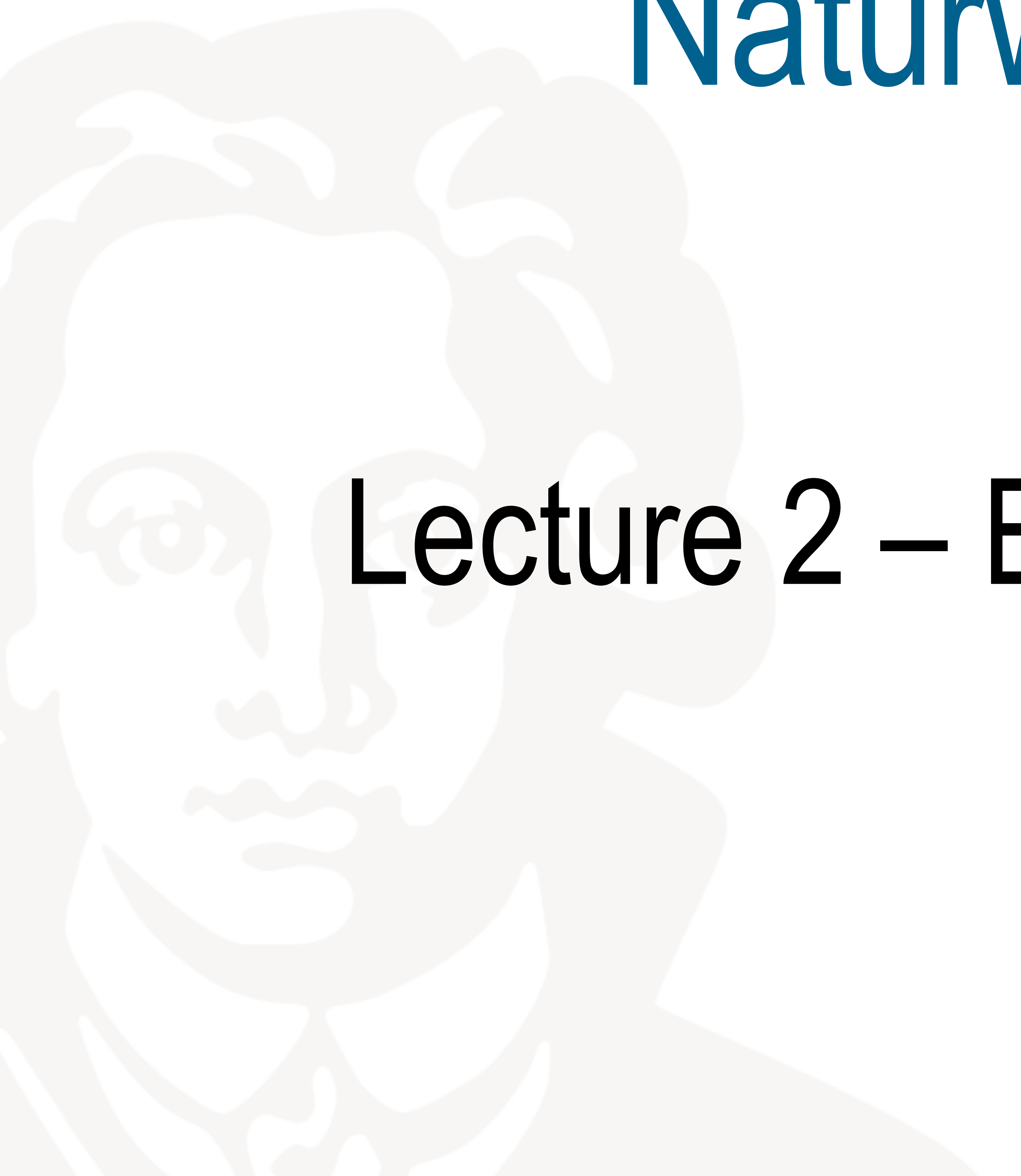
Prof. Dr. Gemma Roig
M.Sc. Alperen Kantarcı
M.Sc. Gamze Akyol

Programmieren für Studierende der Naturwissenschaften



Lecture 2 – Elementary Data Types and Control Structures

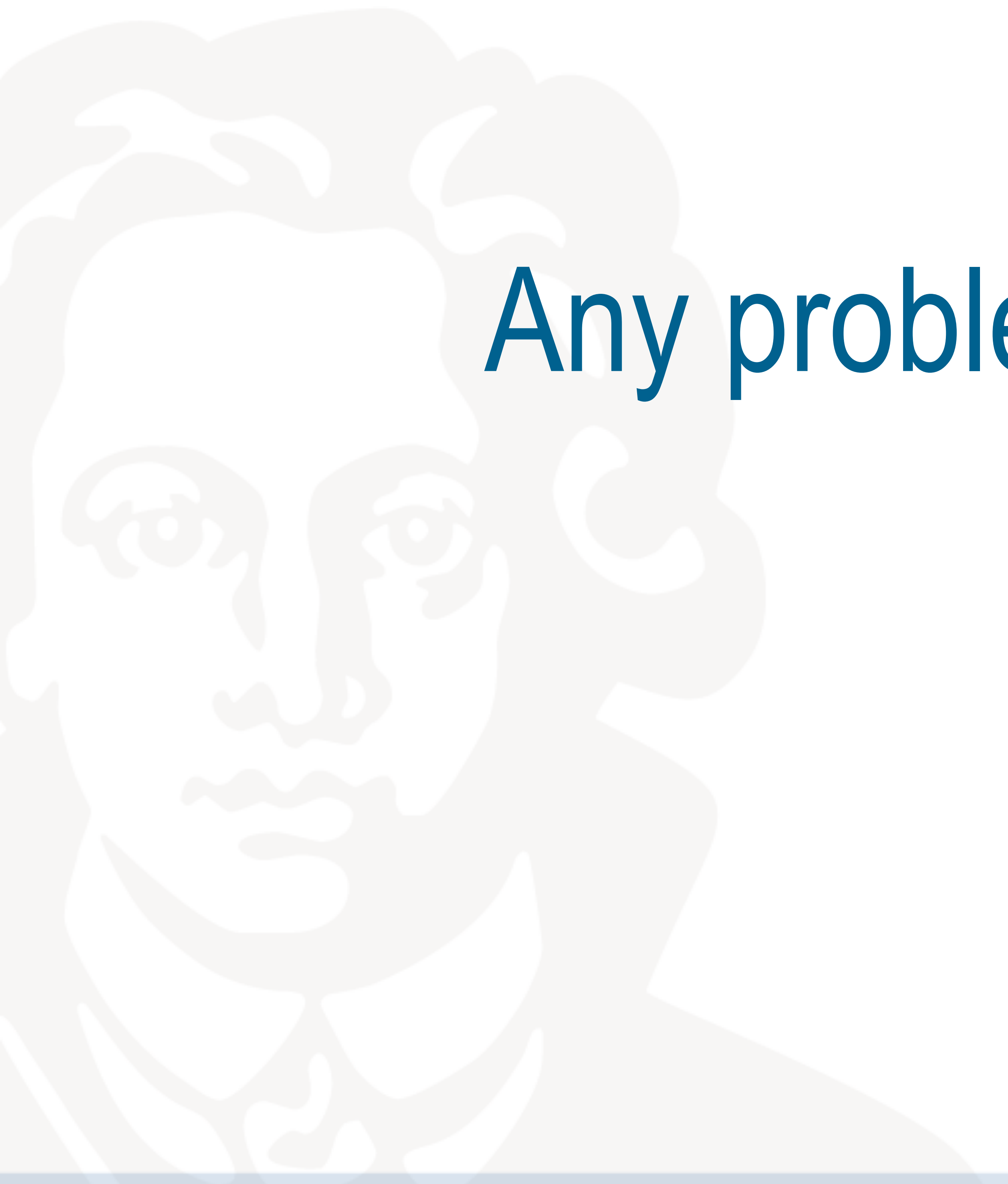
Contents

- L1: Basics of programming
  P1: Exercise 1 and help to installments.
- L2: Elementary data types and control structures
  P2: Exercises
- L3: Aggregated data types
  P3: Exercises
- L4: Aggregated data types and functions
  P4: Exercises
- L5:Testing,error messages and self-help
  P5: Exercises

Contents

- L6: External Packages, Introduction NumPy and SciPy
  P6: Exercises
- L7:External Packages 2
  P7: Exercises
- L8: Handling external data and visualization
  P8: Exercises
- L9: Design of algorithms
  P9: Exercises (not graded) and independent work in small groups

GOETHE UNIVERSITÄT FRANKFURT AM MAIN



Before starting

Any problems regarding the organization and content?

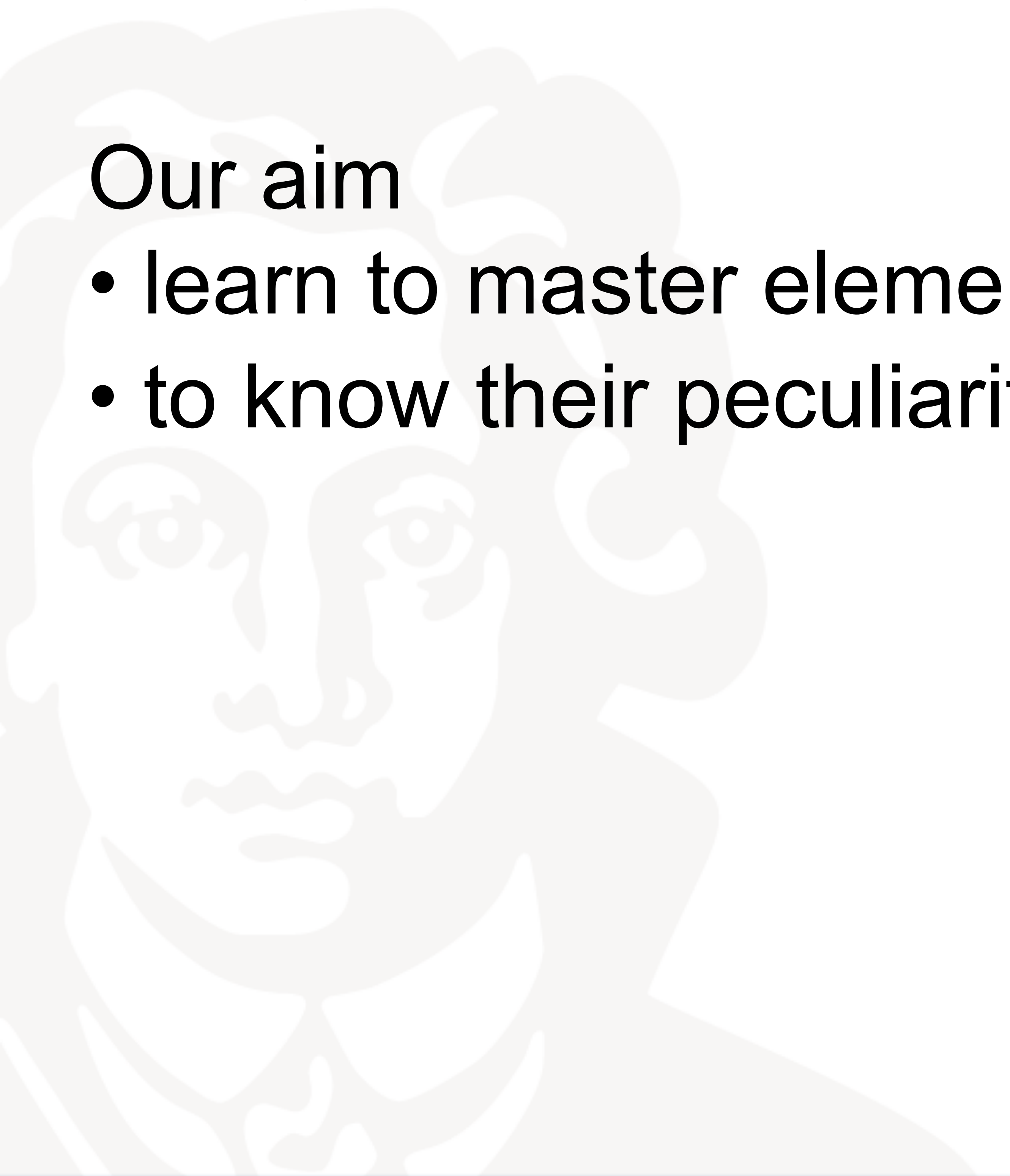
Review of variables

A variable can be conceptually considered as a container in memory

- Three attributes: Name, Type, Value

1. Name: a unique word in a namespace (here initially the entire program) under which the variable can be addressed in the program text
2. Type: Describes the type of the variable to indicate the possible set of operations
3. Value: Content of the variable (depends on the type)
   Example: DIX (surname of Otto DIX or Roman numeral 509?)

Elementary data types…



- are directly supported by the programming language
  - and can differ from language to language,
- can only accept one value of the own value range
- are discrete and finite

- They can often be handled efficiently by appropriate hardware components

Our aim
- learn to master elementary data types of Python
- to know their peculiarities

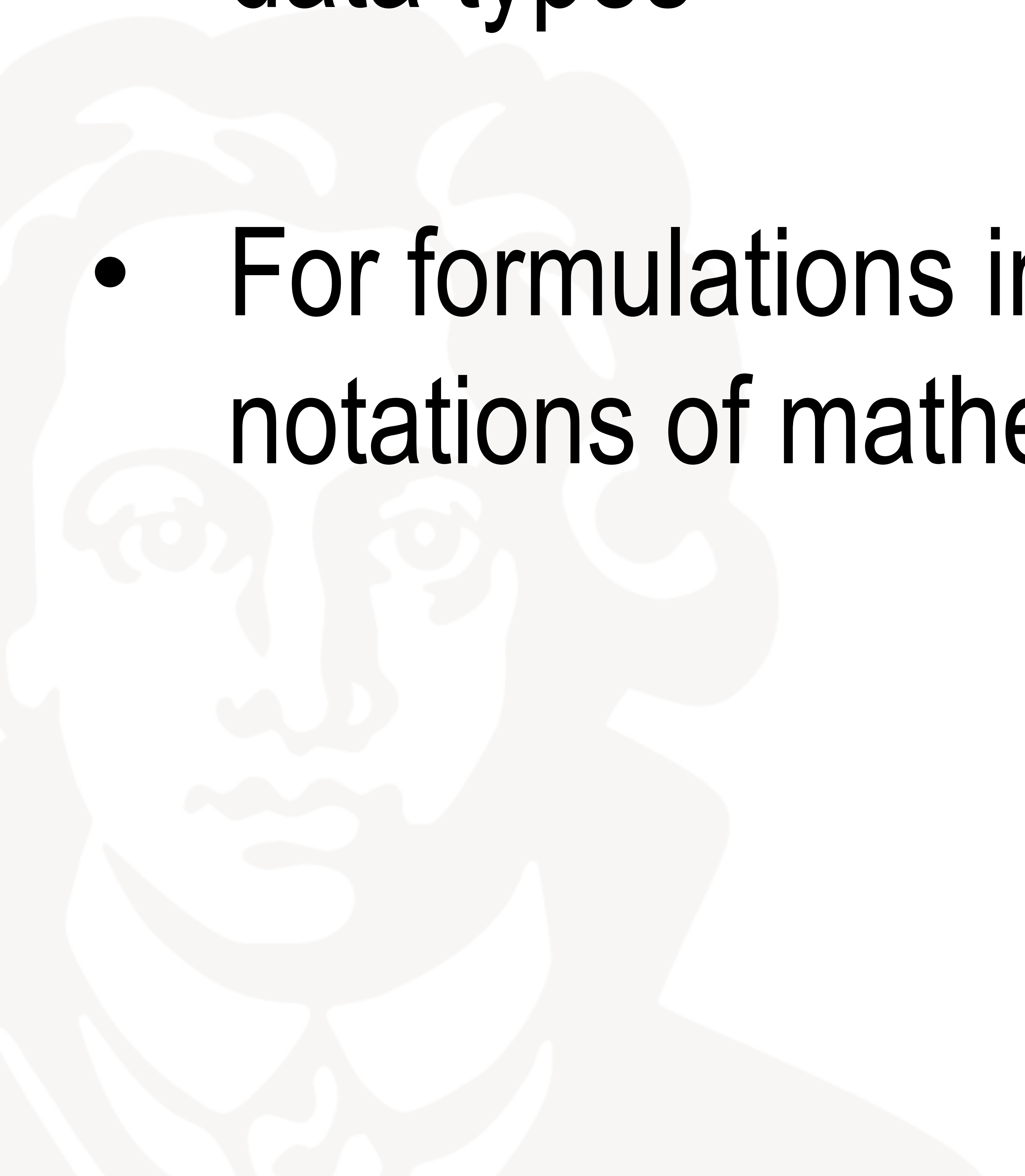
Overview

- Numeric data types
  - Integers (integer)
  - Floating point number

- Boolean data type
- characters? (Strings)

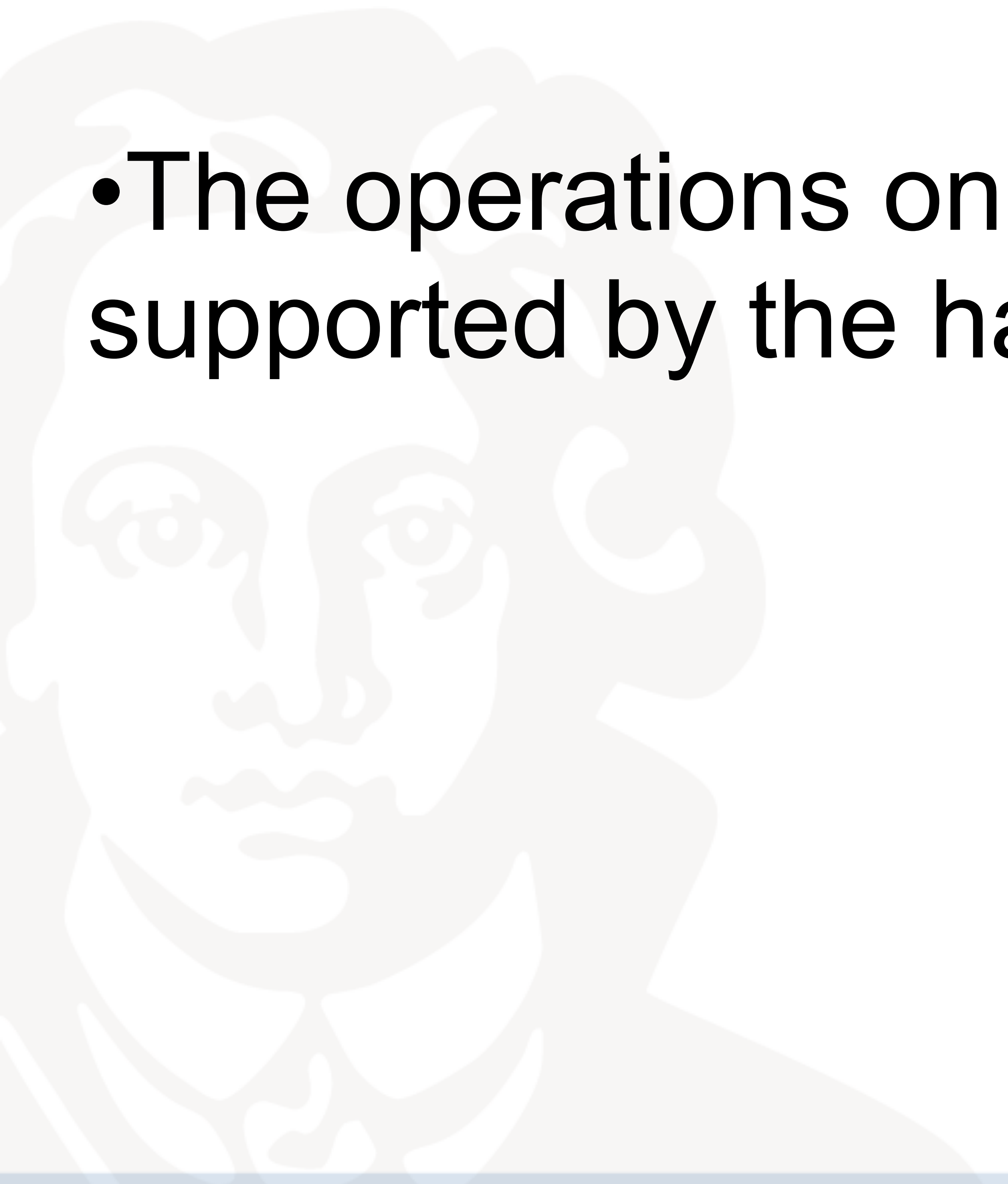
- In addition: operators and operations, respectively
- Type conversion (casting)

Basics



- Numbers and **characters** are undoubtedly elementary data structures
- Every common programming language provides these data types as elementary data types
- For formulations in programming languages, one tries to keep the usual notations of mathematics as far as possible.

Integer Coding



- The binary place value system is used as a basis

- Positive and negative numbers can be represented in this way (two's complement)

- The operations on numbers (addition, subtraction, multiplication, etc.) are supported by the hardware.

Place value systems

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- "Normally" we use the decimal system (numbers in base 10, i.e. the digits from 0 to 9 occur)
- e.g.: 23910 = 2*10^4 + 3*10^3 + 9*10^2 + 1*10^1 + 0*10^0
- However, any natural number can also be represented in a different number system.
- Examples: 1101(2) = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 13(10)

  15(oct)=13(10)

  13:8 = 1 Remainder: 5

  1:8 = 0 Remainder:1

  D(hex)=13(10)
- In the binary system, a character is also called a "bit". This has exactly 2 possible states: on or off (respectively 1 or 0).
- Not included today: Coding of signed numbers (two's complement)

10010001

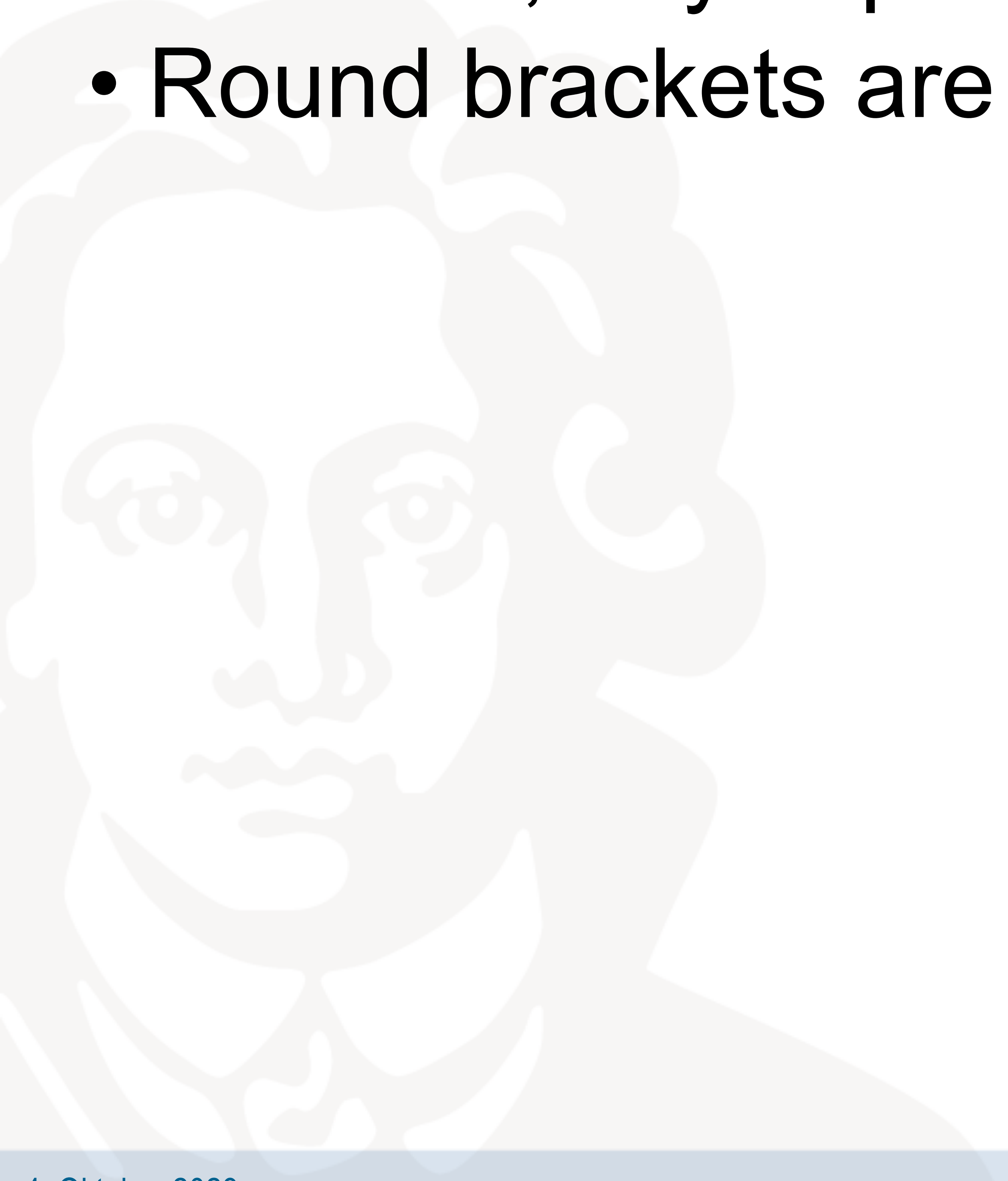
It's clearly one hundred forty five

Obviously, it's ten million ten thousand one

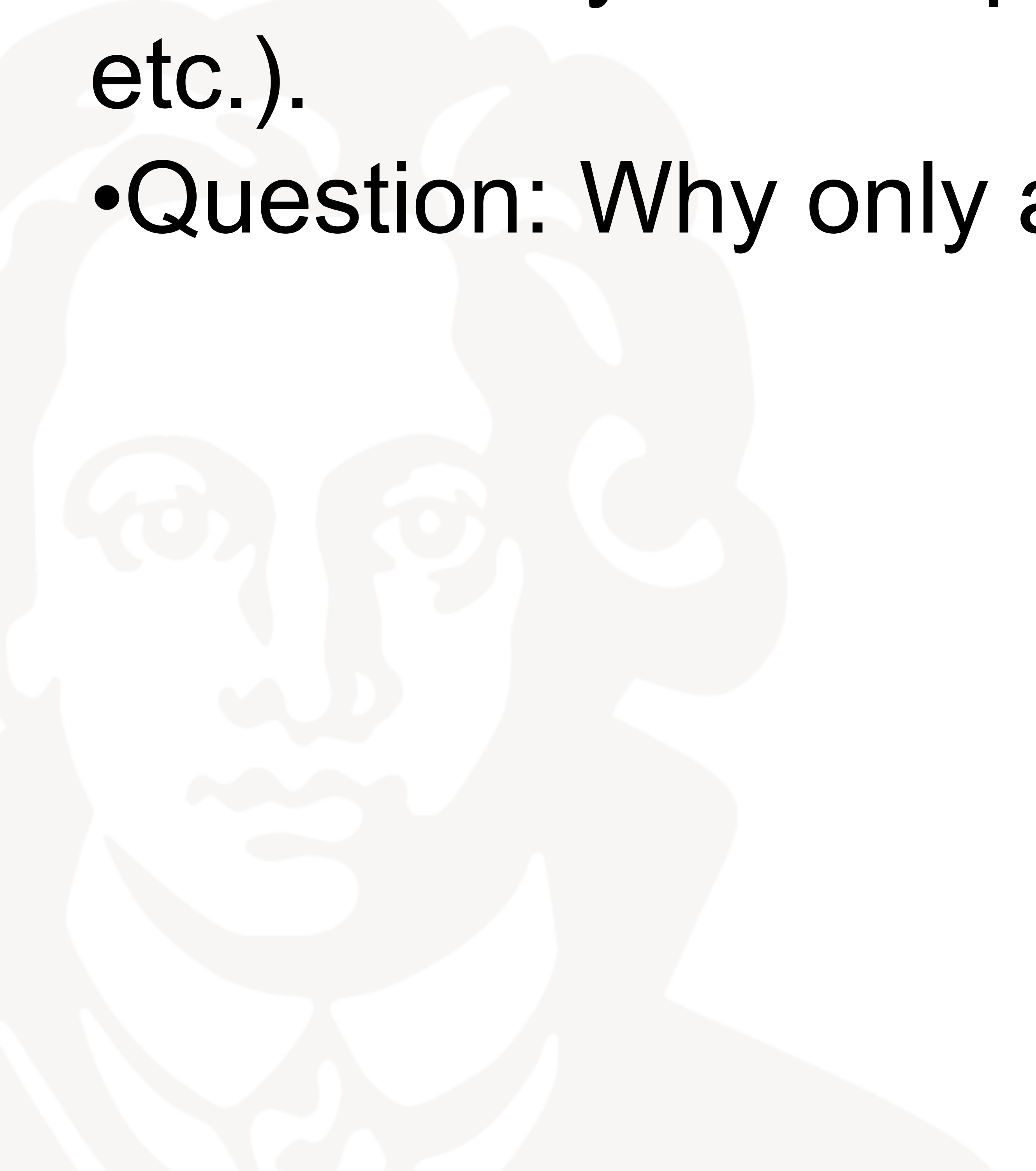
Source - Edlitera

Integer operations

- For integer operands, ALL common operators are defined in Python (+,-, /, etc.)
- Integer division is // and modulo is %
- There are also bit-wise operators (x<<y , x>>y , x&y , x^y , x|y)
- Logical operators (and, or, not)
- With this, any expressions (terms) can be generated as usual in mathematics.
- Round brackets are available to control the evaluation order.



Floating Point Numbers



- Floating point number () A mostly approximated (i.e.approximated) encoding of a rational or real number in a fixed number of bits (mostly 32, 64, more rarely 16, 128 or even 256 bits).
- The set of floating point numbers is a finite subset of the rational numbers, usually extended by some special elements (+infinity, -infinity, NaN (="Not A Number "), -0, etc.).
- Question: Why only approximation? Isn’t any number representable in this way ?

Floating Point Numbers – Display problems

- Limitation of the numbers that can be represented by memory space available per number.

Example:

- I would like to map the number 0.2356
- Assuming I have only a maximum of 2 decimal places available
- No exact representation is possible!
- The best that is possible would be 0.24

Floating Point Numbers – Display problems

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

First of all: Why is this important at all? We just want to program.

- Answer: Error propagation by rounding!

How are floating point numbers displayed on the computer?

Compared to an integer representation, floating point numbers can cover a much larger range of values with the same memory requirements.

- Example:

32 bit two's complement: -2,147*10^9 ≤ z ≤ 2,147* 10^9
32 bit floating point number (IEEE 754): -3,403*10^{38} ≤ z ≤ 3,403* 10^{38}

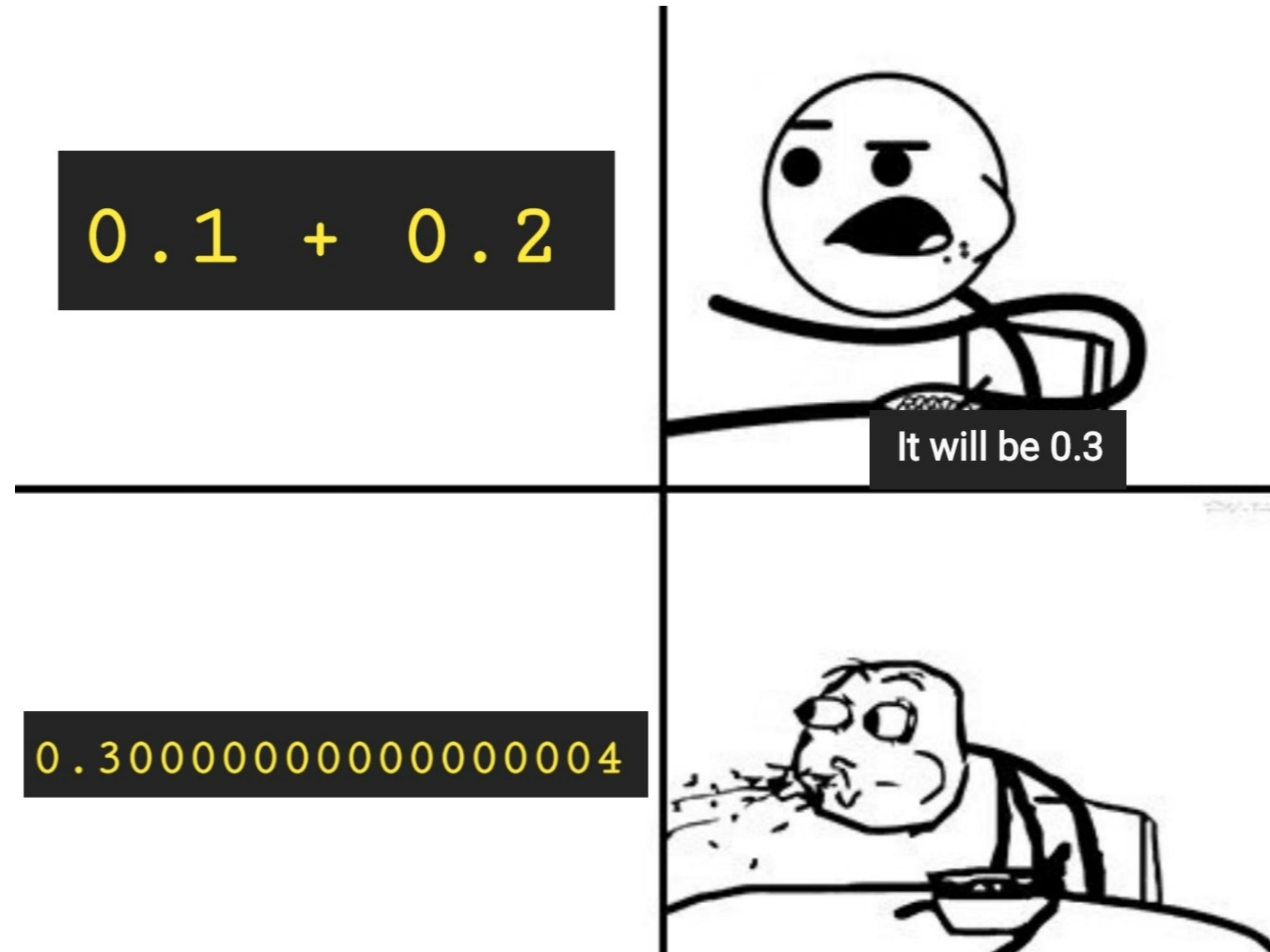
Float Operators

The "normal" operations :`+, -, *, /, **`

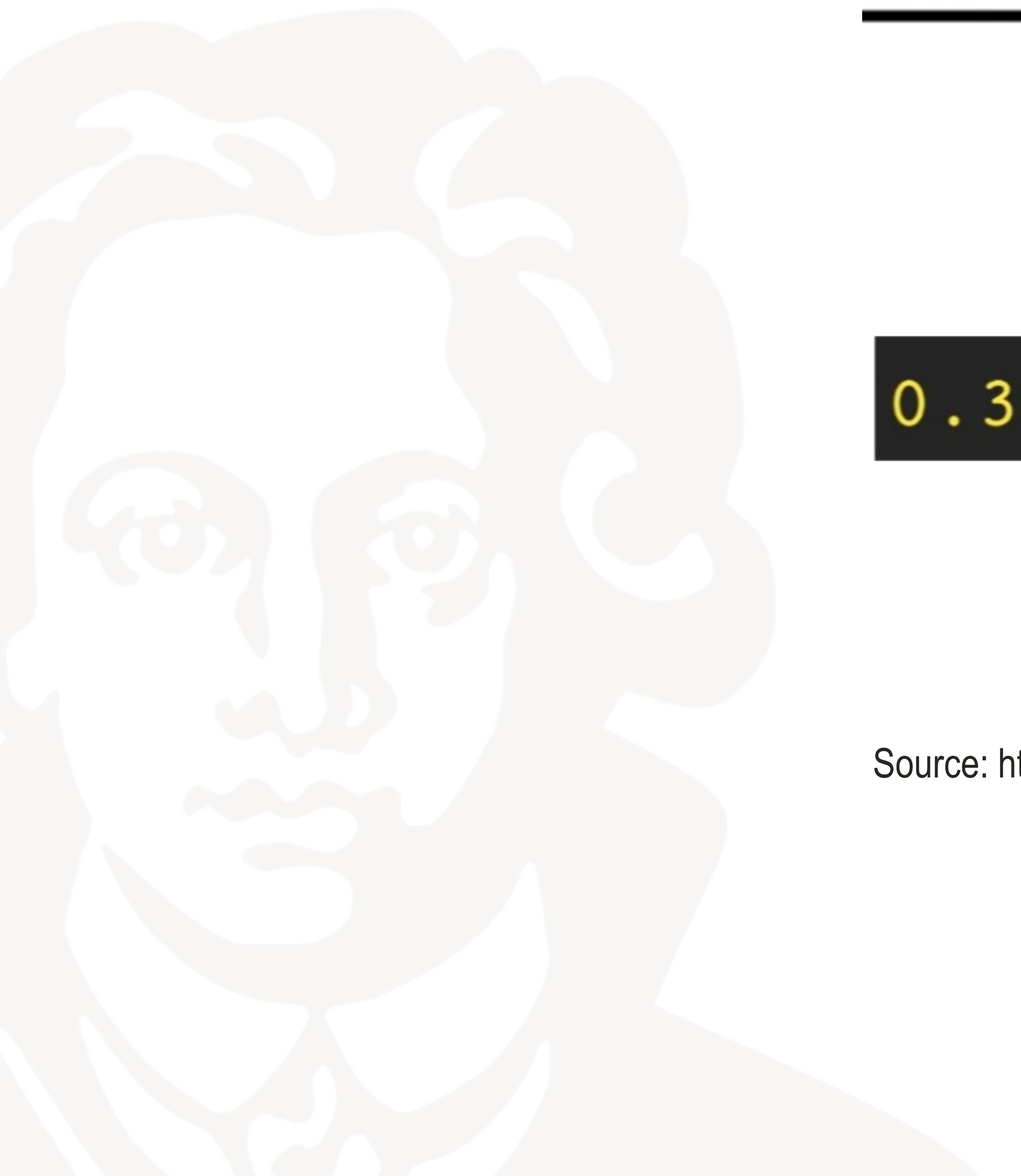
- pow(X,Y[,Z]): X to the power Y [modulo Z].
- round(X [, N]): Default N = 0. Returns a float rounded to N decimal places after the decimal point.

- Also work with floats:
- The integer division X // Y → Returns the "integer" result of a division
- The modulo operator X % Y → Returns the "decimal part" of a division

Float Operators



Source: https://www.reddit.com/r/ProgrammerHumor/comments/q4n3m6/damn_you_floating_point_binary_addition_youve



Boolean data type (truth values)

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- The smallest memory element of a computer is a memory location, a "bit", whose value range is given by two states 0 and 1.

In the following we denote the value 0 as False and 1 as True
The corresponding data type is called "Boolean" (after George Boole).

Special features of Python: Every variable (every expression) can be "interpreted" as boolean

- False for numeric data types if the value is 0
- False for string data types, with length 0
- True for numeric data types if a value $\neq$ 0 is present.
- True string (sequential) data types with length $\neq$ 0 has
- This is convenient in many cases, but has several consequences

Boolean data type (truth values) - Operators

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- Logical operators: and, or, not

Boolean data type (truth values) - Caution

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- Logical operators: and, or, not

```
Python 3.5.3 (default, Apr 22 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more information.
>>> a = True
>>> b = False
>>> a and b
False
>>> a or b
True
>>> a + b
1
>>> a + a
2
>>> b - a
-1
>>> type(a)
<class 'bool'>
>>> type(a-b)
<class 'int'>
```

Boolean data type (truth values) - Comparison

- ... (<, >, <=, >=, !=, ==) return True and False as result

```
Python 3.5.3 (default, Apr 22 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more info
>>> a = 1
>>> b = 2
>>> c = 3
>>> a < b
True
>>> a > b
False
>>> a <= b
True
>>> a + b == c
True
>>> a != b
True
>>>
```

Boolean data type (truth values) - Comparison

- Besides the ones just mentioned, there are also:
  - is
  - is not
- These compare on identity id() (thus whether the same memory location is addressed!).
- Test in the exercise!

```
Python 3.5.3 (default, Apr 22 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more information.
>>> x = [1,2,3]
>>> y = [1,2,3]
>>> x == y
True
>>> x is y
False
>>>
```

Strings

- Strings are written in (above) "quotation marks", whether single ′ or double "

- Here, many programmers experience surprises once, because there is a "chaos" here.

- What chaos exactly?
  - There are still several variants of the apostrophe, see e.g. in Wikipedia, and other special characters
  - The programming language editors (e.g. Idle) of course always get it right
  - But be careful if you **take** code from Word, PowerPoint, or similar!

String Operators

The operators + and * also work with strings, but have a special meaning:

```
>>> s = "Hallo"
>>> t = "Welt"
>>> s + t
'HalloWelt'
>>> s*3
'HalloHalloHallo'
>>>
```

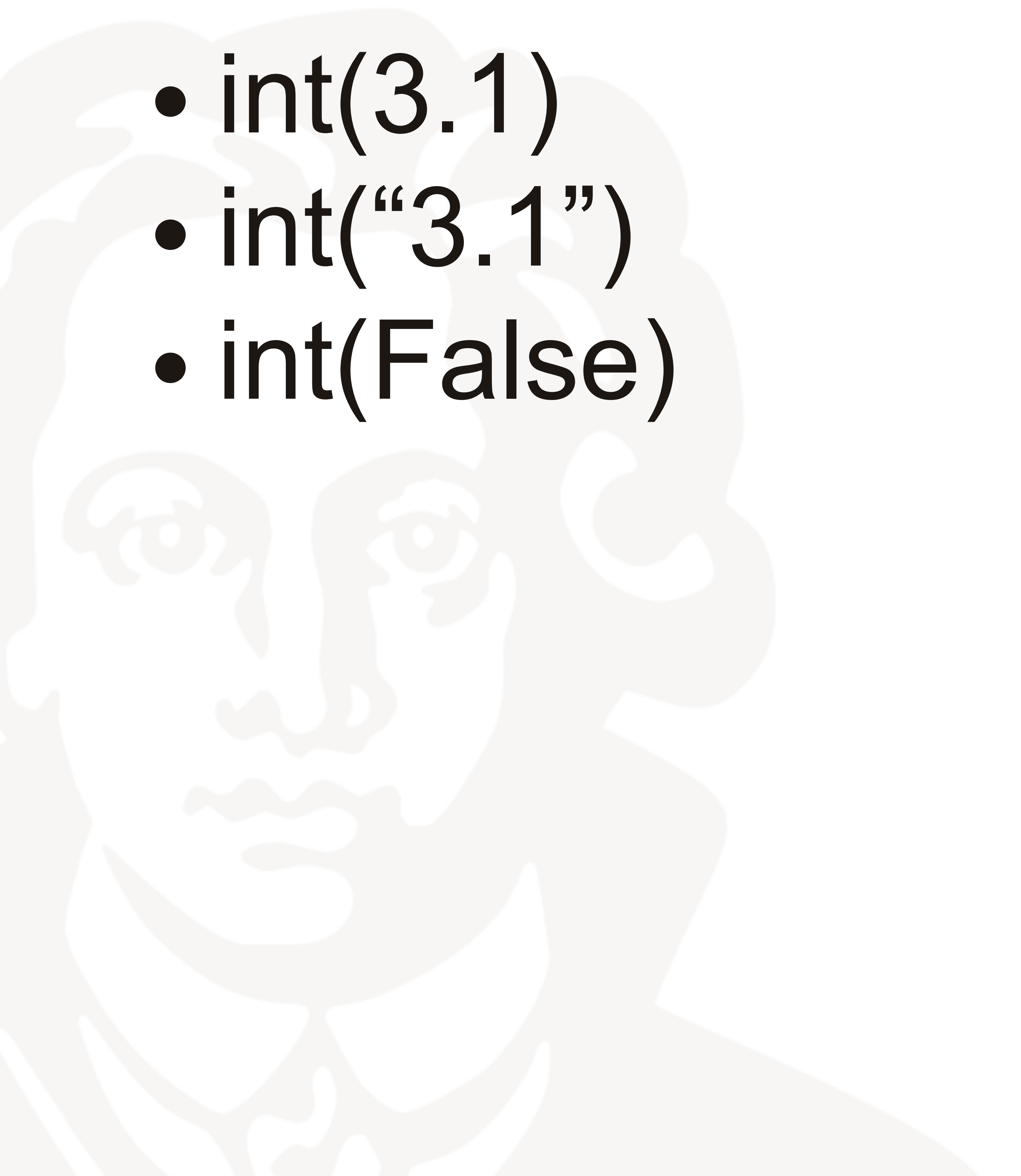
Also the relational operators (>, <, >=, <=, ==, !=, is, is not).

- Attention. <, >, <=, >= order the characters (including digits, spaces, punctuation and special characters) according to the numerical value corresponding to the code point in Unicode, so that, for example, all Latin capital letters are ordered before the lower case "a"

Type Conversion

In Python, data types can be converted into each other (the so-called casting)

- What does Python do at

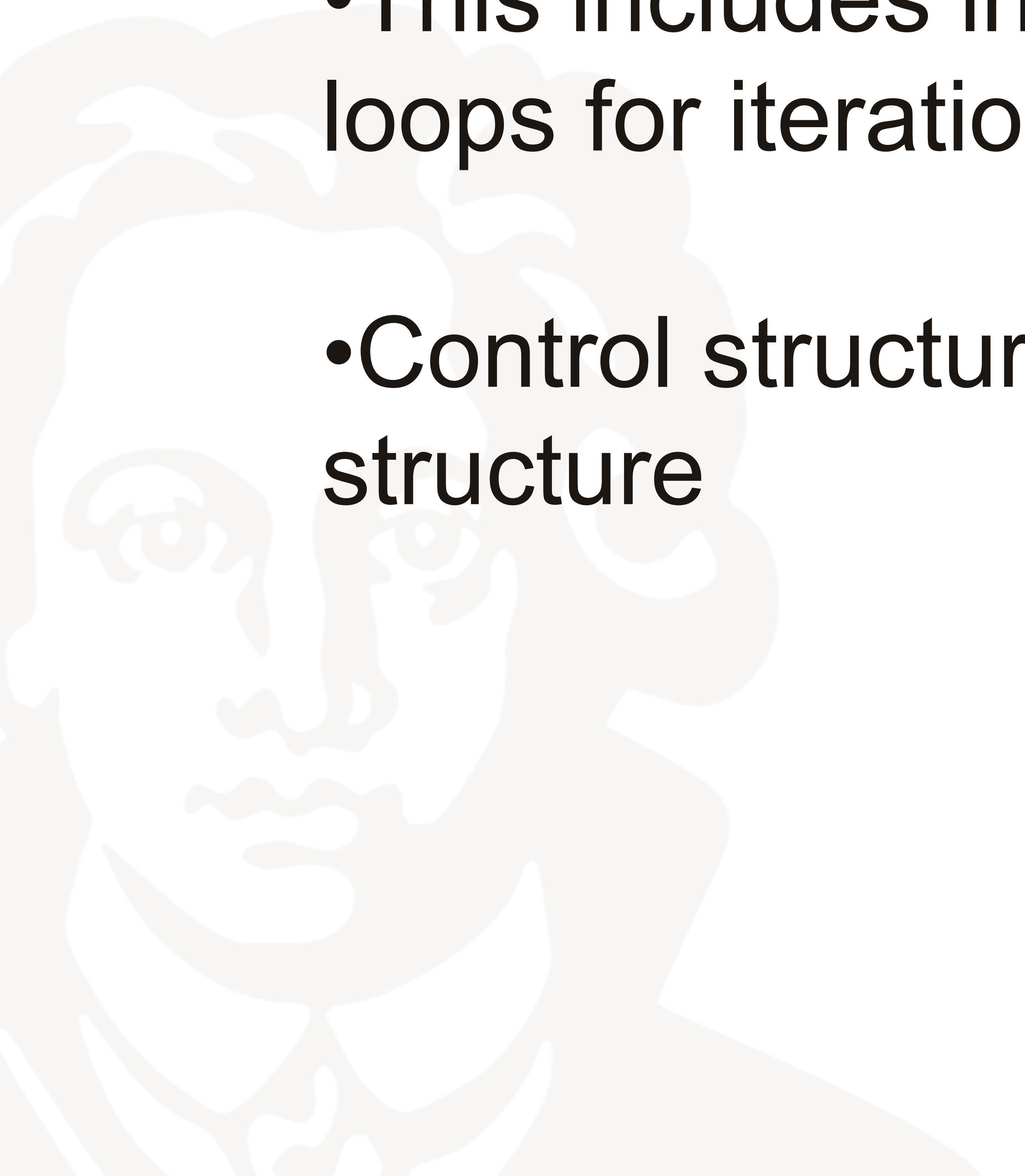


- int(3.1)
- int(“3.1”)
- int(False)

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Control Structures

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

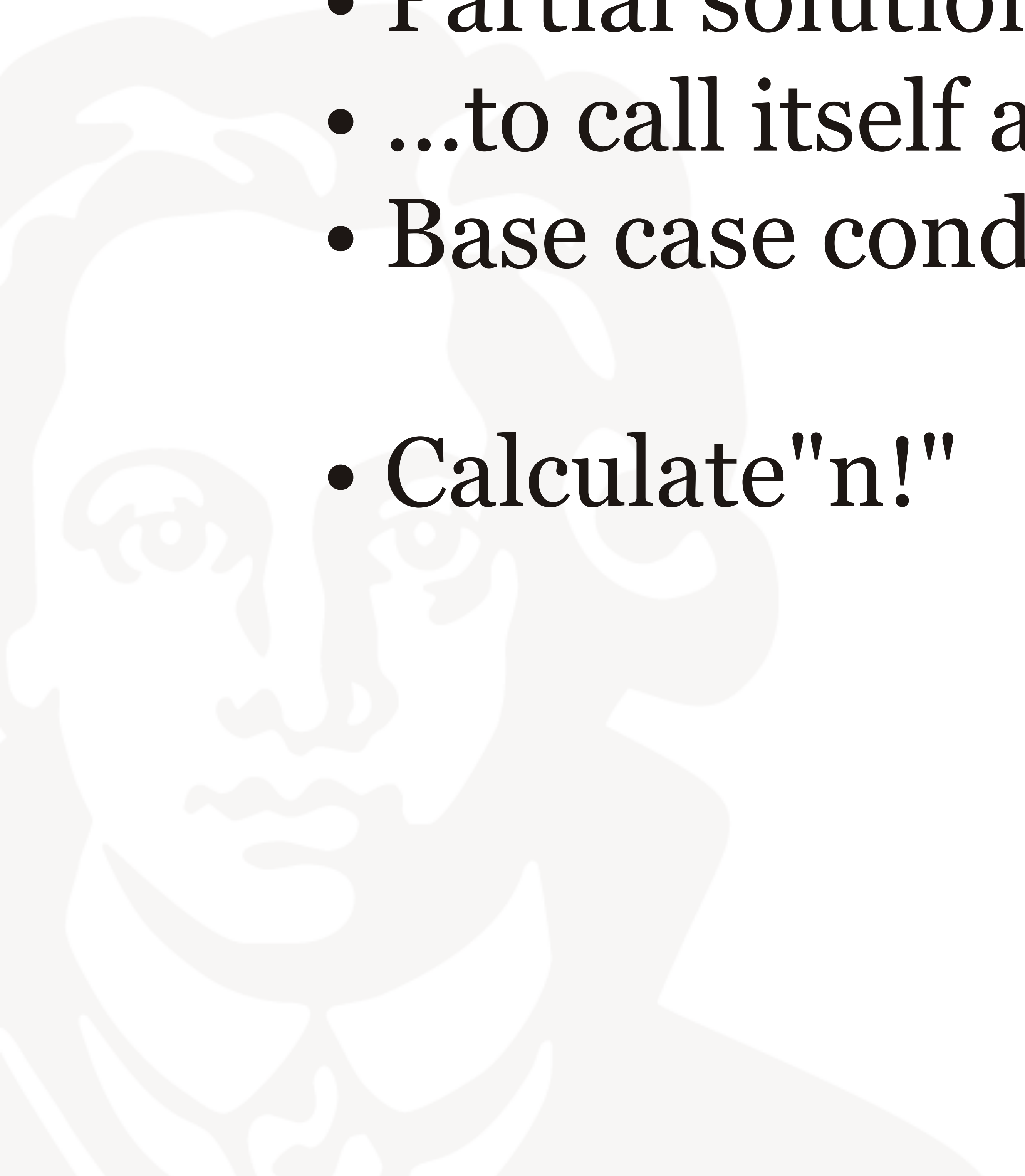


- Case distinction, iteration and recursion are to be grasped as basic mathematical solution methods

- This includes in particular the various forms of case distinctions and loops for iteration

- Control structures of all modern programming languages are similar in structure

Most basic recursion definition



If you want to understand recursion, you must first understand recursion

- Partial solution at each step
- ...to call itself and produce another partial solution.
- Base case condition is required

- Calculate"n!"

Most basic recursion definition

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

If you want to understand recursion, you must first understand recursion

- Partial solution at each step
- ...to call itself and produce another partial solution.
- Base case condition is required

- Calculate"n!"

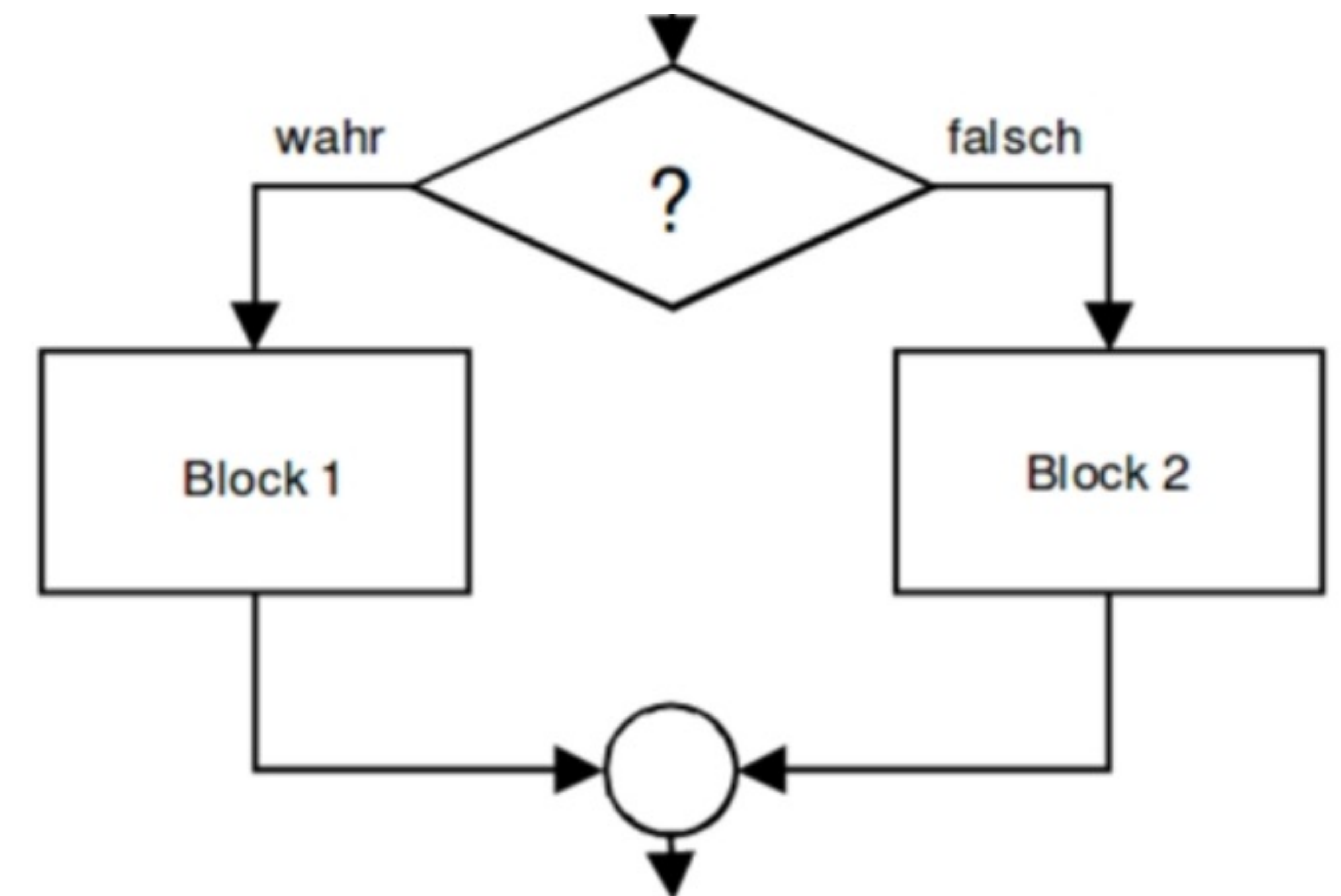
```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)

print("n! for n=10 is: ", factorial(10))
```

Control Structures - Branching

$$fib\, n = \begin{cases} n = 0 & 1 \\ n = 1 & 1 \\ \mathit{else} & fib\,(n-1) + fib\,(n-2) \end{cases}$$

A branch is implemented in all programming languages similar to this:

Generalized:
if <condition>
  then <action sequence>
  else <Alternative Action>
end if



Control Structures – Branching in Python

```
if expression:
    suite
elif expression:
    suite
elif expression:
    suite
...
else:
    suite
```

Important:

- Do not forget colon
- 4 spaces as indentation
- Development environments also accept TAB (but be careful when copying from other sources!)
- "expression" must be evaluated to a Boolean
- "suite" is any instruction

Control Structures – Branching in Python

```
a = 3
b = 2
if a < b:
    pass
elif a == b:
    a += 1
else: a -= 1
print(a)
```

Iteration - Loops

- Iteration is a method of approaching the solution of a "computational problem" in a step-by-step but goal-oriented manner

- It consists of the repeated application of the same calculation procedure

- Often, iteration is done with feedback: The results of one iteration step (or all results achieved so far) are taken as initial values of the next step.

Iteration - Loops

SUM :

$$a = \sum_{i=0}^{n} a_i = a_0 + a_1 + \cdots + a_n$$

PRODUCT:

$$a = \prod_{i=1}^{n} a_i = a_0 + a_1 + \cdots + a_n$$

Iteration - Loops

SUM :

$$a = \sum_{i=0}^{n} a_i = a_0 + a_1 + \cdots + a_n$$

PRODUCT:

$$a = \prod_{i=1}^{n} a_i = a_0 + a_1 + \cdots + a_n$$

Iteration - Loops

- In programming languages, iterative solutions of both types are realized by loops

- A loop repeats a part of the code, called the loop body or loop body, until a termination condition occurs

- Loops that never reach their termination condition or loops that have no termination conditions are called infinite loops

Iteration - Loops

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

The head-driven or pre-test loop:
• first the termination condition is checked before the loop body is run through (usually indicated by
the keyword WHILE (=so long-until).

• **The foot-controlled or rechecking loop:**
only after the loop body has been run through, the termination condition is checked e.g. by a construct REPEAT-UNTIL (for until)

• **The counting loop:**aspecialformofthehead-controlledloop,usuallyimplementedasa FOR(=for) loop.

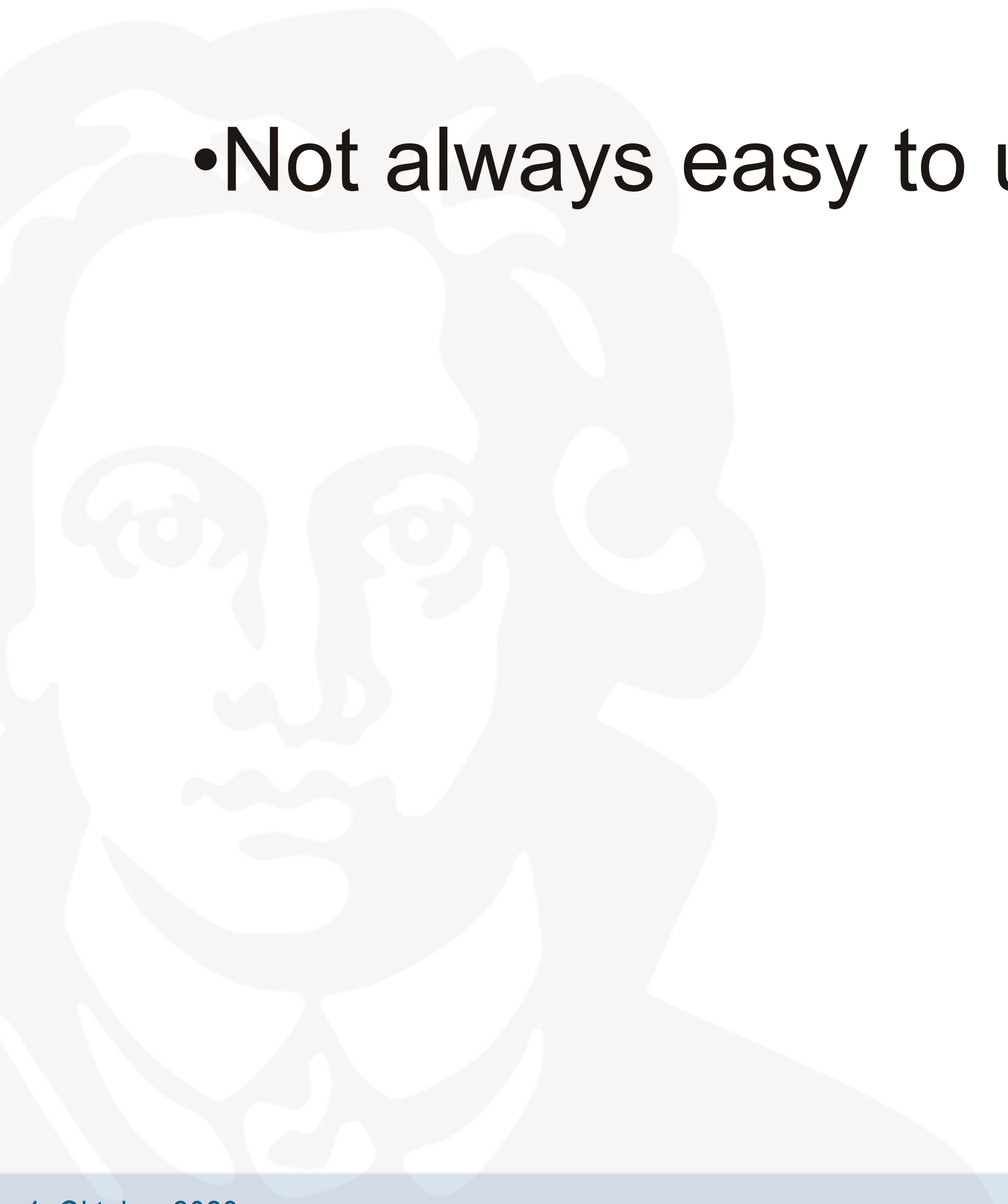
• **The for each loop:** a special form for sequence data types: Means "for each element in the sequence, execute the block exactly once".

Exiting the loop

- To end a loop without reaching the condition (expression == False), use the break statement

- To jump to the next loop iteration (skipping the rest of the loop body) use the continue statement

- Not always easy to understand. Experiment with many print()!



Break & Continue

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

temp.py - /Users/alexanderwolodkin/Documents/temp.py (3.

```python
search_for = eval(input("Geben Sie eine Zahl ein: "))
source = [1, 3, 14, 42]
test = 0

while test < len(source):
    if source[test] == search_for:
        print(search_for, 'liegt  in der Liste "source"')
        break
    print("test =", test, "source[test] =", source[test])
    test +=1

print("Schleife beendet")
```

Python 3.9.0 Shell

```
Geben Sie eine Zahl ein: 14
test = 0 source[test] = 1
test = 1 source[test] = 3
14 liegt  in der Liste "source"
Schleife beendet
```

temp.py - /Users/ale

```python
for i in range(5):
    if i == 2 or i == 3: continue
    print(i)

print("Schleife beendet")
```

```
>>>
============== RESTART: /Users/ale
0
1
4
Schleife beendet
>>> |
```