Python Programming for Life Science Students (PyLifeS)
January - April semester, 2024

A course offered to the students of M.Sc. (Life Sciences)
at the School of Life Sciences, University of Hyderabad and
Ph.D./Integrated Ph.D. (Biology) students at TIFR Hyderabad

Raghunathan Ramakrishnan
ramakrishnan@tifrh.res.in
Tata Institute of Fundamental Research Hyderabad
Hyderabad, India

Course material: https://github.com/raghurama123/pylifes

tifr HYDERABAD | TATA INSTITUTE OF FUNDAMENTAL RESEARCH

Content

References

General References

1. *Understanding Bioinformatics*, Marketa Zvelebil, and Jeremy O. Baum (Garland Science, 2008).

2. *Bioinformatics Algorithms: Design and Implementation in Python*, Miguel Rocha University of Minho, Braga, Portugal Pedro G. Ferreira (Academic Press, 2018).

3. *Computing Skills for Biologists: A toolbox*, Stefano Allesina & Madlen Wilmes (Princeton University Press, 2019).

4. *Python for the Life Sciences: A Gentle Introduction to Python for Life Scientists*, Alexander Lancaster, Gordon Webster (Springer, 2019).

5. William Bo Rothwell, "*Linux for Developers*", Pearson (2018). *See the chapters about GitHub*.

Working with GitHub

This section has been addressed in a practical, hands-on session. If you're unfamiliar with GitHub, allocating approximately 30 minutes to follow through these steps is recommended.

Git and GitHub

- `git` is a commandline program (version control system) which you can access from Linux/Mac terminal.
- GitHub is a platform where you can store and manage your codes/repository
  - interactively on the GitHub website or the GitHub desktop program (*https://desktop.github.com/*) which you can install in your laptop or desktop
  - through `git` commands in a terminal. Check Ref.5 to learn about `git` commands.

Fork the main branch (once)

- Create a GitHub accound and fork the main branch : *https://github.com/raghurama123/pylifes*

Push your changes

- If you have GitHub-deskop installed on a Windows environment, your copy of the repository should be located at `Onedrive/Documents/Github/pylifes`
- You can edit any files present in this folder or create a new file.
- Then, open the GitHub-desktop application, and figure out how to `commit` and `push` your changes

Pull and merge new changes

- Before every class, check your copy of the course content at *https://github.com/USER/pylifes*
- Click the message x `commits behind`
- Click `Create pull request` (which you have to click twice) and click `Merge pull request`

Python basics

You are expected to have the Jupyter Notebook platform installed to work with Python codes. Those with a Windows-based laptop, may install Anaconda *https://docs.anaconda.com/free/anaconda/install/windows/*

Your first Jupyter notebook

- The essential aspects of the python language are collected in *notebooks/PyLifeS01*
- You should read the note book, line-by-line, clarify your doubts.
- Run each line input cell by `shift + enter` or by clicking the run button at the top ▶ Run
- You can type-in any new code, text or comments in this notebook.
- You can also copy the input from one cell and paste in another cell, modify it and run it.
- At the end of each session, remember to save your notebook, and commit/merge to your repository.

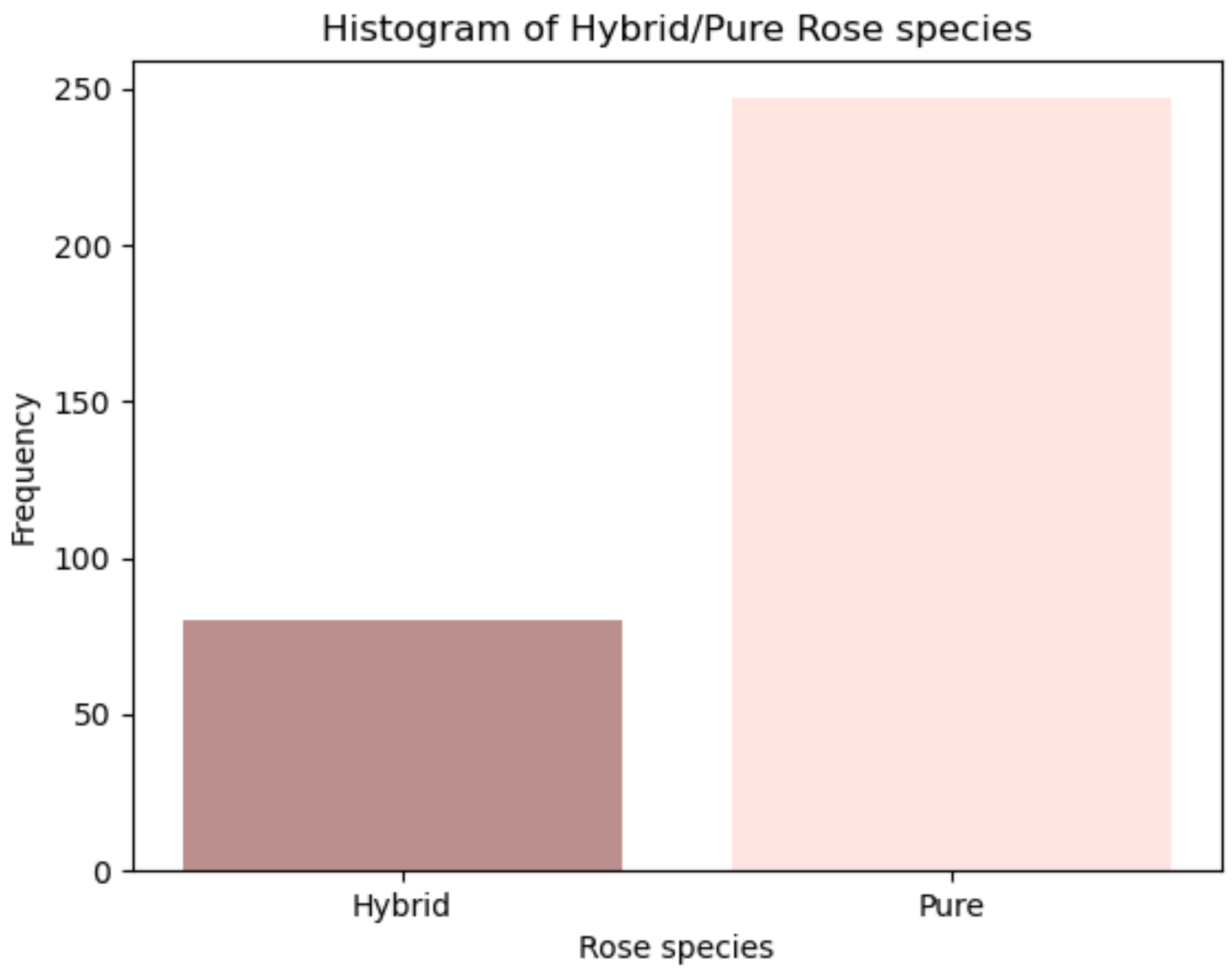
Topics covered

1. Basic datatypes in Python (int, float, complex, string)
2. String manipulation (splitting, splicing)
3. Python list (more commonly encountered datatype during string processing)
4. String concatenation and list appending (building larger strings)
5. For loop (to iterate over a list)
6. While loop (when the number of cycles is unknown apriori)
7. Break vs. continue (gracefully exiting a loop)
8. Functions (writing custom functions, doc string, help)
9. Module (importing modules, math, numpy, numpy arrays.)
10. Input/output (Read from/Write in a file, formatted printing)

Count Roses: A mini project with Pandas

Load the Rose dataset and perform basic statistics

- Genus and species names of all the known species of Rose plant are collected in the file *datasets/Rose.csv*. Let's load this file using the library Pandas and count the number of hybrid species where the Genus and species names are separated by 'x' and pure species.
- We will learn to use some functions in Pandas functions, and attributes of objects created using Pandas, and finally use the library matplotlib to plot a histogram as shown below.
- The necessary python codes are given in *notebooks/PyLifeS02_Rosa*



Self-study Task: Explore Pandas

Exercise 1

The Pandas function `pd.read_csv` was used to read a csv file as a dataframe. The function `dataframe.to_csv` can be used to write a dataframe in a csv file. Extend *notebooks/PyLifeS02_Rosa* by including Python statements to prepare separate csv files for the scientific names of hybrid and pure species of roses.

Exercise 2

Suppose you have a dataframe named df. Explain the Python statement df_copy=df.copy()

Exercise 3

The iris flower dataset containing measurements of sepal length, sepal width, petal length and petal width can be loaded directly in your code from the scikit-learn library.

```
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()

iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

Make a scatterplot of sepal length and sepal width using the following code.

```
import matplotlib.pyplot as plt

iris_df.plot(kind='scatter', x='sepal length (cm)', y='sepal width (cm)')
plt.show()
```

Try the following statements individually and explain the output.

`print(iris_df.describe())`, `print(iris_df.info())`, `print(iris_df.sum())`, `print(iris_df.mean())`, `print(iris_df.median())`, `print(iris_df.max())`, `print(iris_df.min())`

Modeling growth, population, reaction rate and allometry

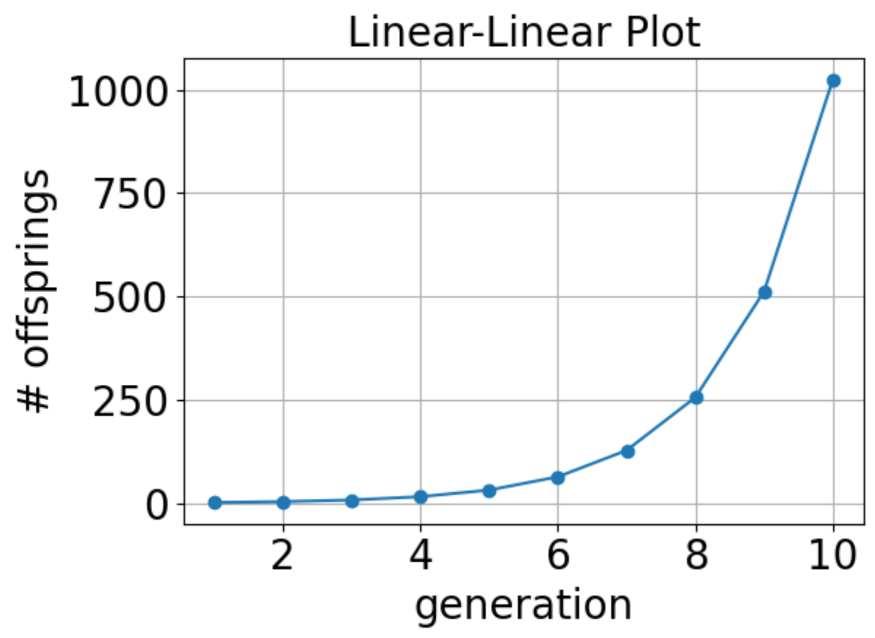
Logarithm

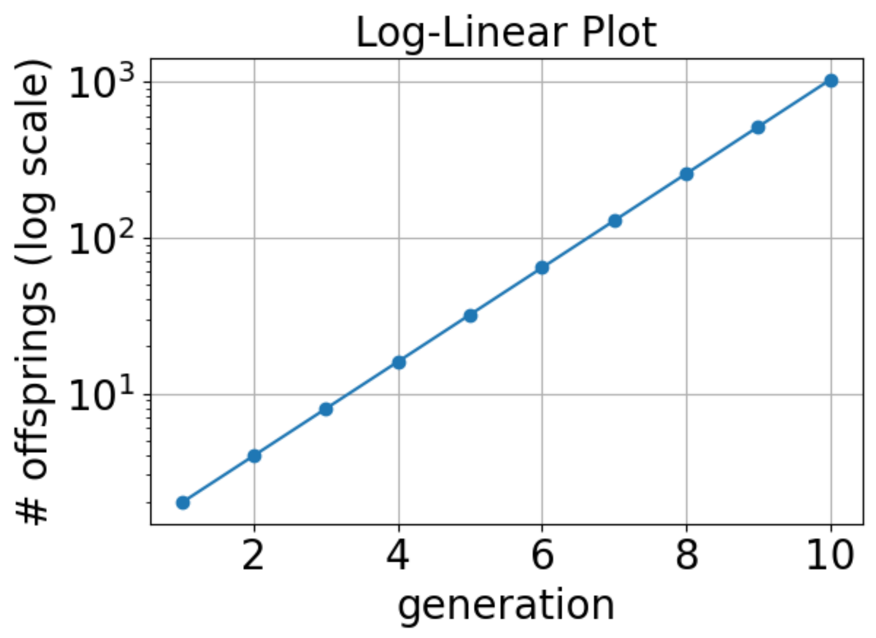
Suppose a bacterial cell divides into two in the first generation. If each offspring divides into two in the second generation, producing four new offsprings, how can we find in which generation the number of new offspings will reach 1 Million?

Answer: In the first generation, we have 2 offsprings, and 2^2=4 offsprings in the second generation. In the *N*-th generation, we have 2^N offsprings. So, want to know when we will have $2^N = 10^6$. We can write this equation by taking log (with base 10) on both sides to get $N\log_{10}2 = 6\log_{10}10 = 6$. So, the answer is $N = 6/\log_{10}2 = 6/0.3010 = 19.93$. So, at the 20th generation, the population of new offsprings will be exceed 1 Million. In Python, you can calculate "log to the base 10" using `numpy.log10()`. However, when you used the matplotlib function `plt.yscale('log')`, the default base is 10.

Plotting in log scale

- The growth of the bacterial colony from the above example is shown *notebooks/PyLifeS03_GrowthFunctions*





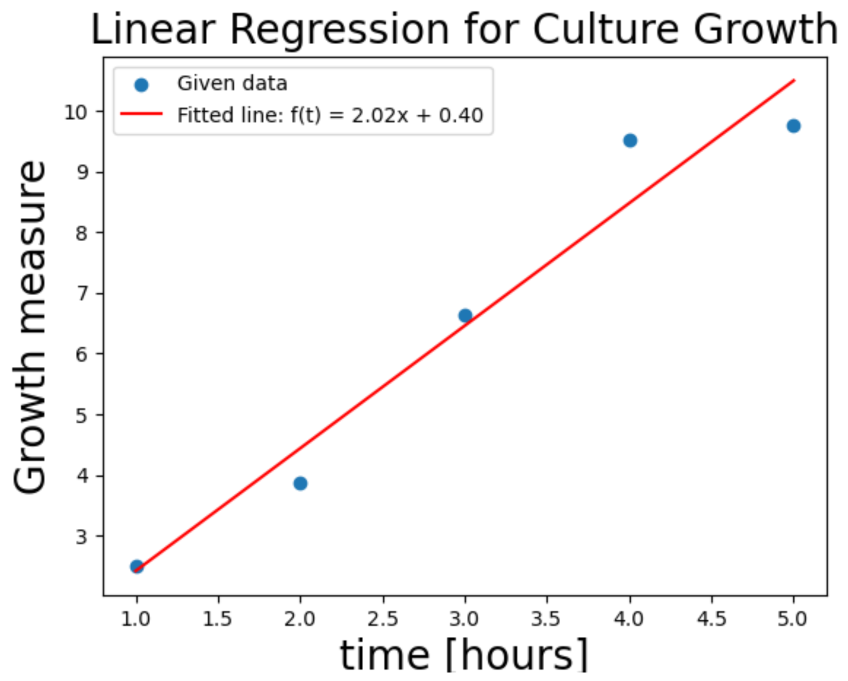
- The plot on the right side is shown for the same data but with the y-axis in log scale. Discuss the advantages of both styles of plots.

log-log vs. log-linear plots

- The function plotted in the previous example is $y = 2^N$. In general, any function of the form $y = Ac^{bx}$ can be plotted in the log-linear style. So, we will have $\log_c(y) = \log_c(A) + bx = y_0 + bx$. We can find the intercept at y_0 (the value of y at $x = 0$) and the slope as b.
- Note that the base to select in this case has to be c. Why?
- If the function we want to plot is of the form $y = Ax^b$, then one can use the log-log style by taking log on both sides to get $\log(y) = \log(A) + b\log(x)$. In this case, it actually does not matter what base is taken for the logarithm, why?

Straight-line fitting

- Suppose we have some measure (population, weight, or some variable that is related to population) of the growth of a culture at 1, 2, 3, 4, 5 hours as 2.497, 3.862, 6.648, 9.523, 9.766 (in some units). Assuming that the growth follows the equation $f(t) = At + B$, let us find A and B using linear regression. In *notebooks/PyLifeS03_GrowthFunctions* we solve this problem using the `LinearRegression` procedure of the library `sklearn` (Scikit-learn).



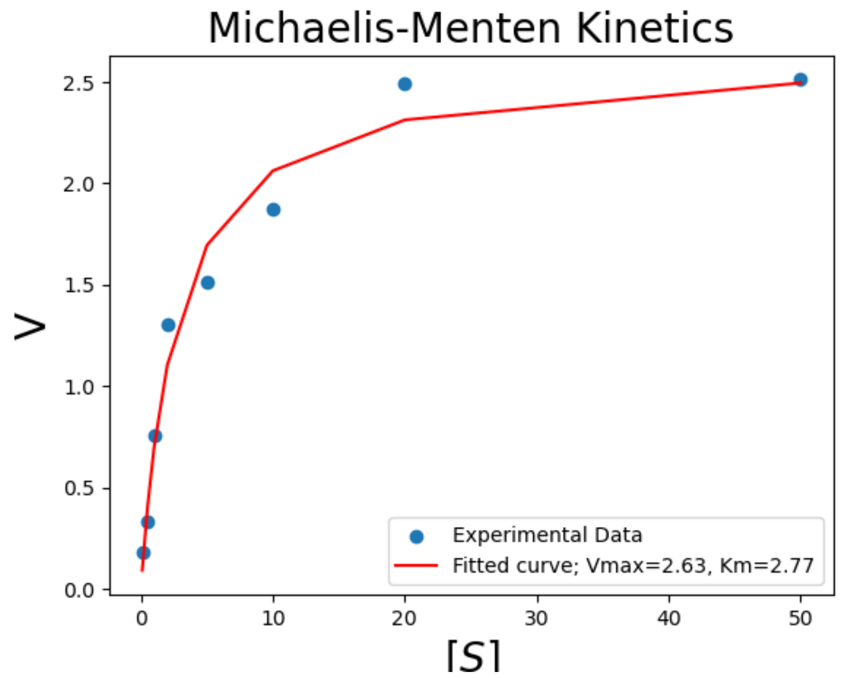
Enzyme activity: Michaelis-Menten equation and Lineweaver-Burk transformation

- In enzyme-catalyzed reactions, the activity of the enzyme (which is the rate of formation of the product) is related to the enzyme concentration according to the Michaelis-Menten equation.

$$\frac{dP}{dt} = V = \frac{V_{\max}[S]}{K_m + [S]}$$

V is the enzyme activity (V for velocity), $[S]$ is the substrate concentration, $V_{\max}$ is the maximum enzyme activity anticipated at a very high concentration of the substrate, and K_m is the Michaelis-Menten constant characteristic of the reaction.

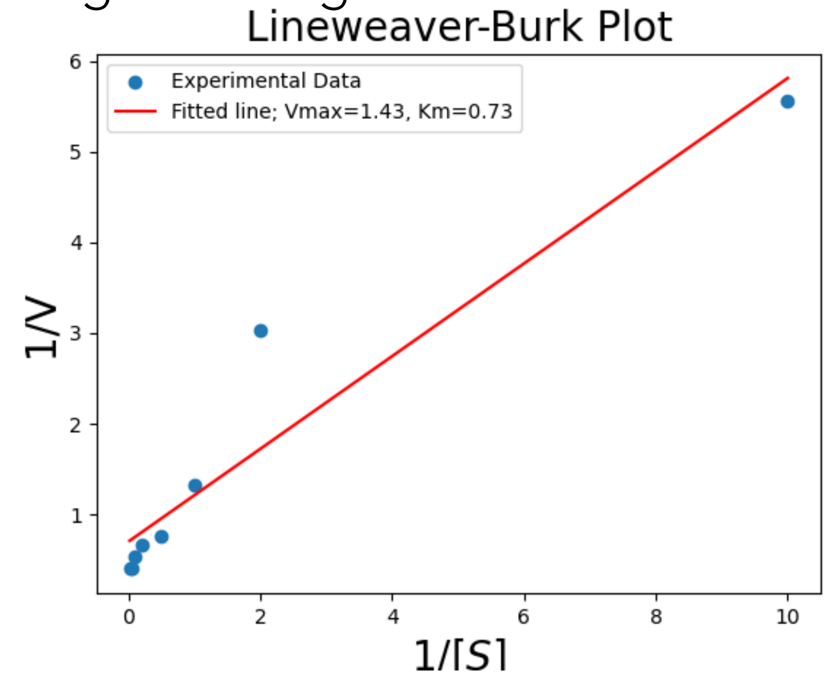
Maud Menten



- To determine, the reaction parameters, $V_{\max}$ and K_m one can directly fit V as a function of $[S]$ using `scipy.optimize.curvefit` as shown in *notebooks/PyLifeS03_GrowthFunctions*.
- A less accurate, but easy to visualize, procedure is to consider the reciprocal of the both sides of the equation resulting in the equation of a straight line that can be fitted using linear regression.

$$\frac{1}{V} = \frac{K_m + [S]}{V_{\max}[S]} = \frac{1}{V_{\max}} + \frac{K_m}{V_{\max}} \frac{1}{[S]}$$

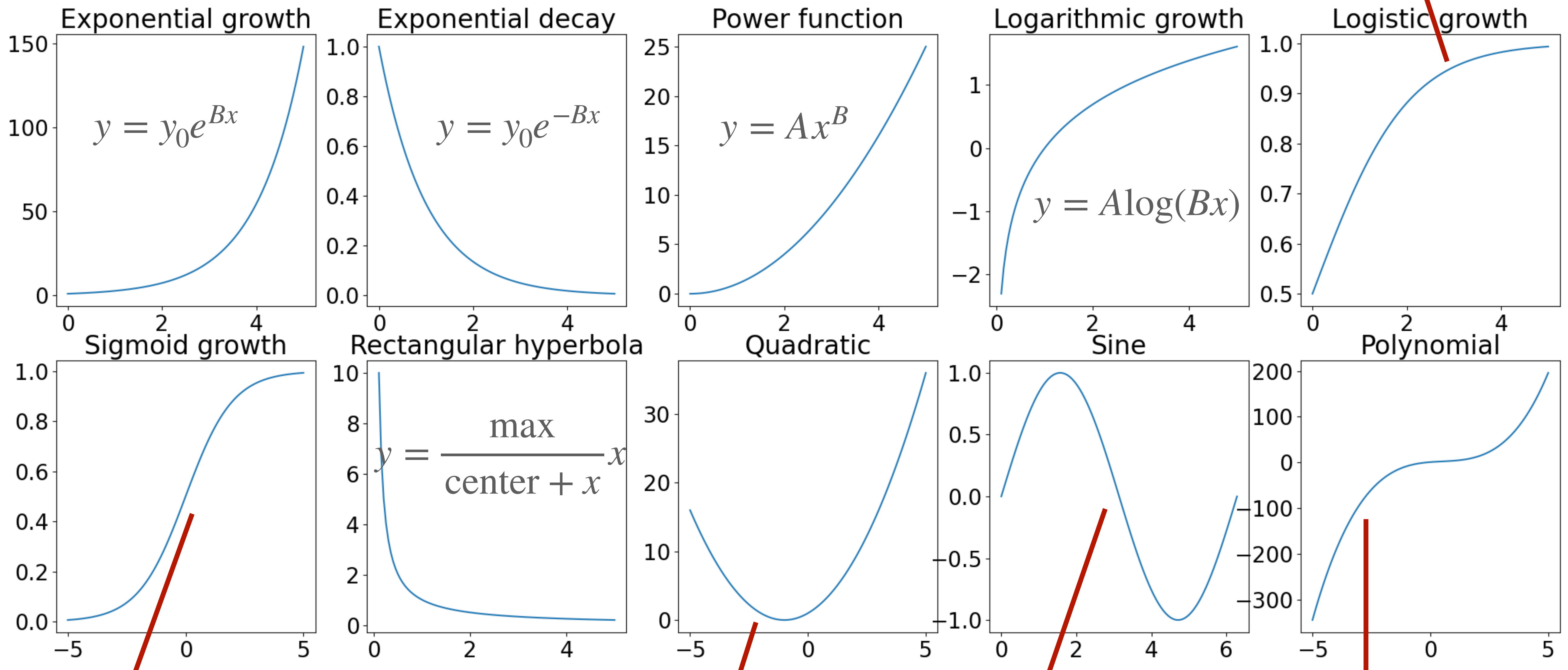
Now the reaction parameters can be interpreted as the slope and the intercept.



- Discuss why the fitted parameters are different in both approaches.

Some interesting functions

$$y = \frac{K}{\left[1 + \left(\frac{K}{y_0} - 1\right) e^{-Rx}\right]}$$



$$y = y_{\min} + \frac{y_{\max} - y_{\min}}{1 + e^{\text{slope}(\text{center} - x)}}$$

$$y = Ax^2 + Bx + C$$

$$y = A + Bx + Cx^2 + Dx^3 + \ldots$$

$$y = y_{\min} + [y_{\max} - y_{\min}]\sin(kx)$$

Exercise 4: Yoda's law

Kyoji Yoda in his 1963 paper showed that the dry weight (w) of certain plants is related to their population within an unit area (i.e. their population density, P) according to $w = AP^B$. Yoda noticed that as the plant weight increased, population density decreased. The population thinned (self-thinning) was rationalized as due to competition for resources. Larger plants were always found sparsely distributed. Some plants even died out for lack of resources as they grew larger. Yoda's self-thinning law could be a reason why the Earth is not entirely populated by weeds. For the data given in *datasets/YodasLaw.csv* estimate the constnts A and B.

Exercise 5: Gompertz function

The age (in days) and weight (in grams) of a bird is given in *datasets/BirdWeight.csv*. Plot this data, and fit the data to the Gompertz function $w = Ae^{-Be^{-Ct}}$ and determine the parameters A, B, and C.

Exercise 6: Fly population

The population of a colony of flies follows the growth function $P = 200/(1 + 7.12e^{-0.21t})$, where P is the number of flies and t is in days.

1. What is the maximum number of flies one can expect in this colony?
2. What was the initial population of the colony?
3. On day 10, what is the estimated population?
4. By what time (in days) one can expect the population to reach 128?

Assignment 1 (due 8 Feb 2024)

Basal metabolic rate (BMR) which is the amount of energy per unit of time required by an organism for biological functioning at rest, and the body weight (w) of 265 mammals are collected in the Appendix of the paper:

M. A. Elgar and P. H. Harvey, "*Basal Metabolic Rates in Mammals: Allometry, Phylogeny and Ecology*" Functional Ecology, Vol. 1, No. 1 (1987), pp. 25-36 DOI: https://doi.org/10.2307/2389354

1. Make a csv file of this dataset.
2. Perform statistical analysis of the dataset and discuss their significance.
3. Fit this data to the allometric scaling law $\mathrm{BMR} = Aw^B$ and determine A and B.
4. Present the original data and the fitted function in graphical form.
5. Repeat the fitting separately for each habitat type (terrestrial, aquatic, arboreal). How different are these models from the global model for all species?

You must email (to ramakrishnan@tifrh.res.in) one zip file with all the necessary files.

Differential equations

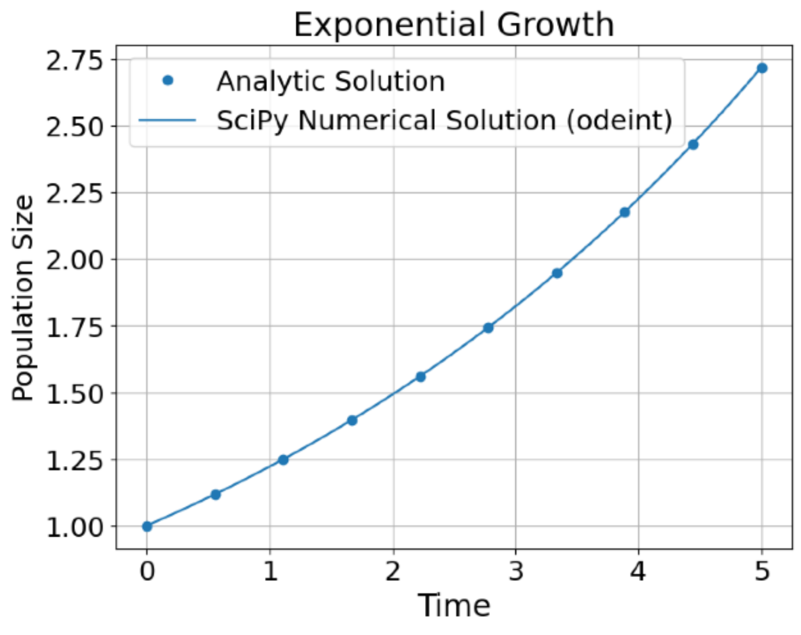
Growth/Decay functions are solutions of differential equations

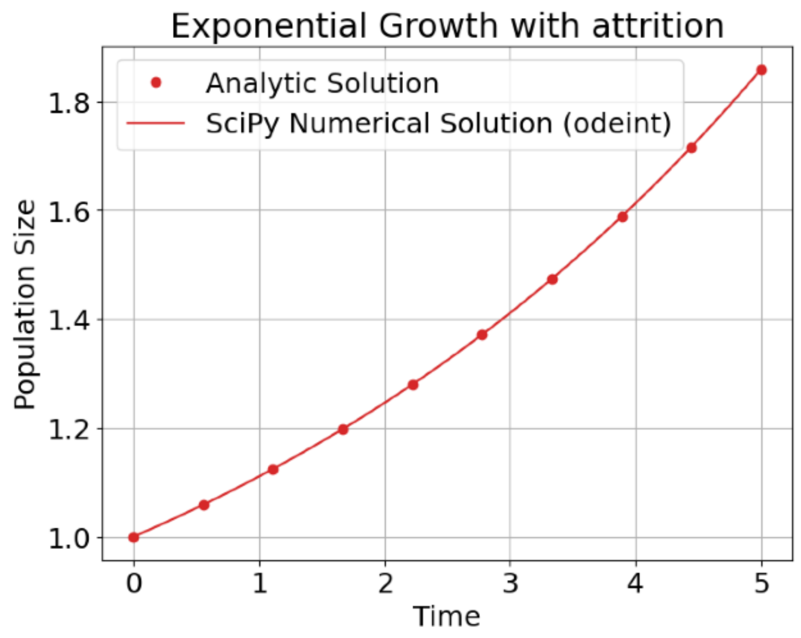
- Consider the exponential growth function $P(t) = P_0 e^{At}$. By substituting $t = 0$, we see that at the initial time, the population is P_0. If the growth rate is positive, $A > 0$, the population will increase exponentially from the initial value. Can the same information be presented in a different form? Yes, in a differential form as follows
$\frac{dP(t)}{dt} = AP(t)$. This equation (the growth equation in the differential form, or the growth differential equation) states that the rate of increase in the population is proportional to the population at that time. In other words, $P(t) = P_0 e^{At}$ is the solution of the differential equation $\frac{dP(t)}{dt} = AP(t)$.
- In some problems, it is easy to define (or set up) a differential equation defining the growth and we can determine the solution numerically using the module `scipy` and the procedure `odeint`. See In *notebooks/PyLifeS04_DifferentialEquations*

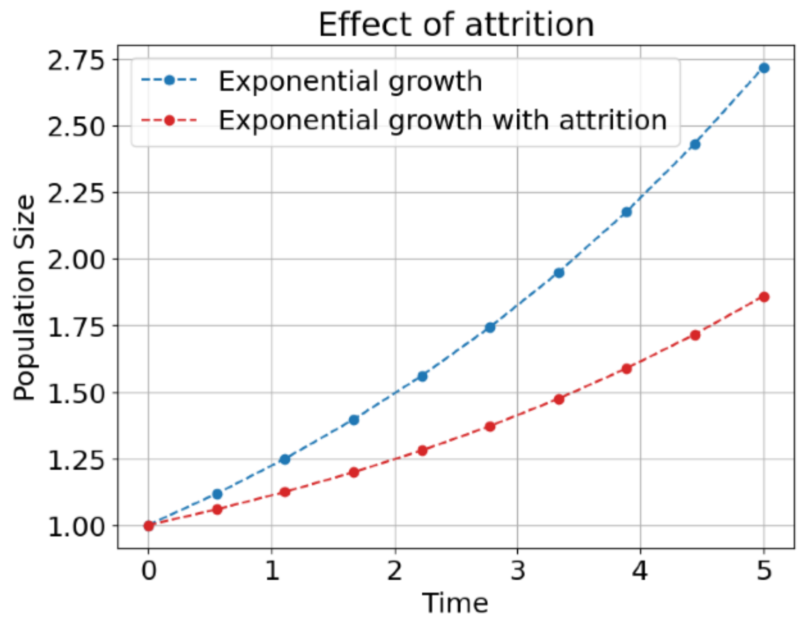
Growth with attrition

- Suppose the population of a species grows exponentially but a certain number of members of the population die at a constant death rate (*e.g.* 3 members in a minute). Then, we need to include a parameter that accounts for the constant decline in the population. The differential equation for this growth process is $\frac{dP(t)}{dt} = AP(t) - B$, where B is the attrition parameter and the negative sign implies that the population decreases.
- The exact solution is $P(t) = (P_0 - B/A)e^{At} + B/A$.
- In *notebooks/PyLifeS04_DifferentialEquations* we compare the analytic solution with the numerical solution calculated using `scipy`.

- The numerical solution obtained using scipy agrees with the closed form analytic solution for both growth models.

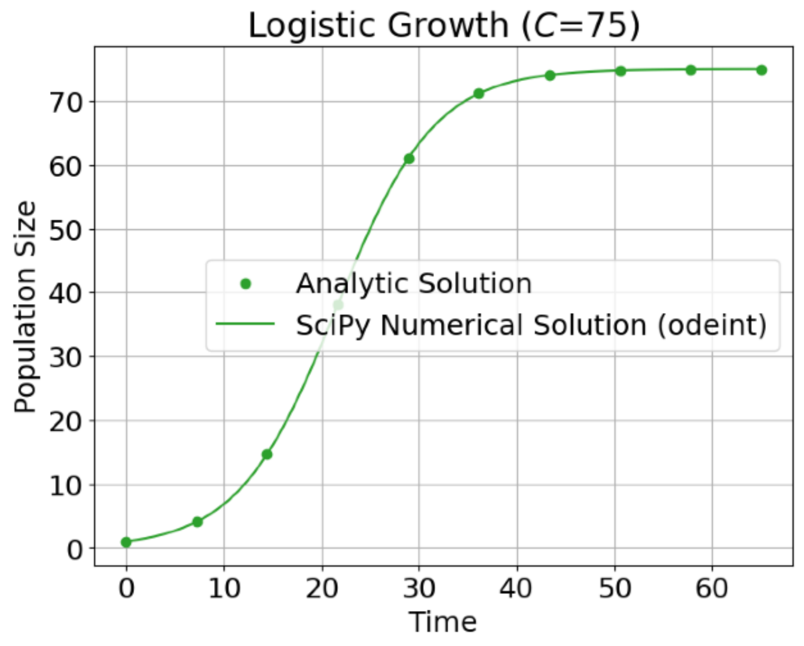






Logistic growth

- Suppose the population of a species grows exponentially at the beginning but the resources in the environment is limited, then the corresponding growth differential equation is $\frac{dP(t)}{dt} = AP(t)\left[1 - \frac{P}{C}\right]$, where C is the carrying capacity of the environment (maximum population size that the environment can support). The exact solution is $P(t) = \frac{C}{1 + \frac{C - P_0}{P_0}e^{-At}}$. For large values of t, e^{-At} approaches zero, hence the population will approach C. In *notebooks/PyLifeS04_DifferentialEquations* we compare the analytic solution with the numerical solution calculated using `scipy`.



Exercise 7: Tumor growth

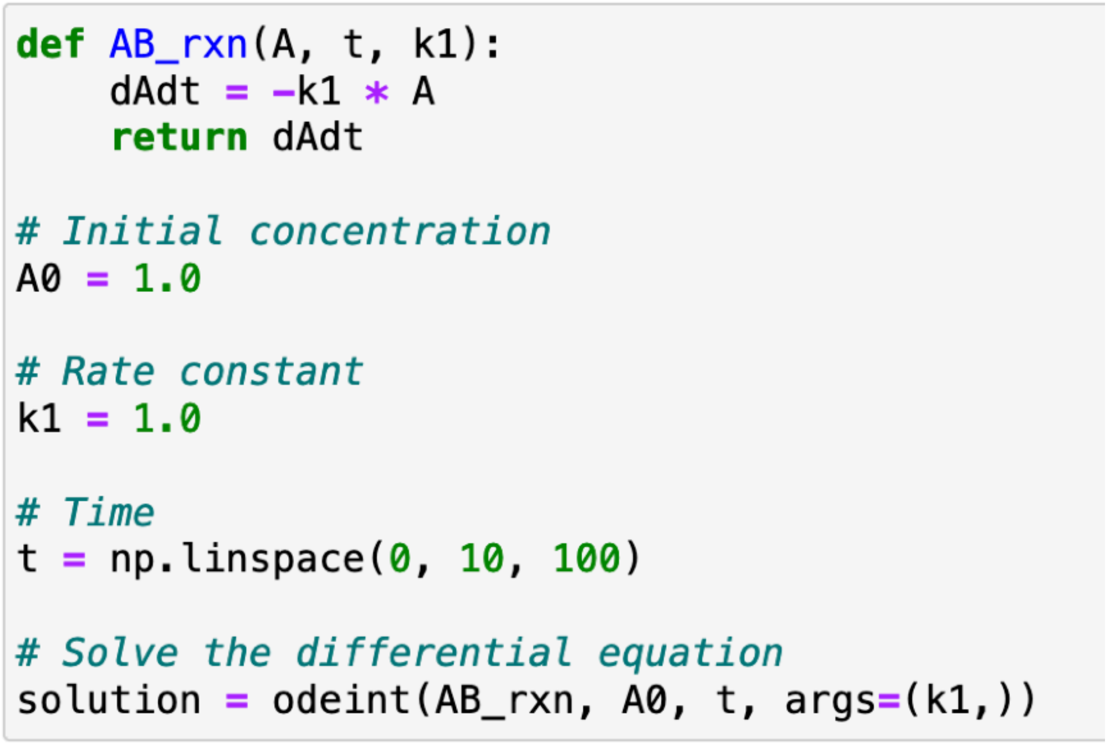
The growth of a tumor (increase in cell count) follows the differential equation $\frac{dP(t)}{dt} = AP(t)\log\left(\frac{C}{P(t)}\right)$, where C is the carrying capacity of the environment (maximum population size that the environment can support). For an initial population, $P_0 = 10$, growth rate $A = 1$, and the carrying capacity $C = 75$, calculate and plot the solution of this differential equation. Does the solution grow faster or slower than the logistic growth with the same parameters?

Systems of coupled differential equations

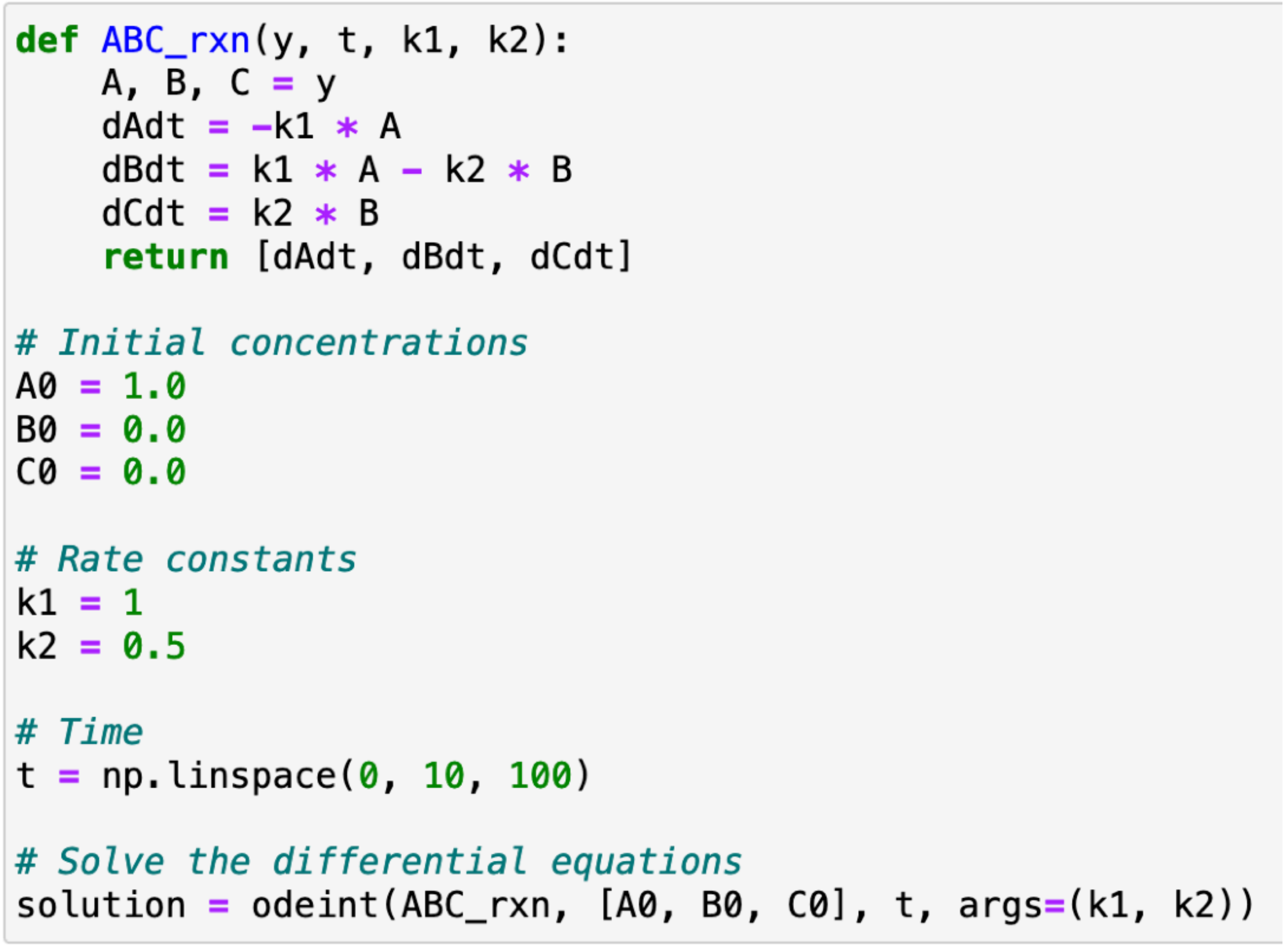
- One of the strengths of the `scipy.odeint` module is that the solution, which is $P(t)$ in the above examples can be a vector $[P_1(t), P_2(t), \ldots, P_n(t)]$. Hence, which very minimal change in the program, one can solve coupled differential equations.
- Suppose we want to study how the concentration of the reactant decreases with time in the reaction, $A \xrightarrow{k} B$, where k is the rate constant, then we have solve the differential equation $\frac{d[A(t)]}{dt} = -k[A(t)]$. Given the initial concentration, $[A_0]$, we can use `scipy.odeint` as in the previous examples to find the solution which will agree with the analytic solution $[A(t)] = [A_0]e^{-kt}$.
- Suppose, we have a 2-step reaction, such as $A \xrightarrow{k_1} B \xrightarrow{k_2} C$. Then how can we proceed? This reaction is described by three differential equations that are dependent on one another.

$$\frac{dA(t)}{dt} = -k_1A(t),\quad \frac{dB(t)}{dt} = k_1A(t) - k_2B(t), \text{ and } \quad \frac{dC(t)}{dt} = k_2B(t)$$

Comparison of codes for A-to-B and A-to-B-to-C reaction models



```python
def AB_rxn(A, t, k1):
    dAdt = -k1 * A
    return dAdt

# Initial concentration
A0 = 1.0

# Rate constant
k1 = 1.0

# Time
t = np.linspace(0, 10, 100)

# Solve the differential equation
solution = odeint(AB_rxn, A0, t, args=(k1,))
```

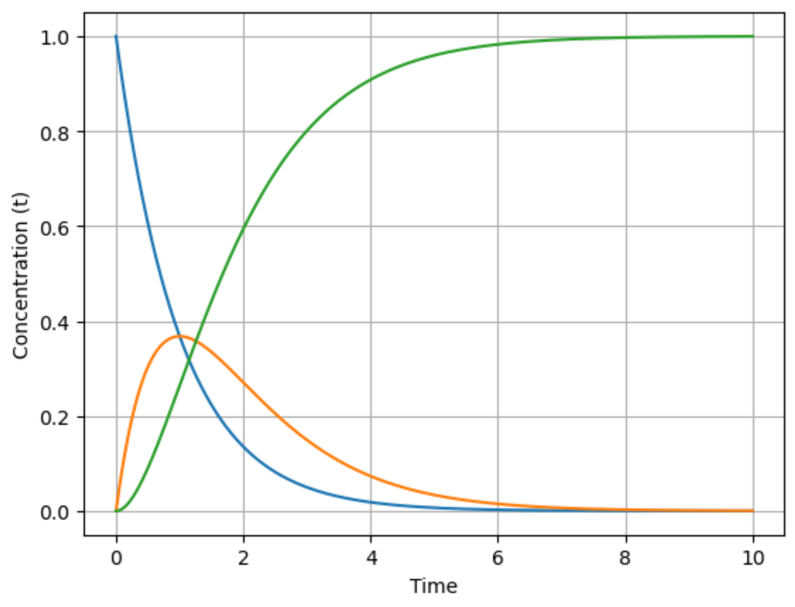


```python
def ABC_rxn(y, t, k1, k2):
    A, B, C = y
    dAdt = -k1 * A
    dBdt = k1 * A - k2 * B
    dCdt = k2 * B
    return [dAdt, dBdt, dCdt]

# Initial concentrations
A0 = 1.0
B0 = 0.0
C0 = 0.0

# Rate constants
k1 = 1
k2 = 0.5

# Time
t = np.linspace(0, 10, 100)

# Solve the differential equations
solution = odeint(ABC_rxn, [A0, B0, C0], t, args=(k1, k2))
```

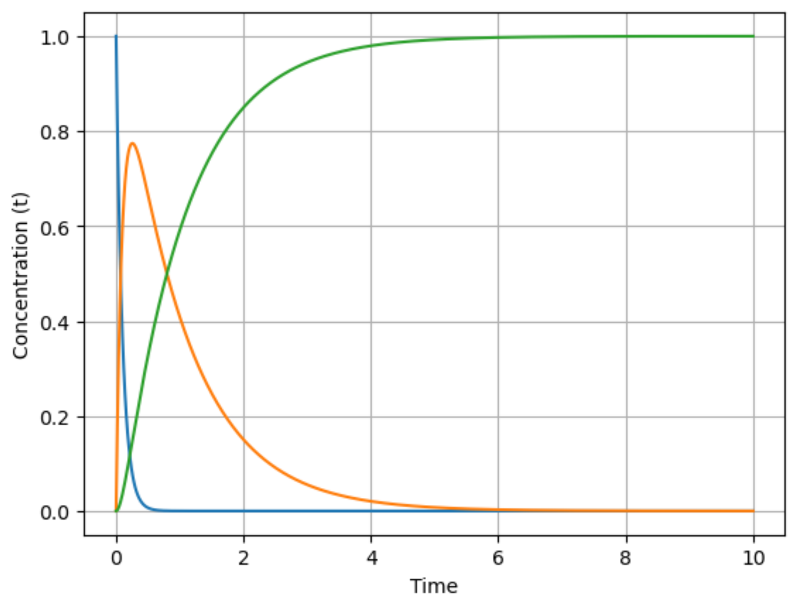
- In *notebooks/PyLifeS05_SystemsOfDifferentialEquations* one can inspect the complete code.
- For the A-to-B reaction, we pass a scalar input (A) and get a scalar output (dAdt). For the 2-step reaction, the input is a vector (y = [A, B, C]) and the output is also a vector [dAdt, dBdt, dCdt].
- So, for a new problem, the only programming effort is to define the initial conditions (concentrations or populations) and the rate constant.

Comparison of codes for A-to-B and A-to-B-to-C reaction models

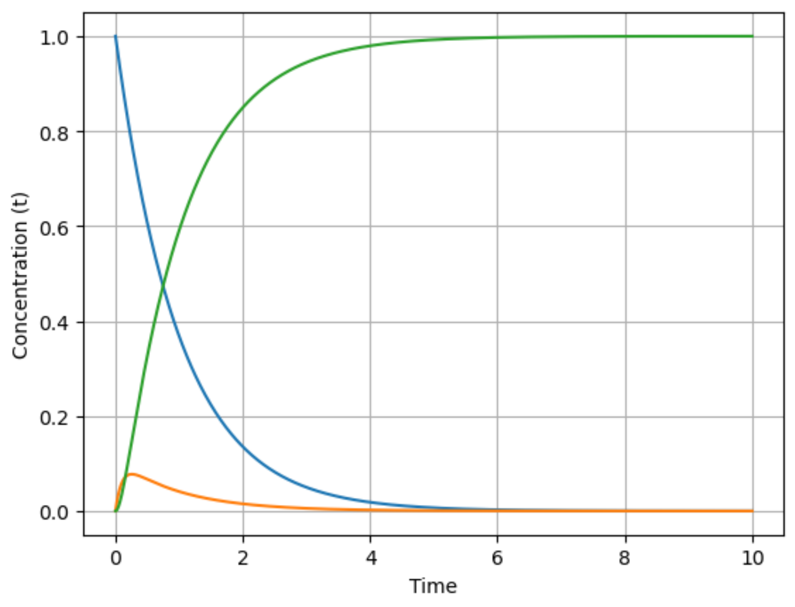
$k_1 = k_2$



$k_1 >> k_2$



$k_1 << k_2$



Exercise 8: Simple infection model for interacting population

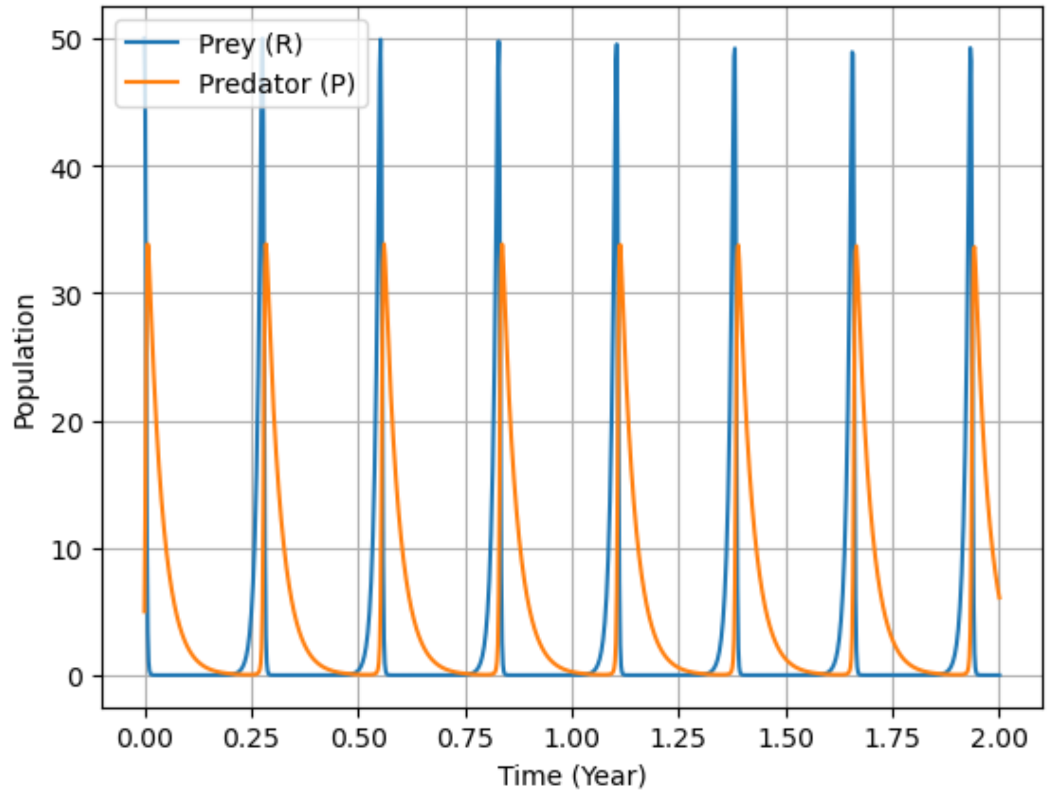
Suppose in a gated community, we divide the population into 2 classes: those infected by a virus (I) and those not infected (N) by the virus. The total population is a constant, $T = I(t) + N(t)$. Rate of increase in the number of infections is given by $\frac{dI(t)}{dt} = \alpha I(t)N(t) - \beta I(t)$, where α is the rate of contact between healthy and infected people, and β is the recovery rate of infected people. Assume that at initial time, there were 10 infected individuals, and solve this equation and plot $I(t)$ and $N(t)$ for 4 weeks using $T = 100$, $\alpha = 0.05$ (i.e. 5 infections in 100 contacts in a week), and $\beta = 4.0$ (i.e. 4 recover in a week' time after infection).

Lotka-Volterra equations for Predator-Prey dynamics

- The Lotka-Volterra model for predator-prey dynamics is a set of differential equations that describe the interactions between two species in an ecosystem, typically a prey species (denoted as R) and a predator species (denoted as P).
- The equations governing this process are as follows:

$$\frac{dR(t)}{dt} = rR(t) - aR(t)P(t) \quad \text{and.} \quad \frac{dP(t)}{dt} = -sP(t) + bR(t)P(t)$$

- r is the reproduction rate of prey
  a is the rate of consumption of prey by predators (interaction)
  s is the death rate of predator
  b is the reproduction rate of predator (as an outcome of consumption of prey, interaction)

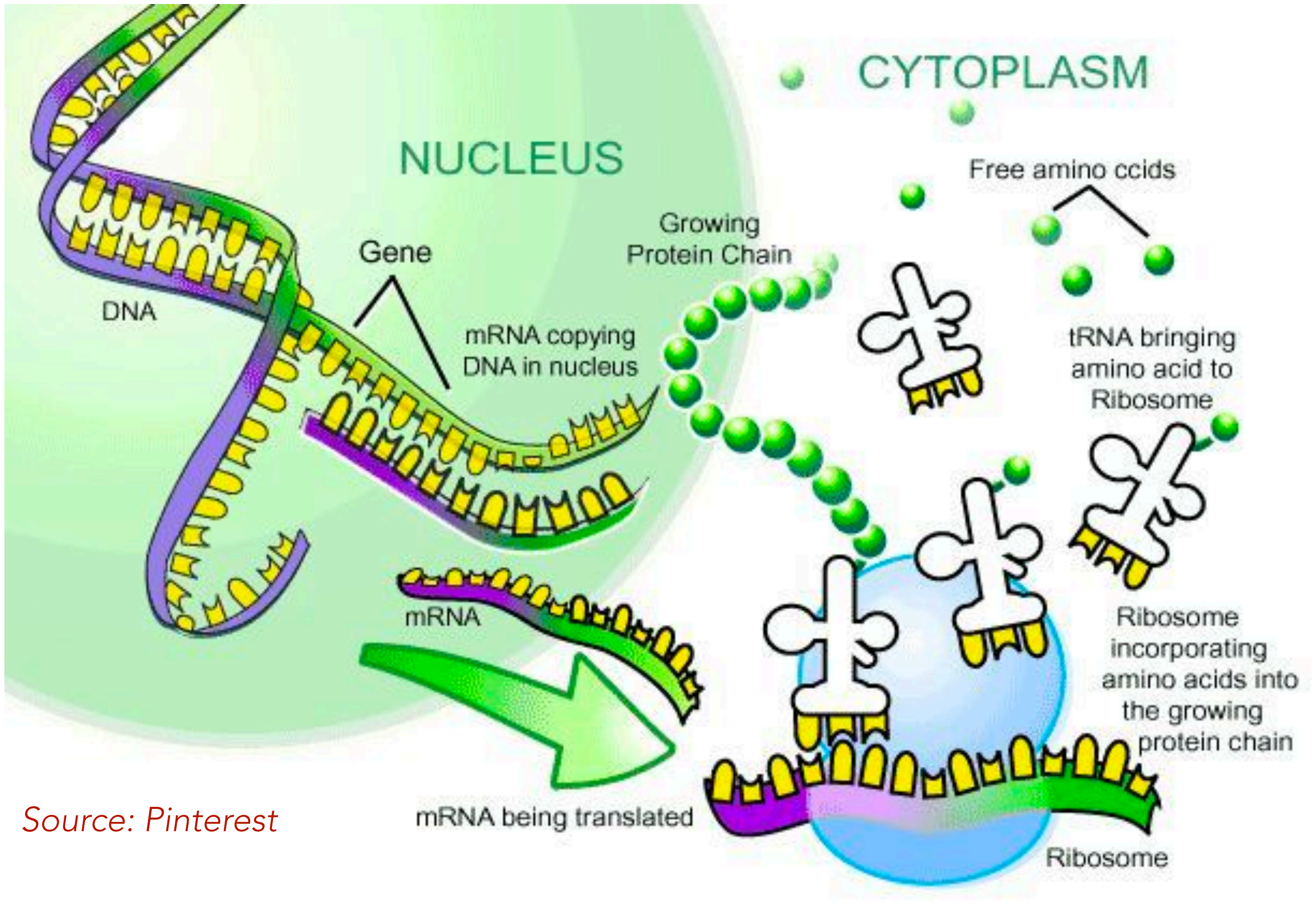


Exercise 9: A hypothetical model for students learning Python through peer-learning

Suppose in a class, there are $T = 30$ students, and let $N(t)$ denote the number of students learning python at a given time. Rate in which students learning Python is given by $\frac{dN(t)}{dt} = \alpha N(t)(T - N(t))$, where α is the rate of learning (due to contact between students who have learned Python, $N(t)$, and the students who are yet to learn Python). Using $\alpha = 0.01$, and $N_0 = 2$ (number of students who knew Python at the initial time), plot $N(t)$ and find out when more than half-of-the class would have learned Python.

Biopython

- Biopython is a very versatile Python module with many procedures and data applicable in bioinformatics.
- You should install biopython in your computer using `pip3 install biopython —user`
- As a simple example, we can see how one can declare a DNA sequence, transcribe it into an RNA sequence, and then translate it into a protein sequence using Biopython. This demonstrates the core processes involved in protein synthesis.


CYTOPLASM
NUCLEUS
Free amino ccids
Gene
Growing Protein Chain
DNA
mRNA copying DNA in nucleus
tRNA bringing amino acid to Ribosome
mRNA
Ribosome incorporating amino acids into the growing protein chain
mRNA being translated
Ribosome

Source: Pinterest

```
from Bio.Seq import Seq

# DNA sequence
dna_sequence = Seq("ATGGCTTAG")

# Transcription (DNA to RNA)
rna_sequence = dna_sequence.transcribe()
print("RNA sequence:", rna_sequence)

# Translation (RNA to protein)
protein_sequence = rna_sequence.translate()
print("Protein sequence:", protein_sequence)
```

- In *notebooks/PyLifeS06_Biopython_01* some basic functionalities of Biopython are demonstrated.