Welcome to Math 124!

Lewis Pan - math.berkeley.edu/~yllpan

Office hour poll

Plan: Your homework is due on Wednesdays 8am, projects due on Friday midnight so

1. One OH on Monday/Tuesday, before hw due
2. One OH on Thursday/Friday before project is due

My time constraints:

- Cannot do 12-1 any day
- Cannot do Tuesday/Thursday 8-9:30, 2-3:30

Office hours will be:

- 9-10:30 Monday
- 2-3 Friday

Logistics for Section

I am flexible on this - I am happy to go over any material people are confused by, give summaries of material, or even give you a headstart on material to come if you feel comfortable with the stuff already covered.

On quizzes, I plan to do this at the end of each section. There will be 6 quizzes, each 15 minutes. Your lowest score will be dropped. If you have DSP arrangements please come and talk to me. The quizzes will be in person but I will upload them after on gradescope.

Unfortunately section is on Wednesday **AFTER** your homework is due. If you have hw questions office hours is probably a better idea, although I would be happy to go over homework solutions in section also.

Julia installation

Julia website: https://julialang.org/downloads/ (https://julialang.org/downloads/)

IJulia: https://julialang.github.io/IJulia.jl/stable/manual/installation/ (https://julialang.github.io/IJulia.jl/stable/manual/installation/)

0. Download julia
1. Open julia in terminal, or whatever environment is native to your OS
2. type in `using Pkg`
3. type in `Pkg.add("IJulia")`

This should install IJulia (only needs to be done once). To open a notebook:

1. `using IJulia`
2. `notebook()`

Alternatively you can just stick to using datahub. The official version the class is using is 1.6.3 - although if you use an older version try to stick to one after 1.2 as the functionality before that is slightly different. I'm on 1.6.2 and everything works fine.

Demo time - LaTeX (pronounced latech)

There are two ways to write an equation,

```
\begin{equation}
    ...
\end{equation}
```

```
$$
    ...
$$
```

$$1 + 1 = 2$$

$$1 + 1 = 2$$

Here is a list of common things:

1. Commands predicated w/ backslash \
2. Subscript, superscript is done with _ ^
3. For example integral sign is \int
4. Summation sign is given by \sum
5. Infinity is \infty
6. Fraction \frac{}{}
7. Power is also ^
8. Greek letters, backslash + spell it out e.g. \pi
9. https://www.overleaf.com/learn/latex/Matrices (https://www.overleaf.com/learn/latex/Matrices)

For example:

$$\int_0^1 x dx = \frac{1}{2}$$

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

For more latex things, google is your best friend - I forget what the commands are all the time. Any questions?

Markdown things:

1. Headers is #
2. Bolding stuff **bold stuff**
3. Italics is *italic stuff*
4. Both ***both***

More demo - Coding!

Basic Types:

1. Integers
2. Float64/Double - Floating point numbers, i.e. decimals (3.14159)
3. Strings - This is basically text ("This is a string.")
4. Arrays - Lists of things

Printing is one of the most important tools - it is extremely helpful for debugging.

```
 println("This stuff here will be printed.")
println(480)
```

In [2]:
```
### Printing a string, if you are inputting the string it needs to be
println("This stuff here will be printed.")

### Printing an integer
println(480)
```

```
This stuff here will be printed.
480
```

The basic arithmetic operations are given by +,-,*,/

```
1+1
2-1
8/3
1.23*3
```

In [3]:
```
1+1
2-1
8/3
1.23*3
```

Out[3]:
```
3.69
```

Why did they not all show up? In jupyter only the last thing evaluated will show up unless you print it explicitly.

In [4]:
```
println(1+1)
println(2-1)
println(8/3)
println(1.23*3)
```

```
2
1
2.6666666666666665
3.69
```

Even though you supply integers, the result will not necessarily stay the same type unlike in some other languages. In this case 8/3 automatically is cast into a float in Julia.

Setting variables is done with = :

```
 var1 = 1;
var2 = 100;
var3 = 1000;
var1 + var2
```

In [5]:
```
var1 = 1;
var2 = 100;
var3 = 1000;
var1 + var2
```

Out[5]:
```
101
```

It is **VERY** good practice to name your variables something descriptive. What I just did is actually terrible practice. If you are working on a long project keeping track of what var1... means is not only tedious, but say you leave it for a while and then pick it up again later, you will forget what it means. I have done this before and have had to deal with a very upset supervisor as a result.

Let's try and define a function now for this mathematical expressions:

$$f(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$$

Function definition is done like so:

```
 function cosApprox( x )
    ...
    return ...
end
```

A function will take in an argument and return another value.

In [1]:
```
function cosApprox( x )
    sum = 1;
    sum -= x^2 / 2;
    sum += x^4 / ( 1*2*3*4 );
    return sum
end
```

Out[1]:
```
cosApprox (generic function with 1 method)
```

In [2]:
```
cosApprox(0.1)
```

Out[2]: 0.9950041666666667

In [3]:
```
cosApprox(0.2)
```

Out[3]: 0.9800666666666666

In [ ]:
```

```