deliver on our promise it's because we're not quite done yet. The real power of the `listings` package lies within defining styles for your code. For instance, would you like line numbers? Nothing easier than that. Add the following to your preamble:

```
\lstset{numbers=left}
```

Within the curly braces of `\lstset` you can define all sorts of things. However, before you click the link in the previous sentence and go off experimenting, let's look at two extremely useful features of `listings` first. One is automatic code highlighting. If you have a piece of Python code and you want keywords, comments, etc. to be highlighted neatly simply tell LaTeX which language it's dealing with:

```
\begin{lstlisting}[language=Python]
\end{lstlisting}
```

Also, you didn't actually need to copy the code from the file, you could have included it directly (remember to upload the file) by doing the following:

```
\lstinputlisting{best_sort.py}
```

Now see if you can make your included code look a little something like the following:

```
def bubble_sort(l: list):
    """ Sort list l in-place super fast. """
    for pass_num in range(len(l)-1,0,-1): # Move back to front.
        for i in range(pass_num):
            if l[i] > l[i+1]:
                temp = l[i]
                l[i] = l[i+1]
                l[i+1] = temp
```

Hint: There is a `firstline` and `lastline` option for `\lstinputlisting`.