Python Data Analysis

2025 – 2026

General informations

Louie Corpe
louie.corpe@cern.ch

Romain Madar
romain.madar@cern.ch

Adrien Auriol
adrien.auriol@cern.ch

Material for the lecture

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

https://moodle2025.uca.fr/course/view.php?id=5715

2. On github platform (free access):

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

Material for the lecture

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

https://moodle2025.uca.fr/course/view.php?id=5715

2. On github platform (free access):

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

Content of the Lecture -- full PDF

There is a lot of information in this lecture. To help you focus on important aspects, each chapter starts 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 types and operations 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 operations for multidimensional data using the full power of numpy. Most of these operations can be performed with existing tools, but it is instructive to do them once with native numpy.

5. Introduction to image processing. Very first steps of image processing (definition, plotting, operation) including basic filter applications (noising, sharpening, border detection).

Other practical examples: Depending on the remaining time (and people's preferences), we can go through different topics among the following ones. Some of them can also be used as projects performed by students.

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

List of Previous Problems with Corrections

- 2019: Analysis of an electric pulse ⟶ problem / correction
- 2020: Ising model (more details on this topic here) ⟶ problem / correction
- 2021: Coupled harmonic oscillators (more details on this topic here) ⟶ problem / correction
- 2022: Random walk ⟶ problem / correction
- 2023: Least action principle ⟶ problem / correction
- 2024: Simulation of bees motion ⟶ problem / 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 run directly within your web browser).

2. Install Python with Anaconda. In order to run Python on your machine, you should install it. I would recommend Anaconda for this, which also includes Jupyter Notebook.

3. Install Git. This is a versioning software that can be installed following these instructions. This whole repository can be *cloned* using the `git clone https://github.com/rmadar/lecture-python` command.

4. Get familiar with notebooks. This represents a nice environment combining code, notes, and plots. This is very powerful to learn something and play with it. You can check out this video or this post.

Material for the lecture

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

https://moodle2025.uca.fr/course/view.php?id=5715

2. On github platform (free access):

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

rmadar Update README	8509a7d · 10 minutes ago	229 Commits
assignment	Getting up to date	2 years ago
data	Getting up to date	2 years ago
documentation	Updating the PDF	2 months ago
exercises	Clearning exercice directory	3 years ago
lectures	Remove the daily notebook for the year to come	13 minutes ago
problems	Rename the exam directory	19 minutes ago
.gitignore	added assignment	6 years ago
README.md	Update README	10 minutes ago
package_version.list	Adding main package version	4 days ago

Content of the Lecture -- full PDF

There is a lot of information in this lecture. To help you focus on important aspects, each chapter starts 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 types and operations 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 operations for multidimensional data using the full power of numpy. Most of these operations can be performed with existing tools, but it is instructive to do them once with native numpy.

5. Introduction to image processing. Very first steps of image processing (definition, plotting, operation) including basic filter applications (noising, sharpening, border detection).

Other practical examples: Depending on the remaining time (and people's preferences), we can go through different topics among the following ones. Some of them can also be used as projects performed by students.

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

List of Previous Problems with Corrections

- 2019: Analysis of an electric pulse ⟶ problem / correction
- 2020: Ising model (more details on this topic here) ⟶ problem / correction
- 2021: Coupled harmonic oscillators (more details on this topic here) ⟶ problem / correction
- 2022: Random walk ⟶ problem / correction
- 2023: Least action principle ⟶ problem / correction
- 2024: Simulation of bees motion ⟶ problem / 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 run directly within your web browser).

2. Install Python with Anaconda. In order to run Python on your machine, you should install it. I would recommend Anaconda for this, which also includes Jupyter Notebook.

3. Install Git. This is a versioning software that can be installed following these instructions. This whole repository can be *cloned* using the `git clone https://github.com/rmadar/lecture-python` command.

4. Get familiar with notebooks. This represents a nice environment combining code, notes, and plots. This is very powerful to learn something and play with it. You can check out 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 run directly within your web browser).

2. Install Python with Anaconda. In order to run Python on your machine, you should install it. I would recommend Anaconda for this, which also includes Jupyter Notebook.

3. Install Git. This is a versioning software that can be installed following these instructions. This whole repository can be *cloned* using the `git clone https://github.com/rmadar/lecture-python` command.

4. Get familiar with notebooks. This represents a nice environment combining code, notes, and plots. This is very powerful to learn something and play with it. You can check out this video or this post.

Skills you're expected to learn

They are listed on top of each chapter, sorted into 3 categories : basic, medium, expert.

Skills you're expected to learn

They are listed on top of each chapter, sorted into 3 categories : basic, medium, expert.

[lecture pdf]

Practical introduction to Python

> **Skills to take away**
>
> - *Basic*: int/float/str, list/dictionary, indexing/slicing, loops, functions, reading/writing files, use a class (attributes, methods), write simplified python code, as defined in the end of this chapter.
> - *Medium*: docstring, comprehension, zip()/enumerate(), `lambda` functions, understand an existing class (search for possible attributes, etc ...)
> - *Expert*: packing/unpacking, parsing file with correct casting, basic plotting, create a new class

1.1 General Information

Python can be installed using Anaconda. Jupyter Notebook (also included with Anaconda) is probably the easiest way to follow this lecture and make your own notes. The goal of this first chapter is to provide a very quick introduction to the basics, but practice is mandatory to get comfortable with Python objects and syntax. You can practice using a web browser only at LearnPython.org. A more complete tutorial (though not interactive) can be found at W3Schools Python Tutorials. I recommend following the last tutorial up to the "Arrays" section.

In Python, there is one instruction per line. Variable assignment is done with =, and indentation is used to

[lecture notebook]

Practical introduction to Python

> **Skills to take away**
>
> - *Basic*: int/float/str, list/dictionary, indexing/slicing, loops, functions, reading/writing files, use a class (attributes, methods), write simplified python code, as defined in the end of this chapter.
> - *Medium*: docstring, comprehension, zip()/enumerate(), `lambda` functions, understand an existing class (search for possible attributes, etc ...)
> - *Expert*: packing/unpacking, parsing file with correct casting, basic plotting, create a new class

General Information

Python can be installed using Anaconda. Jupyter Notebook (also included with Anaconda) is probably the easiest way to follow this lecture and make your own notes. The goal of this first chapter is to provide a very quick introduction to the basics, but practice is mandatory to get comfortable with Python objects and syntax. You can practice using a web browser only at LearnPython.org. A more complete tutorial (though not interactive) can be found at W3Schools Python Tutorials. I recommend following the last tutorial up to the "Arrays" section.

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

Evaluation

Evaluation : ~1h30 **pen & paper exam** in classroom (beginning of november – to be confirmed).

Exam goal: make sure you got

- broad (and useful!) numerical python knowledge
- good coding practices

Exam typical content:

- Lecture questions (concepts, definitions, typical use cases, etc ...)
- Simplified code writing (to be described in the lecture)
- Error findings, code output prediction, etc ...

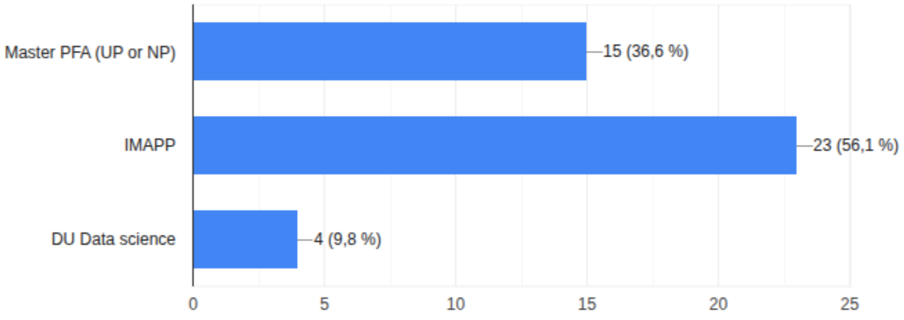
Allowed material :

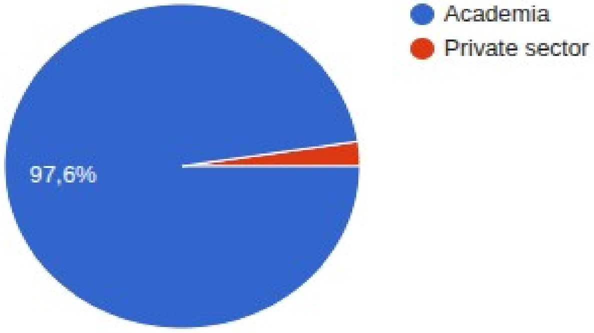
- paper and pen only
- no documentations, no chatGPT
- In particular, **lecture material not allowed**

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).

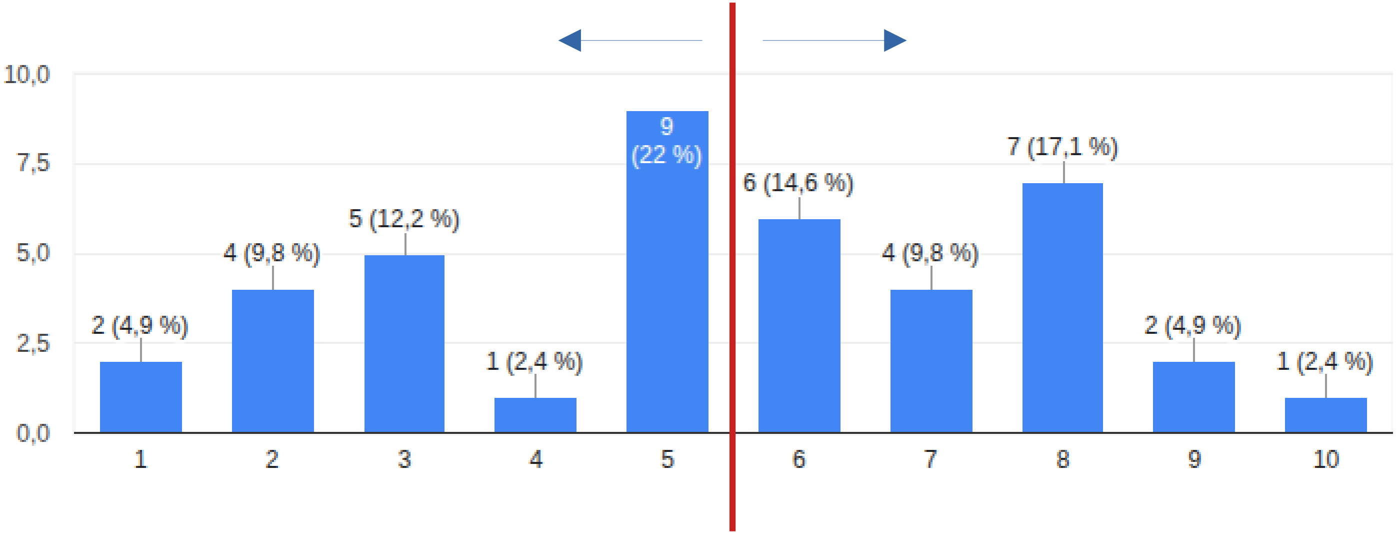
The group

Composition



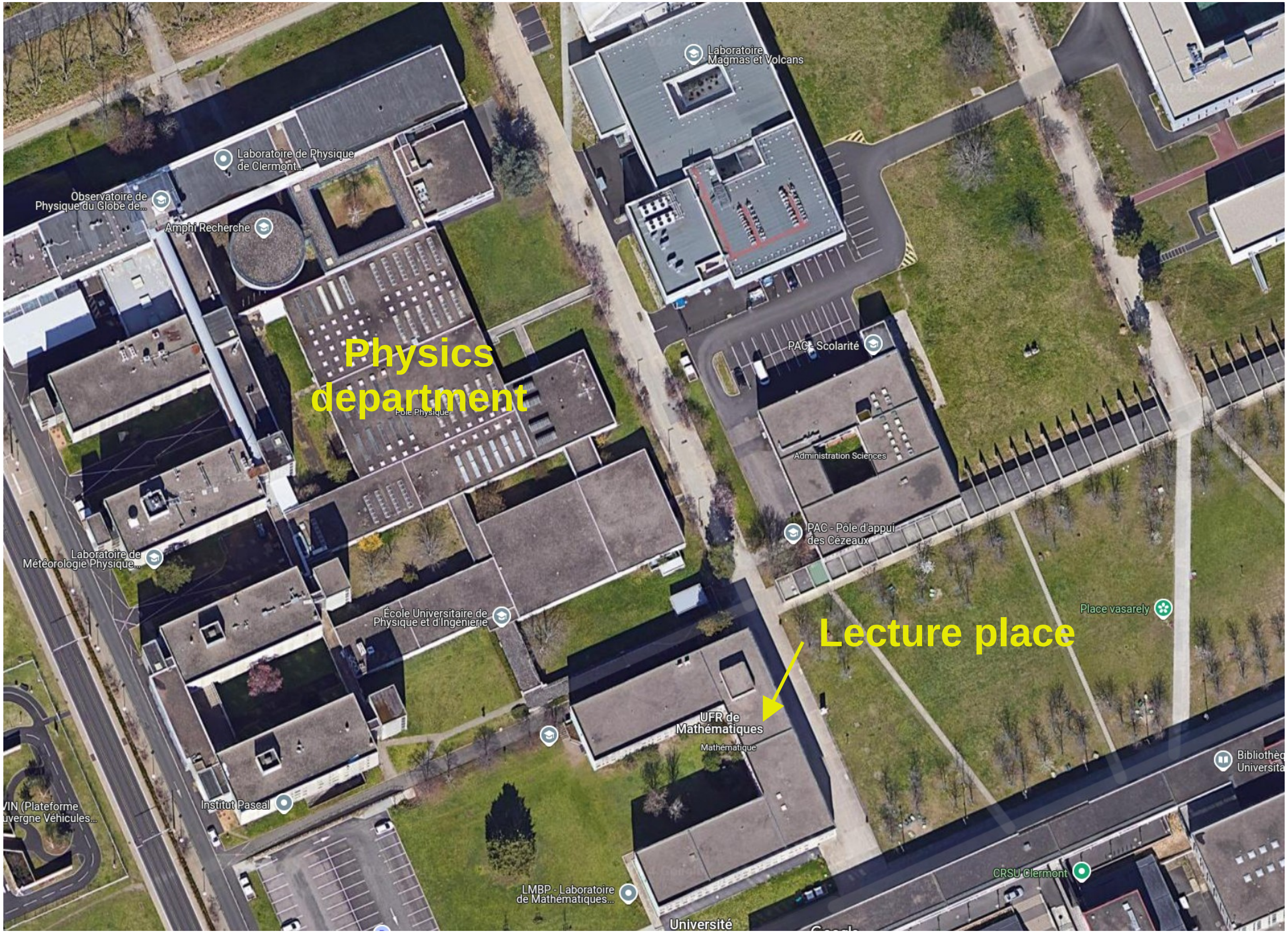


Auto-evaluation of python skills



S38				
Lundi 15/09/2025	Mardi 16/09/2025	Mercredi 17/09/2025	Jeudi 18/09/2025	Vendredi 19/09/2025
Data analysis Python (Romain Madar 1/10 10h00 - 12h30 Intervenant à préciser SCI_004 SCI_002 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar** 3/10 09h30 - 12h30 Intervenant à préciser SCI_008 SCI_007 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar** 5/10 09h30 - 12h30 Intervenant à préciser SCI_008 SCI_007 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar** 7/10 09h30 - 12h30 Intervenant à préciser SCI_004 SCI_002 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar)** 9/10 10h00 - 12h30 Intervenant à préciser SCI_008 SCI_007 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I
Data analysis Python (Romain Madar 2/10 14h00 - 16h30 Intervenant à préciser SCI_008 SCI_007 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar** 4/10 14h00 - 17h00 Intervenant à préciser SCI_008 SCI_007 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar** 6/10 14h00 - 17h00 Intervenant à préciser SCI_004 SCI_002 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar** 8/10 14h00 - 17h00 Intervenant à préciser SCI_008 SCI_007 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I	**Data analysis Python (Romain Madar)** 10/10 14h00 - 16h30 Intervenant à préciser SCI_008 SCI_007 M2 UP gr. 2 M2 UP gr. 1 M2 NP DU I

07h30 08h00- 08h30- 09h00- 09h30- 10h00- 10h30- 11h00- 11h30- 12h00- 12h30- 13h00- 13h30- 14h00- 14h30- 15h00- 15h30- 16h00- 16h30- 17h00- 17h30- 18h00-


Physics department
Lecture place
Laboratoire Magmas et Volcans
Laboratoire de Physique de Clermont...
Observatoire de Physique du Globe de...
Amphi Recherche
PAC - Scolarité
Pôle Physique
Administration Sciences
PAC - Pôle d'appui des Cézeaux
Laboratoire de Météorologie Physique...
École Universitaire de Physique et d'Ingénierie
Place vasarely
UFR de Mathématiques
Mathématique
Institut Pascal
LMBP - Laboratoire de Mathématiques...
CRSU Clermont
Université

Side comment : ChatGPT

Is it needed to learn programing language today,
when tools like ChatGPT exist ?

Let’s try something!

Side comment : ChatGPT

Is it needed to learn programing language today,
when tools like ChatGPT exist ?

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

Let’s try something!

Side comment : ChatGPT

Is it needed to learn programing language today, 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 whole of a diameter d. Write a python function which compute the time it takes for the bucket as function of it's 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/s²).
- $h(t)$ 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:

1. The cross-sectional area of the hole (depending on its diameter d).
2. The cross-sectional area of the bucket.
3. 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:

1. The cross-sectional area of the hole (depending on its diameter d).
2. The cross-sectional area of the bucket.
3. 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 Copy code

```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/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

    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.

    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_pe

# 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

```python
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 ?

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

# Scan over heigh 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]');
```

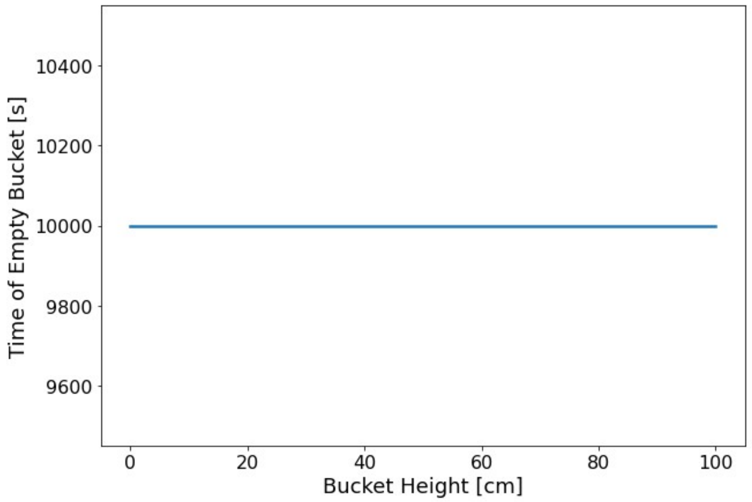
Side comment : ChatGPT

```python
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

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

# Scan over heigh 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

```python
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

```python
# 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

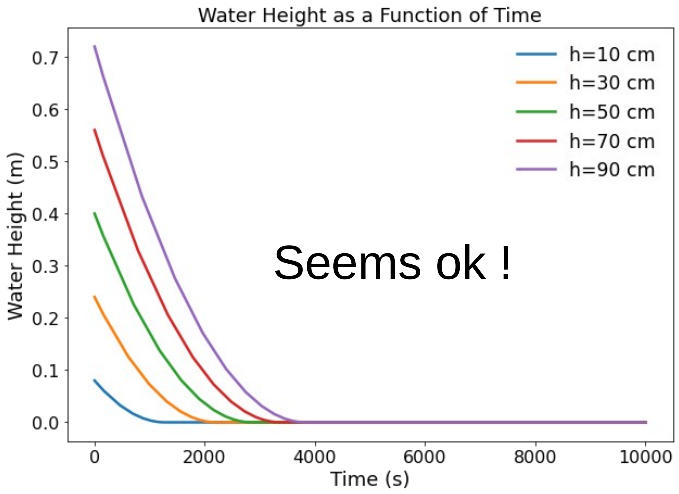
```python
import numpy as np
from scipy.integrate import solve_ivp

# 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

```python
# 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



Seems ok !

Side comment : ChatGPT

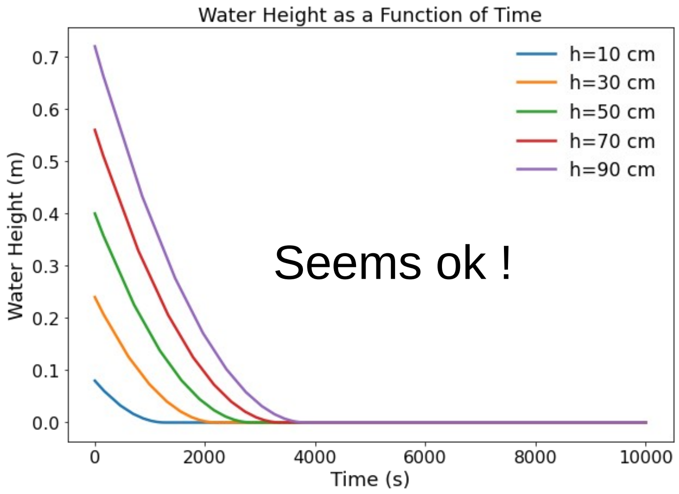
```python
import numpy as np
from scipy.integrate import solve_ivp

# 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

```python
# 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 ...

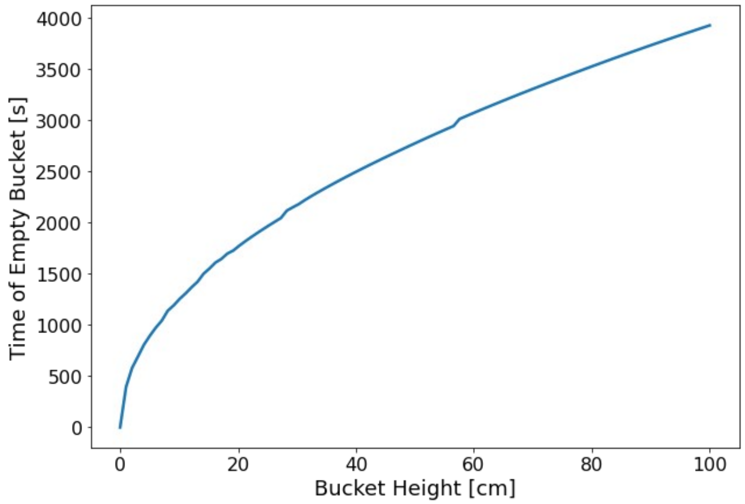
Side comment : ChatGPT

```python
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

```python
# 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.

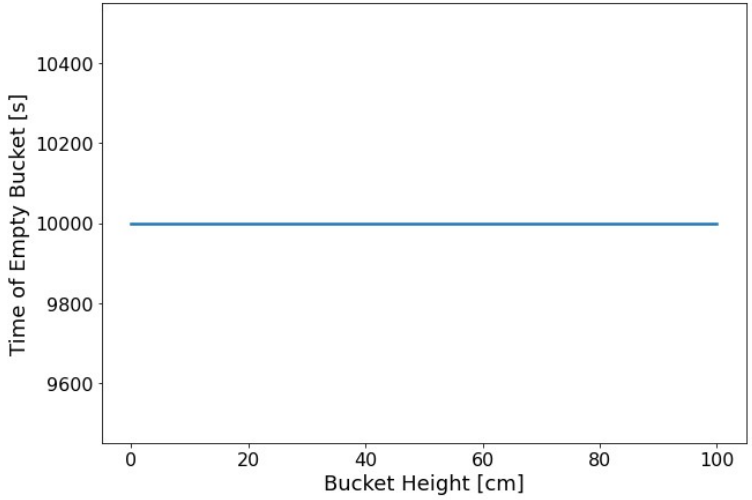
Side comment : ChatGPT

Is it needed to learn programing language today,
when tools like ChatGPT exist ?

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

Well … Yes, it is.

ChatGPT w/o human brain



ChatGPT w/ human brain