

# Python Data Analysis

2024 – 2025

## General informations

**Louie Corpe**

[louie.corpe@cern.ch](mailto:louie.corpe@cern.ch)



**Romain Madar**

[romain.madar@cern.ch](mailto:romain.madar@cern.ch)



# Material for the lecture

1. On **moodle** platform (ENT, UCA account required):

<https://moodle2024.uca.fr/course/view.php?id=7069>

2. On **github** platform (free access):

<https://github.com/rmadar/lecture-python>

# Material for the lecture

1. On **moodle** platform (ENT, UCA account required):

<https://moodle2024.uca.fr/course/view.php?id=7069>

2. On **github** platform (free access):

<https://github.com/rmadar/lecture-python>

## Content of the lecture -- full PDF

There are a lot of information in this lecture. In order to help you to focus on important aspect, each chapter start with a list of expected skills that you should take away, ranked with three levels: *basic*, *medium*, *expert*.

**0. Practical Introduction to Jupyter Notebooks.** This section is not present in the final PDF but is presented during the lecture.

**1. Practical Introduction to Python.** This first section is dedicated to basic object type and operation in python. Functions will also be described but object oriented programming will not be covered.

**2. Introduction to numpy.** Differences between usual python objects and numpy objects will be introduced.

**3. Three tools to know.** This section gives a glimpse of `matplotlib`, `pandas` and `scipy` packages allowing powerful data analysis.

**4. Multidimensional data manipulation.** Non-trivial operation for multidimensional data using the full power of numpy. Most of these operation can be performed with existing tools but it is intrusive to do it once with native numpy.

**5. Introduction to image processing.** Very first steps of image processing (definition, plotting, operation) including basic filters application (noising, sharpen, border detection).

**Other practical examples.** Depending on the remaining time (and the people taste), we can go through different topics among the following ones. Some of them can be also used as a project performed by students.

- Fourier analysis
- Principal component analysis (PCA)
- Random Forest regression
- Gaussian processes

## List of previous exams with corrections

- 2019 : Analysis of an electric pulse --> [exam / correction](#)
- 2020 : Ising model (more details on this topic [here](#)) --> [exam / correction](#)
- 2021 : Coupled harmonic oscillators (more details on this topic [here](#)) --> [exam / correction](#)
- 2022 : Random walk --> [exam / correction](#)

## How to get prepared

**1. Get familiar with python.** I would recommend two links: [w3school tutorial](#) (both basic and complete) and <https://www.learnpython.org> (code can be ran directly within your web browser).

**2. Install python with anaconda.** In order to run python on your own machine, you should install it. I would recommend [anaconda](#) for this, which also includes jupyter-notebook.

**3. Install git.** This is a versioning software which can be installed following these [instructions](#). This whole repository can be *cloned* using `git clone https://github.com/rmadar/lecture-python` command.

**4. Get familiar with notebooks.** This represents a nice environment combining codes, notes and plots. This is very powerful to learn something and play with it. You can checkout [this video](#) or [this post](#).

# Material for the lecture

1. On **moodle** platform (ENT, UCA account required):

<https://moodle2024.uca.fr/course/view.php?id=7069>

2. On **github** platform (free access):

<https://github.com/rmadar/lecture-python>

rmadar Updating the preamble		a4daf84 3 days ago	🕒 157 commits
📁 assignment	Getting up to date	2 weeks ago	
📁 data	Getting up to date	2 weeks ago	
📁 documentation	Updating the preamble	3 days ago	
📁 exam	Getting up to date	2 weeks ago	
📁 exercises	Clearning exercice directory	last year	
📁 lectures	Set the repo up-to-date before 2022 modifications	last year	
📄 .gitignore	added assignment	4 years ago	
📄 README.md	Update README	3 days ago	

## Content of the lecture -- full PDF

There are a lot of information in this lecture. In order to help you to focus on important aspect, each chapter start with a list of expected skills that you should take away, ranked with three levels: *basic*, *medium*, *expert*.

**0. Practical Introduction to Jupyter Notebooks.** This section is not present in the final PDF but is presented during the lecture.

**1. Practical Introduction to Python.** This first section is dedicated to basic object type and operation in python. Fonctions will also be described but object oriented programming will not be covered.

**2. Introduction to numpy.** Differences between usual python objects and numpy objects will be introduced.

**3. Three tools to know.** This section gives a glimpse of `matplotlib`, `pandas` and `scipy` packages allowing powerful data analysis.

**4. Multidimensional data manipulation.** Non-trivial operation for multidimensional data using the full power of numpy. Most of these operation can be performed with existing tools but it is intrusive to do it once with native numpy.

**5. Introduction to image processing.** Very first steps of image processing (definition, plotting, operation) including basic filters application (noising, sharpen, border detection).

**Other practical examples.** Depending on the remaining time (and the people taste), we can go through different topics among the following ones. Some of them can be also used as a project performed by students.

- Fourier analysis
- Principal component analysis (PCA)
- Random Forest regression
- Gaussian processes

## List of previous exams with corrections

- 2019 : Analysis of an electric pulse --> [exam / correction](#)
- 2020 : Ising model (more details on this topic [here](#)) --> [exam / correction](#)
- 2021 : Coupled harmonic oscillators (more details on this topic [here](#)) --> [exam / correction](#)
- 2022 : Random walk --> [exam / correction](#)

## How to get prepared

**1. Get familiar with python.** I would recommand two links: [w3school tutorial](#) (both basic and complete) and <https://www.learnpython.org> (code can be ran directly within your web browser).

**2. Install python with anaconda.** In order to run python on your own machine, you should install it. I would recommand [anaconda](#) for this, which also includes jupyter-notebook.

**3. Install git.** This is a versioning software which can be installed following these [instructions](#). This whole repository can be *cloned* using `git clone https://github.com/rmadar/lecture-python` command.

**4. Get familiar with notebooks.** This represents a nice environnement combining codes, notes and plots. This is very powerful to learn something and play with it. You can checkout [this video](#) or [this post](#).

# Lecture structure & technicals

## A typical day:

- presentation of a new chapter, with some little exercises for you to practice
- practical sessions with larger exercises

**You need to have a running notebook with a proper python environment**

(2 options : UCA computer, your laptop)

### How to get prepared

**1. Get familiar with python.** I would recommend two links: [w3school tutorial](#) (both basic and complete) and <https://www.learnpython.org> (code can be ran directly within your web browser).

**2. Install python with anaconda.** In order to run python on your own machine, you should install it. I would recommend [anaconda](#) for this, which also includes jupyter-notebook.

**3. Install git.** This is a versioning software which can be installed following these [instructions](#). This whole repository can be *cloned* using `git clone https://github.com/rmadar/lecture-python` command.

**4. Get familiar with notebooks.** This represents a nice environnement combining codes, notes and plots. This is very powerful to learn something and play with it. You can checkout [this video](#) or [this post](#).

# Skills and Evaluations

[lecture pdf]

## Chapter 1

### Practical Introduction to Python

#### Skills to take away

- *basic*: int/float/str, list/dictionary, indexing/slicing, loops, functions, reading/writing files
- *medium*: docstring, comprehension, zip()/enumerate(), sorting dictionary
- *expert*: packing/unpacking, parsing file with correct casting, basic plotting

#### 1.1 General information

Python can be installed using [anaconda](#). [Jupyter-notebook](#) (also coming with anaconda) is probably the easiest way to follow this lecture and make your own notes. The goal of this first chapter is to give a *very quick*

[lecture notebook]

jupyter  
nbviewer

JUPYTER FAQ

lecture-python / lectures

### Practical Introduction to Python

#### Skills to take away

- *basic*: int/float/str, list/dictionary, indexing/slicing, loops, functions, reading/writing files
- *medium*: docstring, comprehension, zip()/enumerate(), sorting dictionary
- *expert*: packing/unpacking, parsing file with correct casting, basic plotting

#### General information

Python can be installed using [anaconda](#). [Jupyter-notebook](#) (also coming with anaconda) is probably the easiest way to follow this lecture and make your own notes. The goal of this first chapter is to give a *very quick introduction* basis, but practice is mandatory to get comfortable with python objects and syntax. Practicing is possible with a web browser only using <https://www.learnpython.org>. A more complete tutorial (but not interactive) can be found in [w3school python tutorials](#). I recommend to follow the last tutorial up to *Arrays*.

In python, there is one instruction per line. Variable assignment is done with `=`, indentation is used to group instructions together under a loop or a condition block: there is no bracket (like in C++) or equivalent. Comments (i.e. uninterpreted text) start with `#`. Importation of external modules or function can be done with three different ways: `import module`, `import module as m` or `from module import this_function`.

In the following example, the result of the command will be printed so that people can check that the computer is doing what is expected. The instruction `print(x)` will print the content of `x`. If several variables are printed, it is convenient to use `print('x={}' and 'y={}'.format(x, y))` syntax that will print `x` and `y` in bracket fields with one command - even if they have different types.

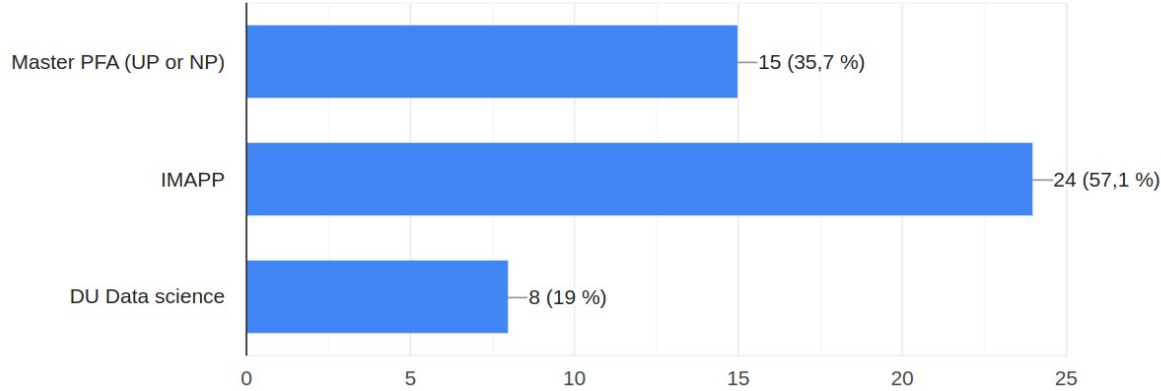
**Evaluation** : 2h exam on a computer in classroom (beginning of november for the masters, end of september for DUs).

**Allowed material** : *only* official python documentations of tools (**lecture material not allowed**)

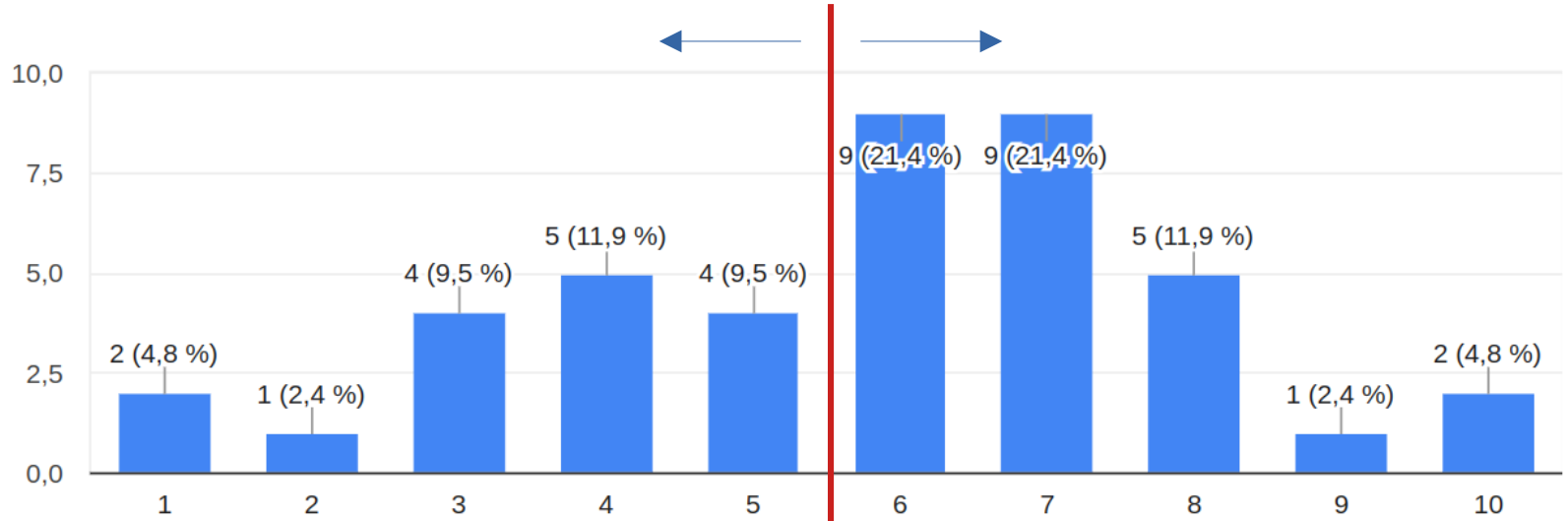
**Caution**: in case of doubt about your answers, you might be interviewed to explain what you wrote and why (it happened in the past that some people might have used unauthorized ressources).

# The group

## Composition

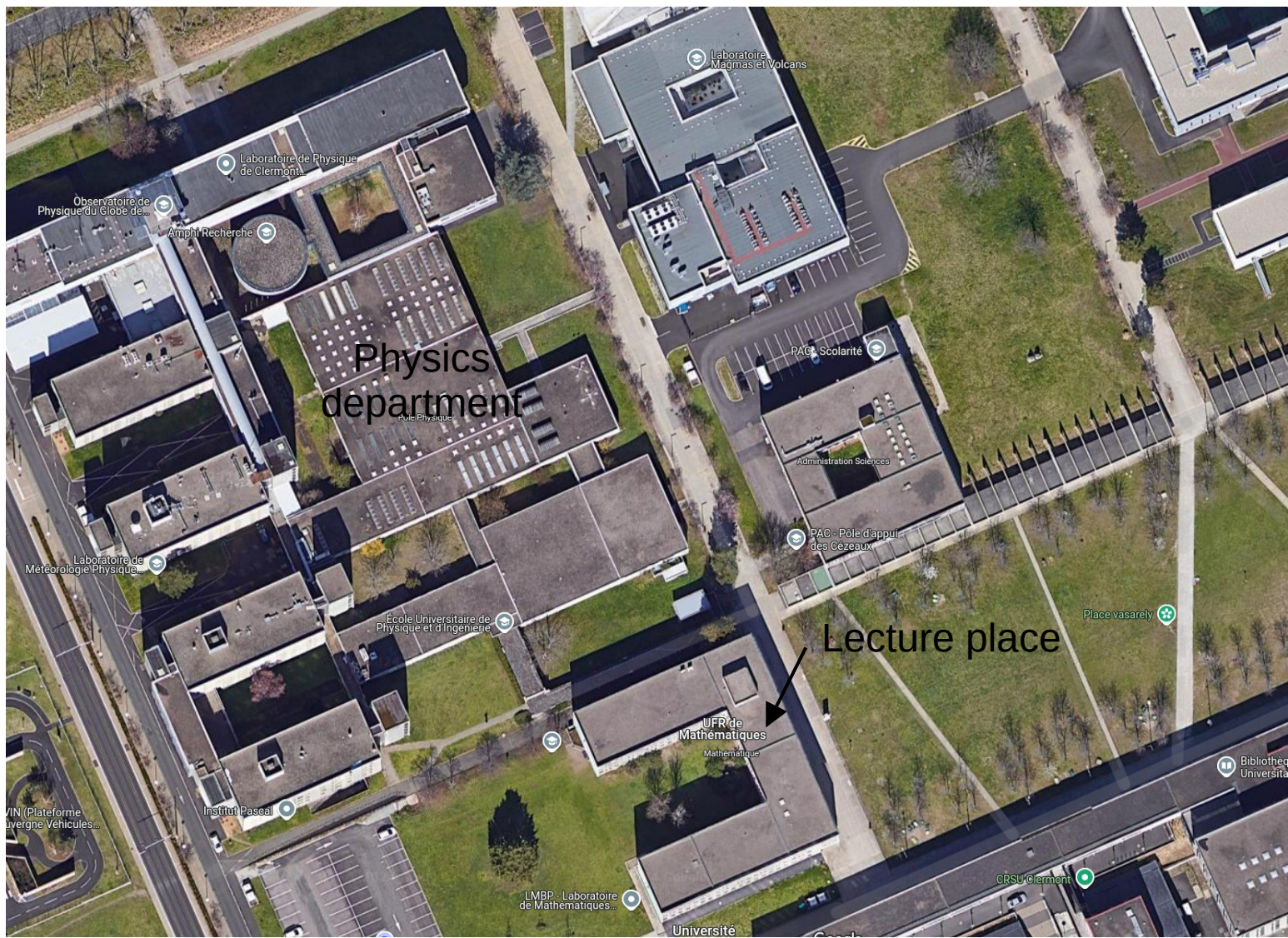


## Auto-evaluation of python skills



	S37				
	Lundi 09/09/2024	Mardi 10/09/2024	Mercredi 11/09/2024	Jeudi 12/09/2024	Vendredi 13/09/2024
07h30					
08h00-					
08h30-					
09h00-					
09h30-					
10h00-					
10h30-					
11h00-					
11h30-					
12h00-					
12h30-					
13h00-					
13h30-					
14h00-					
14h30-					
15h00-					
15h30-					
16h00-					
16h30-					
17h00-					





# Side comment : ChatGPT

How **needed** is it to **learn** programming language,  
when tools like **ChatGPT** exist ?

**Let's try something!**

# Side comment : ChatGPT

How **needed** is it to **learn** programming language,  
when tools like **ChatGPT** exist ?

**Let's try something!**

Here is the problem : a circular bucket of N liters is filled at x% but has a hole of a diameter d. Write a python function which compute the time it takes for the bucket as function of it's height.



# Side comment : ChatGPT

How **needed** is it to **learn** programming language, when tools like **ChatGPT** exist ?

**Let's try something!**

Here is the problem : a circular bucket of  $N$  liters is filled at  $x\%$  but has a hole of a diameter  $d$ . Write a python function which compute the time it takes for the bucket as function of its height.

To solve this problem, we need to take a few factors into account, specifically how fluid flows out of a hole at the bottom of a bucket. This problem relates to Torricelli's law, which describes the speed of fluid flowing out of an orifice under the force of gravity. The relevant physics tells us that:

$$v(t) = \sqrt{2gh(t)}$$

Where:

- $v(t)$  is the speed of the fluid exiting the hole as a function of height.
- $g$  is the acceleration due to gravity (9.81 m/s<sup>2</sup>).
- $h(t)$  is the height of the water above the hole at time  $t$ .

To compute the time it takes for the bucket to empty as a function of its height, we need to consider:

- The cross-sectional area of the hole (depending on its diameter  $d$ ).
- The cross-sectional area of the bucket.
- The rate at which the water height decreases over time.

## Derivation:

The volume flow rate of water out of the hole is given by:

$$Q = A_h v(t) = A_h \sqrt{2gh(t)}$$

Where:

- $A_h = \frac{\pi d^2}{4}$  is the area of the hole.
- $v(t)$  is the velocity of water exiting the hole (using Torricelli's law).

The volume of water in the bucket at height  $h$  is  $V = A_b h$ , where  $A_b$  is the cross-sectional area of the bucket (assuming it's cylindrical).

To compute the time it takes for the bucket to empty as a function of its height, we need to consider:

- The cross-sectional area of the hole (depending on its diameter  $d$ ).
- The cross-sectional area of the bucket.
- The rate at which the water height decreases over time.

## Derivation:

The volume flow rate of water out of the hole is given by:

$$Q = A_h v(t) = A_h \sqrt{2gh(t)}$$

Where:

- $A_h = \frac{\pi d^2}{4}$  is the area of the hole.
- $v(t)$  is the velocity of water exiting the hole (using Torricelli's law).

The volume of water in the bucket at height  $h$  is  $V = A_b h$ , where  $A_b$  is the cross-sectional area of the bucket (assuming it's cylindrical).

The change in water volume over time is equal to the outflow, which gives the following differential equation:

$$\frac{dV}{dt} = -A_h \sqrt{2gh}$$

Substituting  $V = A_b h$ , we get:

$$A_b \frac{dh}{dt} = -A_h \sqrt{2gh}$$

Now we solve this equation for  $\frac{dh}{dt}$ :

$$\frac{dh}{dt} = -\frac{A_h}{A_b} \sqrt{2g\sqrt{h}}$$

We can integrate this equation to find the time it takes for the bucket to empty from a given initial height  $h_0$ .

## Python Implementation

```
python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Constants
g = 9.81 # Acceleration due to gravity (m/s^2)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of
    its initial height and fill percentage.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g. 80)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m^2)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m^2)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0 # Stop when the bucket is empty
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the bucket (1 meter)
bucket_radius = 0.3 # Radius of the bucket (30 cm)
fill_percentage = 80 # Bucket is 80% filled

# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

# Side comment : ChatGPT

```
import numpy as np
from scipy.integrate import solve_ivp

# Constants
g = 9.81 # Acceleration due to gravity (m/s²)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of height.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g., 80 for 80%)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m²)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m²)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0 # Stop when the bucket is empty
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the bucket (1 meter)
bucket_radius = 0.5 # Radius of the bucket (50 cm)
fill_percentage = 80 # Bucket is 80% filled

# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

Time to empty the bucket: 10000.00 seconds

Is the result correct ?

# Side comment : ChatGPT

```
import numpy as np
from scipy.integrate import solve_ivp

# Constants
g = 9.81 # Acceleration due to gravity (m/s²)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of height.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g., 80 for 80%)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m²)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m²)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0 # Stop when the bucket is empty
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the water in the bucket (1 meter)
bucket_radius = 0.5 # Radius of the bucket (50 cm)
fill_percentage = 80 # Bucket is 80% filled

# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

Time to empty the bucket: 10000.00 seconds

```
# Shortcut for the function f(h) = t
f = lambda h: emptying_time(diameter, h/100., bucket_radius, fill_percentage)[0]

# Scan over height values
hs = np.linspace(0, 100, 100)

# For each, compute the time
ts = [f(h) for h in hs]

# Plot time vs height
plt.plot(hs, ts)
plt.xlabel('Bucket Height [cm]')
plt.ylabel('Time of Empty Bucket [s]');
```

Is the result correct ?

# Side comment : ChatGPT

```
import numpy as np
from scipy.integrate import solve_ivp

# Constants
g = 9.81 # Acceleration due to gravity (m/s²)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of height.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g., 80 for 80%)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m²)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m²)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0 # Stop when the bucket is empty
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the bucket (1 meter)
bucket_radius = 0.5 # Radius of the bucket (50 cm)
fill_percentage = 80 # Bucket is 80% filled

# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

Time to empty the bucket: 10000.00 seconds

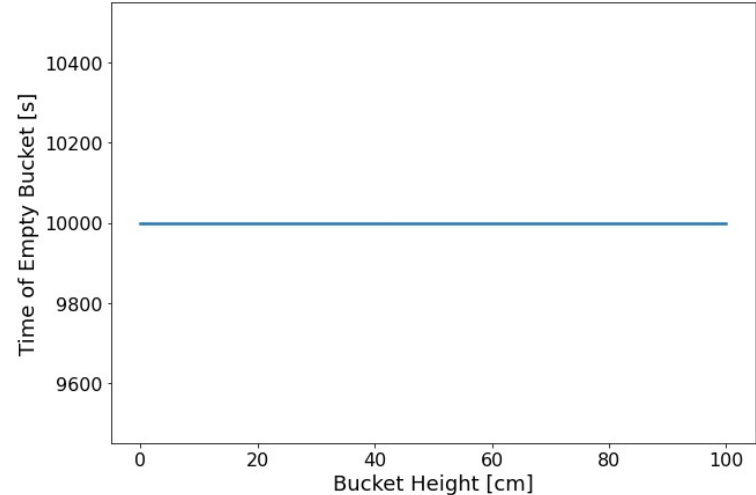
**Is the result correct ? NO**

```
# Shortcut for the function f(h) = t
f = lambda h: emptying_time(diameter, h/100., bucket_radius, fill_percentage)[0]

# Scan over height values
hs = np.linspace(0, 100, 100)

# For each, compute the time
ts = [f(h) for h in hs]

# Plot time vs height
plt.plot(hs, ts)
plt.xlabel('Bucket Height [cm]')
plt.ylabel('Time of Empty Bucket [s]');
```



According to chatGPT, a bucket of 1cm or 1m will empty in 1000s ...

# Side comment : ChatGPT

```
import numpy as np
from scipy.integrate import solve_ivp

# Constants
g = 9.81 # Acceleration due to gravity (m/s²)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of height.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g., 80 for 80%)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m²)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m²)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0 # Stop when the bucket is empty
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the water in the bucket (1 meter)
bucket_radius = 0.5 # Radius of the bucket (50 cm)
fill_percentage = 80 # Bucket is 80% filled

# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

Time to empty the bucket: 10000.00 seconds

```
: # Shortcut for the function f(h) = sol
f = lambda h: emptying_time(diameter, h/100., bucket_radius, fill_percentage)[1]

# Get the differential equation solution for different buckets heights
hs = np.arange(10, 100, 20)
sols = [f(h) for h in hs]

# Plotting height over time for different bucket height
for sol, h in zip(sols, hs):
    plt.plot(sol.t, sol.y[0], label=f'h={h:.0f} cm')
plt.xlabel('Time (s)')
plt.ylabel('Water Height (m)')
plt.title('Water Height as a Function of Time')
plt.legend();
```

Let's investigate  $h(t)$  for different bucket heights

Is the result correct ? NO



# Side comment : ChatGPT

```
import numpy as np
from scipy.integrate import solve_ivp

# Constants
g = 9.81 # Acceleration due to gravity (m/s²)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of height.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g., 80 for 80%)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m²)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m²)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the bucket (1 meter)
bucket_radius = 0.5 # Radius of the bucket (50 cm)
fill_percentage = 80 # Bucket is 80% filled

# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

Time to empty the bucket: 10000.00 seconds

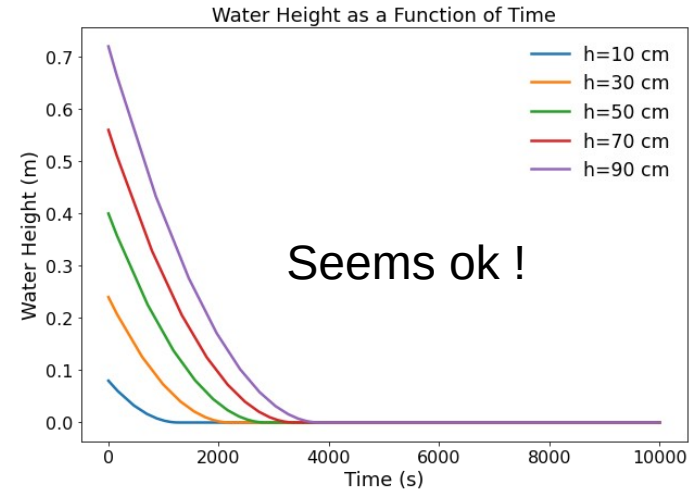
Is the result correct ? NO

```
# Shortcut for the function f(h) = sol
f = lambda h: emptying_time(diameter, h/100., bucket_radius, fill_percentage)[1]

# Get the differential equation solution for different buckets heights
hs = np.arange(10, 100, 20)
sols = [f(h) for h in hs]

# Plotting height over time for different bucket height
for sol, h in zip(sols, hs):
    plt.plot(sol.t, sol.y[0], label=f'h={h:.0f} cm')
plt.xlabel('Time (s)')
plt.ylabel('Water Height (m)')
plt.title('Water Height as a Function of Time')
plt.legend();
```

Let's investigate  $h(t)$  for different bucket heights



# Side comment : ChatGPT

```
import numpy as np
from scipy.integrate import solve_ivp

# Constants
g = 9.81 # Acceleration due to gravity (m/s²)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of height.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g., 80 for 80%)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m²)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m²)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the bucket (1 meter)
bucket_radius = 0.5 # Radius of the bucket (50 cm)
fill_percentage = 80 # Bucket is 80% filled

# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

Time to empty the bucket: 10000.00 seconds

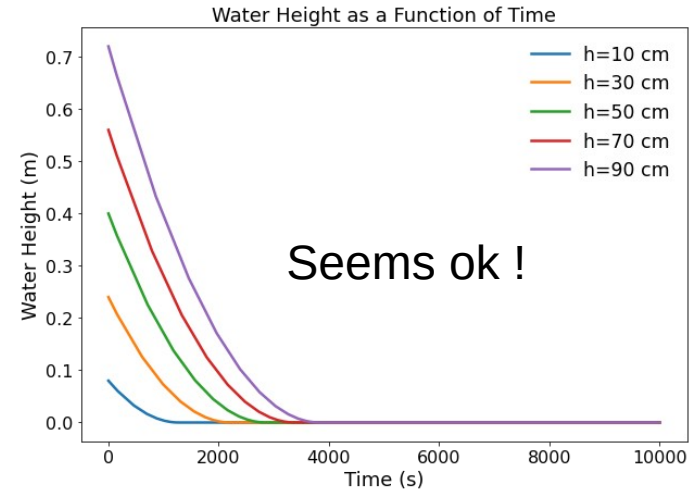
Is the result correct ? NO

```
# Shortcut for the function f(h) = sol
f = lambda h: emptying_time(diameter, h/100., bucket_radius, fill_percentage)[1]

# Get the differential equation solution for different buckets heights
hs = np.arange(10, 100, 20)
sols = [f(h) for h in hs]

# Plotting height over time for different bucket height
for sol, h in zip(sols, hs):
    plt.plot(sol.t, sol.y[0], label=f'h={h:.0f} cm')
plt.xlabel('Time (s)')
plt.ylabel('Water Height (m)')
plt.title('Water Height as a Function of Time')
plt.legend();
```

Let's investigate  $h(t)$  for different bucket heights



But ChatGPT gives the last value, instead of the first one equal to (sufficiently close to) zero ...

# Side comment : ChatGPT

```
import numpy as np
from scipy.integrate import solve_ivp

# Constants
g = 9.81 # Acceleration due to gravity (m/s²)

def emptying_time(diameter, height_initial, bucket_radius, fill_percentage):
    """
    Function to compute the time it takes for the bucket to empty as a function of height.

    Parameters:
    - diameter: Diameter of the hole in meters (d)
    - height_initial: Initial height of the water in the bucket (h_0)
    - bucket_radius: Radius of the bucket in meters (R)
    - fill_percentage: Initial fill percentage of the bucket (as a percentage, e.g., 80 for 80%)

    Returns:
    - Time for the bucket to empty from the initial height.
    """

    # Area of the hole
    A_h = np.pi * (diameter / 2)**2 # Area of the hole (m²)

    # Cross-sectional area of the bucket
    A_b = np.pi * bucket_radius**2 # Cross-sectional area of the bucket (m²)

    # Initial fill height (percentage of the bucket)
    h_initial = height_initial * (fill_percentage / 100.0)

    # Function for the rate of change of height (from Torricelli's law)
    def dhdt(t, h):
        if h <= 0:
            return 0 # Stop when the bucket is empty
        return - (A_h / A_b) * np.sqrt(2 * g * h)

    # Solve the differential equation using scipy's solve_ivp
    sol = solve_ivp(dhdt, [0, 10000], [h_initial], method='RK45', rtol=1e-6)

    # Extract the time when the bucket empties (height goes to zero)
    empty_time = sol.t[-1]

    return empty_time, sol

# Parameters
diameter = 0.01 # Diameter of the hole (10 mm)
height_initial = 1.0 # Initial height of the bucket (1 meter)
bucket_radius = 0.5 # Radius of the bucket (50 cm)
fill_percentage = 80 # Bucket is 80% filled

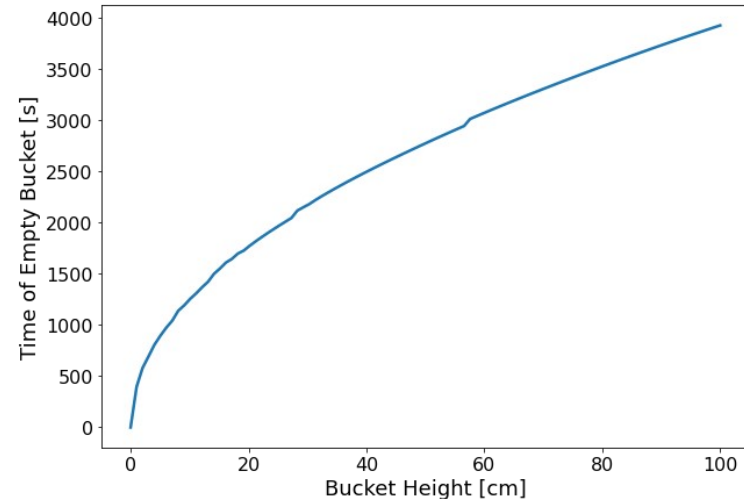
# Compute the emptying time
time_to_empty, sol = emptying_time(diameter, height_initial, bucket_radius, fill_percentage)

# Print the result
print(f"Time to empty the bucket: {time_to_empty:.2f} seconds")
```

Time to empty the bucket: 10000.00 seconds

## Patch to get the correct time

```
# Patch : extract the time when the bucket empties (height goes to zero)
eps = 1e-1 * h_initial # 1 per mille of the initial height of water
tCloseToZero = sol.t[np.abs(sol.y[0])<eps]
if tCloseToZero.size>0:
    empty_time = tCloseToZero[0]
else:
    empty_time = -1
```



Is the result correct ? Now, yes.

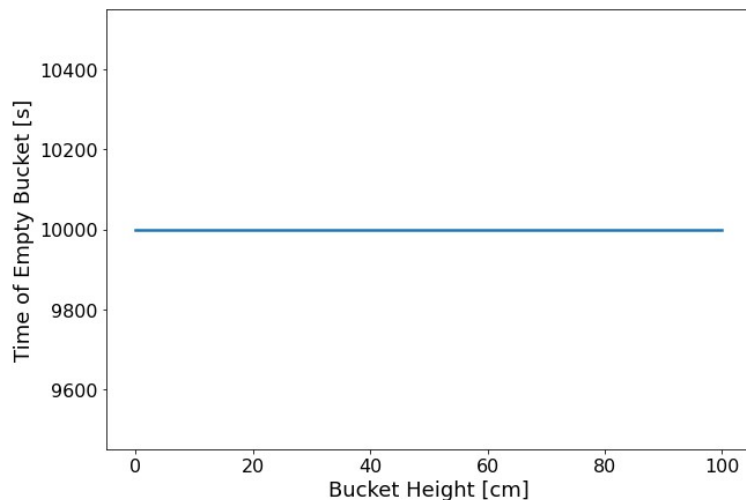
# Side comment : ChatGPT

How **needed** is it to **learn** programming language, when tools like **ChatGPT** exist ?

**It is needed.**

Here is the problem : a circular bucket of  $N$  liters is filled at  $x\%$  but has a hole of a diameter  $d$ . Write a python function which compute the time it takes for the bucket as function of it's height.

ChatGPT w/o human brain



ChatGPT w/ human brain

