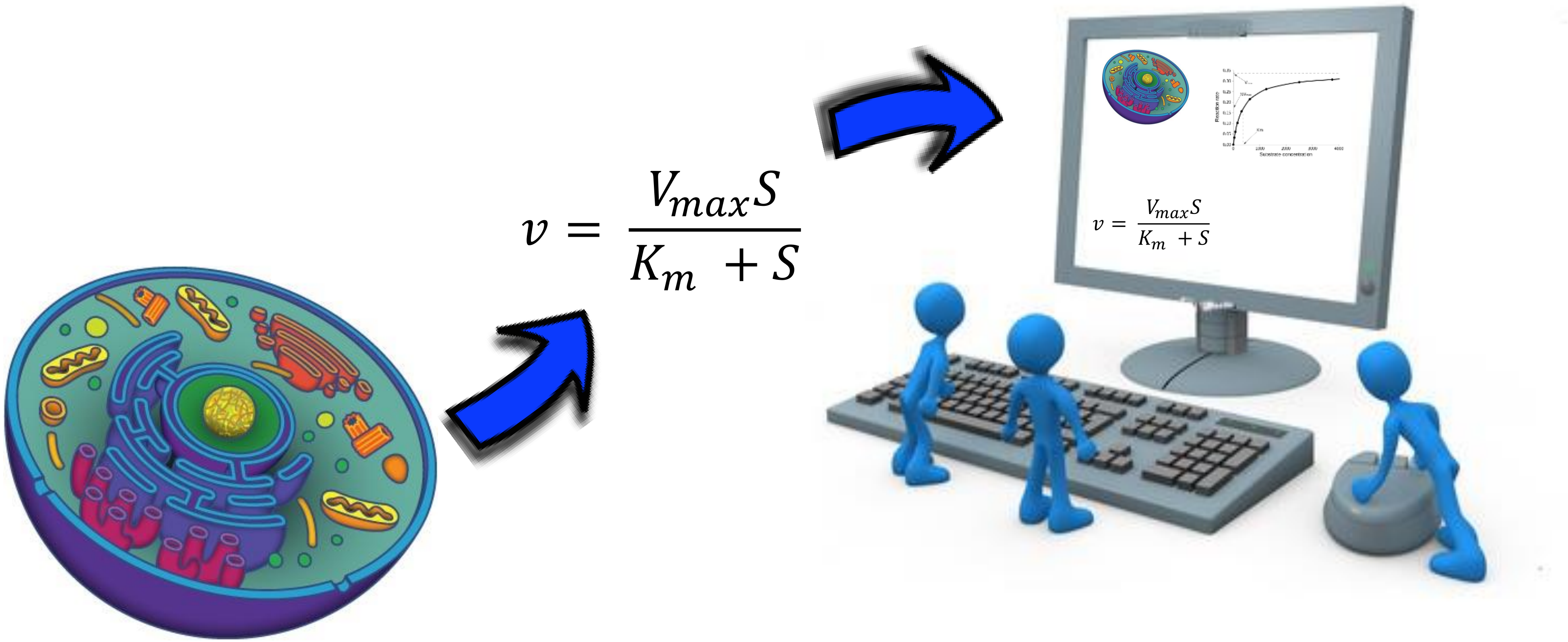
Institute of Quantitative and Theoretical Biology

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

M4453: Introduction to Molecular Systems Biotechnology

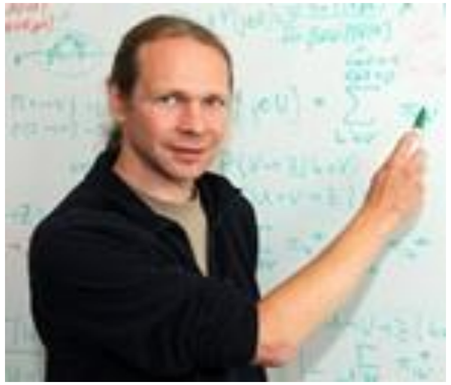


$$v = \frac{V_{max}S}{K_m + S}$$

Quantitative und Theoretische Biologie
Heinrich-Heine-Universität Düsseldorf
Universitätsstraße 1
Gebäude: 25.32
Etage/Raum: 03.26

Our Team

Institute for Quantitative and Theoretical Biology



Jun. Prof. Dr. Oliver Ebenhöh
(Head)

Antonella
(Post-Doc)
succurro@hhu.de

Ovidiu
(Post-Doc)

Simon
(System Admin.)

Mara Schuff
(Secretary)

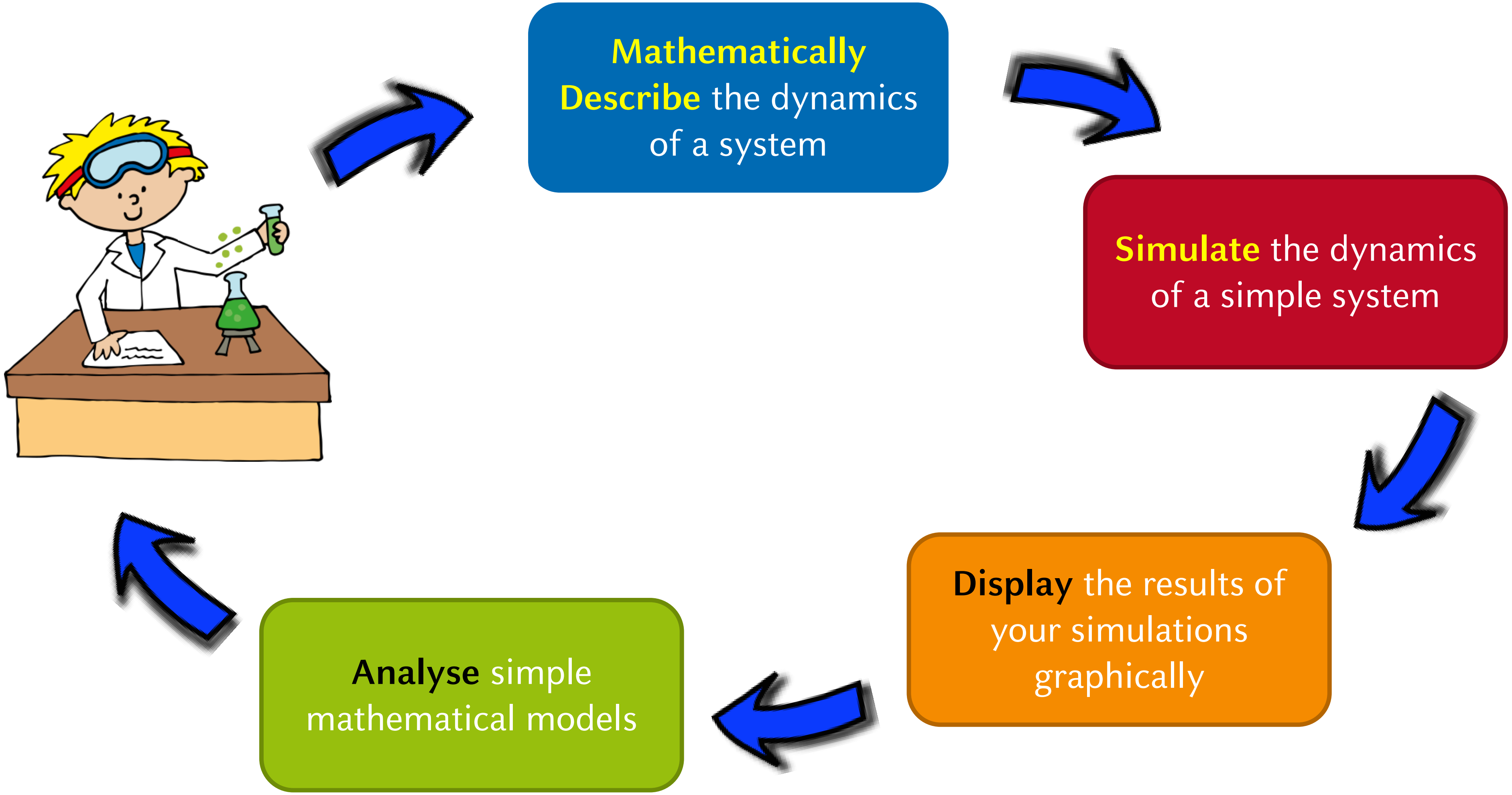
Dr. Steffi Spelberg
(Manager, AccliPhot)

Fiona
(PhD Student)

Anna
(PhD Student)
We built on her slides from last year

Suraj
(PhD Student)
sharma@hhu.de

Aim of the course



Mathematically Describe the dynamics of a system

Simulate the dynamics of a simple system

Display the results of your simulations graphically

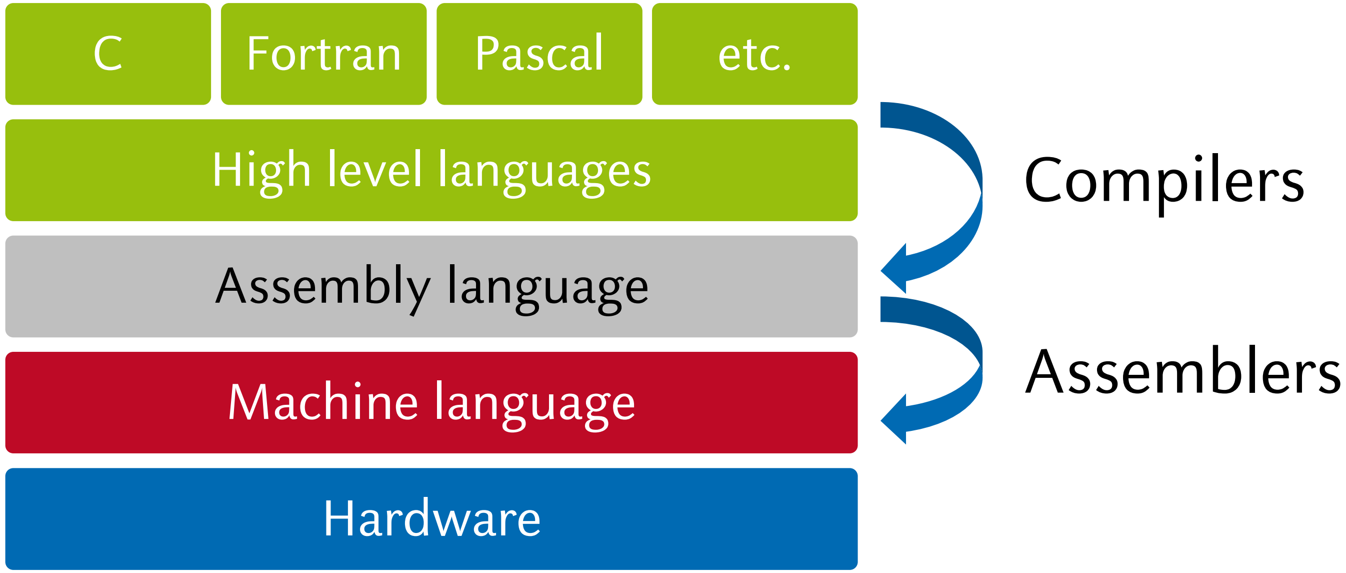
Analyse simple mathematical models

What is programming?

- **Programming** is a process of preparing an (a set of) instruction(s) to perform a particular task

What is programming?

- **Programming** is a process of preparing an (a set of) instruction(s) to perform a particular task



Programming platforms

Popular platforms within scientific community

MATLAB
The Language of Technical Computing

Wolfram
Mathematica

Why use Python?

Because, Python is

- Open source
- runs everywhere (Windows, Mac OS, Linux, Android, etc.)
- friendly and easy to use
- Modular
- Object oriented
- Supported by big community

Who uses Python?

Organizations using Python

- Web Development: Google, Yahoo
- Games: Battlefield 2, Civilization 4, Star Trek Bridge Commander
- Graphics: Walt Disney Feature Animation
- Software Development: Nokia, Red Hat
- Science: The National Research Council of Canada, Los Alamos National Laboratory Theoretical Physics Division, NASA
- Government: USA's Central Investigation Agency (CIA)

Source: https://wiki.python.org/moin/OrganizationsUsingPython

Characteristics of Python

- Indentation is important to explain the scope of the operation

  For example:

  - semicolons in C and MATLAB, or
  - braces in java like wise

- Indexing starts from 0 (only in MATLAB it starts from 1)

Good Programming Practices

1. Comments and/or function description
   - Code is for the machine and comment is for the user

2. Variable naming
   - should be self-explanatory
     - but, can also depend on the scope of its usage

Getting started with python programming

- Use Integrated Development Environments (IDEs) for code development
  - IDE is a software application that allows you to:
    - Write a code
    - Execute a code
    - Debug a code
    - Interpret a code

    Comprehensive facilities for software development
  - Available IDEs: Pycharm (https://www.jetbrains.com/pycharm/), Emacs, IDLE, etc.

- or, do it old-school way ☺
  - Script in any editor → Save your file as (.py) → run your code using the terminal

Installation

Installing python

- Step by step procedure
  - On Windows: http://docs.python-guide.org/en/latest/starting/install/win/
  - On MAC: http://docs.python-guide.org/en/latest/starting/install/osx/
  - On Linux: http://docs.python-guide.org/en/latest/starting/install/linux/

Installing Pycharm

- Step by step procedure
  - https://www.jetbrains.com/pycharm-edu/quickstart/installation.html

Useful sources

- Python Software Foundation:

https://www.python.org/

- How to Think Like a Computer Scientist:
http://openbookproject.net/thinkcs/python/english2e/

- Scientific Tool for Python:
http://wiki.scipy.org/SciPy

- 2D Plotting Library:
http://matplotlib.org/

- 46 Simple Python Exercises:
http://www.ling.gu.se/~lager/python_exercises.html

- http://pythonforbiologists.com/

Home work

Scared of programming? No reason ☺ !

Start playing:

https://studio.code.org/

PYTHON

Introduction to programming

Getting started

Semantics and Syntax

```
x = 5
```

We are assigning value 5 to a variable called x. This is called semantics.

Different Programming language -> different Syntax for the same semantics:

Python

```
x = 5
```

R

```
x <- 5
```

Pascal

```
x := 5;
```

Syntax is a set of rules that defines how a program should be written. It is a language-specific constraint on how we express semantics.

Object Oriented Programing in short

The concept

- The “object” is a data structure characterized by **attributes** (variables) and **methods** (functions)
- A class is a blueprint to “produce” many similar objects
- Objects are said to be instances of classes
- Instances of the same class differ in their attributes values
- The actions that an object can perform are defined by its methods

Python ingredients

1. Data Types:
   - Numbers
   - Strings
   - Lists
   - Tuples
   - Dictionaries
2. Variables
3. Operators
4. If Statement
5. For Loop
6. While Loop
7. Functions
8. Classes

Data types

- Python has five standard data types:
  - Number
  - String
  - List
  - Tuple
  - Dictionary

Strings

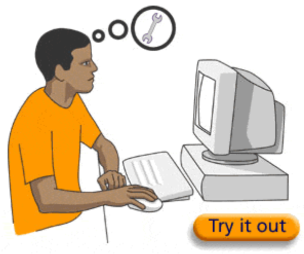
- A sequence of characters

```
>>> my_string = ‘Suraj’
>>> your_string = ‘Student’
```

- Any characters:

```
>>> try_that = ‘sUr@J_§HarMa!’
```

- Python offers you several built-in methods. Investigate what they do:
  - my_string.count(‘a')
  - my_string.find(‘a')
  - my_string.lower()
  - my_string.upper()
  - my_string.replace('a', 'b')
  - my_string.strip()



Lists

- The most basic data structure in Python is the **sequence**.
- Each element of a sequence is assigned a number - its position or index.
  Remember: The first index is 0, the second index is 1

```
>>> my_list = [1,2,3,4,5]
>>> my_list
[1, 2, 3, 4, 5]
>>> print(my_list)
[1, 2, 3, 4, 5]
>>> my_list[0]
1
>>> my_list[-1]
```

Lists

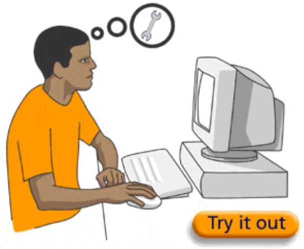
- Python offers set of built-in methods that can be applied to list. Find them using dir() function

```
>>> dir(my_list)
[… 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']
```

Tuples

- Tuples are almost identical to lists but in contrast to the latter, they cannot be altered/changed.

Try it.



```
>>> my_list = [1,2,3,4,5]
>>> my_tuple = (1,2,3,4,5)
```

Hint: You can either try to append it or change selected value

Dictionaries

- A dictionary is another type of a container that can store any number of Python objects.

```
>>> dictionary = {'English': 'dictionary', 'Deutsch':
'Wörterbuch', 'Polski': 'Słownik', 'Italiano': 'dizionario'}
>>> dictionary
{'Italiano': 'dizionario', 'English': 'dictionary', 'Polski':
'Słownik', 'Deutsch': 'Wörterbuch'}
```

- How is dictionary in Italian?

Variables

- Variables are nothing but reserved memory locations to store values.
- Python supports dynamic name resolution (late binding), which binds method and variable names during program execution
- Python interprets and declares variables only when they are set equal to …

```
>>> a = 5
>>> type(a)
int
>>> b = “class”
>>> a = b
>>> type(a)
str
```

Operators and statements

- Every programming language has a set of operators, so do python:
  - Assignment: =
  - Arithmetic: well known: + , -, *, /
    and quite new: %, **, //, ./
  - Comparison: >, <
  - Logical: and, in, or, not
  - Increment/decrement: +=, -=

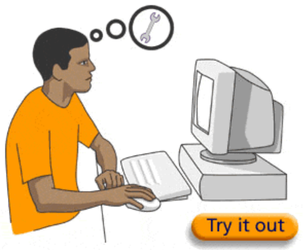
Exercise

1. Define a list containing the age of all of your family members.
2. Find the index of the oldest person. Store this value in a variable.
3. Find the index of the youngest person. Store this value in a variable.
4. Delete the youngest person from the list.
5. Add number 27 to your list.
6. Change the second number in the list to 14.
7. Revert the order of the list. Assign this new list to variable name.
8. Create new list with only last two elements of your original list.
9. Concatenate two sorted lists into a new list. Assign this new list to variable name.

If statement

- Syntax: colon(:) after the condition, action in the new indented line

```
if condition:
    do something


else:
    do something different
```



If statement

- Syntax: colon(:) after the condition, action in the new, indented line

```
if condition:
    do something

else:
    do something different
```

or

```
if condition:
    do something

elif other condition:
    do something else

else:
    do something different
```

If statement

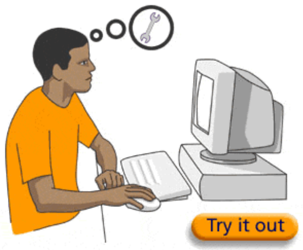
- Syntax: colon(:) after the condition, action in the new, indented line

```
if condition:
   do something

else:
   do something different
```

or

```
if condition:
   do something

elif other condition:
   do something else

else:
   do something different
```

- Write a command that will append the list with number two only if it has an odd length. Example: list = [2, 3, 4] should be appended

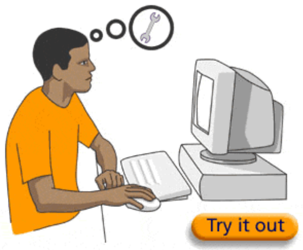


If statement

- Syntax: colon(:) after the condition, action in the new, indented line

- Write a command that will append the list with number two only if it has an odd length. Example: list = [2, 3, 4] should be appended

```
length = len(mylist)

if length%2 != 0:
    my_list.append(2)

else:
    my_list.append(1)
```



For loop

- If you wish to repeat some code certain number of times you will just loop through that code for desired number of times.
- Syntax: for how long:

___do something

For loop

- If you wish to repeat some code certain number of times you will just loop through that code for desired number of times.
- Syntax: for how long:

___do something

```
a = 0
for i in range(5):
        a = i + 1
        print(a)
```

For loop

- If you wish to repeat some code certain number of times you will just loop through that code for desired number of times.
- Syntax: for how long:

  ___do something

```
a = 0
for i in range(5):
        a = i + 1
        print(a)
```

```
fruits = ['banana', 'apple', 'mango']
for fruit in fruits:
    print('Current fruit :', fruit)
```

While loop

- You may also repeatedly execute a targeted statement as long as some given condition is true (you don’t know how many times)

- Syntax: while expression:

  ____action

```
count = 0
while (count < 9):
   print('The count is:', count)
   count += 1
```

While loop

- You may also repeatedly execute a targeted statement as long as some given condition is true (you don’t know how many times)

- Syntax: while expression:

  ____action

```
count = 0
while (count < 9):
   print('The count is:', count)
   count += 1
```

- Be careful: a loop becomes infinite if a condition never becomes false.

While loop

- You may also repeatedly execute a targeted statement as long as some given condition is true (you don’t know how many times)
- Syntax: while expression:

  ____action

```
count = 0
while (count < 9):
   print('The count is:', count)
   count += 1
```

- Be careful: a loop becomes infinite if a condition never becomes false.
- Try to write a loop that will never break.
  Now try to stop it.

Functions

- Any piece code that you think, you will use again then you should probably put it in a function.
- Syntax: def name_of_function(arguments_it_takes):

___action

```
def first_function():
   print(‘my first function’)

def second_function(a):
   a = a + 5
   return a
```

Functions

- Any piece code that you think, you will use again then you should probably put it in a function.
- Syntax: def name_of_function(arguments_it_takes):

  ___action

```
def first_function():
   print(‘my first function’)

def second_function(a):
   a = a + 5
   return a
```

- How to call your function?

Functions are always called by their name, followed with parenthesis and arguments inside. Example: `first_function(), second_function(7)`

Loops: exercises

1. Define a function to find all ages of your family members which are more than 20.
   1. Now change this code so the results will be stored in a list.
   2. Now select from this list only even numbers.

   Hint: Use combination of an if statement with a for/while loop
2. Define a function to:
   1. Define a list of names of your family members
   2. that returns the length of the longest name from the list.

Loops: exercises

1. Define a function to find all ages of your family members which are more than 20.
   1. Now change this code so the results will be stored in a list.
   2. Now select from this list only even numbers.

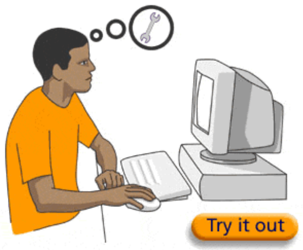
Hint: Use combination of an if statement with a for/while loop

```
agelist1 = [25, 15, 45, 19, 36] # list of ages
agelist2 = [23, 13, 45, 39, 56] # list of ages

newlist = [] # empty list

def findage(agelist):
   for i in agelist:
      if i > 20 and i%2 == 0:
         newlist.append(i)
      else:
         pass
   print (newlist)

findage(agelist1)
findage(agelist2)
```

Loops: exercises

2. Define a function to:
    1. Define a list of names of your family members
    2. that returns the length of the longest name from the list.

```python
namelist1 = ['Oliver', 'Antonella', 'Suraj'] # list of names

emptylist = []
def findlongestname(namelist):
   for i in namelist:
      emptylist.append(len(i))
   longestname = namelist[emptylist.index(max(emptylist))]
   print (longestname)

findlongestname(namelist1)
```



Modules and packages

- Python comes with a library of standard modules.
- Some modules are built into the interpreter; these provide access to operations that are not part of the core of the language but are nevertheless built in:
  - We have used so far for instance: len(), type(), dir()
- Packages are collection of modules.

Commonly used python packages

- Numpy (Numerical python): is a fundamental package for scientific computing. It contains a wide range functions that includes operations on n-dimensional arrays, linear algebra, random number capabilities, etc.
- Scipy (Scientific python): is more comprehensive package that allows functions like integration, image processing, etc.
- Matplotlib: is a python 2-D plotting library

How to use these packages while coding?

```
Example:

>>> import numpy
>>> numpy.arange(5)
```

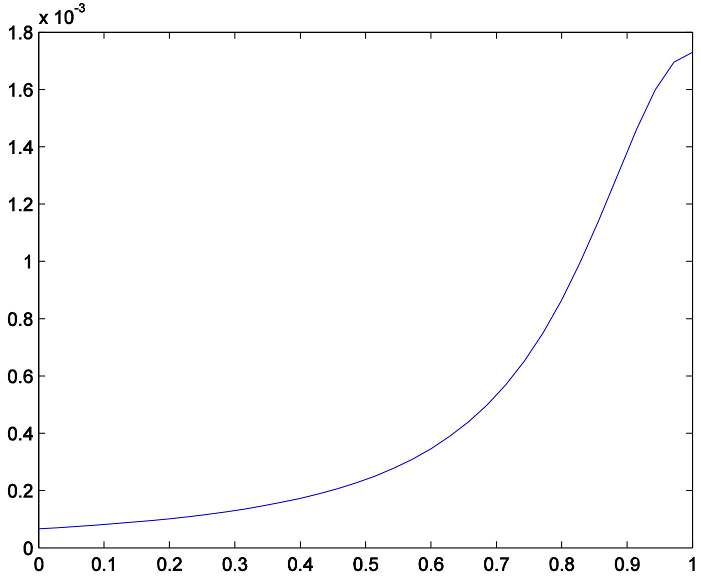
GRAPHICAL PACKAGE

You can display the results graphically

Using graphical package

How to represent my data

~~from pylab import *~~



Create a figure.

Create your data. We need X and Y values.

```
X = [1,2,3]          import numpy as np
X = range()          X = np.linspace()
                     X = np.arange()
```

Use the pylab package to plot your results.

```
import pylab as pl
pl.plot()
```

Show results of screen

```
pl.show()
```

Using graphical package

How to represent my data

1. Plot function f(x) = x.
2. Plot sine and cosine functions on the same plot.
3. Change the colours of the plot to red and blue.
4. Add legend, title and axes names.
5. Try to plot two plots next to each other.

scipy-lectures.github.io/intro/matplotlib/matplotlib.html

SIMULATING THE DYNAMICS OF A SYSTEM

Solving differential equations

First differential equation

Simple logistic growth equation

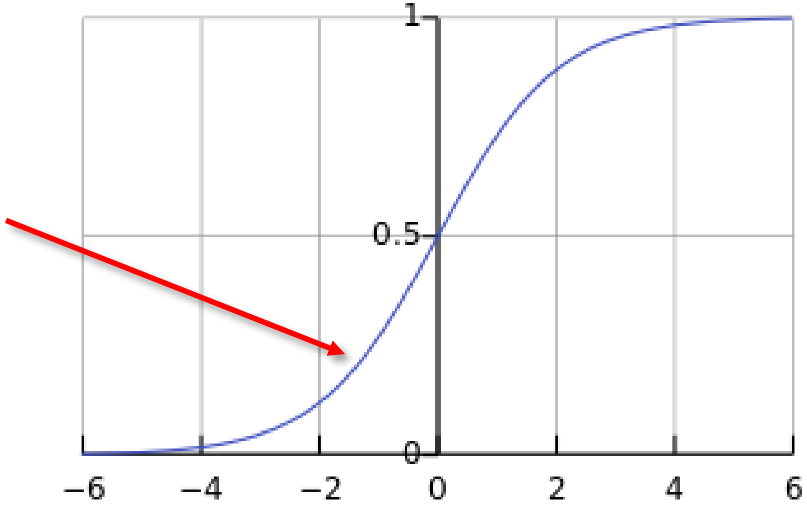
- Letting N represent population size and t represent time, this model is formalized by the differential equation:

$$\frac{dN}{dt} = rN(1 - \frac{N}{k})$$

N represents population size; r defines the growth rate; k is carrying capacity

- The function is a sigmoid curve.

The initial stage of growth is approximately exponential; as saturation begins, the growth slows, and at maturity, growth stops.



First differential equation

Tasks for today

- Goal: Your goal is to write two Python scripts to solve two one-dimensional problems: the logistic equation and the logistic equation with punishment for low population.

How to start:

- Mathematical description
- Initial conditions
- Parameter set
- Necessary tools for integration and plotting

Before we start

Import

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

We will be using functions that are not built in as a standard one. Therefore we need to import specific packages.

SOLVING DIFFERENTIAL EQUATIONS

You can simulate simple dynamic systems

First differential equation

Simple Logistic Growth Equation

```
def logistic(y, t, r, K):
        """ returns population  growth """
        dY = r * y[0] * (1 – y[0]/ float(K))
        return
```

Note that t is not used by the function. Why do we supply it as an argument?

First differential equation

Simple Logistic Growth Equation

```
def logistic(y, t, r, K):
        """ returns population  growth """
        dY = r * y[0] * (1 - y[0]/ float(K))
        return
```

Note that t is not used by the function. Why do we supply it as an argument?

Provide values of r and K and starting population size.

```
params = (0.3, 10)
Y = [1]
```

What is the type of the params value?

First differential equation

Simple Logistic Growth Equation: integration

```
growth = scipy.integrate.odeint(func, y0, t, args=(), …)
```

First differential equation

Simple Logistic Growth Equation: integration

```
growth = scipy.integrate.odeint(func, y0, t, args=(), …)
```

```
t = range(0,1000)
growth = scipy.integrate.odeint(logistic, y, t, args=params)

plt.plot(t, growth)
plt.show()
```

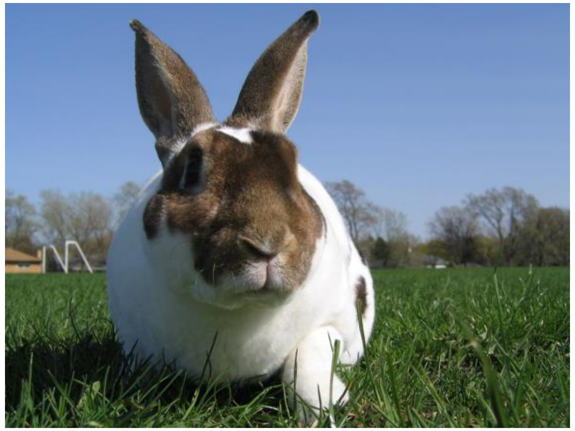
SOLVING COUPLED DIFFERENTIAL EQUATIONS

Lotka-Volterra Model

Coupled differential equations

Lotka-Volterra Model

- Lets consider population of two species that interact.
- One is a predator and second a prey.
- How to describe their dependent dynamics?



Coupled differential equations

Lotka-Volterra Model

- b is the natural growing rate of rabbits, when there are no wolfs
- d is the natural dying rate of rabbits, due to predation
- c is the natural dying rate of wolfs, when there are no rabbits
- f is the factor describing how many caught rabbits let create a new wolf

Coupled differential equations

Lotka-Volterra Model

- b is the natural growing rate of rabbits, when there are no wolfs
- d is the natural dying rate of rabbits, due to predation
- c is the natural dying rate of wolfs, when there are no rabbits
- f is the factor describing how many caught rabbits let create a new wolf

```
rabbits  →  dr/dt = b*r - d*r*w
  wolfs  →  dw/dt = -c*r + f*b*w*r
```

Coupled differential equations

Task

- Solve the Lotka-Volterra model (also known as the predator-prey) and create plots of the evolution of the population for following cases:
  - a. r = d = c = f = 1 for variety of initial conditions
  - b. r = 1, d = 0.1, c = 1.5, f = 0.75 and t = 1000, for R = 10 and W = 5
- What does it mean that the population size is stable over the time?
- Play with parameters and initial conditions so different species will survive.

Coupled differential equations

Lotka-Volterra Model

```
from numpy import *
import scipy.integrate
import matplotlib as plt

# Set parameters
a = 1.
b = 0.1
c = 1.5
d = 0.75
def dX_dt(X, t=0):
    """ Return the growth rate
    of fox and rabbit populations. """
      drdt = a*X[0] - b*X[0]*X[1]
      dwdt = -c*X[1] + d*b*X[0]*X[1]

    return array([drdt, dwdt])
```

Use X = [r, w] to describe the state of both populations

Coupled differential equations

Integration

```
X = [5, 10]
t = range(0,1000)
population = scipy.integrate.odeint(dX_dt, X, t)

plt.plot(t, population)
plt.show()
```

SOLVING DIFFERENTIAL EQUATIONS

Exercise: Enzyme Kinetics

Mathematical description

Enzyme kinetics

- Irreversible reactions

$$A \xrightarrow{v_1} B \xrightarrow{v_2} C$$

- Reversible reactions

$$A \underset{v_2}{\overset{v_1}{\rightleftarrows}} B \underset{v_4}{\overset{v_3}{\rightleftarrows}} C$$

$$\boxed{\frac{dy}{dt} = rate_{production} - rate_{consumption}}$$

- In simple case,

$$\boxed{\boldsymbol{rate(v) = k_i S}}$$

- In enzyme catalysed reactions, which is the common case in biological science, it is defined by Michaelis-Menten (MM) kinetics

$$\boxed{\boldsymbol{rate(v) = \frac{V_{max} S}{K_m + S}}}$$

where, $\boldsymbol{k_i}$ is the rate of catalysis/conversion , $\boldsymbol{V_{max}}$ is max catalytic velocity, $\boldsymbol{K_m}$ is MM constant and $\boldsymbol{S}$ is the substrate

Mathematical description

Enzyme kinetics

- Irreversible reactions

$$A \xrightarrow{v_1} B \xrightarrow{v_2} C$$

$$\boxed{\frac{dy}{dt} = rate_{production} - rate_{consumption}}$$

$$\frac{dA}{dt} = -v_1$$

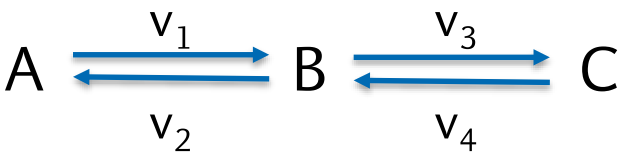
$$\frac{dB}{dt} = v_1 - v_2$$

$$\frac{dC}{dt} = v_2$$

Mathematical description

Enzyme kinetics

- Reversible reactions



$$\boxed{\frac{dy}{dt} = rate_{production} - rate_{consumption}}$$

$$\frac{dA}{dt} = -v_1 + v_2$$

$$\frac{dB}{dt} = v_1 - v_2 - v_3 + v_4$$

$$\frac{dC}{dt} = v_3 - v_4$$

Initial conditions & Parameters

Initial concentration of Metabolites

- Define initial concentration of the metabolites

Parameters

- Define parameter of the system

Tools for integration and plotting in python

Tools/packages for integration

```
>>> import scipy.integrate
>>> scipy.integrate.odeint(function, y0, t, args=(), …)
```

Tools/ packages for Plotting

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(X, Y)
```

SOLVING DIFFERENTIAL EQUATIONS

You can simulate simple dynamic systems

Mathematical description of a system

Description of enzyme kinetics

```
import numpy as np

def calculateconcentration(y, t, k1, k2):
    """
    returns concentration of metabolites over time
    y: the initial concentration of the metabolites(A, B)
    t: the different points
    k1: rate constant of A→B
    k2: rate constant of B→C
    """
    v1 = k1*y[0]
    v2 = k2*y[1]
    dAdt = -v1
    dBdt = v1 – v2
    dCdt = v2
    dydt = np.array([dAdt, dBdt, dCdt])
    return dydt
```

Note that ‘**t**’ is not used by the function, then why did we give it as an argument?

Mathematical description of a system

Integration

```python
import numpy as np
import scipy

def integratefunction(k1, k2):
        """
        returns the final concentration of metabolites after integration
        k1: rate constant of A→B
        k2: rate constant of B→C
        """
        t = np.linspace(0,100, 1000)
        y0 = np.array([100, 0, 0])
        yfinal = scipy.integrate.odeint(calculateconcentration, y0, t, args = (k1, k2))
        return t, yfinal
```

Visualize your results

Plot your integration

```python
import matplotlib.pyplot as plt

def plotresults(X, Y):
        """
        Plots of concentration curve of metabolites during integration
        X: variable values on x-axis
        Y: variable values on y-axis
        """
        plt.plot(X, Y)
        return plt.show()
```

Visualize your results

Plot your integration

```
import matplotlib.pyplot as plt

def plotresults(X, Y):
        """
        Plots of concentration curve of metabolites during integration
        X: variable values on x-axis
        Y: variable values on y-axis
        """
        for i in range(len(Y[0]):
                plt.plot(X, Y[:, i], label = ‘Y’+str(i))
        return plt.show()
```

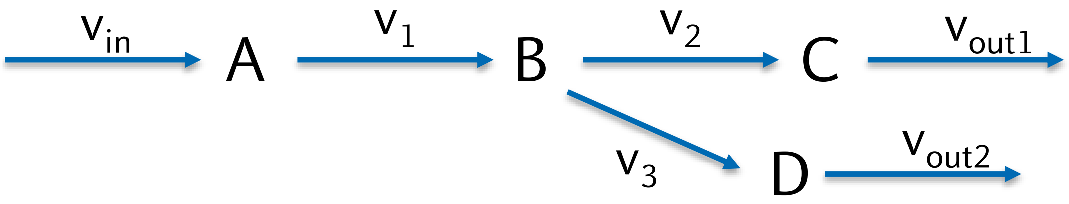
Your script all together

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

def calculateconcentration(y, t, k1, k2):
        ...
        return dydt

def integratefunction(k1, k2):
        ...
        return t, yfinal

def plotresults(X, Y):
        ...
        return plt.show()

t, yfinal = integratefunction(0.1, 0.2)
plotresults(t, yfinal)
```

Exercise

Task 1

- Use this network



For v_{out1} and v_{out2} the k_{out1}, k_{out2} = 1.0, 1.0 (constant)

- Now, use Michaelis-Menten equation to define the rate equation

$$\boxed{\boldsymbol{rate(v) = \frac{V_{max}S}{K_m + S}}}$$

- Vary v_{in} by keeping rest parameter values fixed in the system and integrate till steady-state is reached

- Plot the dynamics of steady-state concentrations of each metabolites in the system