first class demo

python is slow

```
import numpy as np

def f(x, y, z):
  return np.exp(-x**2-y**2-z**2)

def main():

    N = 500
    xv = np.linspace(0.0, 2.0, N)
    dx = xv[2]-xv[1]

    ans = 0.0
    for x1 in xv:
        for x2 in xv:
            for x3 in xv:
                ans += f(x1, x2, x3) * dx**3
    print(ans)
    return ans

main()
```

VS

```
function f(x, y, z)
    ret1 = exp(-x^2-y^2-z^2)
    return ret1
end

function main()

    N = 500
    xv = range(0.0, 2.0, length=N)
    dx = xv[2]-xv[1]

    ret = 0.0
    for x1 in xv, x2 in xv, x3 in xv
        ret += exp(-x1^2-x2^2-x3^2) * dx^3
    end

    print(ret)
    return ret
end
```

```
main()
```

The output:

```
time julia test2.jl

0.6910931690128204
-------------------------------------------------------
Executed in  937.50 millis    fish           external
   usr time    1.88 secs      0.22 millis    1.88 secs
   sys time    0.89 secs      1.03 millis    0.89 secs

time python3 test2.py

0.6910931690128224


-------------------------------------------------------
Executed in   80.36 secs    fish           external
   usr time   81.11 secs  221.00 micros   81.11 secs
   sys time    1.24 secs  971.00 micros    1.24 secs
```

side effects

```
xarr = [1, 2, 3]
yarr = [2, 4, 6]

a = xarr
a += yarr
```

In python (but **not** in Julia), this would have modified xarr. But matrix side effect is still there:

```
Amat = [1 2 3; 4 5 6]
Bmat = Amat
Bmat[:,1] = [7 11]
```

what would Bmat be? What happens to Amat?

logistic curve

see class notes