

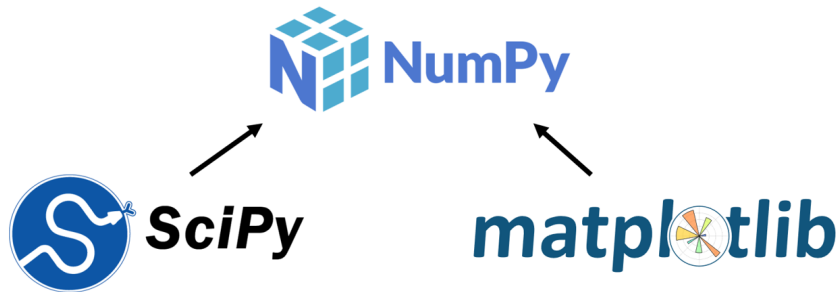
# Programming for Essential Digital Skills, Part 2

Pieter Kler

2025

## Chapter 9 - Mathematics and plotting

# Data science with Python



Mathematical tasks:

- Root finding
- Optimization
- Integration
- Linear algebra

Visualization and plotting:

- Function plotting
- Data visualization
- 3D-plotting

# SciPy: Scientific Computing with Python

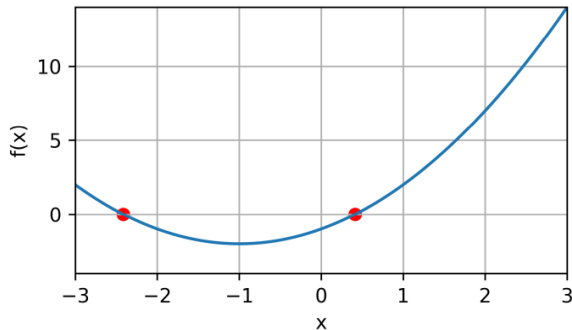
## Subpackages

SciPy is organized into subpackages covering different scientific computing domains. These are summarized in the following table:

Subpackage	Description
<code>cluster</code>	Clustering algorithms
<code>constants</code>	Physical and mathematical constants
<code>fft</code>	Discrete Fourier transforms
<code>fftpack</code>	Fast Fourier Transform routines (legacy)
<code>integrate</code>	Integration and ordinary differential equation solvers
<code>interpolate</code>	Interpolation and smoothing splines
<code>io</code>	Input and Output
<code>linalg</code>	Linear algebra
<code>ndimage</code>	N-dimensional image processing
<code>odr</code>	Orthogonal distance regression
<code>optimize</code>	Optimization and root-finding routines

# Root finding

Consider  $f(x) = x^2 + 2x - 1$ . A **root**  $x$  of the function  $f$  is a point that satisfies  $f(x) = 0$ .



Solving the equation  $f(x) = 0$  using `fsolve()`

## Solving the equation $f(x) = 0$ using `fsolve()`

```
# Import optimize package from SciPy

# Define f as Python function

# Use fsolve() to solve f(x) = 0 with initial guess

# Print the found root
```

Solving the equation  $f(x) = 0$  using `fsolve()`



## Solving the equation $f(x) = 0$ using `fsolve()`

```
import scipy.optimize as optimize

def f(x):
    return x**2 + 2*x - 1

guess = 3
f_zero = optimize.fsolve(f,guess)[0]

print("A root of the function f is given by", f_zero)
```

A root of the function f is given by 0.41421356237309503

## Solving the equation $f(x) = 3$

Suppose we want to solve  $f(x) = 3$ . **Question:** How to do this with root finding?

## Solving the equation $f(x) = 3$

Suppose we want to solve  $f(x) = 3$ . **Question:** How to do this with root finding?

- If we define  $g(x) = f(x) - 3$ , then  $g(x) = 0$  if and only if  $f(x) = 3$ .

## Solving the equation $f(x) = 3$

Suppose we want to solve  $f(x) = 3$ . **Question:** How to do this with root finding?

- If we define  $g(x) = f(x) - 3$ , then  $g(x) = 0$  if and only if  $f(x) = 3$ .

```
def g(x):  
    return f(x) - 3  
  
guess = 4  
f_zero = optimize.fsolve(g,guess)[0]  
  
print("A number x satisfying f(x) = 3, is given by", f_zero)
```

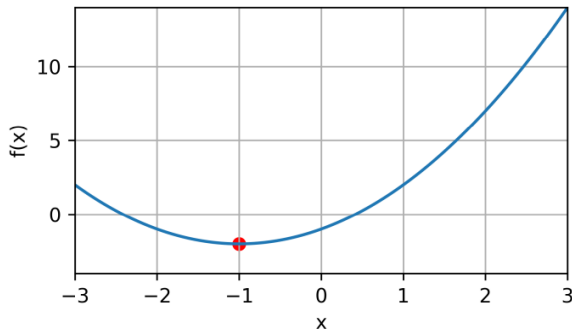
A number x satisfying  $f(x) = 3$ , is given by 1.2360679774998171

## Solving the equation $f(x) = c$

```
def solve_eq(f,c,guess):  
    # This function returns the solution to  $f(x) = c$  using  
    # fsolve() on the function  $g(x) = f(x) - c$   
  
    def g(x):  
        return f(x) - c  
  
    x = optimize.fsolve(g,guess)[0]  
    return x
```

## Minimizing a function $f$

Consider  $f(x) = x^2 + 2x - 1$ . **Minimum of  $f$**  is a point  $x$  for which  $f(x)$  is smallest.



## Computing a minimum of $f$ using `fmin()`

```
import scipy.optimize as optimize

def f(x):
    return x**2 + 2*x - 1

guess = 1
minimum = optimize.fmin(f,guess)

print('The minimum of the function f is attained at x = ', minimum)
```

Optimization terminated successfully.

Current function value: -2.000000

Iterations: 19

Function evaluations: 38

The minimum of the function  $f$  is attained at  $x = [-1.]$

## Computing a minimum of $f$ using `fmin()`

```
import scipy.optimize as optimize

def f(x):
    return x**2 + 2*x - 1

guess = 1
minimum = optimize.fmin(f,guess,disp=False)[0]

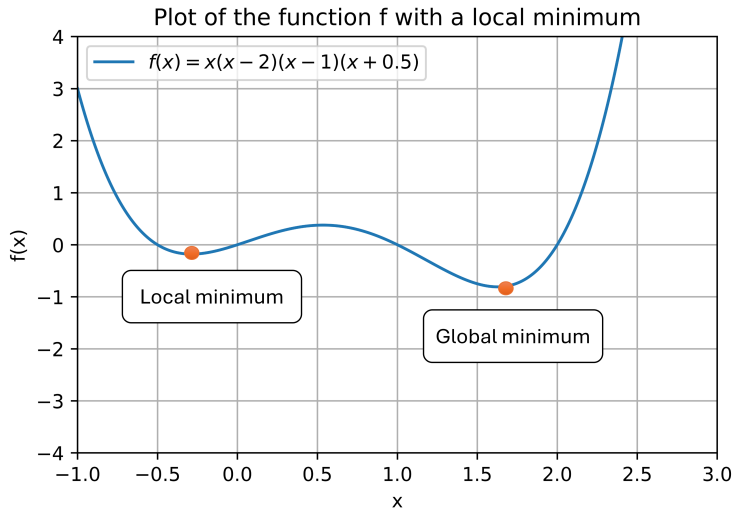
print('The minimum of the function f is attained at x = ', minimum)
```

The minimum of the function  $f$  is attained at  $x = -1.00000000000000018$



## Local vs. global minima

`fmin()` might return a “local” minimum, which is not the true minimum of the function.



# Matplotlib: Data visualization

Matplotlib is a package that can be used for data visualization

- For this we use the `matplotlib.pyplot` (sub)package ...
- ... which we usually import under the name `plt`

```
import matplotlib.pyplot as plt
```

# How are functions plotted in Python?

- 1 Create a vector of  $x$ -values, e.g.,

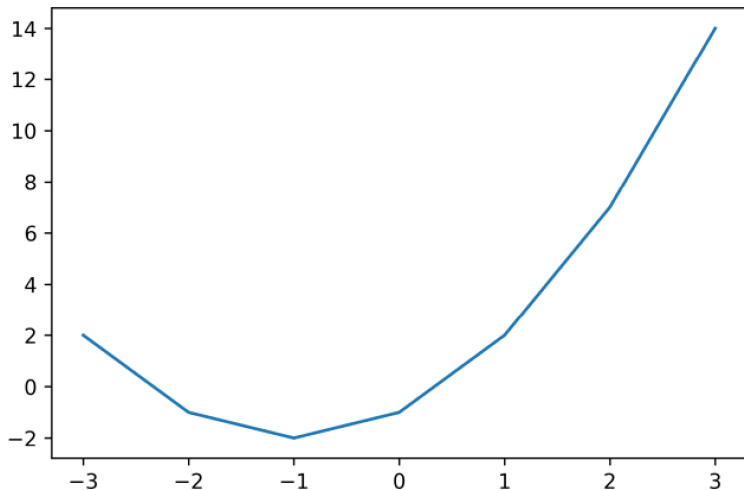
$$x = [-3, -2, -1, 0, 1, 2, 3].$$

- 2 Compute the function values

$$[f(-3), f(-2), f(-1), f(0), f(1), f(2), f(3)] = [2, -1, -2, -1, 2, 7, 14].$$

- 3 Draw the points  $(x_i, f(x_i))$  and connect them with line segments.

## Resulting Python plot



## Plotting a “smooth” line

Increase the number of points in  $x$  to get a smoother line using `np.linspace()`.

- Command `np.linspace(a,b,k)` gives array with  $k$  evenly spaced points in interval  $[a, b]$ ; first point is  $a$  and last point  $b$ .
- **Question:** Which points does `np.linspace(0,1,11)` create?

## Plotting a “smooth” line

Increase the number of points in  $x$  to get a smoother line using `np.linspace()`.

- Command `np.linspace(a,b,k)` gives array with  $k$  evenly spaced points in interval  $[a, b]$ ; first point is  $a$  and last point  $b$ .
- **Question:** Which points does `np.linspace(0,1,11)` create?

## Plotting a “smooth” line

Increase the number of points in  $x$  to get a smoother line using `np.linspace()`.

- Command `np.linspace(a,b,k)` gives array with  $k$  evenly spaced points in interval  $[a, b]$ ; first point is  $a$  and last point  $b$ .
- **Question:** Which points does `np.linspace(0,1,11)` create?

```
import numpy as np
```

```
a = 0
```

```
b = 1
```

```
k = 11
```

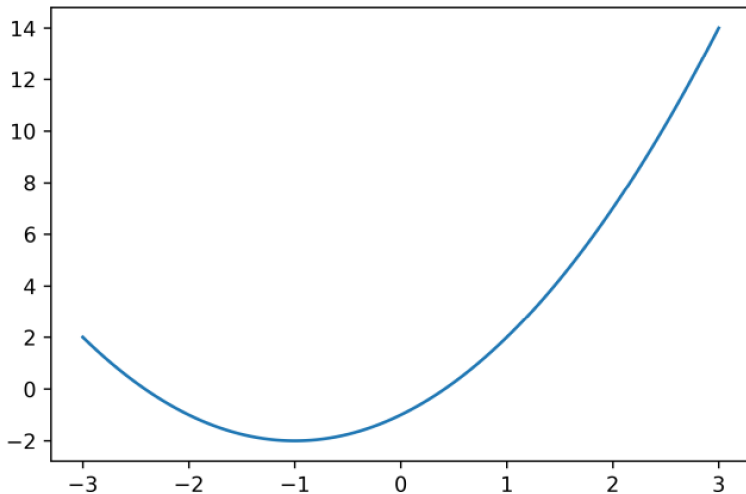
```
x = np.linspace(a,b,k)
```

```
print(x)
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
```

## Resulting “smoothed” Python plot

Using `x = np.linspace(-3,3,600)`

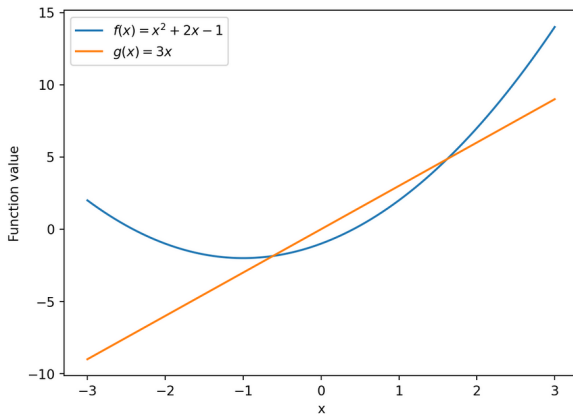




## Adding legend to plot

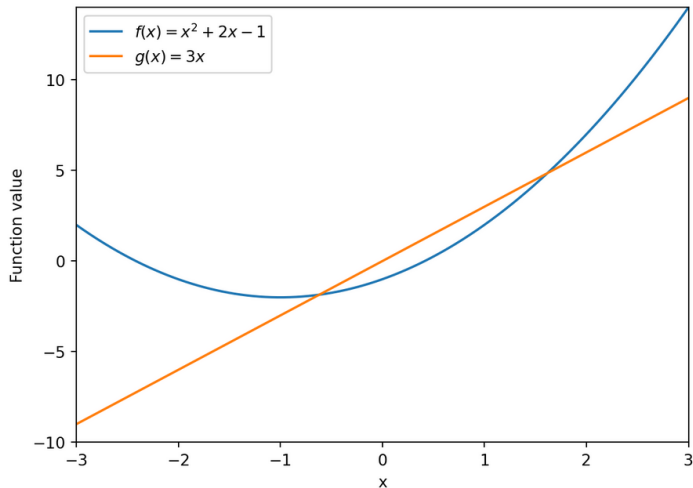
Use label-argument in `plt.plot()` in combination with `plt.legend()` at the end ...

- ... and `plt.xlabel('x')` and `plt.ylabel('Function value')` for axis labels.



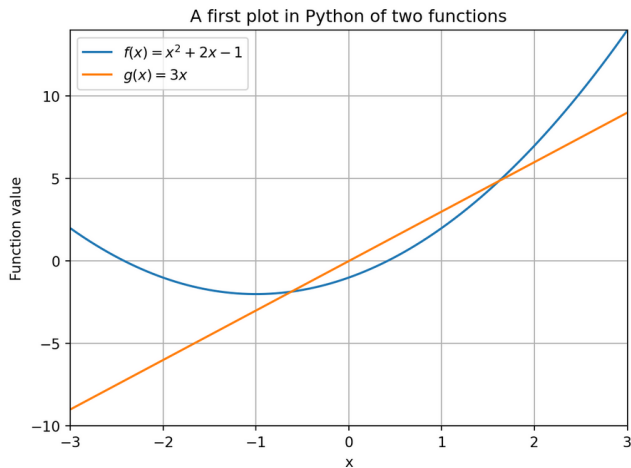
## Fixing axes ranges

Use `plt.xlim(-3,3)` and `plt.ylim(-10,14)` to fix range of horizontal/vertical axis, resp.



## Adding title and grid

- Use `plt.title('A first plot of two functions')` to add title
- Use `plt.grid()` to add grid.



## Classroom Exercise 1

Consider the function  $f(x) = \frac{9}{10}x^4 - 3x^3 - \frac{7}{2}x^2 + 12x + 3$ .

- Plot this function with horizontal axis range  $[-6, 6]$ , and vertical axis range  $[-15, 15]$ .
- Find four roots of this function with `fsolve()` by trying out different initial guesses.
- Find a minimum of this function with `fmin()` by using initial guesses  $-1$  and  $2$ . Are both solutions actual minima of the function?

## Chapter 10 - Object oriented programming

# Object oriented programming

Every student is registered in TiU's digital administration. Keeps track of:

- Personal information
- Course registrations
- Study progress

# Object oriented programming

Every student is registered in TiU's digital administration. Keeps track of:

- Personal information
- Course registrations
- Study progress

In programming terms, every student is considered an **object** in the administration.

# Object oriented programming

Every student is registered in TiU's digital administration. Keeps track of:

- Personal information
- Course registrations
- Study progress

In programming terms, every student is considered an **object** in the administration.

- Creating an object is done with a `Class` in Python.



# Attributes and methods

When a student joins the university, an object is **initialized** for them containing their personal information, for example,

- Name
- Age
- Student number (or student ID)

# Attributes and methods

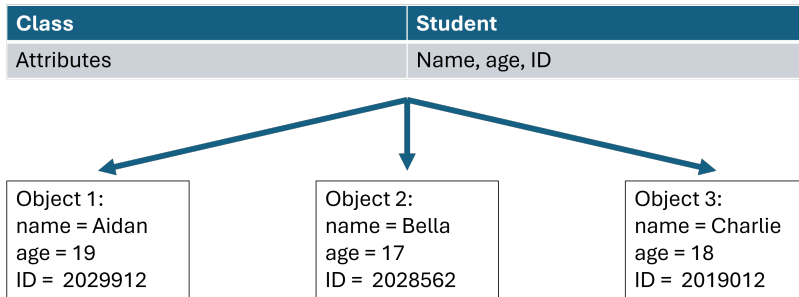
When a student joins the university, an object is **initialized** for them containing their personal information, for example,

- Name
- Age
- Student number (or student ID)

These pieces of information are called the **attributes** of the object.

## Student example

We will create a class `Student` whose objects are students with attributes: name, age, (student) ID.



## Student example: methods

To request information of a student, and to document their study progress, we can define Python functions in our class, which are called **methods**.

## Student example: methods

To request information of a student, and to document their study progress, we can define Python functions in our class, which are called **methods**.

- Initializing an object is done with the built-in `__init__()` method.

## Student example: methods

To request information of a student, and to document their study progress, we can define Python functions in our class, which are called **methods**.

- Initializing an object is done with the built-in `__init__()` method.

## Student example: methods

To request information of a student, and to document their study progress, we can define Python functions in our class, which are called **methods**.

- Initializing an object is done with the built-in `__init__()` method.

Further (self-defined) methods could be:

## Student example: methods

To request information of a student, and to document their study progress, we can define Python functions in our class, which are called **methods**.

- Initializing an object is done with the built-in `__init__()` method.

Further (self-defined) methods could be:

- Checking if a student is an adult (for legal reasons)



## Student example: methods

To request information of a student, and to document their study progress, we can define Python functions in our class, which are called **methods**.

- Initializing an object is done with the built-in `__init__()` method.

Further (self-defined) methods could be:

- Checking if a student is an adult (for legal reasons)
- Checking exam registration

## Student example: methods

To request information of a student, and to document their study progress, we can define Python functions in our class, which are called **methods**.

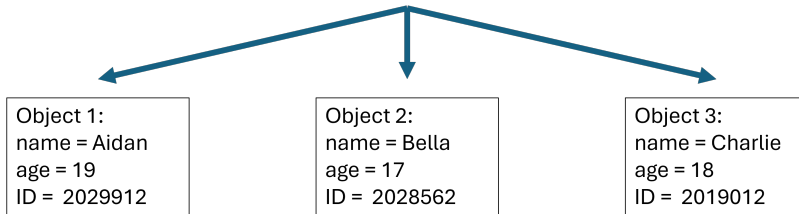
- Initializing an object is done with the built-in `__init__()` method.

Further (self-defined) methods could be:

- Checking if a student is an adult (for legal reasons)
- Checking exam registration
- Keeping track of list of registered courses

## Student example: overview

Class	Student
Attributes	Name, age, ID
Methods	Initialize object, check if adult, check exam registration, keep track of registered courses



A **class** in Python creates **objects** with **attributes**, and can use **methods** to investigate and modify these objects.

# Initializing objects

Initialization is always done with `__init__()` function.

```
class Student:
    # This function initializes an object of the class Student
    # by setting the attributes (name, age and ID) of an object.
    def __init__(self,name,age,student_number):
        self.name = name
        self.age = age
        self.ID = student_number
```

The argument `self` should be thought of as the object that we want to create.

- The use of `self` as a variable name for this is standard in Python

## Methods

Methods are Python functions in a class. `__init__()` is also a method of the class.

```
class Student:
    def __init__(self,name,age,student_number):
        self.name = name
        self.age = age
        self.ID = student_number

    # Check if student is adult
    def adult(self):
        if self.age >= 18:
            return print(self.name," is an adult")
        else:
            return print(self.name,"is not an adult")
```

- Every method has first input argument `self`.
- Attribute `attribute_name` can be accessed in method using `self.attribute_name`.

## Additional inputs and changing attributes

Methods can require additional input arguments beside `self`.

- See `reg_check()` example in course document.

Methods can be used to manipulate attributes.

- See `addCourse()/delCourse` example in course document.

## Adding course to course list

We add a **new attribute** `courses` that is an (initially empty) list.

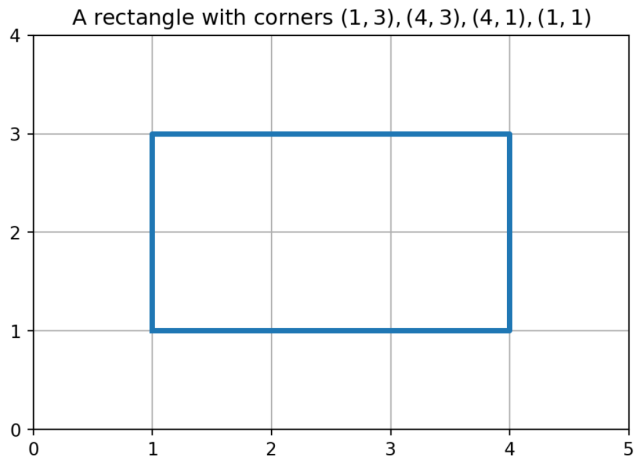
```
class Student:
    def __init__(self, name, age, student_number):
        self.name = name
        self.age = age
        self.ID = student_number
        self.courses = []
```

Write method `addCourse()` that can add (append) course names to this list.

```
def addCourse(self, course_name):
    return self.courses.append(course_name)
```

## Mathematical example: Rectangles

We create a class `Rectangle` whose objects are rectangles in a two-dimensional plane.

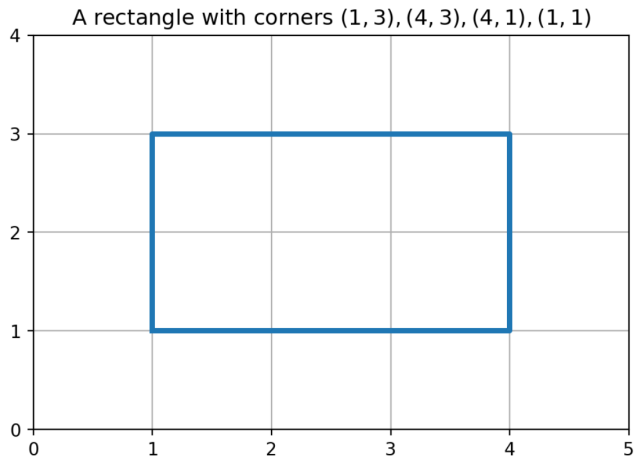


How to model a rectangle? What should the attributes be?



## Mathematical example: Rectangles

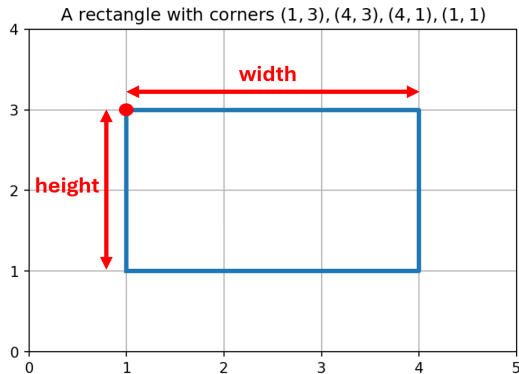
We create a class `Rectangle` whose objects are rectangles in a two-dimensional plane.



How to model a rectangle? What should the attributes be? Depends on desired methods...

## Rectangle attributes

To uniquely determine a rectangle, it suffices to know: Upper-left corner point, width and height.



## Rectangle \_\_init\_\_() method

```
class Rectangle:
    # Here corner is the upper-left corner point which
    # should be a list containing the x- and y-coordinate.
    def __init__(self, corner, height, width):
        self.corner= corner
        self.height = height
        self.width = width

rectangle1 = Rectangle([1,3],2,3)

# Print upper left corner of the rectangle
print(rectangle1.corner)
```

[1, 3]

## Rectangle methods: Area and circumference

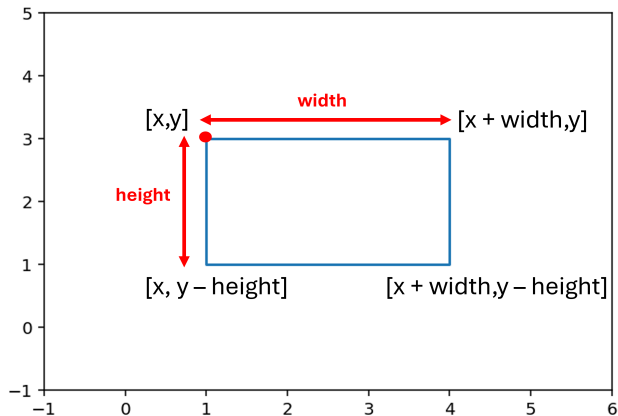
```
class Rectangle:
    # Initialize rectangle by providing upper-left corner, width and height
    def __init__(self, corner, width, height):
        self.corner = corner
        self.height = height
        self.width = width

    # Compute area = width*height of rectangle
    def area(self):
        return self.height*self.width

    # Compute circumference = 2*width + 2*height of rectangle
    def circumference(self):
        return 2*self.width + 2*self.height
```

## Rectangle methods: Compute all corner points

Given upper left corner  $[x, y]$ , width and height, we can compute the coordinates of the remaining corner points.



## Rectangle methods: Compute all corner points (cont'd)

```
class Rectangle:
    # Initialize rectangle by providing upper-left corner, width and height
    def __init__(self, corner, height, width):
        self.corner= corner
        self.height = height
        self.width = width

    # Compute corner points: the output has the points in the
    # order [upper-left, upper-right, lower-right, lower-left]
    def corners(self):
        up_right = [self.corner[0] + self.width, self.corner[1]]
        low_right=[self.corner[0]+self.width,self.corner[1]-self.height]
        low_left = [self.corner[0], self.corner[1] - self.height]
        return [self.corner, up_right, low_right, low_left]
```

## Rectangle methods: Plotting

You can find the `plotting()` method in the course document. The idea is as follows:

- Determine all corner points using the `corners()` method.
- Plot the points in order: upper-left, upper-right, lower-right, lower-left, upper-left.

## Rectangle methods: Plotting

You can find the `plotting()` method in the course document. The idea is as follows:

- Determine all corner points using the `corners()` method.
- Plot the points in order: upper-left, upper-right, lower-right, lower-left, upper-left.

The `plot()` function plots these five points and connects them with a line segment (resulting in a rectangle shape).

- Plot list with all  $x$ -coordinates against list with all  $y$ -coordinates of the five points.