Numerical Approaches for Quantum Mechanics

Raghunathan Ramakrishnan
ramakrishnan@tifrh.res.in

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

30 December 2021

TIFR Centre for Interdisciplinary Studies

The presentation and the codes are available for download at https://github.com/raghurama123/NumQM

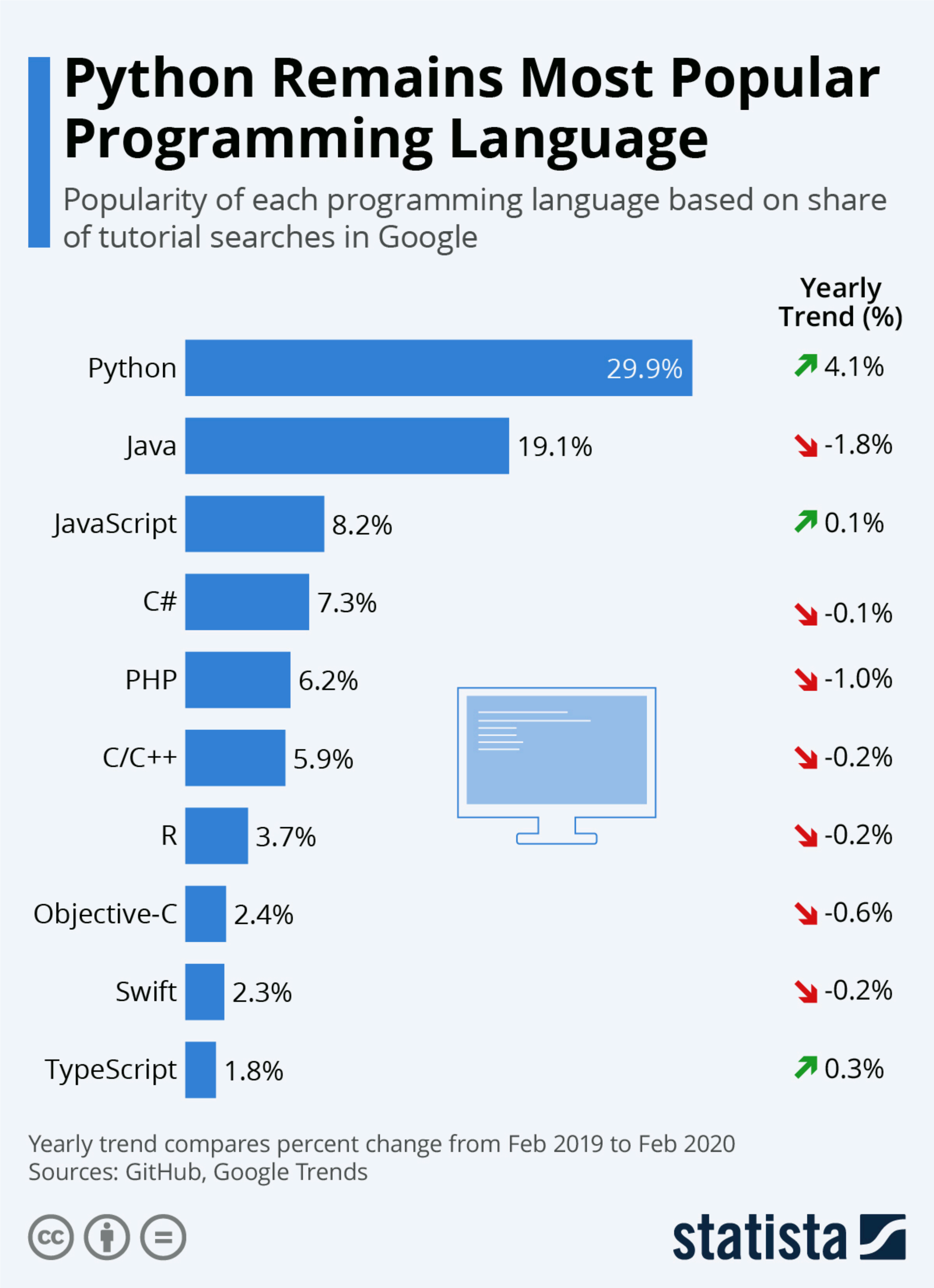
Purpose of this presentation

- To give a flavour for numerical modelling of quantum mechanical problems.
- To demonstrate how a bit of programming can clarify important concepts.
- To highlight the connections between concepts learned in quantum mechanics and the mathematics encountered in numerical computation.
- To provide sample programs that you can download, modify and enhance your understanding.

Why Numerical Approaches?

- Except for simple cases, quantum mechanical problems are mathematically intensive.
- Numerical modelling provides quick solutions to analytically complicated problems.
- When combined with visualisation of results, computer modelling provides deep insights.
- One can perform virtual experiments that may not even be feasible to perform in a laboratory.
- Free software can help fast-track coding efforts.
- Possible to combine to symbolic computation with numerical computation.

Choosing a programming language



Python Remains Most Popular Programming Language

Popularity of each programming language based on share of tutorial searches in Google

Yearly trend compares percent change from Feb 2019 to Feb 2020
Sources: GitHub, Google Trends

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

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

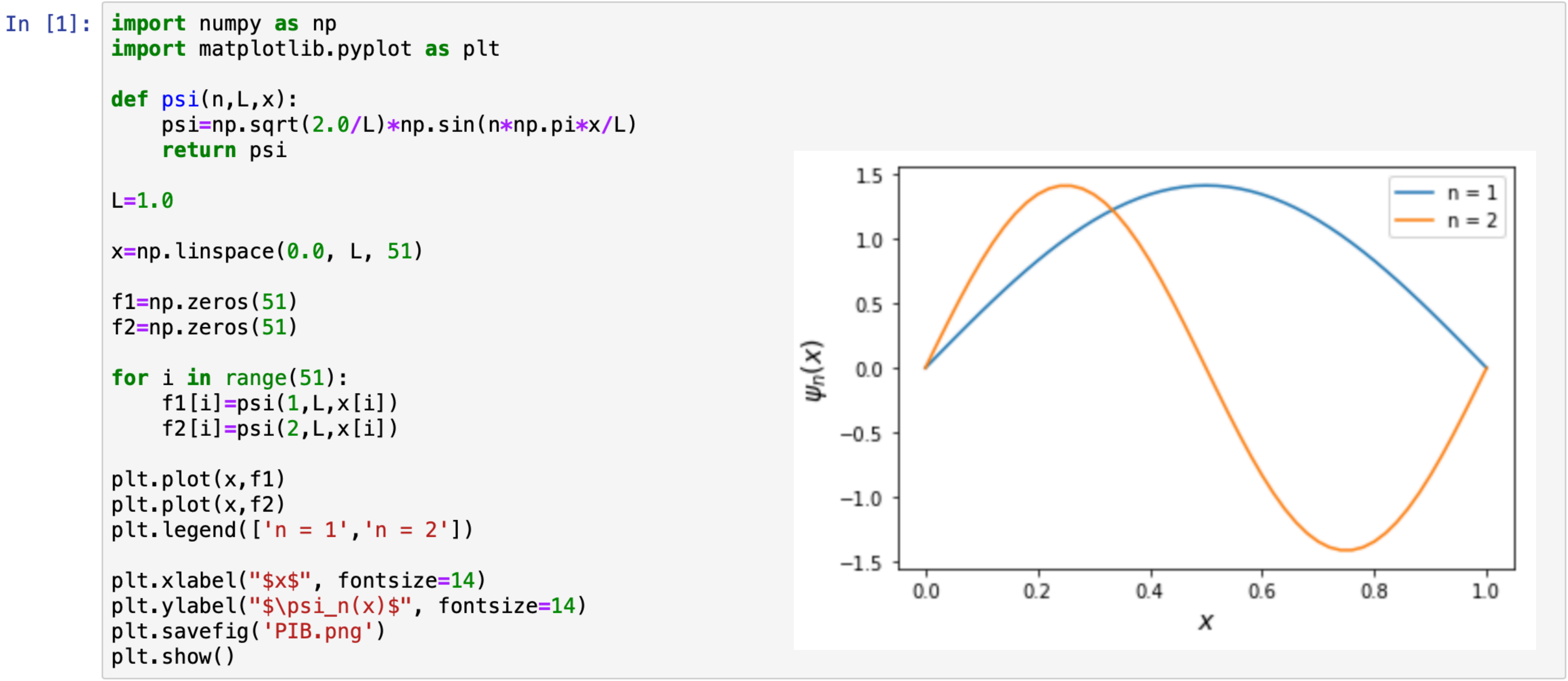
Don’t be shy to ask around.

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

Mathematics and Numerical Methods

Equations	Root-finding (Newton-Raphson, simplex, etc.)
Functions	Optimization: minimization (Newton-Raphson, simplex, etc.)
Function approximation	Least-squares regression (Gaussian elimination, LU decomposition, etc.), interpolation
Eigenvalue problems	Similarity transformation (Jacobi method), Iterative method (power method, Lanczos, etc.)
Differentiation/Integration	Finite-derivatives, trapezoidal method, quadratures, etc.
Differential equations	Euler method, Runge-Kutta, etc.

Let’s begin with a plot



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

        def psi(n,L,x):
            psi=np.sqrt(2.0/L)*np.sin(n*np.pi*x/L)
            return psi

        L=1.0

        x=np.linspace(0.0, L, 51)

        f1=np.zeros(51)
        f2=np.zeros(51)

        for i in range(51):
            f1[i]=psi(1,L,x[i])
            f2[i]=psi(2,L,x[i])

        plt.plot(x,f1)
        plt.plot(x,f2)
        plt.legend(['n = 1','n = 2'])

        plt.xlabel("$x$", fontsize=14)
        plt.ylabel("$\psi_n(x)$", fontsize=14)
        plt.savefig('PIB.png')
        plt.show()
```

Non-stationary states and their time-evolution

Recall:

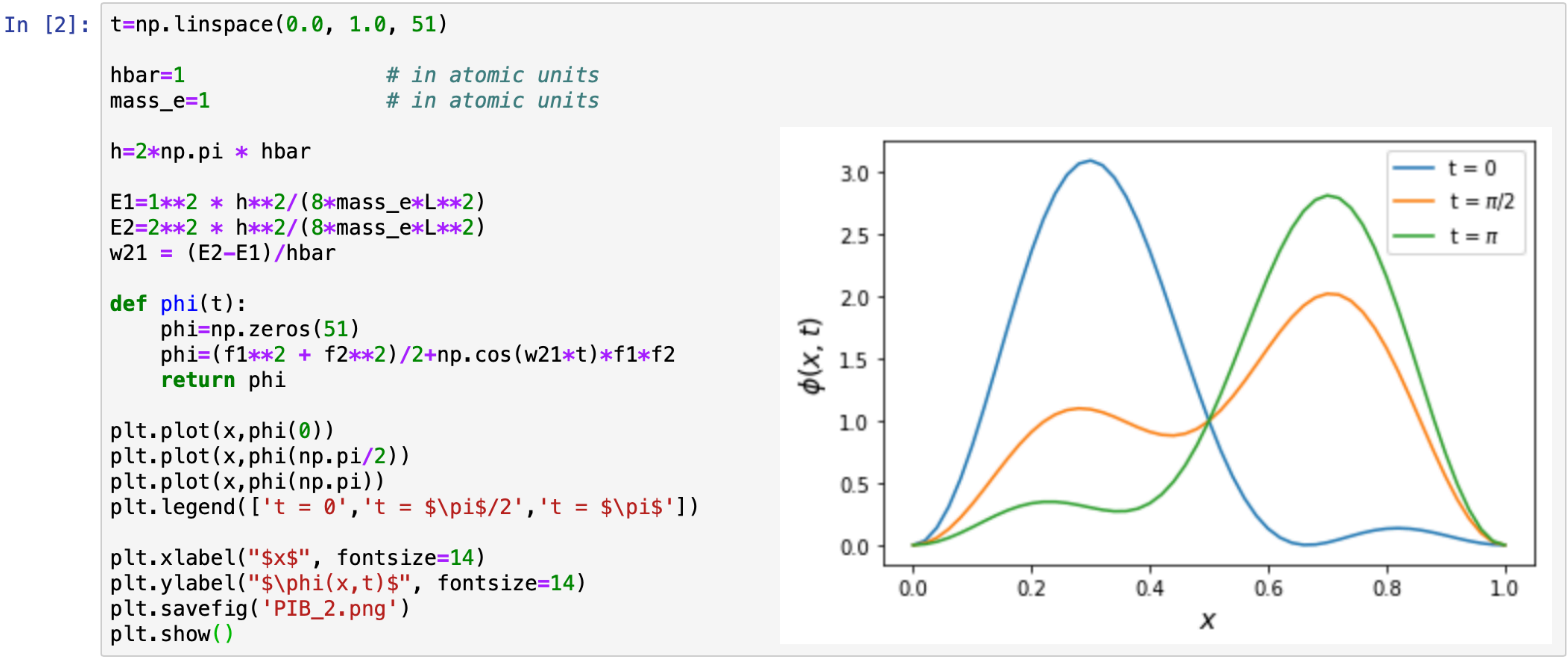
- ❖ Eigenstates of a system are stationary. When a system is in one of the eigenstates, all quantum mechanical observables (expectation values, $\langle n | \hat{O} | n \rangle$) are conserved in time.

- ❖ Linear combinations of eigenfunctions are not stationary. Consider $\phi(x) = \left[\psi_m(x) + \psi_n(x)\right]/\sqrt{2}$

$$\phi(x,t) = \exp\left(-it\hat{H}/\hbar\right)\left[\psi_m(x) + \psi_n(x)\right]/\sqrt{2} = \Big[\exp\left(-itE_m/\hbar\right)\psi_m(x) + \exp\left(-itE_n/\hbar\right)\psi_n(x)\Big]/\sqrt{2}$$

$$\begin{aligned}
|\phi(x,t)|^2 &= \phi^*(x,t)\phi(x,t) \\
&= \left(|\psi_m(x)|^2 + |\psi_n(x)|^2\right)/2 + \mathrm{Real}\left[\exp(-it\omega_{mn})\psi_n^*(x)\psi_m(x)\right]; \qquad \omega_{mn} = (E_m - E_n)/\hbar \\
&= \left(|\psi_m(x)|^2 + |\psi_n(x)|^2\right)/2 + \cos(\omega_{mn}t)\psi_n^*(x)\psi_m(x)
\end{aligned}$$

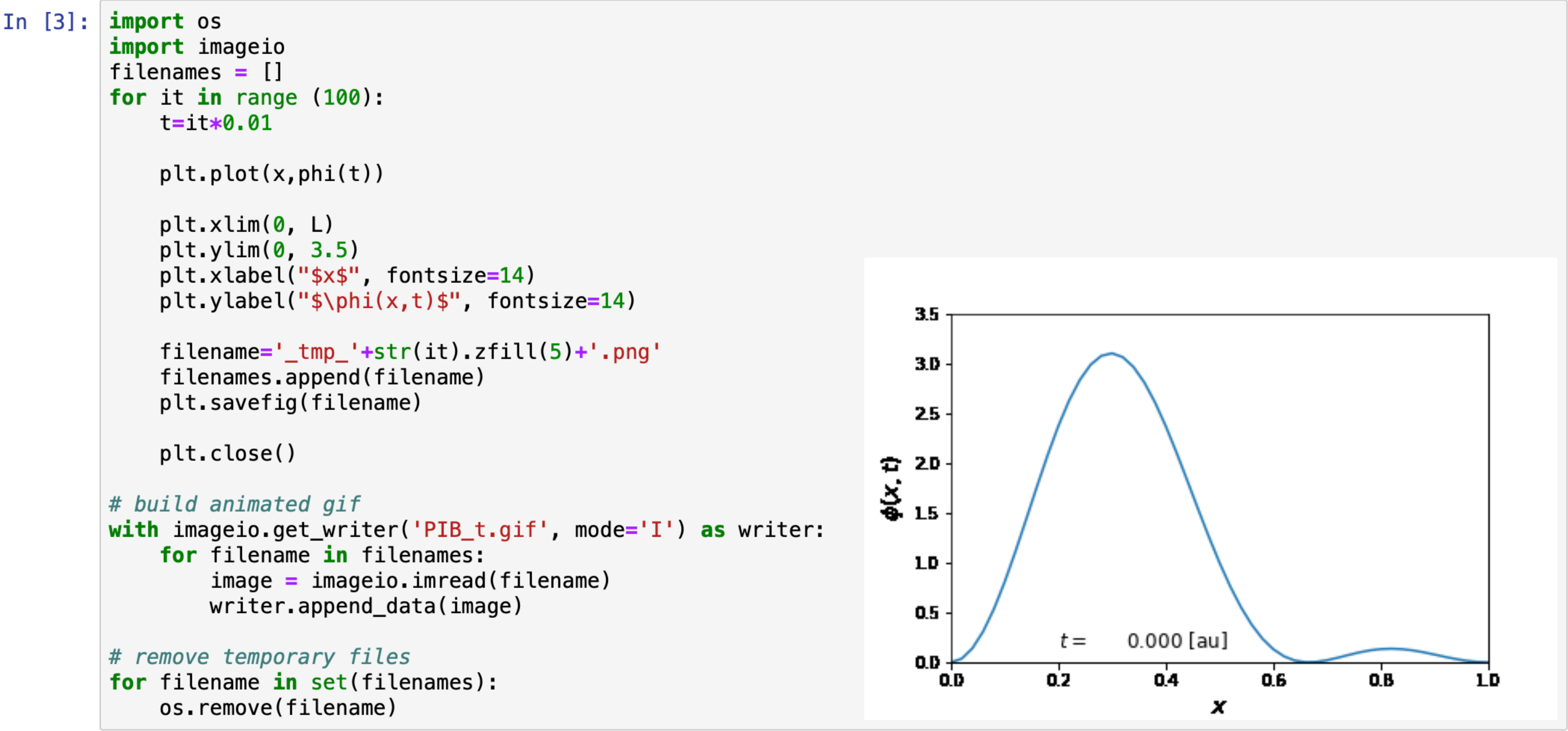
Let's watch the wave function evolve in time



In [2]:

```
t=np.linspace(0.0, 1.0, 51)

hbar=1                  # in atomic units
mass_e=1                # in atomic units

h=2*np.pi * hbar

E1=1**2 * h**2/(8*mass_e*L**2)
E2=2**2 * h**2/(8*mass_e*L**2)
w21 = (E2-E1)/hbar

def phi(t):
    phi=np.zeros(51)
    phi=(f1**2 + f2**2)/2+np.cos(w21*t)*f1*f2
    return phi

plt.plot(x,phi(0))
plt.plot(x,phi(np.pi/2))
plt.plot(x,phi(np.pi))
plt.legend(['t = 0','t = $\pi$/2','t = $\pi$'])

plt.xlabel("$x$", fontsize=14)
plt.ylabel("$\phi(x,t)$", fontsize=14)
plt.savefig('PIB_2.png')
plt.show()
```

Let’s watch the wave function evolve in time



In [3]:

```
import os
import imageio
filenames = []
for it in range (100):
    t=it*0.01

    plt.plot(x,phi(t))

    plt.xlim(0, L)
    plt.ylim(0, 3.5)
    plt.xlabel("$x$", fontsize=14)
    plt.ylabel("$\phi(x,t)$", fontsize=14)

    filename='_tmp_'+str(it).zfill(5)+'.png'
    filenames.append(filename)
    plt.savefig(filename)

    plt.close()

# build animated gif
with imageio.get_writer('PIB_t.gif', mode='I') as writer:
    for filename in filenames:
        image = imageio.imread(filename)
        writer.append_data(image)

# remove temporary files
for filename in set(filenames):
    os.remove(filename)
```

Wavefunctions in x and p representations

Suppose a system is in a state denoted by the ket $|n\rangle$, then the corresponding wave function of the system is obtained by representing the ket in the position or momentum eigen kets.

$$\psi_n(x_0) = \langle x_0 | n \rangle; \qquad \phi_n(p_0) = \langle p_0 | n \rangle$$

The definition of the basis kets $|x_0\rangle$ and $|p_0\rangle$ has to be taken as one of the postulates of quantum mechanics. They are related to one another by Fourier transformation.

	x-representation	p-representation
$\lvert x\rangle$	$\langle x_0 \lvert x\rangle = \delta(x - x_0)$	$\langle p_0 \lvert x\rangle = \dfrac{1}{\sqrt{2\pi\hbar}} \exp(-ip_0x/\hbar)$
$\lvert p\rangle$	$\langle x_0 \lvert p\rangle = \dfrac{1}{\sqrt{2\pi\hbar}} \exp(ipx_0/\hbar)$	$\langle p_0 \lvert p\rangle = \delta(p - p_0)$

Changing representations

$$
\begin{aligned}
\phi_n(p_0) &= \langle p_0 | n \rangle \\
&= \langle p_0 | \hat{I} | n \rangle \\
&= \langle p_0 | \int dx_0 | x_0 \rangle \langle x_0 | n \rangle \\
&= \int dx_0 \langle p_0 | x_0 \rangle \langle x_0 | n \rangle \\
&= \int dx_0 \langle p_0 | x_0 \rangle \psi_n(x_0)
\end{aligned}
$$

$$
= \frac{1}{\sqrt{2\pi\hbar}} \int dx_0 \exp(-ip_0 x_0/\hbar) \psi_n(x_0)
$$

$$
\begin{aligned}
\psi_n(x_0) &= \langle x_0 | n \rangle \\
&= \langle x_0 | \hat{I} | n \rangle \\
&= \langle x_0 | \int dp_0 | p_0 \rangle \langle p_0 | n \rangle \\
&= \int dp_0 \langle x_0 | p_0 \rangle \langle p_0 | n \rangle \\
&= \int dp_0 \langle x_0 | p_0 \rangle \phi_n(p_0)
\end{aligned}
$$

$$
= \frac{1}{\sqrt{2\pi\hbar}} \int dp_0 \exp(ip_0 x_0/\hbar) \phi_n(p_0)
$$

$\hat{I} = \int dx_0 | x_0 \rangle \langle x_0 | = \int dp_0 | p_0 \rangle \langle p_0 |$ is called as the completeness relation

Thank you for your attention!

ramakrishnan@tifrh.res.in