

Lecture 1

Introduction to Variables and Control Statements

Jan Tekülve

jan.tekuelve@ini.rub.de

Computer Science and Mathematics
Preparatory Course

21.09.2018

Course Formalities

Goals:

- ▶ Learning Basic-Programming with Python
- ▶ Refreshing Elementary Mathematical Concepts

Concept:

- ▶ Each lecture will be split into a theoretical explanation and a programming session
- ▶ On the last day (05.10.) there will be an ungraded “test”

Overview

1. Motivation

2. Programming

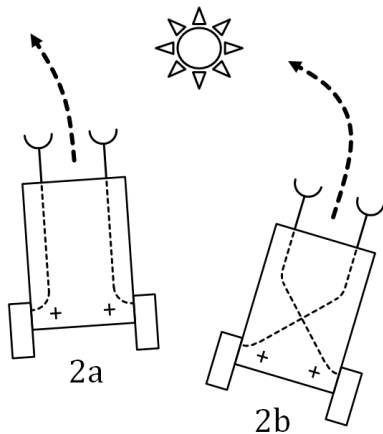
- Set up
- Data Types
- Control Statements
- Utilities

3. Math

- Number Systems

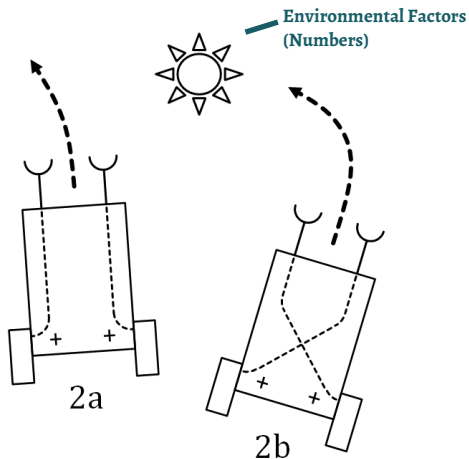
4. Tasks

Motivation: Modeling a cognitive agent



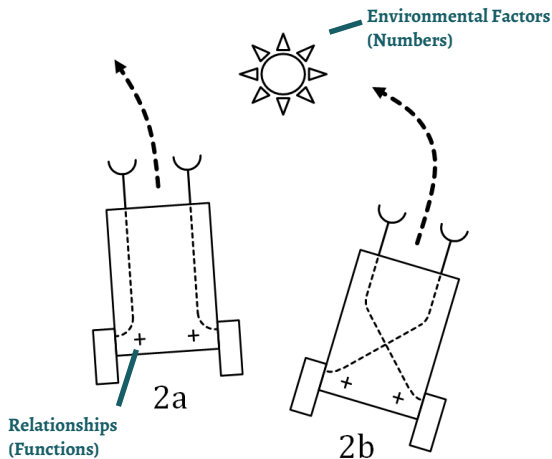
Braitenberg Vehicles

Motivation: Modeling a cognitive agent



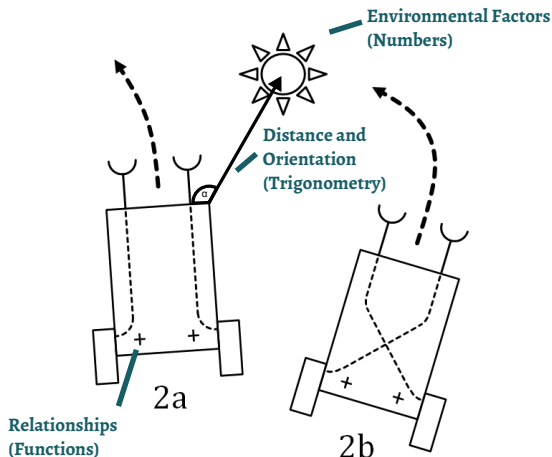
Braitenberg Vehicles

Motivation: Modeling a cognitive agent



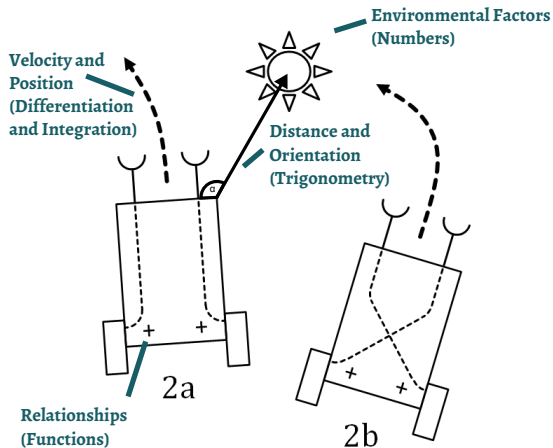
Braitenberg Vehicles

Motivation: Modeling a cognitive agent



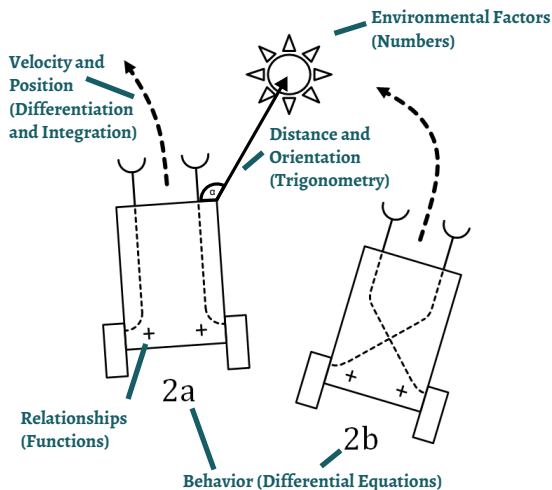
Braitenberg Vehicles

Motivation: Modeling a cognitive agent



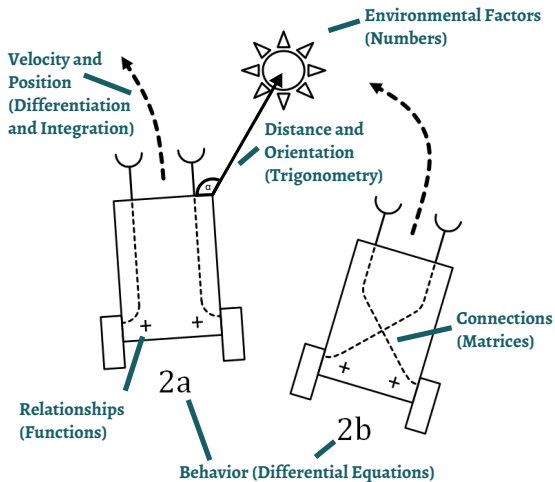
Braitenberg Vehicles

Motivation: Modeling a cognitive agent



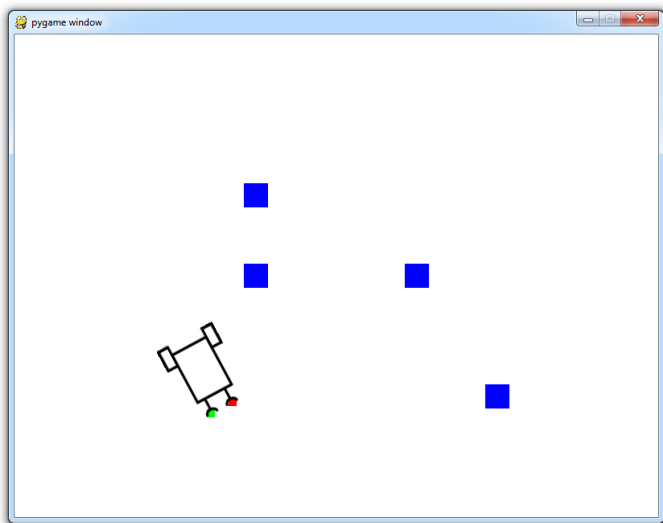
Braitenberg Vehicles

Motivation: Modeling a cognitive agent



Braitenberg Vehicles

Programming Goal



Course Structure

#	Date	Title	Topics
1	21.09.	Variables and Control Statements	<i>Number Systems, Data Types, Control Statements</i>
2	24.09.	Functions in Math and Programming	<i>Function Types and Properties, Plotting Functions, Lists</i>
3	25.09.	Coordinate Systems	<i>Vectors, Trigonometry, The Pygame Module</i>
4	26.09.	Sequences	<i>Sequences and Series, Limits, Recursiveness</i>
5	27.09.	Differentiation	<i>Derivative Definition, Calculating Derivatives, Numerical Differentiation, File-Input/Output</i>

Course Structure

#	Date	Title	Topics
6	28.09.	Integration	<i>Geometrical Definition, Calculating Integrals, Numerical Integration</i>
7	01.10.	Differential Equations	<i>Properties of Differential Equations, Euler Approximation, Braitenberg Vehicle</i>
8	02.10.	Matrices	<i>Matrix Addition, Matrix Multiplication, Basic Neural Networks</i>
	03.10.	HOLIDAY	
9	04.10.	Make a Wish Lecture	<i>Individual Wishes, Finish Programming</i>
10	05.10.	Repetition Lecture and "Test"	<i>Repetition of Core Concepts and Ungraded Test</i>

1. Motivation

2. Programming

- Set up
- Data Types
- Control Statements
- Utilities

3. Math

- Number Systems

4. Tasks

Python

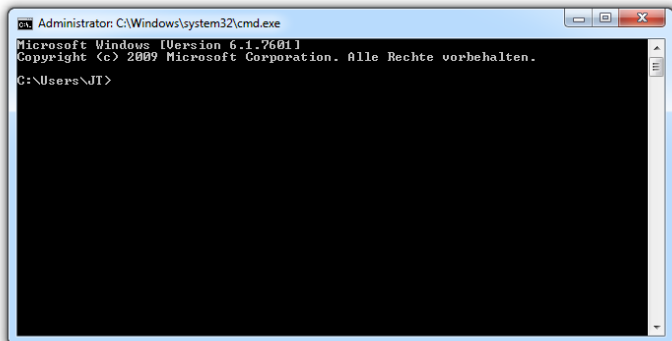
Why Python?

- ▶ It is simple but high level
- ▶ It is interpreted “on the fly”
- ▶ It is the state of the art scripting language

Setting up Python

- ▶ Download Python3 from: <https://www.python.org/downloads/>
- ▶ To code you will need a simple text editor (e.g. Notepad)
- ▶ For further info look here: <https://docs.python.org/3/>

Excursion: Terminal/Command Prompt

A screenshot of a Windows Command Prompt window. The title bar reads "Administrator: C:\Windows\system32\cmd.exe". The window content shows the following text:

```
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.  
C:\Users\JT>
```

The Terminal/Command Prompt offers a simple Input/Output Interface

Excursion: Terminal/Command Prompt

Open the Terminal/Prompt:

Windows

Click on Start and type *cmd*

Mac OS

Click on the Spotlight Icon and type
terminal

Starting Python

- ▶ Type *python --version* into the terminal and press return
- ▶ If you have version 3: **You are fine!**
- ▶ If you have version 2: **You are fine, but be aware!**
- ▶ If an error message pops up: **Follow me along in the next steps!**

Setting up the Path variable

Windows

1. Find out the
/path/where/python/is/located/
2. Access 'System Settings' from your Control Panel.
3. Click on the 'Advanced' tab.
4. Click on the 'Environmental Variables' button on the bottom of the screen
5. Locate 'Path' under the 'System Variables' section and Click on 'Edit'
6. At the end type ';' followed by the python installation path

Mac

1. Open '/etc/paths'
2. Add the line at the end:

```
/usr/local/bin
```

3. Save the file

If it does not work:

1. Let's fix this later
2. For now open:
<https://trinket.io/features/python3>

Setting Up

Set up your personal environment

- ▶ Create a folder for your python projects
- ▶ Open the terminal/command-prompt and navigate to folder using the `cd` command

```
cd /path/to/your/created/folder
```

- ▶ Inside the folder create a file called *helloworld.py* and open it

Hello World

- ▶ Write the following line into the file:

```
print("Hello World!")
```

- ▶ Type the following in the terminal/command prompt

```
python helloworld.py
```

and press return to execute the script

Variables

- ▶ Variables are the elementary building block of every program

Variables

- ▶ Variables are the elementary building block of every program
- ▶ Variables store information of various type:

```
greeting = "Hello, Hello!" # String Type  
print(greeting) # prints "Hello, Hello!"  
#Comments can be written after a leading #
```

Variables

- ▶ Variables are the elementary building block of every program
- ▶ Variables store information of various type:

```
greeting = "Hello, Hello!" # String Type
print(greeting) # prints "Hello, Hello!"
#Comments can be written after a leading #
```

- ▶ Variables are assigned via '='

```
a = 5 # Integer Type
b = 3.0 # Float Type
c = a # c is 5
d = b + c # d is 8.0
```

Operations on Data Types

► Operations on Numbers

`2+2 #4`

`50-5*6 #20`

`(50-5*6)/4 #5.0`

`8/5 #1.6`

`17/3 #5.666666666666667`

`17//3 #5 Integer Division`

`17%3 #2 Rest of the Division`

► Operations on Strings

`'Wo' + 'rd' #'Word' or "Word"`

`'Isn't' # This results in an error!`

`'Isn\'t' #'Isn't' Use \ to escape characters`

Control Statements

- ▶ if-Statement

```
x = 3.5
if x > 0 : #Indentation organizes blocks
    print("x is positive!")#Indent with 4 spaces
print("Program is finished!")
```

Control Statements

▶ if-Statement

```
x = 3.5
if x > 0 : #Indentation organizes blocks
    print("x is positive!")#Indent with 4 spaces
print("Program is finished!")
```

▶ else-statement

```
x = 3.5
if x > 0 : #Indentation organizes blocks
    print("x is positive!")#Indent with 4 spaces
else :
    print("x is not positive!")
print("Program is finished!")
```

Control Statements

- ▶ else if-statement

```
x = 3.5
if x > 0 : #Indentation organizes blocks
    print("x is positive!") #Indent with 4 spaces
elif x < 0 :
    print("x is negative!")
else:
    print("x is zero!")
print("Program is finished!")
```

Variable Scope

- ▶ Python code is organized in blocks by indentation (4 spaces)

```
a = 3  
b = 4  
if a > 2:
```

```
    c = a + b  
    b = 1  
    if c > 5:
```

```
        print(a)
```

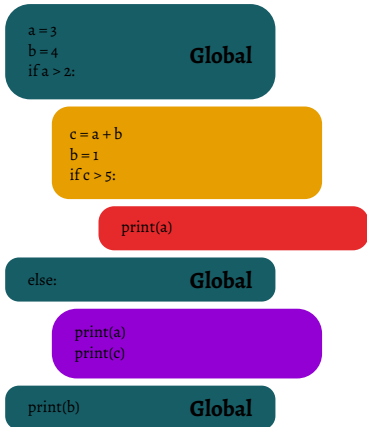
```
else:
```

```
    print(a)  
    print(c)
```

```
print(b)
```

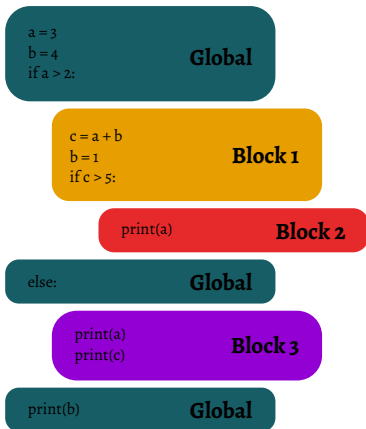
Variable Scope

- ▶ Python code is organized in blocks by indentation (4 spaces)
- ▶ Variables defined in the global scope are available at all positions in the code below its definition



Variable Scope

- ▶ Python code is organized in blocks by indentation (4 spaces)
- ▶ Variables defined in the global scope are available at all positions in the code below its definition
- ▶ Variables defined in a block are available in the block and all blocks inside it



Variable Scope

► Example

```
a = 3 # Global Scope
b = 4
if a > 2 :
    c = a + b # Block 1
    b = 1
    if c > 5:
        print(a) # Block 2
else : # Global
    print(a) # Block 3
    print(c) # If a <= 2 this will result in an error
print(b) # '1' or '4' if a <= 2
```

While Loops

- ▶ Print the numbers from 1 to 10

```
a = 0
while a < 10 :
    a = a +1 # Increase a by 1
    print(a)
```

- ▶ Be careful with the exit condition

```
a = 0
while a < 10 :
    print(a) # Prints 0 until the end of time
```

You can kill the running program by pressing Ctrl+C

Boolean Statements

► Examples

`3 > 2 #True, greater than`

`3 < 3 #False, less than`

`3 <= 3 # True, equal or less than`

`4 == 5 # False, == checks equality`

`4 != 5 # True, != is the opposite of ==`

`"ello" in "Hello" # True, only works for sequence types`

`"hel" not in "Hello" # True, "in" is case sensitive`

Boolean Statements

▶ Examples

```
3 > 2 #True, greater than
```

```
3 < 3 #False, less than
```

```
3 <= 3 # True, equal or less than
```

```
4 == 5 # False, == checks equality
```

```
4 != 5 # True, != is the opposite of ==
```

```
"ello" in "Hello" # True, only works for sequence types
```

```
"hel" not in "Hello" # True, "in" is case sensitive
```

▶ Boolean Variables

```
test = 7
```

```
isGreaterThanOne = test > 1
```

```
if isGreaterThanOne:
```

```
    print("The number is Greater than 1!")
```

User Input

- ▶ Use input to prompt the user

```
person = input('Enter your name: ')\nprint('Hello ' + person)
```

User Input

- ▶ Use input to prompt the user

```
person = input('Enter your name: ')\nprint('Hello ' + person)
```

- ▶ Invalid Data Types

```
inputValue = input('Please enter a number: ')\nresult = 5 + inputValue # This results in an error!
```

User Input

- ▶ Use input to prompt the user

```
person = input('Enter your name: ')\nprint('Hello ' + person)
```

- ▶ Invalid Data Types

```
inputValue = input('Please enter a number: ')\nresult = 5 + inputValue # This results in an error!
```

- ▶ Variables might need to be *type casted*

```
result = 5 + float(inputValue)\n#This works if an actual number was typed
```

Type Casting

- ▶ Implicit Typecast

```
a = 1.0 #float
```

```
b = 2 #int
```

```
c = a + b #3.0 float
```

Type Casting

► Implicit Typecast

```
a = 1.0 #float
b = 2 #int
c = a + b #3.0 float
```

► Explicit Typecasts

```
d = float(b) #2.0
e = 3.7
f = int(3.7) #3 Any floating point is cut off
g = str(e) #String '3.7'
h = int(g) # This results in an error!
i = float(g) # 3.7
print('Variable i is: ' +str(i)) #Print expects strings
```

Useful built-in Functions

- ▶ Rounding and Absolute Value

```
a = 3.898987897897
b = round(a,3) #3.899
c = abs(-3.2) #|-3.2| = 3.2
t = type(c) #t is <class 'float'>
test = t is float # True
```

- ▶ The math module

```
import math #Import makes a module available
squareTwo = math.sqrt(2) # $\sqrt{2}$ 
power = math.pow(3,4) #  $3^4$ 
exponential = math.exp(4) # $e^4$ 
piNumber = math.pi #3.14159265359
```

1. Motivation

2. Programming

- Set up
- Data Types
- Control Statements
- Utilities

3. Math

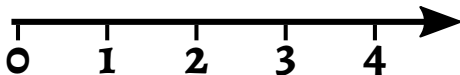
- Number Systems

4. Tasks

Definitions

Number Systems

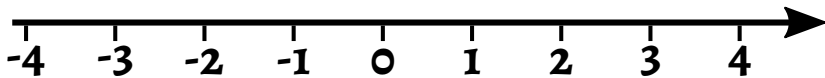
- ▶ **Natural Numbers:** $\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}$
- ▶ **Integer Numbers:** $\mathbb{Z} =$
- ▶ **Rational Numbers:** \mathbb{Q}
- ▶ **Real Numbers:** \mathbb{R}



Definitions

Number Systems

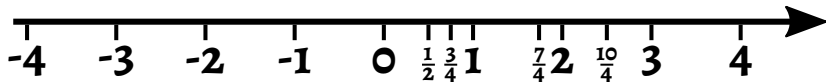
- ▶ **Natural Numbers:** $\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}$
- ▶ **Integer Numbers:** $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- ▶ **Rational Numbers:** \mathbb{Q}
- ▶ **Real Numbers:** \mathbb{R}



Definitions

Number Systems

- ▶ **Natural Numbers:** $\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}$
- ▶ **Integer Numbers:** $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- ▶ **Rational Numbers:** $\mathbb{Q} = \frac{a}{b}$, where $a, b \in \mathbb{Z}$ and $b \neq 0$
- ▶ **Real Numbers:** \mathbb{R}



Real Numbers

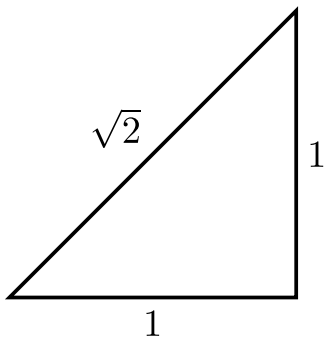
- ▶ Between two rational numbers is an infinite amount of rational numbers

Real Numbers

- ▶ Between two rational numbers is an infinite amount of rational numbers
- ▶ However: $\sqrt{2}$ is not a rational number

Real Numbers

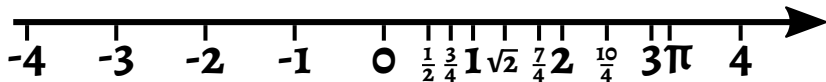
- ▶ Between two rational numbers is an infinite amount of rational numbers
- ▶ However: $\sqrt{2}$ is not a rational number
- ▶ The irrational number $\sqrt{2} = 1.4142135 \dots$ is part of the real world:



Definitions

Number Systems

- ▶ **Natural Numbers:** $\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}$
- ▶ **Integer Numbers:** $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- ▶ **Rational Numbers:** $\mathbb{Q} = \frac{a}{b}$, where $a, b \in \mathbb{Z}$ and $b \neq 0$
- ▶ **Real Numbers:** $\mathbb{R} = \mathbb{Q} + \text{irrational numbers}$



Definitions

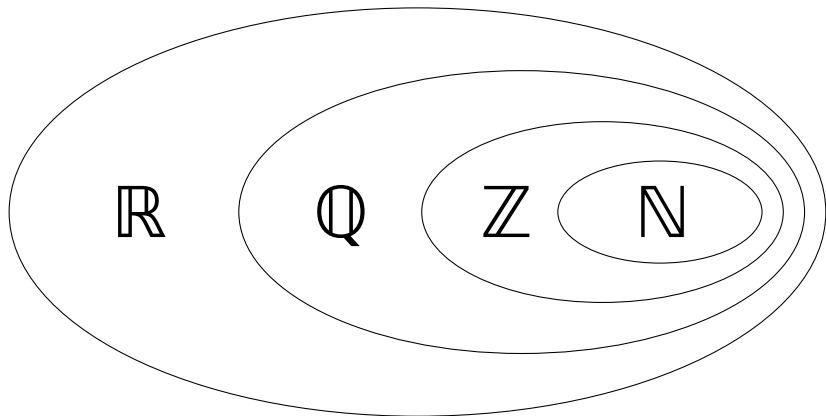
Number Systems

- ▶ **Natural Numbers:** $\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}$
- ▶ **Integer Numbers:** $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- ▶ **Rational Numbers:** $\mathbb{Q} = \frac{a}{b}$, where $a, b \in \mathbb{Z}$ and $b \neq 0$
- ▶ **Real Numbers:** $\mathbb{R} = \mathbb{Q} + \text{irrational numbers}$

Honorable Mention

- ▶ **Complex Numbers:** $\mathbb{C} = a + ib$, where $a, b \in \mathbb{R}$ and $i = \sqrt{-1}$

Number Systems



Tasks

1. Write a Script that determines whether a given Input number is an Integer or Rational Number. Print the result to the console.
 - ▶ Use python's *input* function to retrieve the input number. Assume that the user types only numbers.
 - ▶ Use type casting to convert into the correct data types
 - ▶ Be aware that the rationals include all integers, but not the other way round.
2. Write a Script that finds a fraction $\frac{a}{b}$, where $a, b \in \mathbb{I}$ that resembles the first four digits of $\sqrt{2} = 1.4142$. Use a *Brute Force* approach that tries out possible values for a and b until a solution is found.
 - ▶ Start with $a = 1$ and $b = 1$
 - ▶ Use a loop that exits when the rounded $\frac{a}{b}$ equals 1.4142
 - ▶ In each loop iteration adapt the values for a and b by incrementing them or setting them to 1.

Lecture Slides/Material

Type the following URL to access the lecture slides:

https://www.ini.rub.de/teaching/courses/c_science_math