TS_Logarithme_decimal

February 19, 2020

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
```

```
In [3]: %matplotlib inline
```

0.1 Données (concentration en ion hydronium)

```
In [68]: pH = np.arange(0, 15, 0.1)
         concentration = np.array([10**(-k) for k in pH])
```

0.2 Ph en fonction de la concentration à partir d'un graphique à échelle linéaire en abscisse et logarithmique en ordonnée

```
In [69]: pH
```

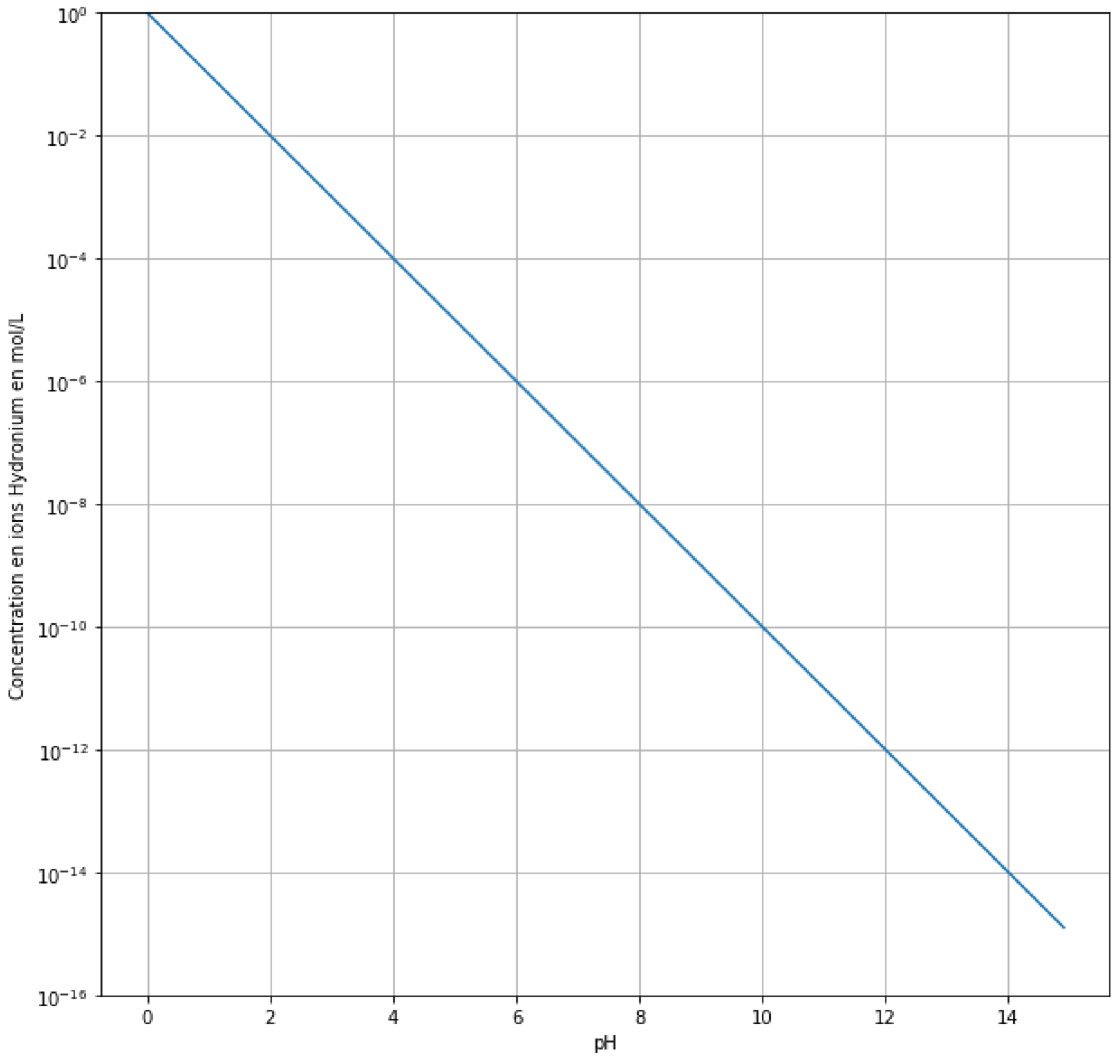
```
Out[69]: array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
                 1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,
                 2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ,  3.1,  3.2,
                 3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,  4. ,  4.1,  4.2,  4.3,
                 4.4,  4.5,  4.6,  4.7,  4.8,  4.9,  5. ,  5.1,  5.2,  5.3,  5.4,
                 5.5,  5.6,  5.7,  5.8,  5.9,  6. ,  6.1,  6.2,  6.3,  6.4,  6.5,
                 6.6,  6.7,  6.8,  6.9,  7. ,  7.1,  7.2,  7.3,  7.4,  7.5,  7.6,
                 7.7,  7.8,  7.9,  8. ,  8.1,  8.2,  8.3,  8.4,  8.5,  8.6,  8.7,
                 8.8,  8.9,  9. ,  9.1,  9.2,  9.3,  9.4,  9.5,  9.6,  9.7,  9.8,
                 9.9, 10. , 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.8, 10.9,
                11. , 11.1, 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 11.8, 11.9, 12. ,
                12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8, 12.9, 13. , 13.1,
                13.2, 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9, 14. , 14.1, 14.2,
                14.3, 14.4, 14.5, 14.6, 14.7, 14.8, 14.9])
```

```
In [70]: concentration
```

```
Out[70]: array([1.00000000e+00, 7.94328235e-01, 6.30957344e-01, 5.01187234e-01,
                3.98107171e-01, 3.16227766e-01, 2.51188643e-01, 1.99526231e-01,
                1.58489319e-01, 1.25892541e-01, 1.00000000e-01, 7.94328235e-02,
                6.30957344e-02, 5.01187234e-02, 3.98107171e-02, 3.16227766e-02,
                2.51188643e-02, 1.99526231e-02, 1.58489319e-02, 1.25892541e-02,
                1.00000000e-02, 7.94328235e-03, 6.30957344e-03, 5.01187234e-03,
```

```
       3.98107171e-03, 3.16227766e-03, 2.51188643e-03, 1.99526231e-03,
       1.58489319e-03, 1.25892541e-03, 1.00000000e-03, 7.94328235e-04,
       6.30957344e-04, 5.01187234e-04, 3.98107171e-04, 3.16227766e-04,
       2.51188643e-04, 1.99526231e-04, 1.58489319e-04, 1.25892541e-04,
       1.00000000e-04, 7.94328235e-05, 6.30957344e-05, 5.01187234e-05,
       3.98107171e-05, 3.16227766e-05, 2.51188643e-05, 1.99526231e-05,
       1.58489319e-05, 1.25892541e-05, 1.00000000e-05, 7.94328235e-06,
       6.30957344e-06, 5.01187234e-06, 3.98107171e-06, 3.16227766e-06,
       2.51188643e-06, 1.99526231e-06, 1.58489319e-06, 1.25892541e-06,
       1.00000000e-06, 7.94328235e-07, 6.30957344e-07, 5.01187234e-07,
       3.98107171e-07, 3.16227766e-07, 2.51188643e-07, 1.99526231e-07,
       1.58489319e-07, 1.25892541e-07, 1.00000000e-07, 7.94328235e-08,
       6.30957344e-08, 5.01187234e-08, 3.98107171e-08, 3.16227766e-08,
       2.51188643e-08, 1.99526231e-08, 1.58489319e-08, 1.25892541e-08,
       1.00000000e-08, 7.94328235e-09, 6.30957344e-09, 5.01187234e-09,
       3.98107171e-09, 3.16227766e-09, 2.51188643e-09, 1.99526231e-09,
       1.58489319e-09, 1.25892541e-09, 1.00000000e-09, 7.94328235e-10,
       6.30957344e-10, 5.01187234e-10, 3.98107171e-10, 3.16227766e-10,
       2.51188643e-10, 1.99526231e-10, 1.58489319e-10, 1.25892541e-10,
       1.00000000e-10, 7.94328235e-11, 6.30957344e-11, 5.01187234e-11,
       3.98107171e-11, 3.16227766e-11, 2.51188643e-11, 1.99526231e-11,
       1.58489319e-11, 1.25892541e-11, 1.00000000e-11, 7.94328235e-12,
       6.30957344e-12, 5.01187234e-12, 3.98107171e-12, 3.16227766e-12,
       2.51188643e-12, 1.99526231e-12, 1.58489319e-12, 1.25892541e-12,
       1.00000000e-12, 7.94328235e-13, 6.30957344e-13, 5.01187234e-13,
       3.98107171e-13, 3.16227766e-13, 2.51188643e-13, 1.99526231e-13,
       1.58489319e-13, 1.25892541e-13, 1.00000000e-13, 7.94328235e-14,
       6.30957344e-14, 5.01187234e-14, 3.98107171e-14, 3.16227766e-14,
       2.51188643e-14, 1.99526231e-14, 1.58489319e-14, 1.25892541e-14,
       1.00000000e-14, 7.94328235e-15, 6.30957344e-15, 5.01187234e-15,
       3.98107171e-15, 3.16227766e-15, 2.51188643e-15, 1.99526231e-15,
       1.58489319e-15, 1.25892541e-15])
```

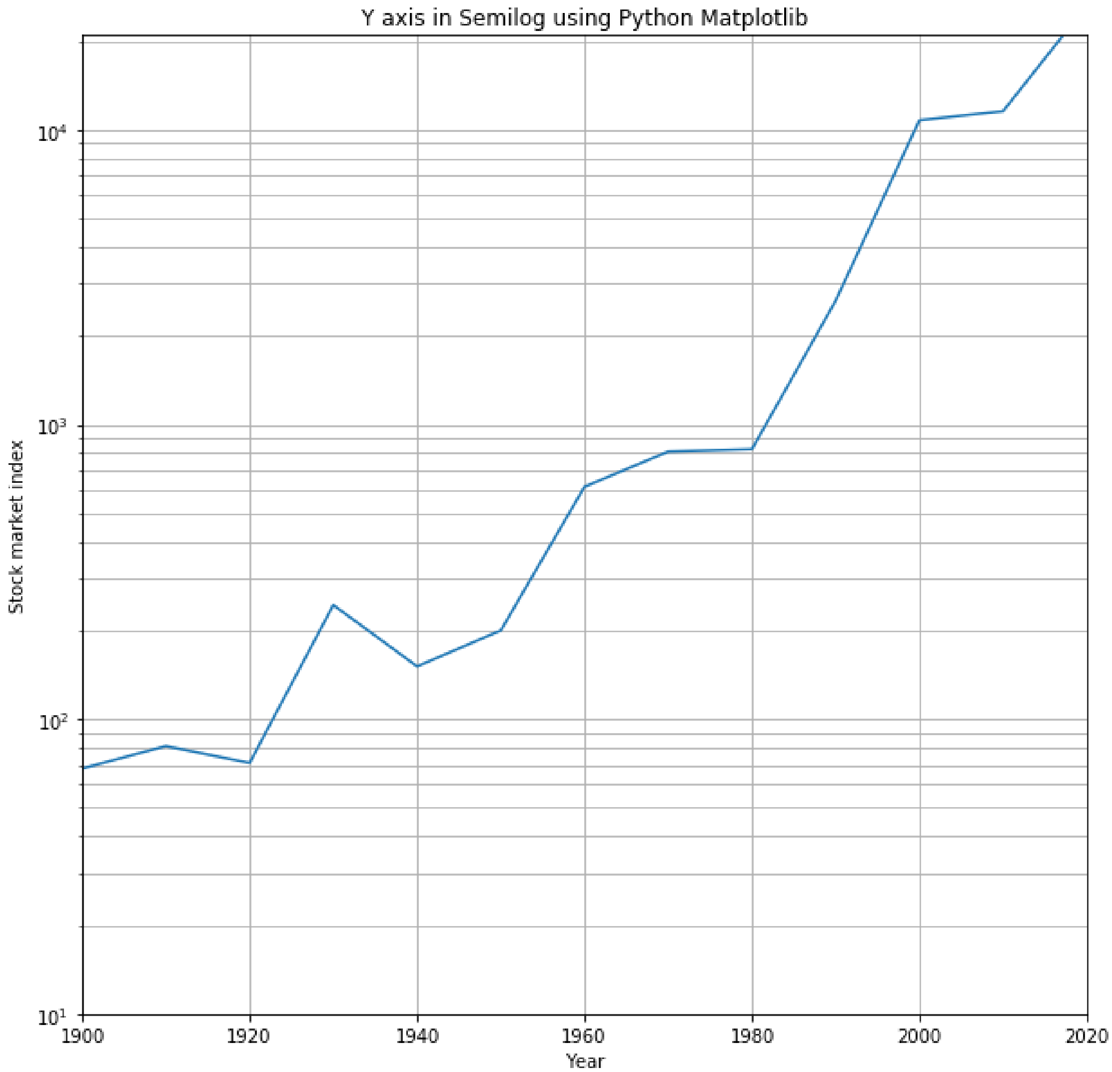
In [72]:
```
plt.figure(figsize=(10,10))
plt.xlabel('pH')
plt.ylabel('Concentration en ions Hydronium en mol/L')
plt.ylim([10**(-16), 1])
plt.grid(True)
plt.semilogy(pH, concentration)
plt.show()
```



0.3 Un autre exemple

```
In [58]: plt.figure(figsize=(10,10))

         # Year data for the semilog plot

         years       = [1900, 1910, 1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010


         # index data - taken at end of every decade - for the semilog plot

         indexValues = [68, 81, 71, 244, 151, 200, 615, 809, 824, 2633, 10787, 11577, 20656]
```

```python
# Display grid
plt.grid(True, which="both")

# Linear X axis, Logarithmic Y axis
plt.semilogy(years, indexValues )
plt.ylim([10,21000])
plt.xlim([1900,2020])

# Provide the title for the semilog plot
plt.title('Y axis in Semilog using Python Matplotlib')

# Give x axis label for the semilog plot
plt.xlabel('Year')

# Give y axis label for the semilog plot
plt.ylabel('Stock market index')

# Display the semilog plot
plt.show()
```


Y axis in Semilog using Python Matplotlib
Stock market index
10^4
10^3
10^2
10^1
1900
1920
1940
1960
1980
2000
2020
Year