Computer-based Exercises in Physical Chemistry

Raghunathan Ramakrishnan
ramakrishnan@tifrh.res.in

Tata Institute of Fundamental Research
TIFR Center for Interdisciplinary Sciences
Hyderabad, INDIA

26 December 2021

TIFR Centre for Interdisciplinary Studies

Why do we need

It’s easy to plot and appreciate chem

Why should I learn about computers?

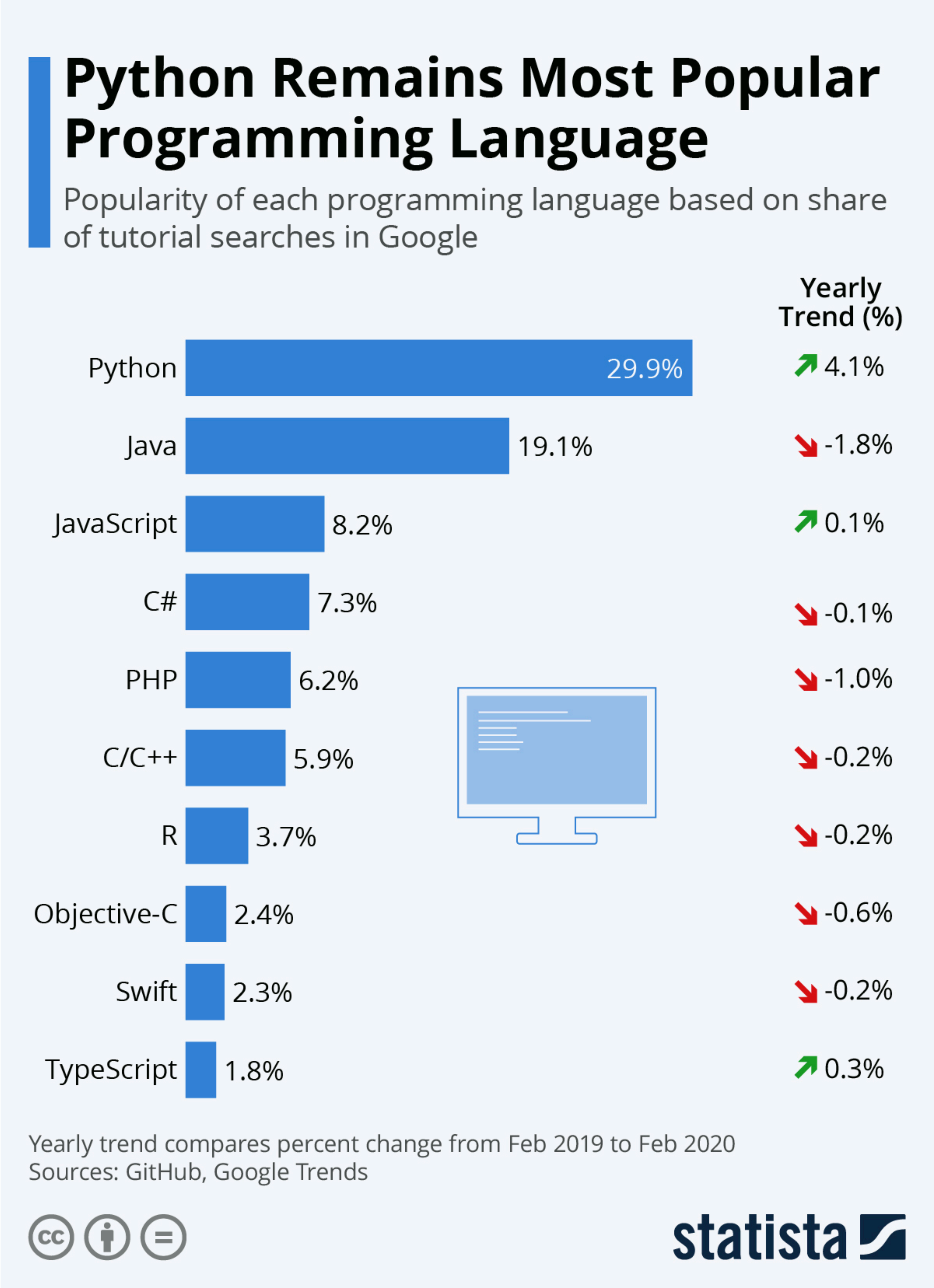
Often equations are difficult to solve manually.
Computer is a versatile equipment where one can perform several virtual experiments relevant for chemistry.

I want to have Python in my computer but I don’t know how to install it. What to do?

<u>Don’t be shy to ask around.</u>

Take help from friends, teachers, or research scholars in your institute.

Choosing a programming language



Q: What's the best programming language to learn for science student with no previous programming experience

A: Python

Python is

- free
- easy to reference in the internet
- has a lot of libraries for visualisation, numerical methods, and data-analysis
- more libraries means less coding effort *so that one can focus on the research problem at hand*

SciPy

NumPy

matplotlib

IP[y]: IPython Interactive Computing

pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

scikit learn

Why should I learn about computers?

Plotting function
Simple Statistics
Solving simple problems
Solving Advanced problems

Mathematics and Numerical Methods

Plotting function
Simple Statistics
Solving simple problems
Solving Advanced problems

A warm up problem

1.0 atm of nitrosyl chloride is introduced into a reaction vessel. The compound dissociates into nitric oxide and chlorine according to the reaction

$$2NOCl(g) \rightleftharpoons 2NO(g) + Cl_2(g)$$

If the equilibrium constant of the reaction is known to be 2.18, find the partial pressure of the gases at equilibrium

Answer

At equilibrium, $P_{\text{NOCl}} = 1 - 2x$, $P_{\text{NO}} = 2x$, and $P_{\text{Cl}_2} = x$.

Equilibrium-constant implies $\dfrac{P_{\text{NO}}^2 P_{\text{Cl}_2}}{P_{\text{NOCl}}^2} = \dfrac{(2x)^2 x}{(1-2x)^2} = K_{\text{eq.}} = 2.18$

The expression can be rearranged as a cubic equation $4x^3 - 8.72x^2 + 8.72x - 2.18 = 0$ which needs to be solved to determine the value of x (from which the partial pressures can be calculated)

Since $1 \geq P_{\text{NOCl}} \geq 0$ we know that $1 \geq 1 - 2x \geq 0$ or $x \geq 1/2$.

Cubic equation

We know how to solve a quadratic equation and find two solutions.

We are taught to rearrange the cubic equation into simple forms like (for example)

$(x - d)(ax^2 + bx + c) = 0$, in order to find the third solution.

Cubic equation

We know how to solve a quadratic equation and find two solutions.

We are taught to rearrange the cubic equation into simple forms like (for example)

$(x - d)(ax^2 + bx + c) = 0$, in order to find the third solution.

Now, let's see if a computer and Python can help us!

Graphical solution of the cubic equation

In [1]:

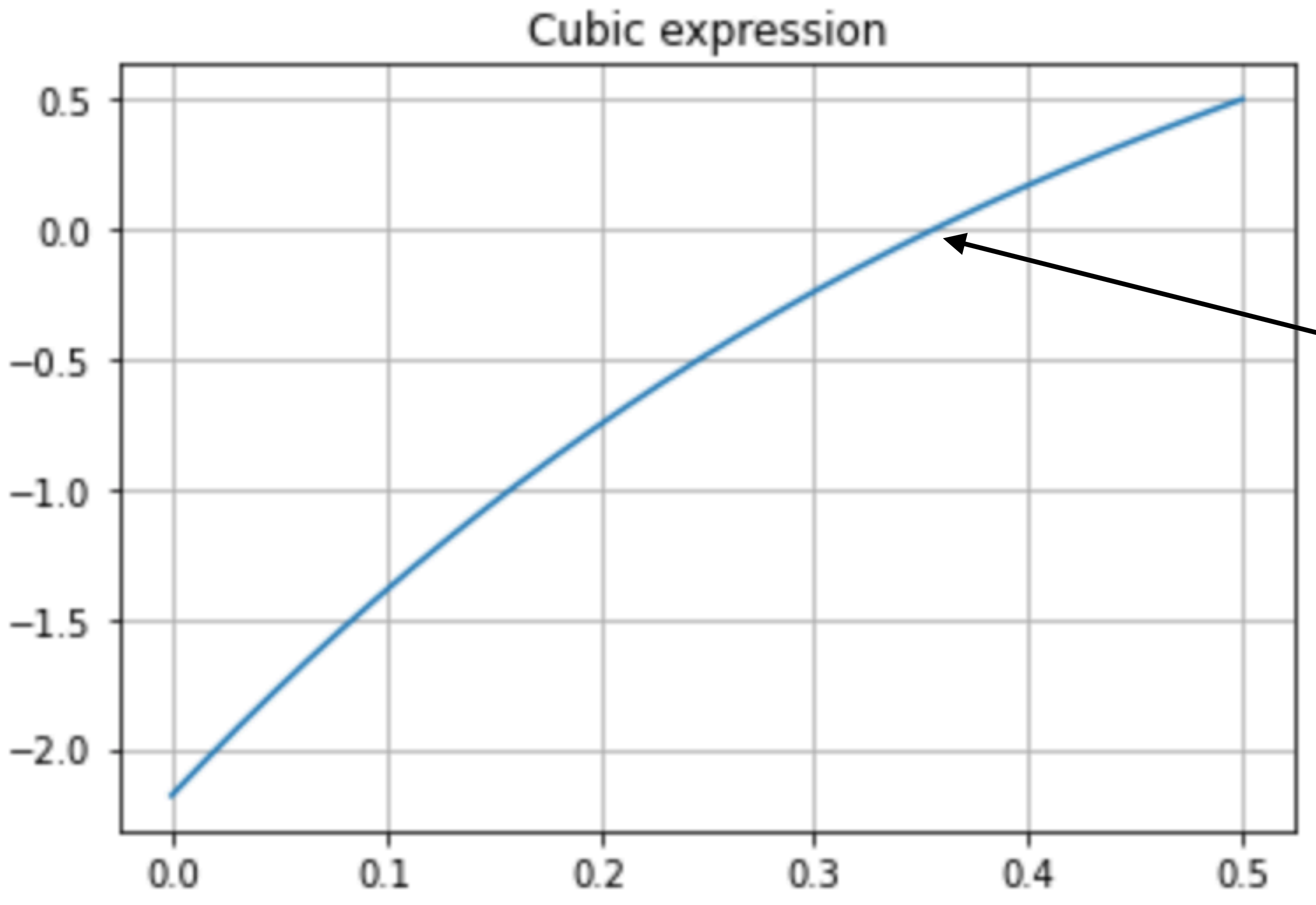
```
import numpy as np
import matplotlib.pyplot as plt

#=== x-values
x_min=0.0
x_max=0.5
x_grids=501
x=np.linspace(x_min, x_max, x_grids)

#=== f(x) values
f=4*x**3 - 8.72*x**2 + 8.72*x - 2.18

#=== plot
plt.plot(x,f)
plt.title('Cubic expression')
plt.grid()
plt.show()
```

x-range is fixed between 0 and 1/2 (using our previous knowledge of the problem)



The solution seems to be around x= 0.35

Numerical solution using secant method

In [2]:
```
from scipy import optimize

def f(x):
    f=4*x**3 - 8.72*x**2 + 8.72*x - 2.18
    return f

xguess=0.1
solution=optimize.root_scalar(f, fprime=None, x1=xguess+0.01, method='secant', bracket=None, x0=xguess, \
                              options={'tol':1e-6, 'maxiter':100})
solution
```

Out[2]:
```
      converged: True
           flag: 'converged'
 function_calls: 7
     iterations: 6
           root: 0.3560857818097863
```

- Secant method is a modified version of the Newton-Raphson method.
- Newton-Raphson requires that the derivative of the function is also known.
- In secant method, the derivative is approximated numerically using a finite-step (hence, finite-derivative)
- So, along with x0 (the initial guess for the root), another point x1 must also be specified.
- The derivative for the first iteration is estimated as $f'(x_0) \approx (f(x_1) - f(x_0))/(x_1 - x_0)$

The answer

At equilibrium, $P_{\mathrm{NOCl}} = 1 - 2x$, $P_{\mathrm{NO}} = 2x$, and $P_{\mathrm{Cl_2}} = x$.

```
In [3]: x=solution.root
        print("The partial pressure of NOCl is ",1-2*x)
        print("The partial pressure of NO is ",2*x)
        print("The partial pressure of Cl2 is ",x)

The partial pressure of NOCl is  0.28782843638042743
The partial pressure of NO is  0.7121715636195726
The partial pressure of Cl2 is  0.3560857818097863
```

The IPython notebook can be downloaded from https://github.com/raghurama123/Comp_PhysChem_Basic

The answer

At equilibrium, $P_{\mathrm{NOCl}} = 1 - 2x$, $P_{\mathrm{NO}} = 2x$, and $P_{\mathrm{Cl_2}} = x$.

```
In [3]: x=solution.root
        print("The partial pressure of NOCl is ",1-2*x)
        print("The partial pressure of NO is ",2*x)
        print("The partial pressure of Cl2 is ",x)

The partial pressure of NOCl is  0.28782843638042743
The partial pressure of NO is  0.7121715636195726
The partial pressure of Cl2 is  0.3560857818097863
```

Are you surprised that the sum of all partial pressures exceed the initial pressure of 1 atm?

A problem in integration

Debye's theory of molar heat capacity (Debye-T^3 law) of a monoatomic crystal states

$$\overline{C}_V(T) = 9R\left(\frac{T}{\Theta_\mathrm{D}}\right)^3 \int_0^{\Theta_\mathrm{D}/T} \frac{x^4 e^x}{(e^x-1)^2} dx$$

where R is the gas constant and $\Theta_\mathrm{D} = 309$ K is the Debye temperature. Given that for copper, , find the molar heat capacity at $T = 90$ K.

Experimentally measured value is $14.49\,\mathrm{J}\cdot K^{-1}\cdot\mathrm{mol}^{-1}$

Numerical integration using quadrature

In [1]:
```
import numpy as np
from scipy import integrate

R=8.314     # gas constant in J K^-1 mol^-1

Theta_D=309 # Debye Temperature of copper in K

T = 90     # given K at which we want molar heat capacity

def fn_I(x):
    fn_I= x**4 * np.exp(x) / (np.exp(x)-1)**2
    return fn_I

# For fine-tuning the integration settings see the scipy documenation
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.quadrature.html

Integral=integrate.quadrature(fn_I, 0.0, Theta_D/T)

print("The value of the integral is: ",Integral[0], " with a numerical error of ", Integral[1])

CV=9*R*(T/Theta_D)**3*Integral[0]

print("Heat capacity of Cu at T = 103 K is: ",CV, " J mol^-1 K^-1")
```

lower limit

upper limit

```
The value of the integral is:  7.9788851408858035  with a numerical error of  6.324787538147802e-08
Heat capacity of Cu at T = 103 K is:  14.751861725666032  J mol^-1 K^-1
```

Agrees well with the experimentally measured value: $14.49\,\mathrm{J}\cdot K^{-1}\cdot\mathrm{mol}^{-1}$