

# Introduction to Python Programming



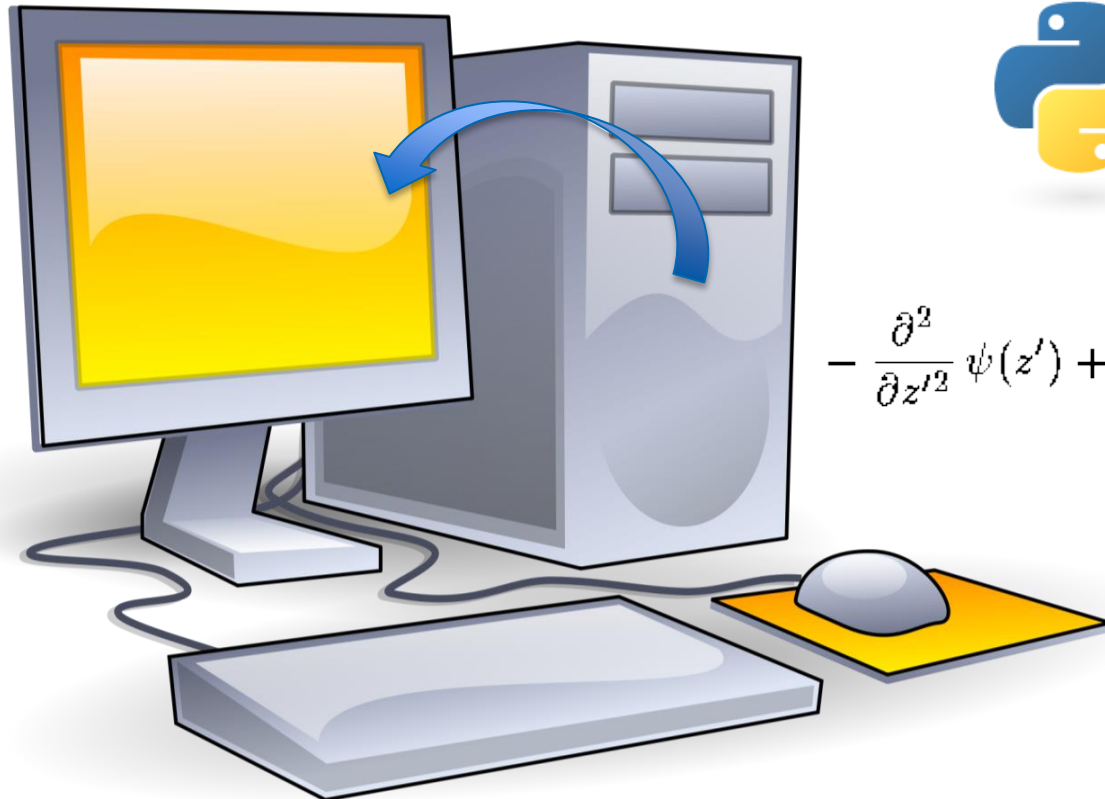
**Anna  
Matuszyńska**



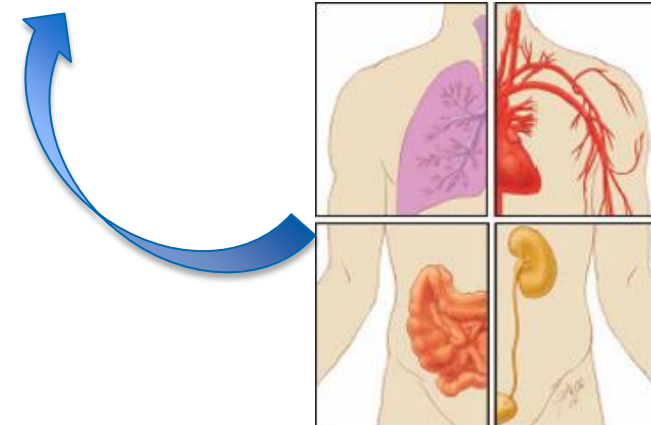
**Oliver Ebenhöf**  
oliver.ebenhoeh@hhu.de



**Ovidiu Popa**  
ovidiu.popa@hhu.de



$$-\frac{\partial^2}{\partial z'^2} \psi(z') + z' \psi(z') = E' \psi(z')$$



*Heinrich Heine*

HEINRICH HEINE  
UNIVERSITÄT DÜSSELDORF

## Learning outcomes

- You are familiar with simple mathematical models that describe the prototype of biological systems
- You can **describe** simple dynamic systems with equations
- You are able to **simulate** simple dynamic systems using Python language
- You can **display** the results graphically in different ways and interpret
- You are able to **analyze** simple mathematical models

You prove it with your final project and during the exam

## Programming is like...

- ... **cooking**: you prepare a list of variables (**ingredients**) and their data values (**amount**) and a list of stack manipulation instructions (**recipe**) so someone else can repeat it and obtain the same result (**delicious dish**)
- ... **training a dog**: A computer is a **simple** device that knows how to **remember** things and how to **look up** those memories. It starts out with a blank slate of memory and **relies** on a human to **tell** it what to do with that memory.

From the Outdoor Kitchen of: **WATSON SUPPLY**

### Blackened Sea Scallops Serves 4

20	Fresh Sea Scallops	<b>Directions</b> Heat iron skillet in oven, add oil. Dredge scallops in seasoning and sear each side 4-5 minutes or until done. Place scallops on plate and squeeze lemon juice, serve immediately.
¼ cup	Chef Paul Prudhomme's Blackened Redfish Magic	
3 Tbsp	Olive Oil	
2	Lemons, Halved	

QUALITY SOLUTIONS FOR *Outdoor Living*



- 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:

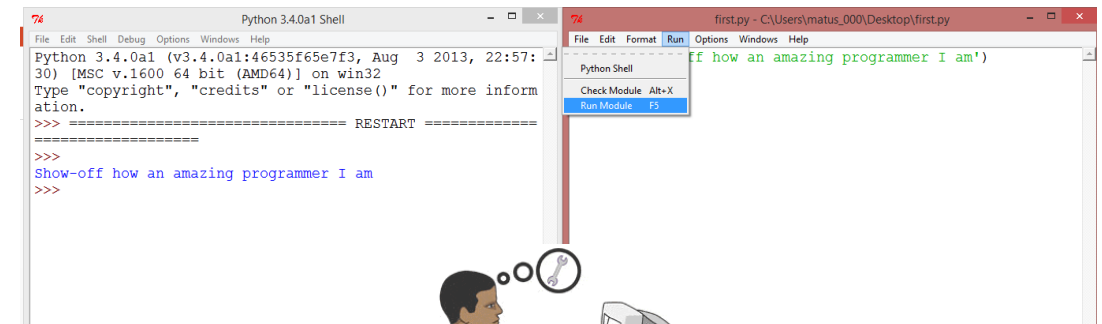
[http://www.ling.gu.se/~lager/python\\_exercises.html](http://www.ling.gu.se/~lager/python_exercises.html)

- <http://pythonforbiologists.com/>

## Editor and interpreter

- We will use the **Python IDLE**: the text editor for code execution
- **IDLE** is the Python IDE (integrated or interactive development environment).
- It is a software application that provides us comprehensive facilities for proper software development. It consists of:

- Source code editor
- Build automation tools
- Debugger
- Interactive interpreter (Python shell window)



# PYTHON

## Gentle introduction to programming

## Syntax and semantics

```
x = 5
```

- The meaning of this term is to assign value 5 to a variable called x. We call it **semantics**.
- Programming languages offer different ways to provide the same semantics:

Python

```
x = 5
```

R

```
x <- 5
```

Pascal

```
x := 5;
```

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

## All we need

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



## Comments

### Code Tells You **How**, Comments Tell You **Why**

- One line: # (hash) sign
- Multiple lines: 3 single ' ' ' or double " " " quotes before and after

Ask yourself: "What questions would be asked by someone looking at this code for the first time?" and write the comment so it includes the answer.

## Data types

- Python has five standard data types:
  - **Number**
  - String
  - List
  - Tuple
  - Dictionary
- For example: my age would be stored as a numeric value, my name as set of letters, and address as mixture of numbers and letters (alphanumeric characters)

## Strings

- A sequence of characters

```
>>> my_string = 'Anna'  
>>> your_string = 'Student'
```

- **Any** characters:

```
>>> try_that = 'a2na_matuszYnska!'
```

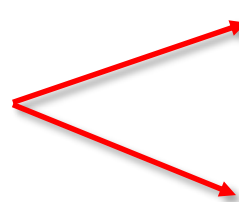
- Python offers you several built-in methods. Investigate what they do:
  - my\_string.count('a')
  - my\_string.find('n')
  - my\_string.lower()
  - my\_string.upper()
  - my\_string.replace('a', 'b')
  - my\_string.strip()



## 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

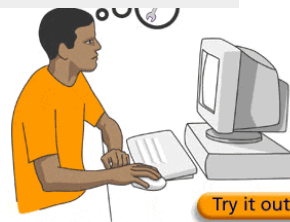
```
>>> list = [1,2,3,4,5]
>>> list
[1, 2, 3, 4, 5]
>>> print(list)
[1, 2, 3, 4, 5]
>>> list[0]
1
>>> list[-1]
```



## Lists

- We may wish to **count** number of characters in the list, **sort** it ascending, **delete** certain position or **append**.
- As for strings, Python offers set of built-in **methods** that can be applied to it. Find them using `dir()` function
- `dir(my_object)` is a provided method that attempt to return a list of valid **attributes** for that object.

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



Try it out

## Tuples

- Tuples are almost identical to lists but in contrary to the latter, they cannot be changed. Try it.

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



```
>>> list = [1,2,3,4,5]
>>> tuple = (1,2,3,4,5)
```

## Dictionaries

- A dictionary is mutable and is another type of a **container** that can store any number of Python objects.
- Dictionary in English: Wörterbuch in German

```
>>> 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

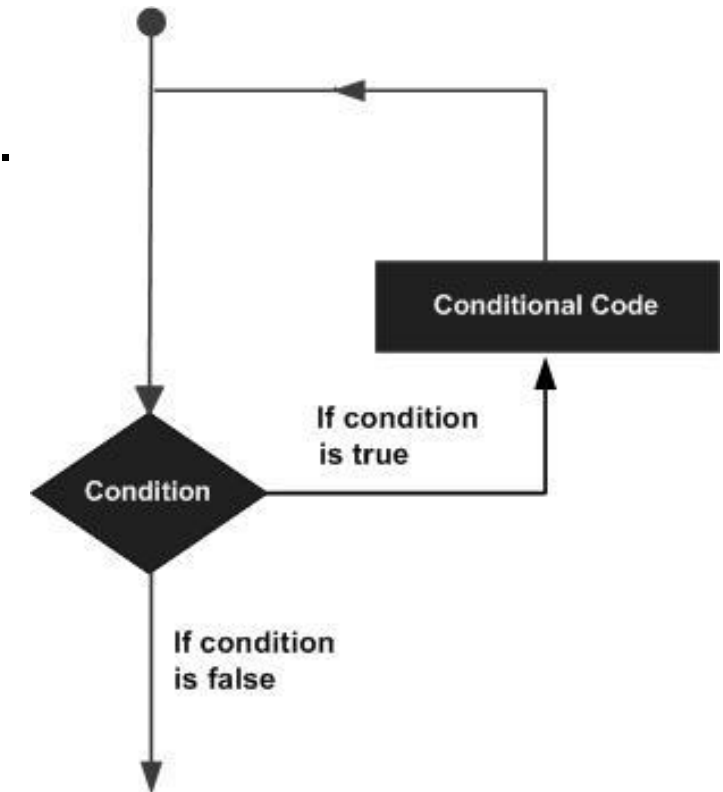
- With every programming language we have operators:
  - Assignment: =
  - Arithmetic: well known: + , - , \* , /  
and quite new: %, \*\*, //, ./
  - Comparison: >, <
  - Logical: and, in, or, not
  - Increment/decrement: +=, -=
- Python, among others, include:

If, for, while, try, class, def, with, pass, assert, yield, import, print



## If statement

- The heart of programming logic
- It is a **conditional** that when satisfied **activates** some part of code
- We use **if statement** all the time:  
If I pass it, I will come for a beer. If not, I won't come.  
If I will fail, my parents will kill me.  
If I will make all exercises I will learn.  
If I won't pay the bill, they will cut off my electricity.



## If statement

- **Syntax:** `:` colon after the condition, action in the new, **indented** line  
**if** for first condition,  
        action  
**elif** for second condition and more,  
**else** for the last condition

```
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



## 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:

\_\_\_\_\_action

- **How long** can be expressed in many ways.
- Examples:

“Increase variable a by 5, **for 5 times**”

“Print **all elements** of the list”

```
a = 4
for a in range(5):
    a = a + 5
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
```

- 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

- Tired of repeating pieces of code? Any code that you think you will ever use again, you should probably put in a function.
- **Syntax:** `def name_of_function(arguments it takes):`

        action

```
def my_first():  
    print('first function')  
  
def my_second(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:** `my_first()`, `my_second(7)`

# Exercises

## Lists: exercises

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





## Loops: exercises

1. Find all numbers dividable by three between 0 and 100.

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

**Hint:** There is built in method that will quickly find numbers that divided by 3 return no reminder

2. Now change this code so the results will be stored in a list.

**Hint:** You can declare an empty list called **results** before the loop.

3. Now select from this list only even numbers.



## Functions: exercises

1. Define a function that takes a list of numbers as an argument and returns the largest of them. Use the **if statement** that you have just learned.  
**Comment:** Python has a built in method called `max()`. Don't use it here.
2. Write a function that takes a number and returns a list of its digits.
3. Write a function that checks whether an element occurs in a list.
4. Define two functions that sums and multiplies respectively all the numbers in a list of numbers.  
**Example:** `my_sum([2, 3, 4])` should return 9
5. Define a function `find_longest_word()` that takes a list of words and returns the length of the longest one.

More: [http://www.ling.gu.se/~lager/python\\_exercises.html](http://www.ling.gu.se/~lager/python_exercises.html)

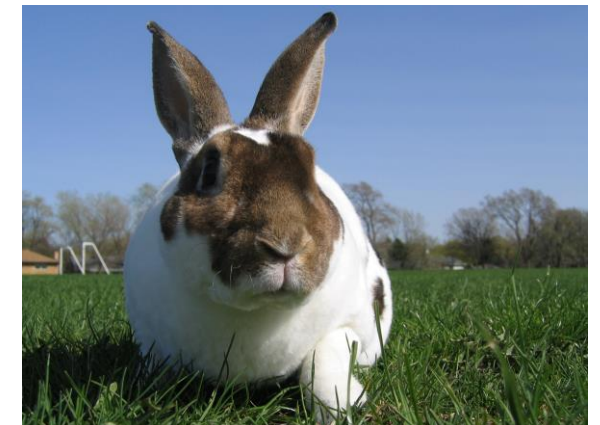
## Fibonacci Numbers

We will consider an **idealized biological system** published in Liber Abaci (1202) by Leonardo of Pisa, known as Fibonacci.

Puzzle: We have rabbit population and wish to know how many rabbits will we have after one year.

Assumptions: a newly born pair of rabbits, one male, one female, are put in a field;

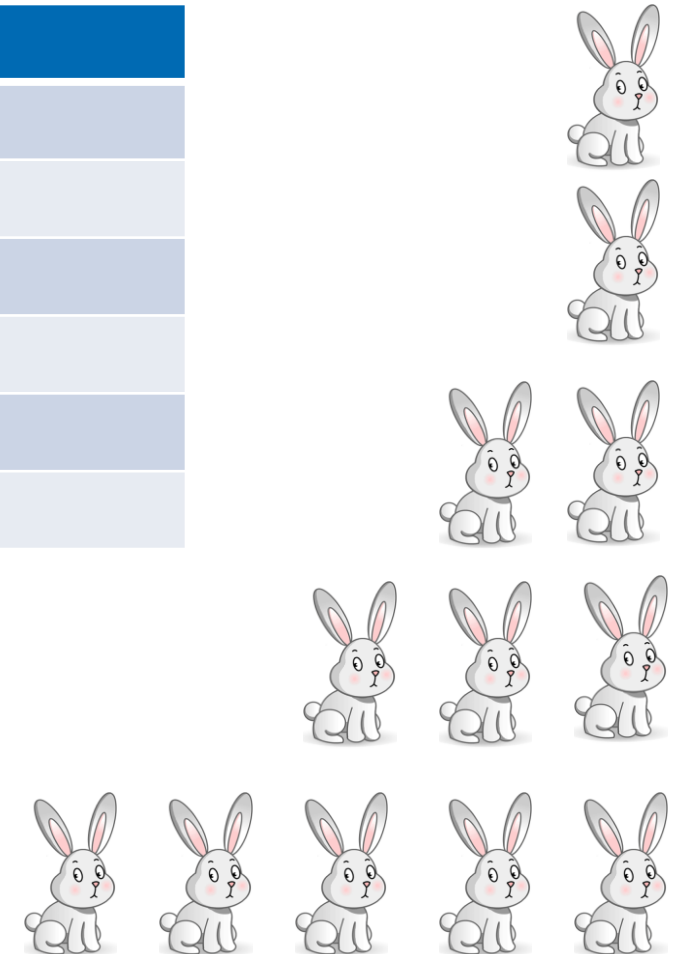
- rabbits are able to mate at the age of one month
- mating pair always produces one new pair
- rabbits never die



## Fibonacci Numbers

Month	Rabbit pairs
0	0
1	1
2	?
3	
4	
5	

$$F_n = F_{n-1} + F_{n-2},$$



## Recursion vs. iteration

- A **recursive** method is a method that calls itself either directly or indirectly

Approach to reduce problem to its simpler version:

1. A simple base case (or cases)
2. A set of rules that reduce all other cases toward the base case

```
if base:
    _____
else:
    _____
```

- An **iterative** method is the act of repeating a process with the aim of approaching a desired goal.

It is using the output from one iteration as the input to the next.

```
if _____:
    result = do input
    input = result } loop
```

### Sum Up:

Iterative Algorithms = Fast Performance but Hard to write  
Recursive Algorithms = Fast to write but Bad performance

## Code

```
def fib_rec(n):  
    """ assumes n an int >= 0  
    returns Fibonacci of n  
    """  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
    else:  
        return fib(n-1) + fib(n-2)  
  
def testFib(n):  
    for i in range(n+1):  
        print 'fib of', i, '=', fib_rec(i)
```

```
def fib_iter(n):  
    a,b = 0, 1  
    for i in range(0,n):  
        a,b = b, a+b  
    return a
```

## There is always another way

- Closed-form expression known as Binet's formula

$$F_n = \frac{1}{\sqrt{5}} \cdot \left( \frac{1 + \sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \cdot \left( \frac{1 - \sqrt{5}}{2} \right)^n .$$

```
from math import sqrt
```

```
def fib(n):
```

```
    F = ((1+sqrt(5))**n - (1-sqrt(5))**n) / (2**n * sqrt(5))
```

```
    return F
```

```
from math import sqrt
```

```
def fib(n): return ((1+sqrt(5))**n - (1-sqrt(5))**n) / (2**n * sqrt(5))
```

## 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()`
- Other modules you need to call in order to use their function:

```
from math import sqrt
```

- **Packages** are collection of modules. They provide a way of structuring Python's module namespace by using "dotted module names".

Over next days we will be using lots of such packages.



## Variations on the Fibonacci numbers: exercise

1. Define a function that instead of a particular Fibonacci number will return the list of the first n Fibonacci numbers.

**Example:** fib\_seq(5) should return 1, 1, 2, 3, 5

2. Define a function that will return Fibonacci numbers between given range.

**Example:** fib\_seq(3, 5) should return 2, 3, 5

```
def fib(n):  
    a, b = 0, 1  
    while a < n:  
        print a  
        a, b = b, a+b  
    print()
```

# GRAPHICAL PACKAGE

You can display the results graphically

## How to represent my data

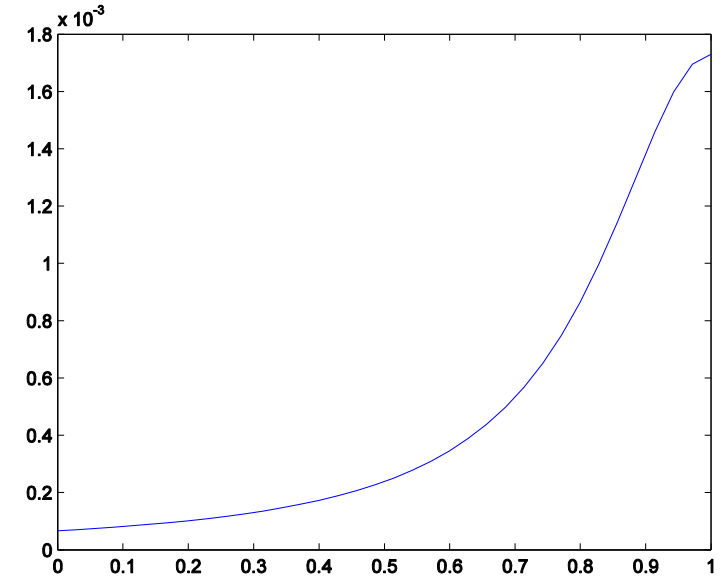
Create a figure.

Create your data. We need X and Y values.

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

## 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](https://scipy-lectures.github.io/intro/matplotlib/matplotlib.html)



## Collection of simple models in python

<https://gitlab.com/ebenhoehe/models-for-teaching/tree/master>