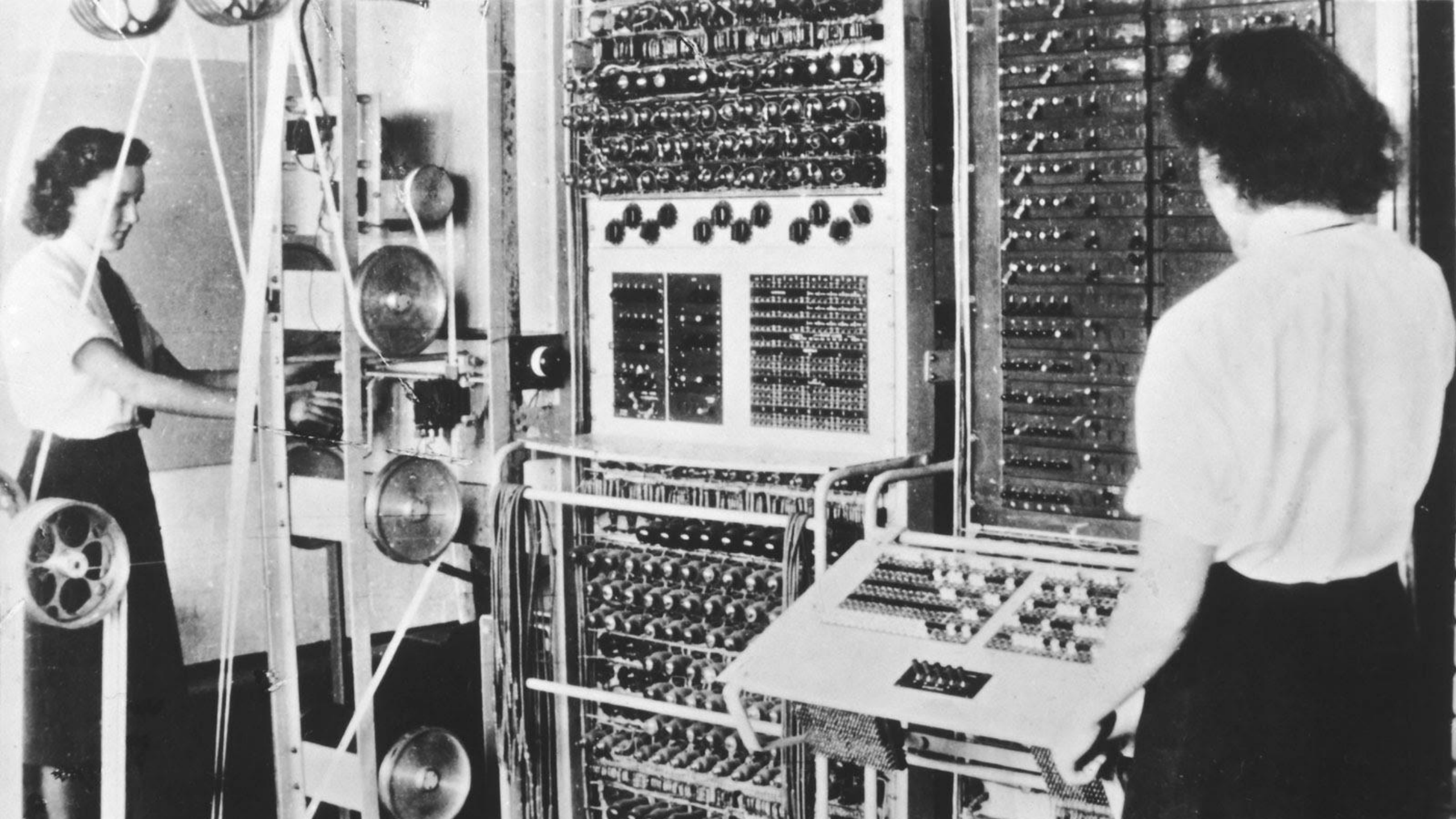
Informatik 1 - Introduction

Prof. Harald Gall

Dr. Carol Alexandru-Funakoshi

University of Zurich, Department of Informatics



What can computers accomplish?

- Driving autonomous cars (somewhat)
- Defeating humans at chess and go
- Searching the World Wide Web
- Rendering photorealistic pictures
- Finding the shortest route from home to work
- Solving Info1 assignments 
- ...

How can computers achieve that?

- A computer performs calculations and remembers the results
  - Computation + State = Getting things done
- The best thing about computers is: they do exactly what they're told

How can computers achieve that?

- A computer performs calculations and remembers the results
  - Computation + State = Getting things done
- The best thing about computers is: they do exactly what they're told
- The worst thing about computers is: they do exactly what they're told

How can computers achieve that?

- A computer performs calculations and remembers the results
  - Computation + State = Getting things done
- The best thing about computers is: they do exactly what they're told
- The worst thing about computers is: they do exactly what they're told
- Probably, you already tell computers what to do by "using" them
  - Programming is just a more direct way to do this!
- In this lecture, you will learn how to program

Goals of the Lecture

At the end of the course, you will be able to...

- analyze problems that you want to solve with software
- design algorithms
- read and write basic programs
- Recognize and apply foundational programming concepts

No programming experience?

- Informatics **1 - Introduction** to programming
- “No” prior knowledge is required or expected
- However, learning to program can be tough for a complete beginner...
  - Programming is 95% "learning by doing"
  - Don't give up if things "don't work", that's part of the experience
  - Programming is basically "solving puzzles"
  - Do the exercises!!!
- Most of what you need to know is taught in the first 4 weeks!

Course Organization

Course Organization

Lecturers	Prof. Dr. Harald Gall, Dr. Carol Alexandru-Funakoshi
Teaching Assistants	Marco Edoardo Palma, Alex Wolf
Lecture Schedule	Tuesday 12:15 - 13:45
Course Info Page	https://www.tiny.uzh.ch/15c
General Questions	OLAT Forum
Personal Questions	info1@lists.ifi.uzh.ch (use your ***@*uzh.ch** address!)
Registration	UZH Module Booking – as soon as possible!
Exercises	Online at https://access.ifi.uzh.ch
Midterm-Exam	**Monday**, 04.11.2024, 18:15 – 20:00, **on-site**
Final Exam	**Thursday**, 19.12.2024, 11:30 – 14:30, **on-site**

Course Staff

- Lecturers:
  - Prof. Dr. Harald C. Gall
  - Dr. Carol V. Alexandru-Funakoshi
- Assistants:
  - Marco Edoardo Palma
  - Alex Wolf
- Teaching Assistants:
  - Jerome Maier
  - Natalia Carrión Giménez
  - Jorge Alejandro Ortiz Valerio
  - Mete Polat
- Tutors:
  - Ceyhun Emre Açikmese
  - Fina Kossmann
  - Matteo Iulian Adam

Course Structure

- Weekly Lecture (livestream & recording on OLAT)
- Weekly Assignments
  - Homework assignments via ACCESS exercise platform
- On-site office hours with tutors (from week #2)
- Exams (online)
  - Midterm exam
  - Final exam
- Bonus System

Course Schedule (subject to change)

Tue, 17.09.2024	12:15 - 13:45	Organizational Details, Command Line, Programming Environments & IDEs
Tue, 24.09.2024	12:15 - 13:45	Python Tutorial Part I - Functions, Types, Data structures
Tue, 01.10.2024	12:15 - 13:45	Python Tutorial Part II - Conditional statements, Dictionaries, Comprehensions
Tue, 08.10.2024	12:15 - 13:45	Python Tutorial Part III - None, Loops, References
Tue, 15.10.2024	12:15 - 13:45	Python Tutorial Part IV - Scope, Classes, Objects
Tue, 22.10.2024	12:15 - 13:45	Python Tutorial Part V - Inheritance
Tue, 29.10.2024	12:15 - 13:45	Preparation for Midterm
Mon, 04.11.2024	**18:10 – 20:00**	**Midterm (on-site at KO2-F-180 + HAH-E-03 + HAH-E-11)**
Tue, 05.11.2024	12:15 - 13:45	Revisiting the Midterm Exam, Exceptions and Debugging, More on Functions
Tue, 12.11.2024	12:15 - 13:45	TBD
Tue, 19.11.2024	12:15 - 13:45	Everything's an Object in Python, Higher-order Functions, Special Methods
Tue, 26.11.2024	12:15 - 13:45	TBD
Tue, 03.12.2024	12:15 - 13:45	Type hints, Git
Tue, 10.12.2024	12:15 - 13:45	Repetition for Assessment Exam
Thu, 19.12.2024	**11:30 - 14:30**	**Assessment Exam (on-site at Messe Oerlikon)**

Assignment Schedule

Date	Assignment released	Assignment due
Tue, 17.09.2024	1	
Tue, 24.09.2024	2	
Tue, 01.10.2024	3	1
Tue, 08.10.2024	4	2
Tue, 15.10.2024	5	3
Tue, 22.10.2024	6	4
Tue, 29.10.2024		5
Tue, 05.11.2024	7	6
Tue, 13.11.2024	8	
Tue, 19.11.2024	9	7
Tue, 26.11.2024	10	8
Tue, 03.12.2024	11	9
Tue, 10.12.2024		10
Tue, 17.12.2024		11

Course Links

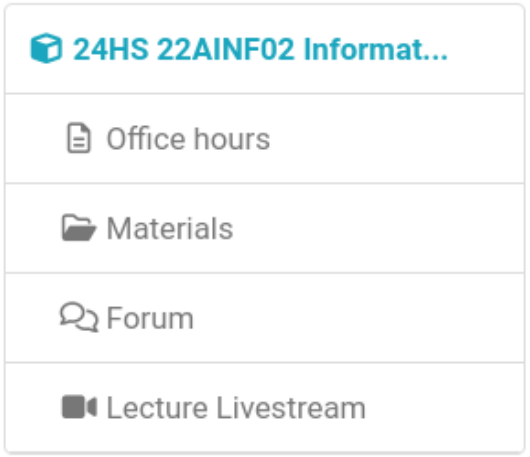
- **Website**: https://www.ifi.uzh.ch/en/seal/teaching/courses/info1.html
- **OLAT** (AINF1166 Informatik I): https://lms.uzh.ch/url/RepositoryEntry/17589469401
- **ACCESS** (Exercises): https://access.ifi.uzh.ch/
- Official **Python documentation**: https://docs.python.org/3/
- MIT course:

  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/

- Book: Introduction to Computation and Programming Using Python

  https://www.amazon.com/Introduction-Computation-Programming-Using-Python/dp/0262529629

OLAT – Central hub for all things Info1



24HS 22AINF02 Informatics I (L+E) (Informatik I)

Welcome to the Informatics I OLAT course. This course will be used for the lecture slides, as a communication hub via the forums, and for the
of the exercises.

The following links point to the other relevant information sources:

- VVZ Entry and Module Booking
- Basic Course Information Page
- Exercise Submission and Correction Tool, "ACCESS"
- Mailing list for contacting the course organizers (use your @uzh email address!), however: please ask content-related questions here in

Presence

The course will be given physically at the University in lecture hall HAH-E-03.

"Office Hours" - from week 2

Ask you questions and receive help!

Time	Where	Tutor	Languages
Mon 10-11	IFI Room: ???	Fina Kossmann	German, English
Thu 12-13	IFI Room: ???	Matteo Adam	German, English
Fri 10-11	IFI Room: ???	Ceyhun Açikmese	German, English

On-site: Office hours at the IFI (Room see OLAT)

You may join during several time slots in the same week!

Weekly Assignments

- Home Exercise
  - Exercises are being published after the lecture
  - Submission deadline before the lecture **two weeks** later

Exams: on-site, bring-your-own-device

- On-Site in rooms KO2-F-180, HAH-E-03, HAH-E11
- Midterm
  - You **MUST** attend; if you do not, you are excluded from the final exam!
  - You do not have to pass
- Final Exam

- You are responsible for your own device. Make sure you have enough charge and that everything is working flawlessly!

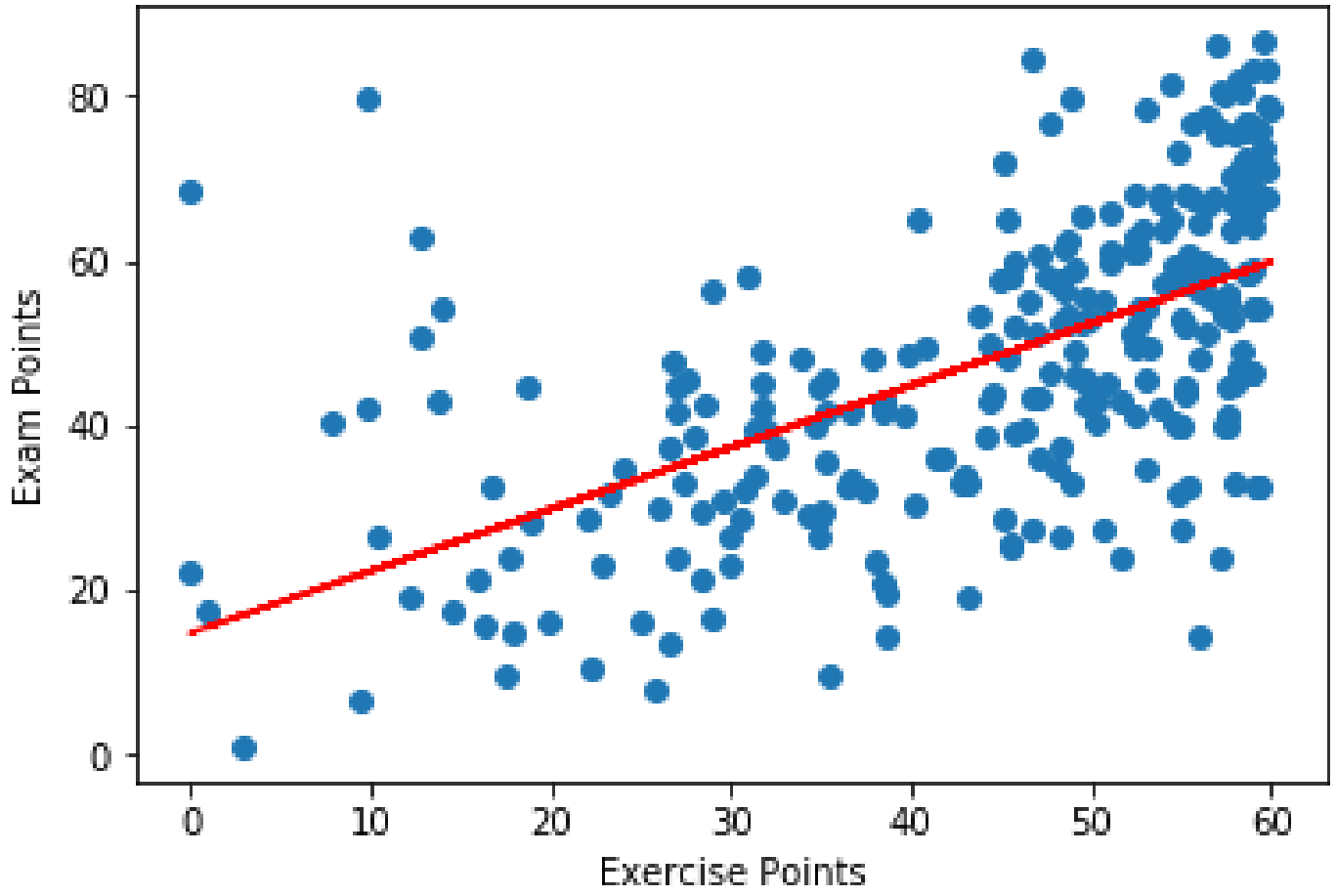
Grading

- Your grade is primarily determined by your **final exam**
- Students that pass the final exam, can further improve their grade in a bonus system. For both the midterm and the assignments, apply the following rule:
  - If you reach at least 65% of the achievable points, every additional 1% gives you 0.01 bonus points up to a maximum of 0.25 (at 90% points achieved)
    - Example: Midterm 74% of points achieved, assignments 87% of points achieved, 5.00 in the final exam ⟶ 0.09 + 0.22 = 0.31 bonus; Final grade: 5.25
    - There will be *roughly* 100-120 points achievable in the assignments
- The bonus **cannot** help you to pass the final exam though!

Tips

- Programming is fun, but it takes time to master it.
- Do not give up when things don't make sense to you right away.
- Make sure that you
  - search the internet whenever encountering (unexpected) error messages
  - ask questions during the lecture and office hours
  - don’t just read code! **Write it** yourself to get more familiar with Python and its functions, your IDE, and get a better feeling for how to approach coding tasks!
- Practice makes perfect, so practice, practice, practice!

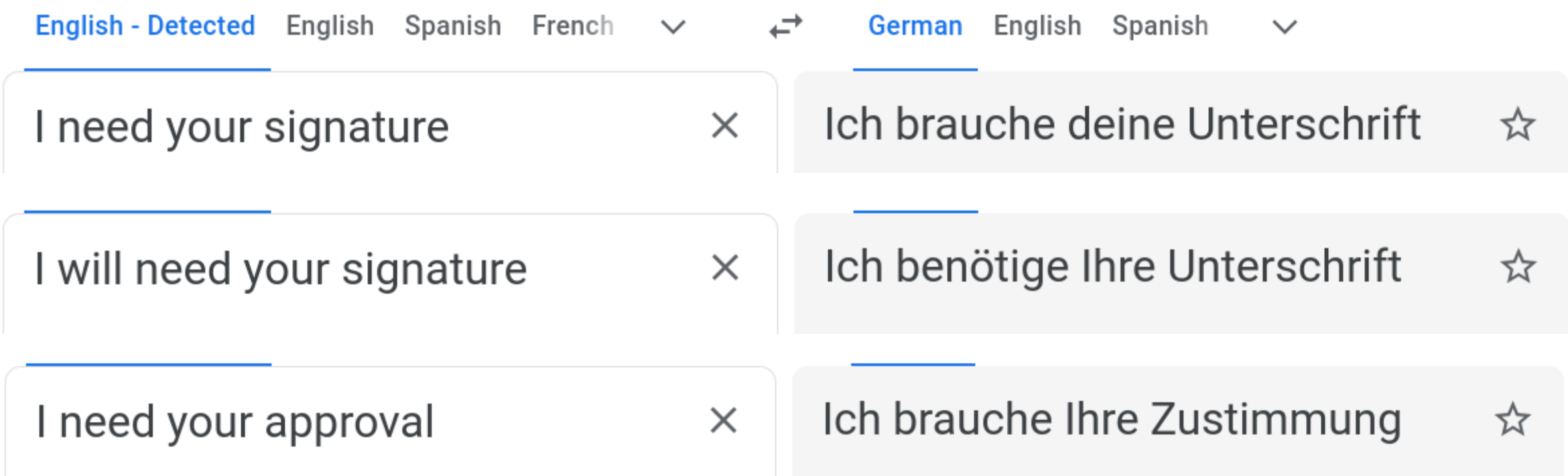
Take the exercises seriously!



We found a clear correlation between the points reached in the exercise and the points reached in the final.

A word about AI

- Using LLM to program without knowing how to program is a lot like using machine translation without knowing the target language



A word about AI

- Using LLM to program without knowing how to program is a lot like using machine translation without knowing the target language
  - It only works well for simple stuff
  - Native speakers **will** know you have no clue
  - You will never do translation work, unless you learn the language
- Unless you learn to program by yourself, LLM will not solve your problems. Don't waste your time.
- In the exams, you won't be allowed to use LLM

Until Next Time

- Registration (!!!)
  - book the module (Modulbuchung)
    - This automatically enrolls you in our OLAT campus course (after <24h)
    - This automatically enrolls you in ACCESS (after <48h)
- Read ALL information on the OLAT main page carefully!
- Work on the first assignment
- Visit the online tutor office hours next week if you need help getting started (installing the Python environment etc.)

Informatik 1
Data and Algorithms

Prof. Harald Gall

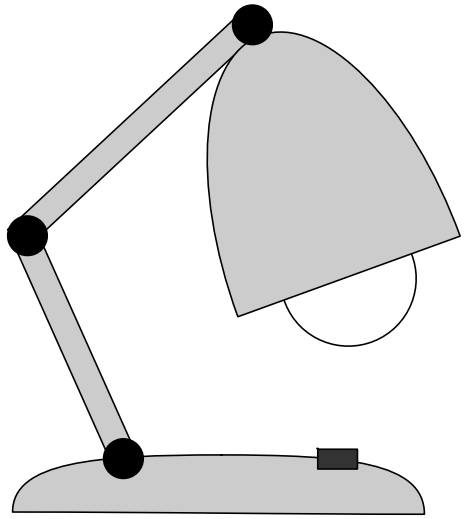
Dr. Carol Alexandru-Funakoshi

University of Zurich, Department of Informatics

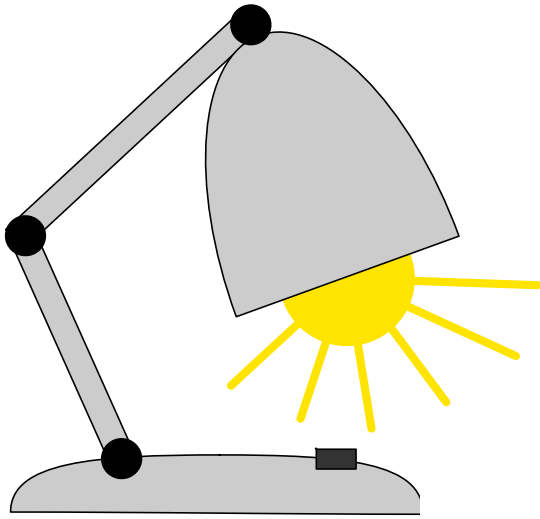
How does a computer represent data?


https://commons.wikimedia.org/wiki/File:Binary_Code.jpg

Computers differentiate between ON and OFF



0



1

Bitte ein Bit

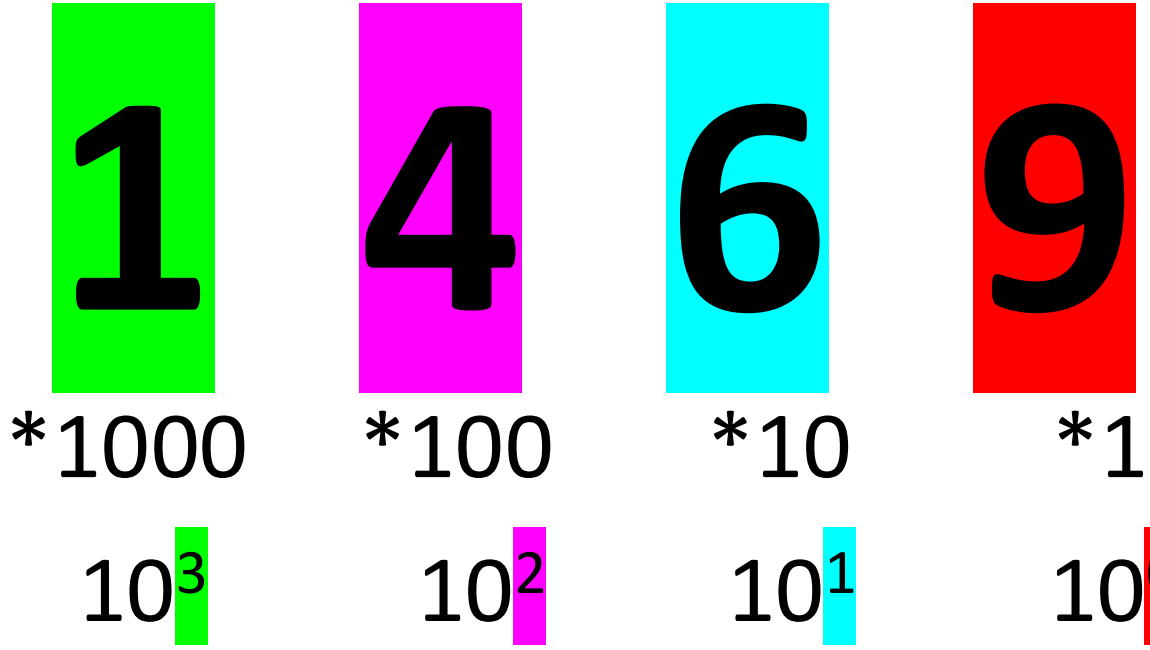
Conventional Counting

1 4 6 9

*1000	*100	*10	*1
10^3	10^2	10^1	10^0

$1*10^3 + 4*10^2 + 6*10^1 + 9*10^0 = 1469$

Conventional Counting



$1*10^3 + 4*10^2 + 6*10^1 + 9*10^0 = 1469$

Binary Counting

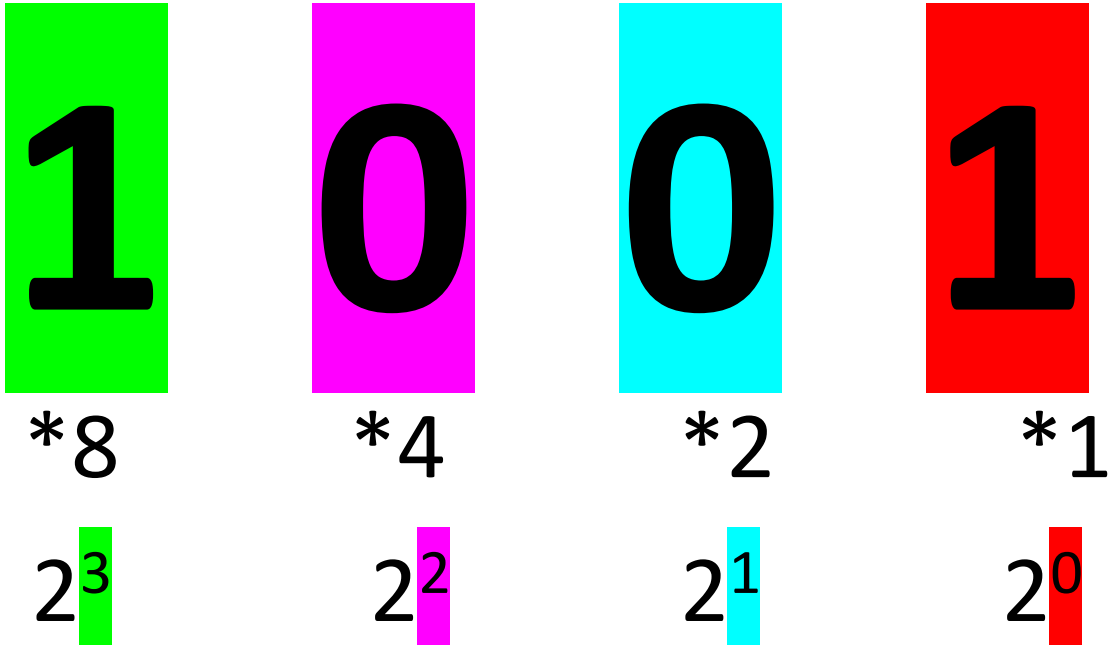
1 0 0 1

*8 *4 *2 *1

2^3 2^2 2^1 2^0

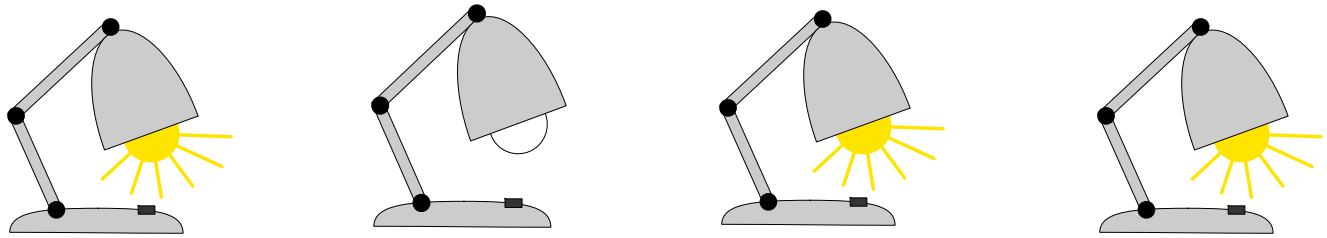
$1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 9$

Binary Counting



$1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 9$

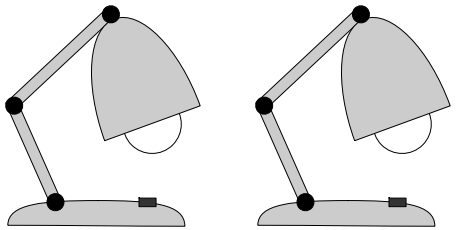
Binary Counting

… 

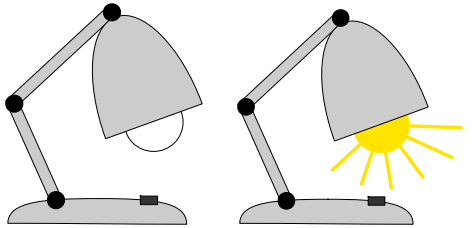
$2^3 = 8$ $2^2 = 4$ $2^1 = 2$ $2^0 = 1$

1x8 + 0x4 + 1x2+ 1x1 = 11

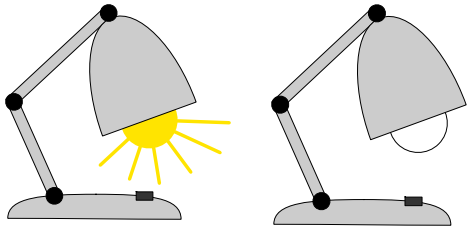
Examples



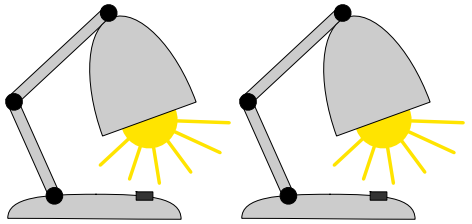
= 0



= 1

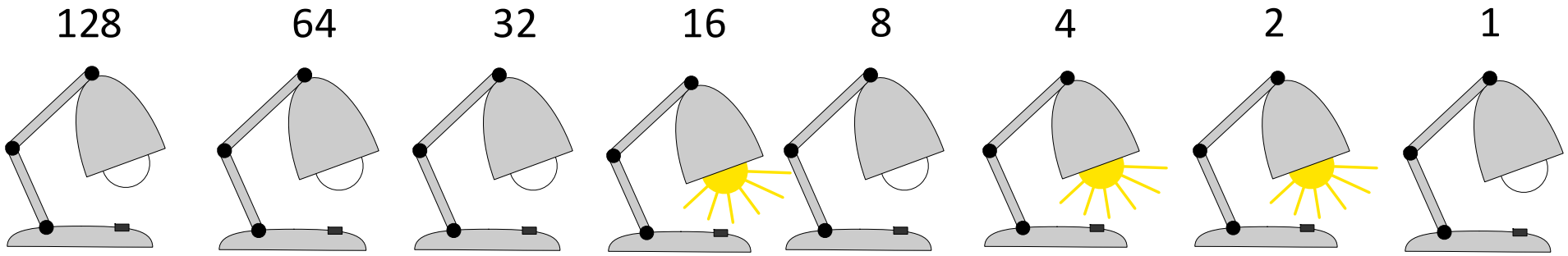


= 2



= 3

1 Byte (= 8 Bit)



Minimum Value? 0
Maximum Value? 255
This Value? 22

What do these bits represent?

- Numbers

Binary	0	1	1101	10010111
Decimal	0	1	11	227
Hexadecimal	0	1	B	E3

What do these bits represent?

- Numbers

Binary	0	1	1101	10010111
Decimal	0	1	11	227
Hexadecimal	0	1	B	E3

- Characters

Binary	100100	1000001	11100110	1001000101010010
Decimal	36	65	230	37202
ASCII	$	A		
UTF-8	$	A	æ	酒
КОИ-7	¤	A		

USASCII code chart

b_7 b_6 b_5 → / Bits ↓	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 Row ↓ / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

1000001 = A

1000010 = B

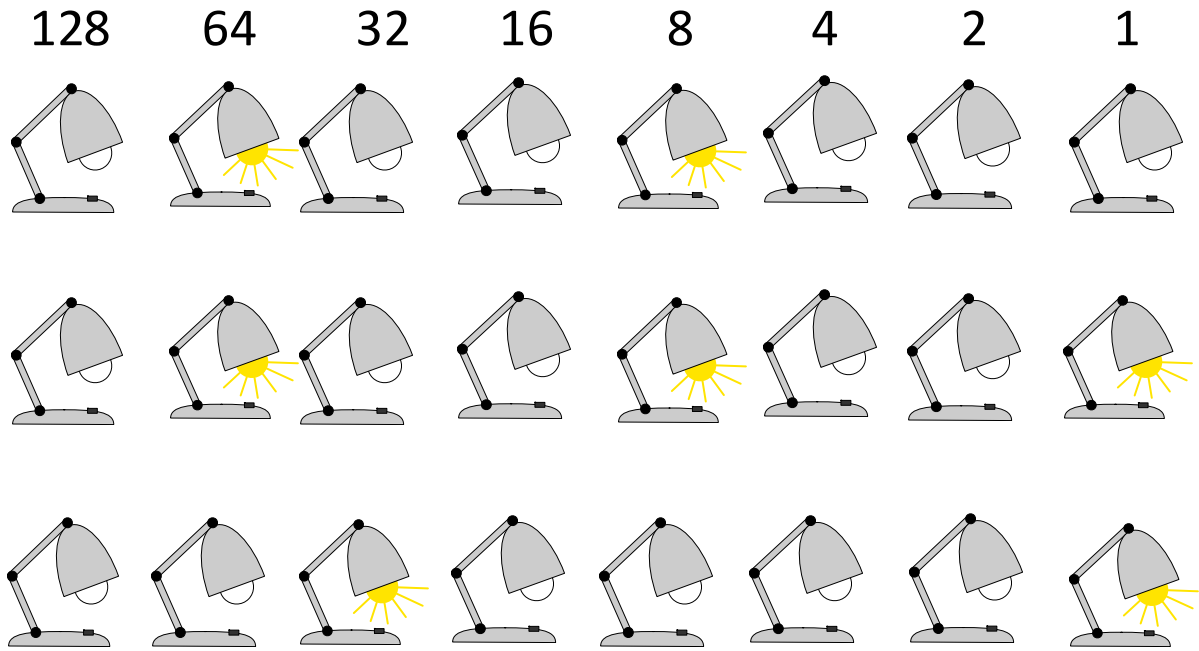
1000011 = C

…

Binary code does not have a meaning *per se*, the meaning depends on the interpretation!

https://en.wikipedia.org/wiki/ASCII#/media/File:USASCII_code_chart.png

Sentences are "strings" of characters



= 72 = H

= 73 = I

= 33 = !

01001000010010010010000 1 = 72,73,33 = HI!

What do these bits represent?

- Numbers

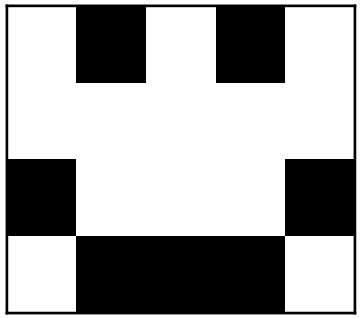
Binary	0	1	1101	10010111
Decimal	0	1	11	227
Hexadecimal	0	1	B	E3

- Characters

Binary	100100	1000001	11100110	1001000101010010
Decimal	36	65	230	37202
ASCII	$	A		
UTF-8	$	A	æ	酒
КОИ-7	¤	A		

- Pictures: 10101111110111010001

1	0	1	0	1
1	1	1	1	1
0	1	1	1	0
1	0	0	0	1



What do these bits represent?

R	196	11000100	#c4595f
G	89	11000100	
B	95	01011111	

- Numbers

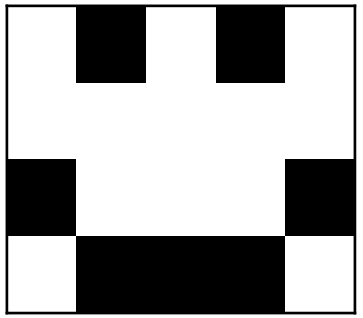
Binary	0	1	1101	10010111
Decimal	0	1	11	227
Hexadecimal	0	1	B	E3

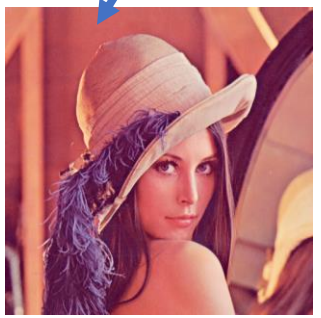
- Characters

Binary	100100	1000001	11100110	1001000101010010
Decimal	36	65	230	37202
ASCII	$	A		
UTF-8	$	A	æ	酒
КОИ-7	¤	A		

- Pictures: 10101111110111010001

1	0	1	0	1
1	1	1	1	1
0	1	1	1	0
1	0	0	0	1





What do these bits represent?

R	196	11000100	#c4595f
G	89	11000100	
B	95	01011111	

- Numbers

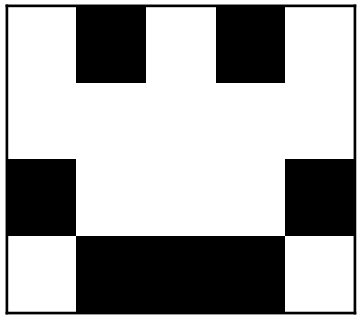
Binary	0	1	1101	10010111
Decimal	0	1	11	227
Hexadecimal	0	1	B	E3

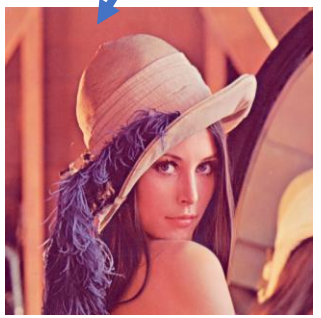
- Characters

Binary	100100	1000001	11100110	1001000101010010
Decimal	36	65	230	37202
ASCII	$	A		
UTF-8	$	A	æ	酒
КОИ-7	¤	A		

- Pictures: 10101111110111010001

1	0	1	0	1
1	1	1	1	1
0	1	1	1	0
1	0	0	0	1



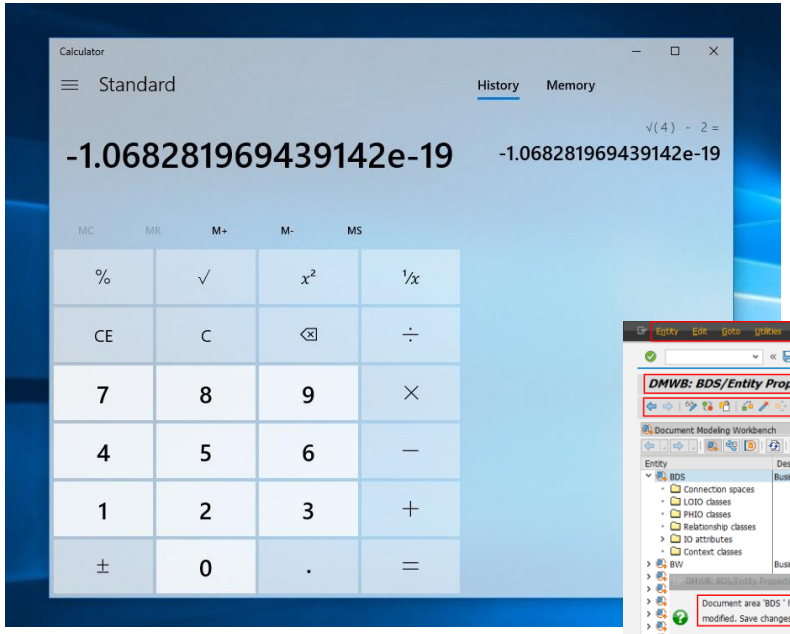


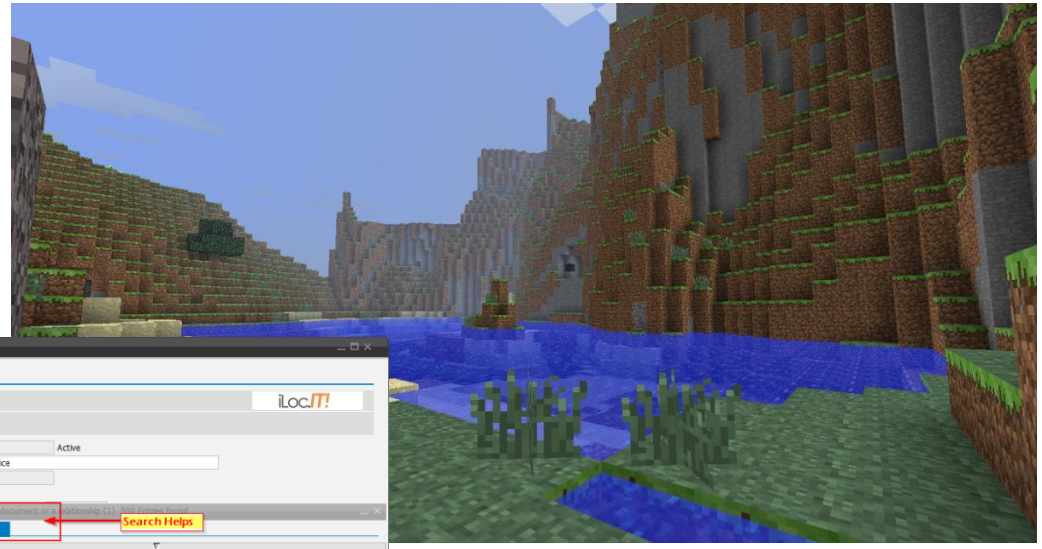
- Machine code:

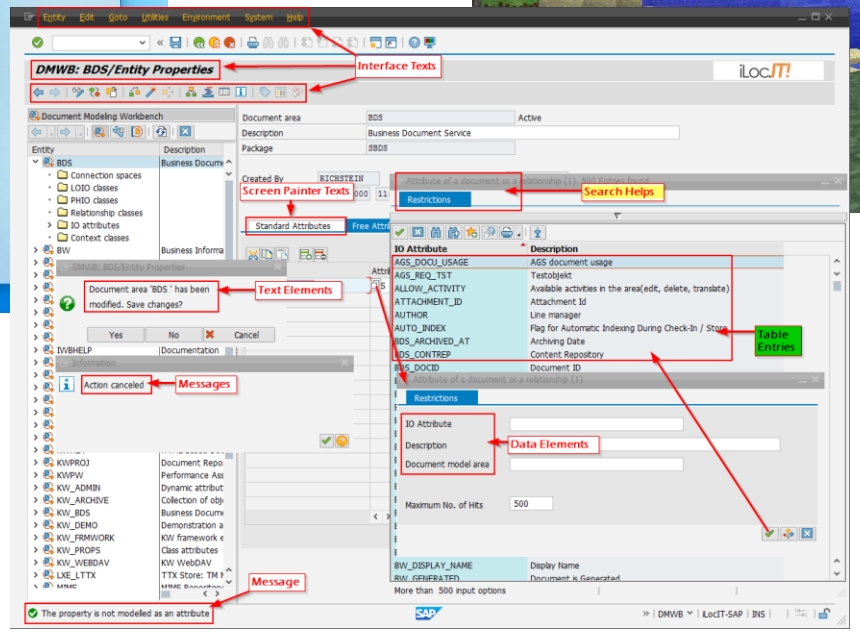
```
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  54 80 04 08 34 00 00 00  |........T...4...|
00000020  00 00 00 00 00 00 00 00  34 00 20 00 01 00 00 00  |........4. .....|
00000030  00 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000040  00 80 04 08 74 00 00 00  74 00 00 00 05 00 00 00  |....t...t.......|
00000050  00 10 00 00 b0 04 31 db  43 b9 69 80 04 08 31 d2  |......1.C.i...1.|
00000060  b2 0b cd 80 31 c0 40 cd  80 48 65 6c 6c 6f 20 77  |....1.@..Hello w|
00000070  6f 72 6c 64                                       |orld|
00000074
```

How to actually instruct a computer to do something?

By clicking on stuff







By giving instructions (typically plain text)

- A programming language is a formal language that specifies a set of instructions (or commands) that can be given to a computer to produce various kinds of output. (Wikipedia)

- Programming Languages define syntax, valid ways to express intent through declarations, statements and expressions.

- The semantics of a program are properties that describe the meaning of the executed instructions.

There are literally 1000s of languages

Clojure [edit]

```
(println "Hello world!")
```

CMake [edit]

```
message(STATUS "Hello world!")
```

This outputs

```
-- Hello world!
```

COBOL [edit]

Using fixed format.

Works with: OpenCOBOL
Works with: Dell Enterprise COBOL

```
       program-id. hello.
       procedure division.
           display "Hello world!".
           stop run.
```

Using relaxed compilation rules, the hello program can become a single DISPLAY statement.

Works with: GnuCOBOL

```
display"Hello, world".
```

```
prompt$ cobc -x -frelax-syntax -free hello.cob
hello.cob: 1: Warning: PROGRAM-ID header missing - assumed
hello.cob: 1: Warning: PROCEDURE DIVISION header missing - assumed

prompt$ ./hello
Hello, world
```

Plan [edit]

This prints HELLO WORLD on operator's console.

```
#STEER          LIST,BINARY
#PROGRAM        HLWD
#LOWER
MSG1A           11HHELLO WORLD
MSG1B           11/MSG1A
#PROGRAM
#ENTRY          0
      DISTY     MSG1B
      SUSWT     2HHH
#END
#FINISH
#STOP
```

Pointless [edit]

Python [edit]

Works with: Python version 2.4

```
print "Hello world!"
```

The same using sys.stdout

```
import sys
sys.stdout.write("Hello world!\n")
```

```
Debug("Hello world!")
```

Oforth [edit]

```
"Hello world!" .
```

Ol [edit]

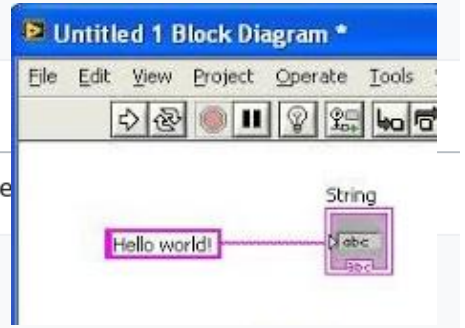
```
(print "Hello world!")
```

Onyx [edit]

```
`Hello world!\n' print flush
```

OOC [edit]

To print a String, either call its println() me




```
main: func {
  "Hello world!" println()
}
```

Or call the free println() function with the String as the argument.

Malbolge [edit]

Long version:

```
('&%:9]!~}|z2Vxwv-,POqponl$Hjig%eB@@>}=<M:9wv6WsU2T|nm-,jc
`CB]V?Tx<uVtT`Rpo3NlF.Jh++FdbCBA@?]!~|4XzyTT43Qsqq(Lnmkj"Fl
```

Short version:

```
(=<`#9]~6ZY32Vx/4Rs+0No-&Jk)"Fh}|Bcy?`=*z]Kw%oG4UUS0/@-ejc
```

Output:

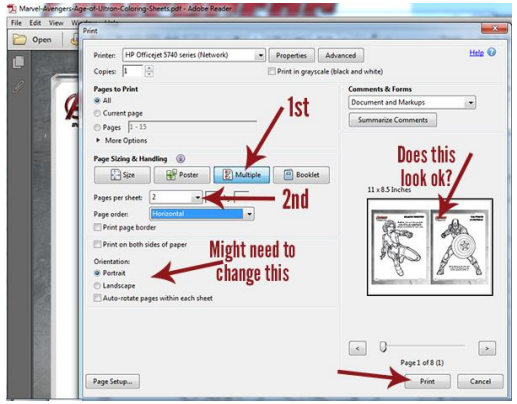
```
HELLO WORLD!
```

https://www.rosettacode.org/wiki/Hello_world/Text

Computer Program

- A computer program is a collection of instructions that perform a specific task when executed by a computer.
  - Programs can run *indefinitely* (“endless loop”).
- You can think of a program as a "recording" or "playbook" of instructions.

Instead of this...



... we just tell the computer what to do step by step ...

```
doc = PDFDoc(input_path + "tiger.pdf")

# Set our PrinterMode options
printerMode = PrinterMode()
printerMode.SetCollation(True)
printerMode.SetCopyCount(1)
printerMode.SetDPI(100);
printerMode.SetDuplexing(PrinterMode.e_Duplex_Auto)

printerMode.SetScaleType(PrinterMode.e_ScaleType_FitToOutPage)

Print.StartPrintJob(doc, printerMode)
```

Algorithm, an informal definition

An algorithm unambiguously defines the steps to perform a particular type of task.

An algorithm takes input parameters, performs a series of computations, and returns an output.

An algorithm must halt eventually.

Cooking recipes resemble the definition of an algorithm. Due to the lack of clear syntax and semantics, their execution is typically not well-defined.

Examples of Algorithms

- Sorting values
- Finding the shortest route from home to university (*shortest path algorithm*)
- Calculating the cost of an assembled good (e.g., car engine)
- Predicting the stock market price
- …?

Algorithm: Find name in phone book

1. Open the phone book
2. As long as you do not see the name, turn the page
3. Remember the number next to the name

This is pseudocode, a list of informal statements in an imaginary programming language that are to be executed.

What happens if the name does not exist?

Algorithm: Find name in phone book, Take 2

1. Open the phone book
2. As long as you do not see the name, turn the page
3. Did you find the name?
   -> Remember the number next to the name
4. Did you come to the end of the phone book?
   -> The name does not seem to be there.

Algorithm: Find name in phone book, Take 3

1. Look at the first letter of the name and remember it (let's call it "x")
2. Open the phone book
3. Skip ahead until the first page with names starting with x
4. As long as you do not see the name, turn the page
5. Did you find the name?

   -> Remember the number next to the name
6. Did you come to the end of the section with names starting with x?

   -> The name does not seem to be there.

Algorithm: Count the number of people

1. num = 0
2. for every person in the room:
    1. num = num + 1

Programs can remember intermediate values by assigning them to variables.

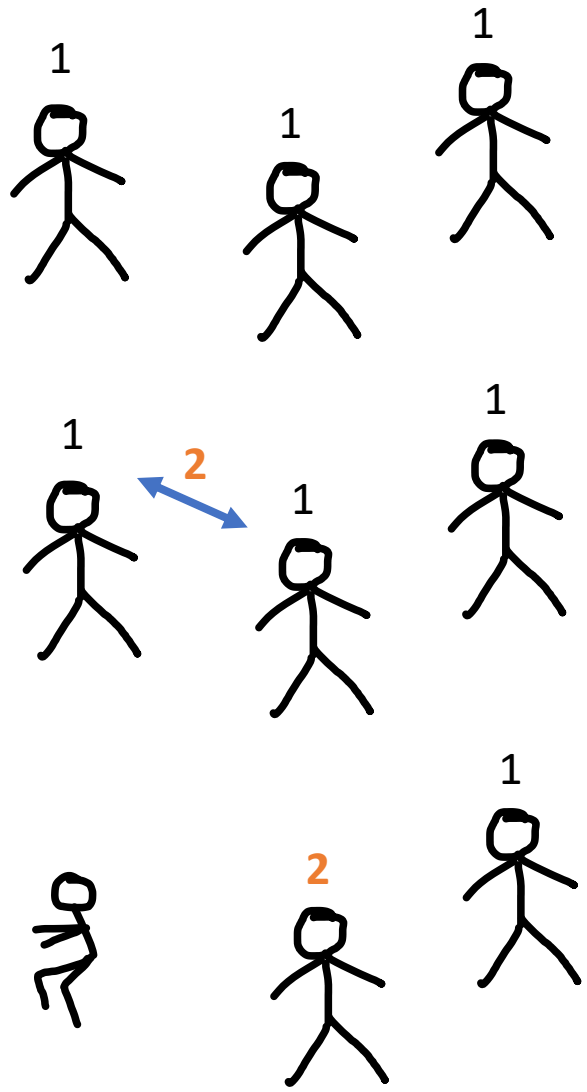
The execution time of this algorithm grows linearly with the number of people to count. The more people in the room, the longer it takes to count them.

Social Algorithm: Counting

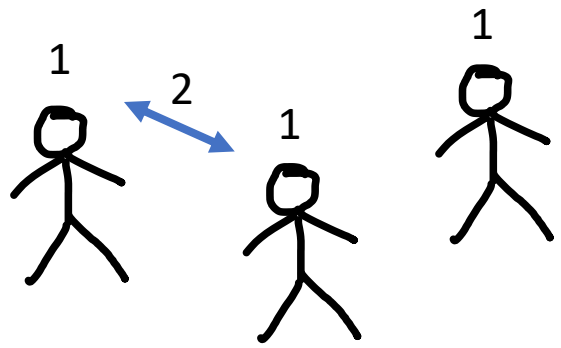
1. All of you stand up and remember the number ”1”

2. Find somebody else in the audience that is standing. Get in contact with each other and add up your numbers.

3. **One of you** sits down, **the other** updates their own number and goes back to step 2

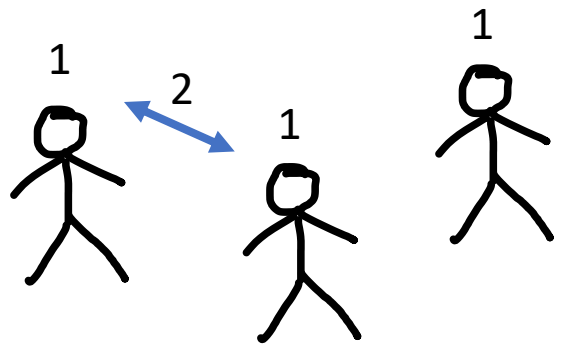


Social Algorithm: Counting



What is different to the first approach?

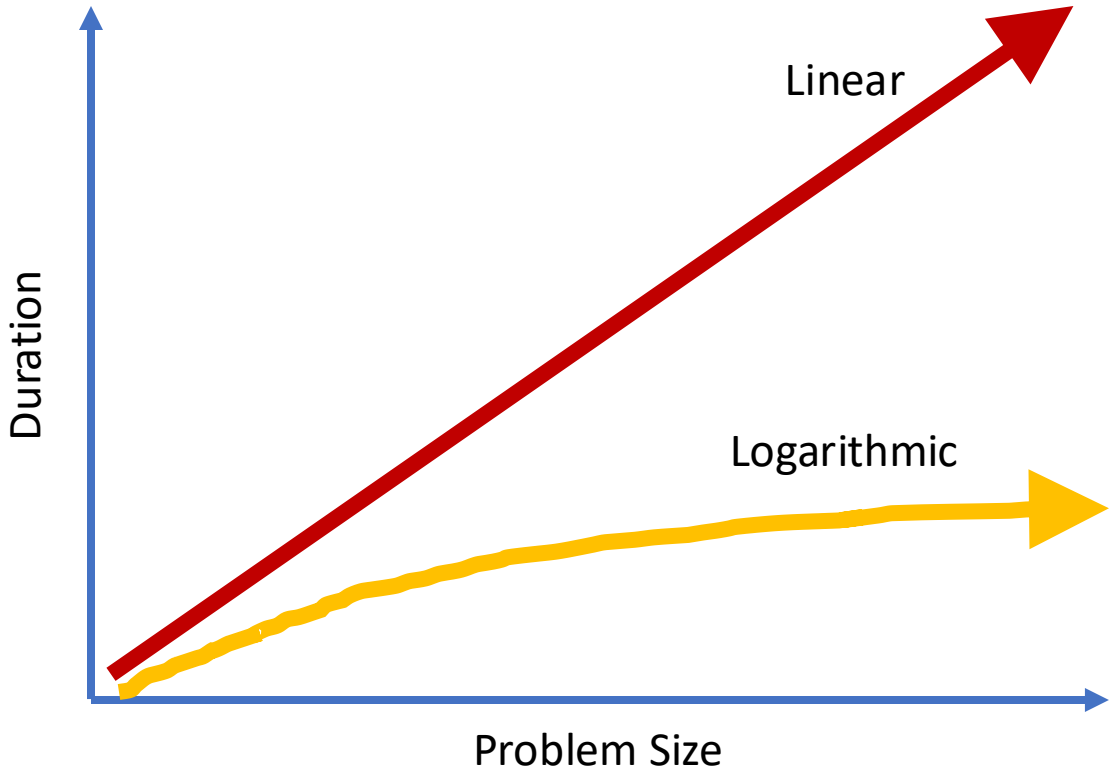
Social Algorithm: Counting



What is different to the first approach?

Division of the problem into smaller pieces, better scaling.

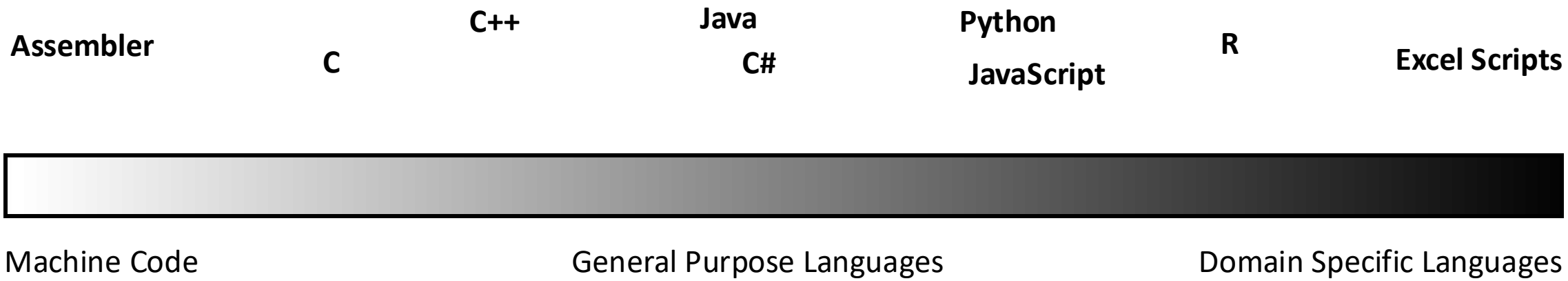
Some algorithms are faster than others



In our "social algorithm", counting a room twice the size would only have taken insignificantly longer!

First steps in Python

We are going to learn Python. Why?



Python is a high-level programming language. It is a general language that features many advanced concepts, but has little syntactical overhead, which makes it easy to learn.

Python Programs are Plain-Text

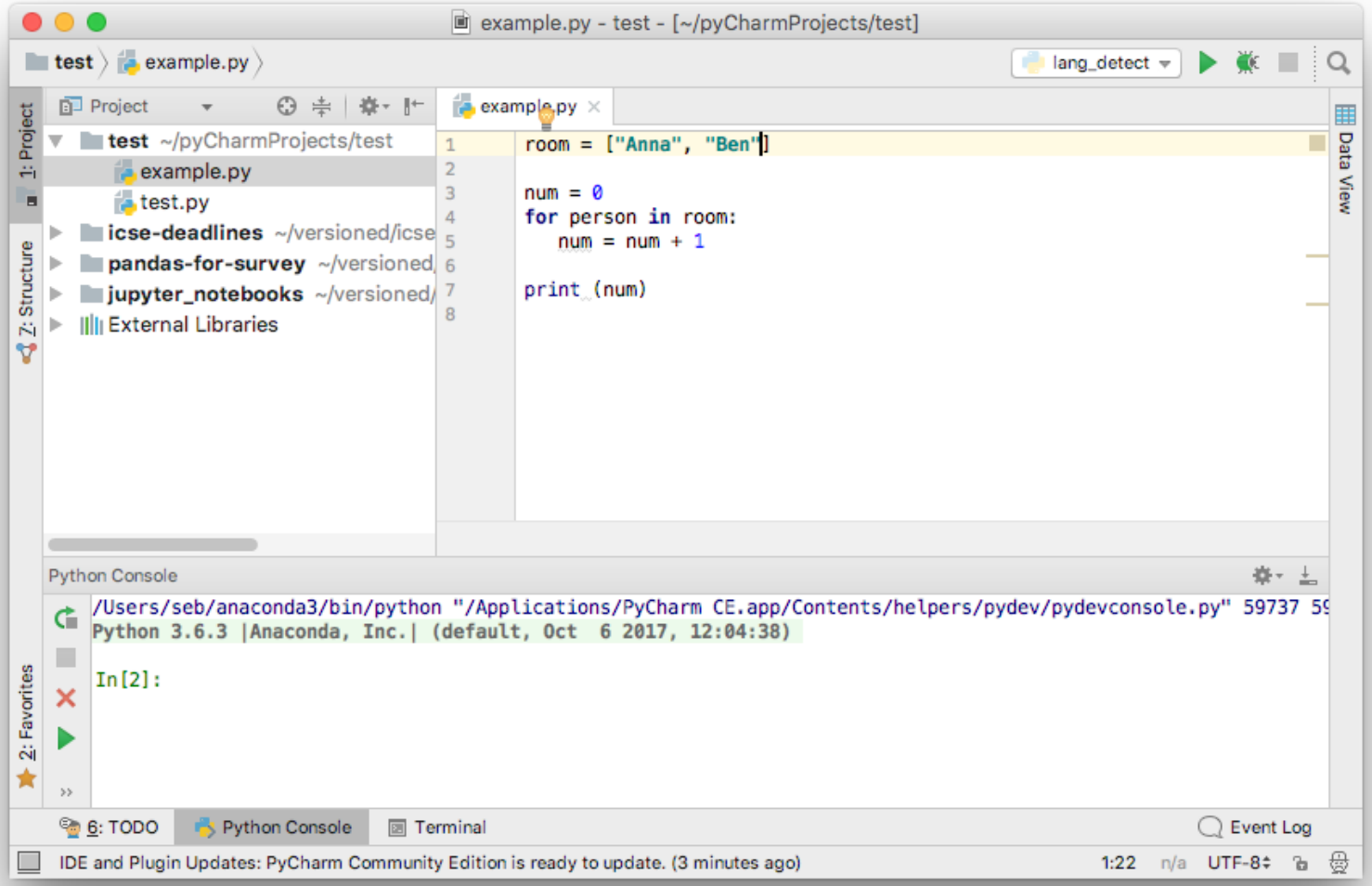
```
room = [“Anna”, “Ben”]
num = 0
for person in room:
    num = num + 1
print (num)
```

```
$ python3 counting.py
2
```

We will learn how to represent data that is more complex than numbers and strings.

Python programs can be executed by running an interpreter.

If you just run 'python3' without specifying a file, you get an interactive session



An Integrated Development Environment *can* make it easier to read, write and execute programs.

Several alternative IDEs exist, such as IDLE, PyCharm or VS Code. A good text editor is also fine.

Find an Editor that you like and get comfortable using the integrated tools (syntax highlighting, debugger, etc.)

Do not use something like Microsoft Word or Apple Pages!!!

You can request input and print on the screen

```
print ("hello world")
msg = "hello world"
print (msg)

name = input()
print ("hello " + name)
```

Use the `print()` command to print information on the screen.

You can print literal values (such as strings) and variables.

Use the `input()` function to request information from the user.

Use `+` to concatenate different strings.

Course Contents

Today's lecture in a nutshell

- Computers represent data in a binary form, combining multiple bits allows to encode numbers, strings of characters, or anything else
- Computers can be instructed through statements
- Computers can remember intermediate computation results
- Algorithms are repeatable recipes to solve a particular problem type
- Different algorithms might exist for the same task
- Programming languages like Python provide means to input/output
- Python programs are plain-text and need to be interpreted

Contents of the Remaining Course

- First 6 weeks: Python Tutorial
  - Data Types, Operators, Expressions
  - Conditional and Iterative Statements
  - Functions
  - Built-In Data Structures: Tuples, Lists, Dictionaries
  - Classes, Objects and Inheritance
  - Special functions and more
- Testing and Debugging
- Information Hiding, Design Principles and Patterns
- Code Organization: Modules and Version Control

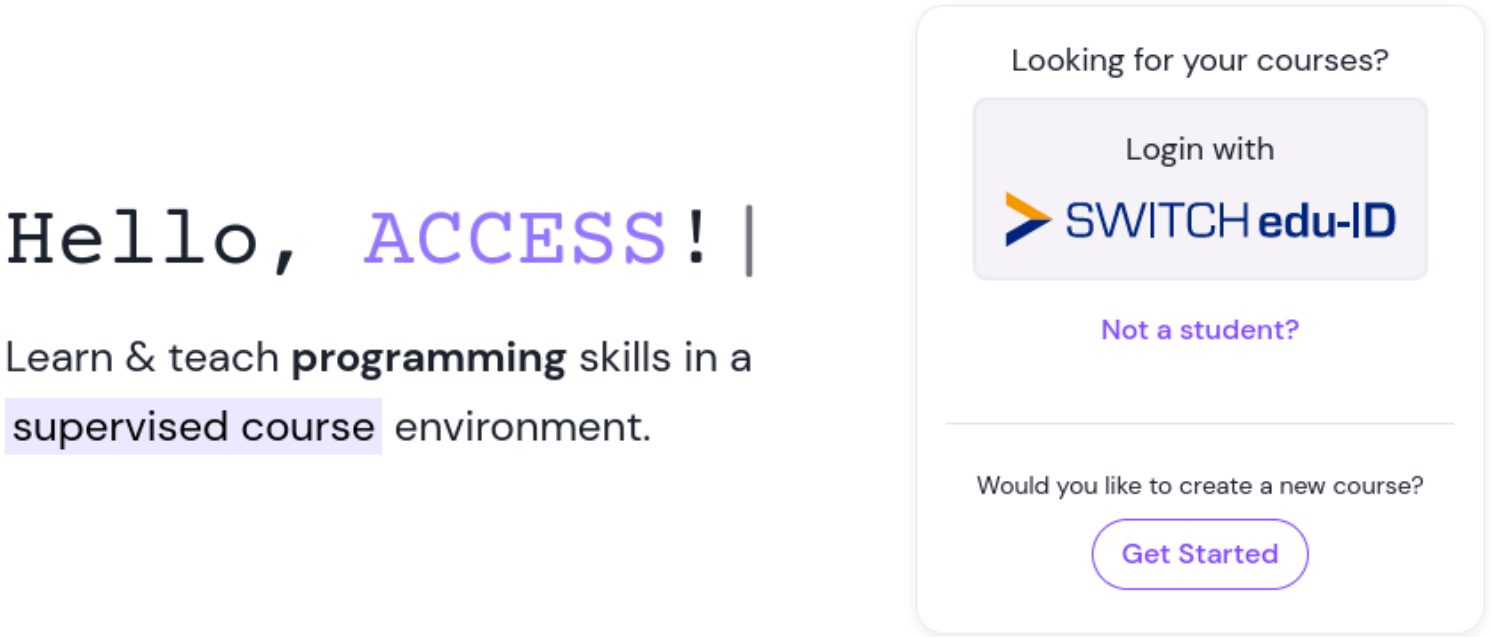
ACCESS

Online Programming Exercises

Dr. Carol Alexandru-Funakoshi

University of Zurich, Department of Informatics

https://access.ifi.uzh.ch



ACCESS.

Welcome, **Carol**!

My Courses 0

No courses found.

Please try reloading this page if this is unexpected.

Notice Board

- Welcome to ACCESS!

Explore ACCESS

More courses
coming soon!

ACCESS.

Welcome, **Carol**!

My Courses 1

Informatics I (Fall 2023)

0 Online

University of Zurich

01-09-2023 ~ 01-02-2024 12 Students

0 / 0 Points

Notice Board

- Welcome to ACCESS!

Explore ACCESS

More courses coming soon!

Informatics I (Fall 2023)

0 Online

University of Zurich

01-09-2023 ~ 01-02-2024 12 Students

Introduction to Programming

⏰ Active Assignments

2	**Getting Started with Python!** 4 tasks	19-01-2023 ~ 03-10-2023 15 DAYS 22 HOURS 48 MINUTES	0 / 3 Points	Show tasks

🔒 Closed Assignments

1	**Hello, World!** 1 task	01-01-2020 ~ 12-01-2020	0 / 1 Points	Show tasks

ASSIGNMENT 1

Getting Started with Python!

4 Tasks | 19-01-2023 ~ 03-10-2023 | Submission Open

15 DAYS 22 HOURS 48 MINUTES

Tasks 4

1	**Hello, World!**	**2** / 2 Submissions left	Me / Avg.	View
2	**Installing Python**	**3** / 3 Submissions left	Me / Avg.	View
3	**Variable assignment**	**1** / 3 Submissions left	Me / Avg.	View
4	**Programming Environments**	**3** / 3 Submissions left	Me / Avg.	View

Instructions

In this task, you take your first steps in working with source code, and you learn how to use ACCESS.

Find your way around ACCESS

Have a look around:

- At the top left, you navigate to different assignments and tasks
- On the left, below these instructions you can see the file tree, showing all the files contained in this task
- On the right, you can see your score, how many attempts you have left, and the history of all your coding activity
- In the middle, you can see the files you've opened. Only some of the files can be edited.
- At the top right, you have three buttons "Test", "Run" and "Submit". Don't click them just yet.
- At the bottom, you have three tabs which corresponds to the buttons at the top right.

To solve assignments in ACCESS, you generally

- Read the instructions carefully. For the most part, the instructions contain everything you need to know to solve a task correctly.
- Open and edit the files as required in the instructions to implement your solution
- Click the "Run" button and observe the output from your script at the bottom in the "Run" tab
- Click the "Test" button to run the provided tests. Note that these are very minimal and do not generally determine whether your solution is correct. You will learn how to write your own tests so that you can take advantage of this feature in the future.
- Submit your solution by clicking "Submit". It will automatically be graded, and you will receive feedback on how much of your solution is correct and what may be remaining issues with it.
- Check the solution hint by looking at the history on the right side. Modify your solution if necessary, re-run, re-test, and re-submit.

You only have a limited number of submission attempts. However, after a set period of time (typically 1 day), you will receive a refill on your attempts. So if you run out, try coming back the next day to continue working on the assignment!

Hello, World!

Traditionally, the first program learned is "Hello, World!". All it does is *print* these two words to the command line. "Printing", in the context of programming, has little to do with printers and just means outputting some text on the screen.

Have a look at the file `task/script.py`. It contains one line of code (starting with `print...`) and some comments (starting with `#`).

Select the `script.py` file and press the "Run" button to execute the code. At the bottom

Files

- instructions_en.md
- task
  - script.py
  - tests.py

script.py | tests.py

```
1  #!/usr/bin/env python3
2
3  # Modify the following code
4  print("This sentece will be printed to the command line.")
5
6
```

Test Run Submit

> **Submission 1**
$ No output

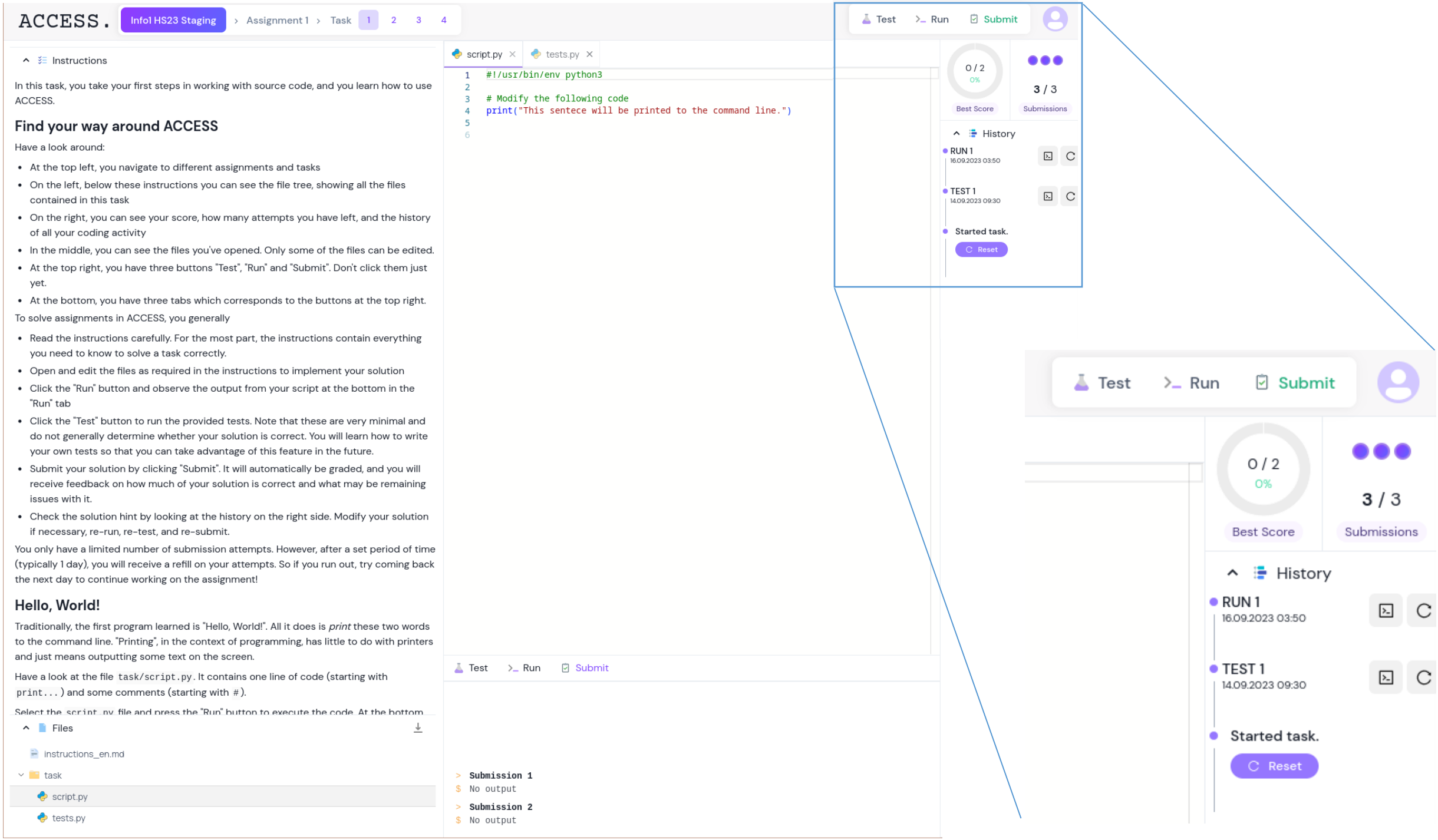
> **Submission 2**
$ No output

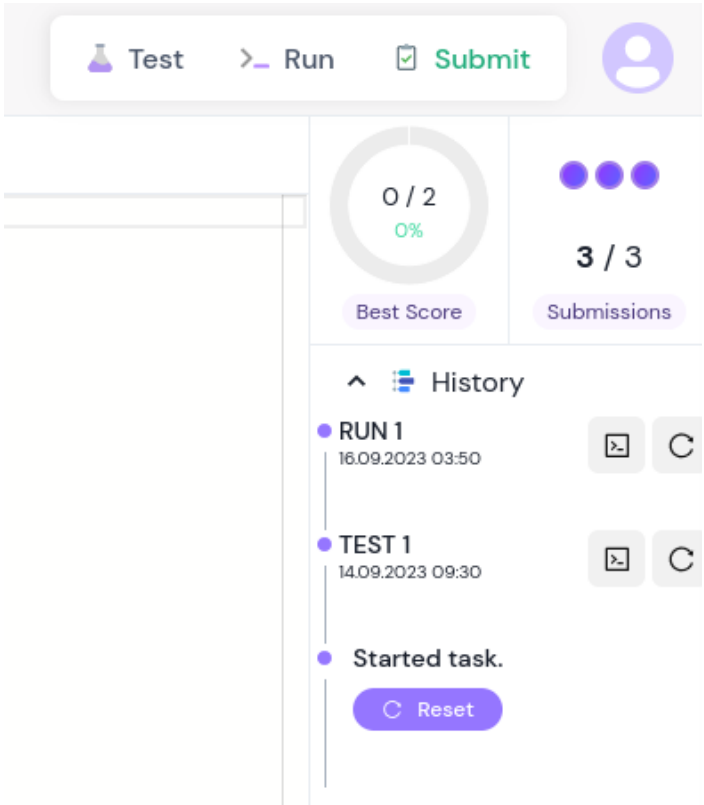
0 / 2
0%
Best Score

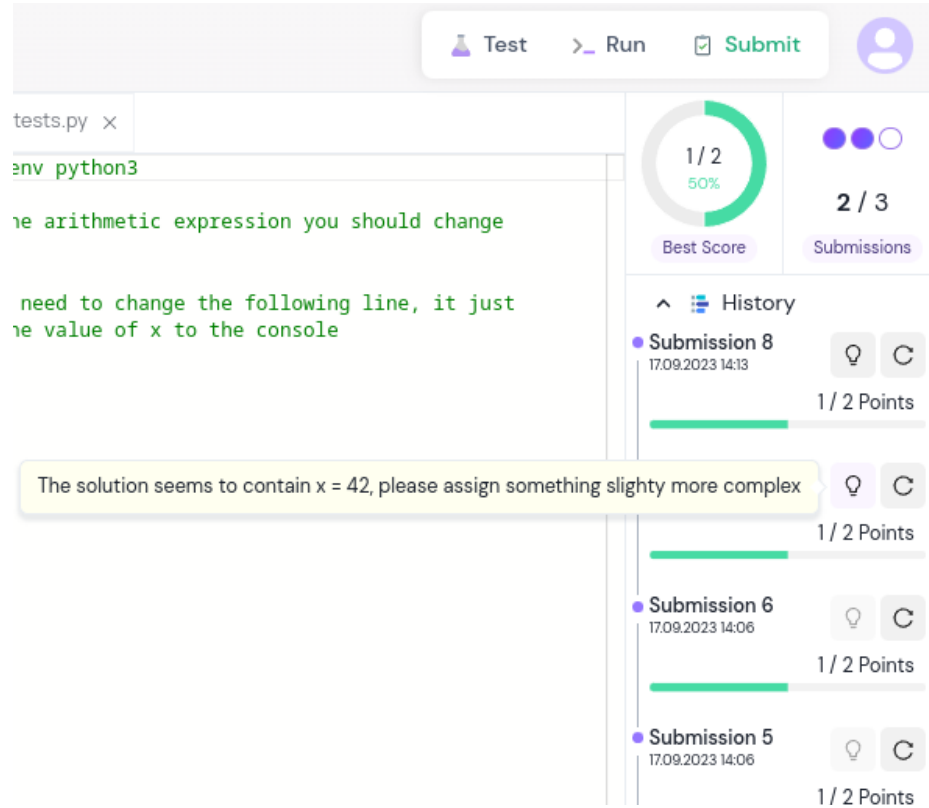
3 / 3
Submissions

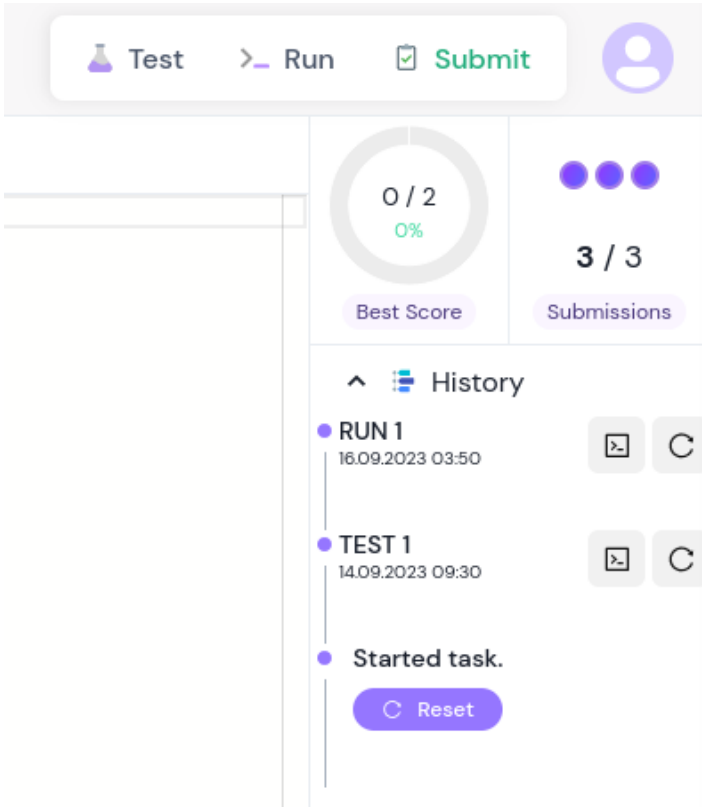
History

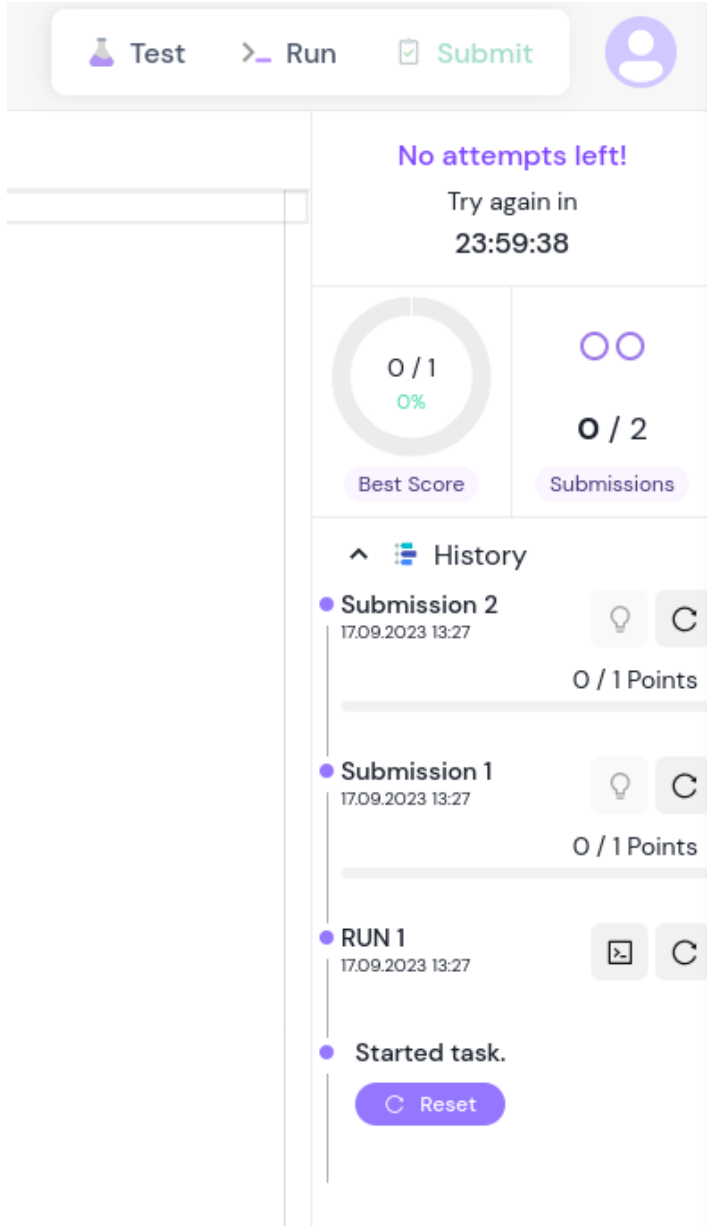
- RUN 1
  16.09.2023 03:50
- TEST 1
  14.09.2023 09:30
- Started task.
  Reset


ACCESS.
Info1 HS23 Staging
Assignment 1
Task 1 2 3 4
Test
Run
Submit
Instructions
In this task, you take your first steps in working with source code, and you learn how to use ACCESS.
Find your way around ACCESS
Have a look around:
At the top left, you navigate to different assignments and tasks
On the left, below these instructions you can see the file tree, showing all the files contained in this task
On the right, you can see your score, how many attempts you have left, and the history of all your coding activity
In the middle, you can see the files you've opened. Only some of the files can be edited.
At the top right, you have three buttons "Test", "Run" and "Submit". Don't click them just yet.
At the bottom, you have three tabs which corresponds to the buttons at the top right.
To solve assignments in ACCESS, you generally
Read the instructions carefully. For the most part, the instructions contain everything you need to know to solve a task correctly.
Open and edit the files as required in the instructions to implement your solution
Click the "Run" button and observe the output from your script at the bottom in the "Run" tab
Click the "Test" button to run the provided tests. Note that these are very minimal and do not generally determine whether your solution is correct. You will learn how to write your own tests so that you can take advantage of this feature in the future.
Submit your solution by clicking "Submit". It will automatically be graded, and you will receive feedback on how much of your solution is correct and what may be remaining issues with it.
Check the solution hint by looking at the history on the right side. Modify your solution if necessary, re-run, re-test, and re-submit.
You only have a limited number of submission attempts. However, after a set period of time (typically 1 day), you will receive a refill on your attempts. So if you run out, try coming back the next day to continue working on the assignment!
Hello, World!
Traditionally, the first program learned is "Hello, World!". All it does is print these two words to the command line. "Printing", in the context of programming, has little to do with printers and just means outputting some text on the screen.
Have a look at the file task/script.py. It contains one line of code (starting with print...) and some comments (starting with #).
Select the script.py file and press the "Run" button to execute the code. At the bottom
Files
instructions_en.md
task
script.py
tests.py
script.py
tests.py
#!/usr/bin/env python3
# Modify the following code
print("This sentece will be printed to the command line.")
Test
Run
Submit
> Submission 1
$ No output
> Submission 2
$ No output
0 / 2
0%
Best Score
3 / 3
Submissions
History
RUN 1
16.09.2023 03:50
TEST 1
14.09.2023 09:30
Started task.
Reset


Test
Run
Submit
0 / 2
0%
Best Score
3 / 3
Submissions
History
RUN 1
16.09.2023 03:50
TEST 1
14.09.2023 09:30
Started task.
Reset

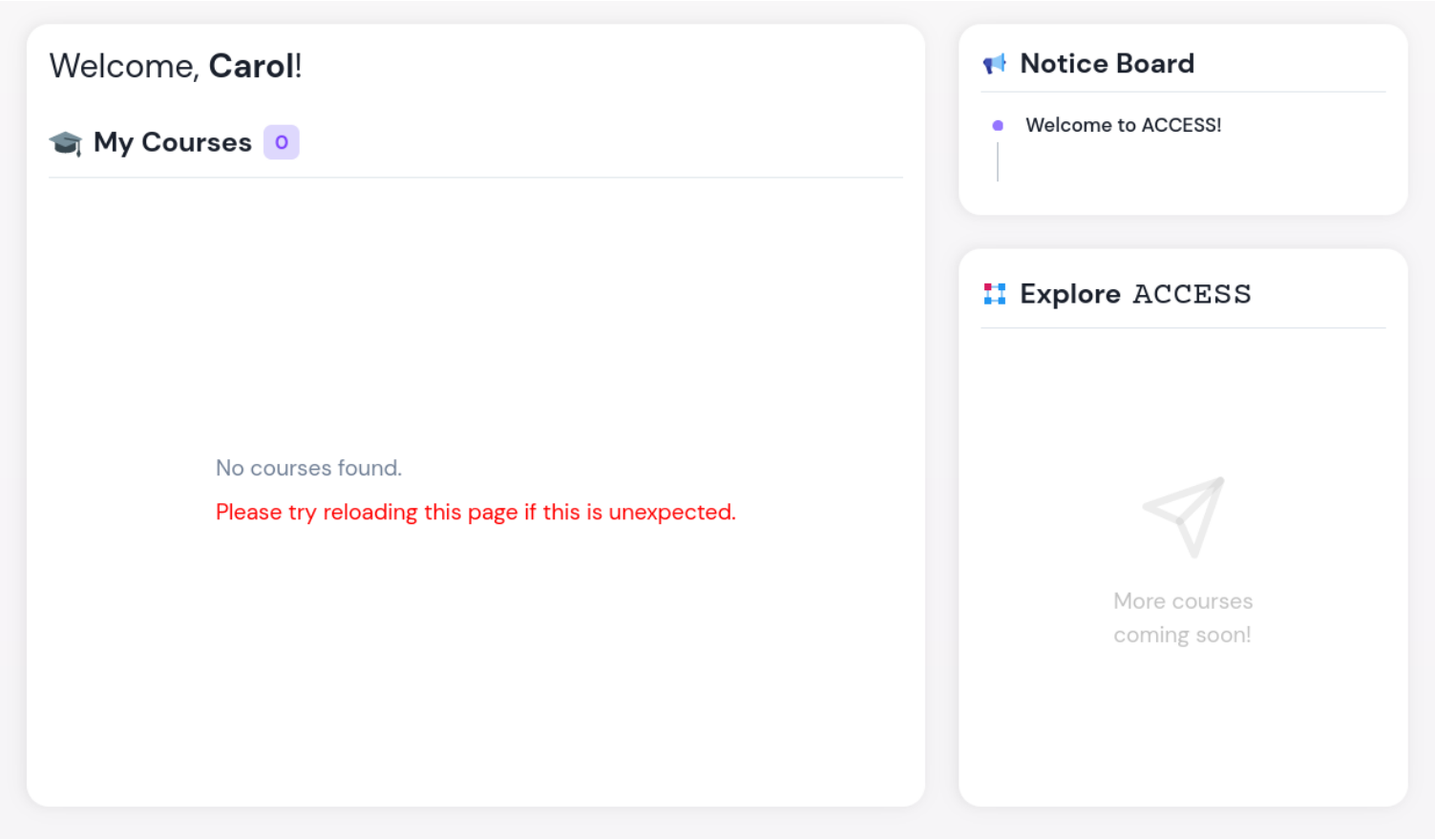

Test
Run
Submit
tests.py
env python3
he arithmetic expression you should change
need to change the following line, it just
he value of x to the console
1 / 2
50%
Best Score
2 / 3
Submissions
History
Submission 8
17.09.2023 14:13
1 / 2 Points
The solution seems to contain x = 42, please assign something slighty more complex
1 / 2 Points
Submission 6
17.09.2023 14:06
1 / 2 Points
Submission 5
17.09.2023 14:06
1 / 2 Points


Test
Run
Submit
0 / 2
0%
Best Score
3 / 3
Submissions
History
RUN 1
16.09.2023 03:50
TEST 1
14.09.2023 09:30
Started task.
Reset


Test
Run
Submit
No attempts left!
Try again in
23:59:38
0 / 1
0%
Best Score
0 / 2
Submissions
History
Submission 2
17.09.2023 13:27
0 / 1 Points
Submission 1
17.09.2023 13:27
0 / 1 Points
RUN 1
17.09.2023 13:27
Started task.
Reset

First time in ACCESS

- https://access.ifi.uzh.ch
- You should automatically gain access *after you book the Module*, but **it may take up to 48 hours**. Reload the page after a few hours if you do not yet see the course.



Important!

- You can *Run* and *Test* as often as you like and even add your own tests. However, you MUST *Submit* your solution in time to be graded.
- Your **BEST** submission will count
- The number of submission attempts is limited, but attempts will refill over time. **It makes sense to work on exercises early!**
- You see the output of Run and Test
  - "Test" tab shows the output of *your* test suite, not ours
  - "Run" tab shows you the output if your script is being run directly
- Every Run, Test and Submit creates a new version, you can freely jump between them. Reset to the template when in doubt.

Python 3 and Programming Environments

Programming locally on your computer

Dr. Carol Alexandru-Funakoshi

University of Zurich, Department of Informatics

Local programming environment required!

- ACCESS isn't everything!
- The midterm and final exams will NOT take place in ACCESS
- On your own machine, you must install:
  - Python 3.x.x (latest or second-latest version)
  - An editor & IDE of your choice & Jupyter Notebook
- You must be able to write, save and run Python scripts locally without using ACCESS.
- If you have problems installing Python, attend office hours

What code editor to use?

You must have **all** of the following three:

1. Simple editor + command line
   - e.g.: Notepad, Notepad++, BBEdit, vim, Emacs, ...
2. IDE that provides a debugger
   - e.g.: IDLE, PyCharm, VS Code, PyDev, (vim, Emacs), ...
3. Jupyter Notebook
   - Install Jupyter Notebook or use JupyterLite
   - Don't use Google Colab

Why and what for?

1. Simple editor + command line
   - To know how it works under the hood
   - If you like to keep it simple
2. IDE that provides a *debugger*
   - For "real" software development
3. Jupyter Notebook
   - Experimentation, visualization, documentation
   - **NOT** application programming

How to install Python?

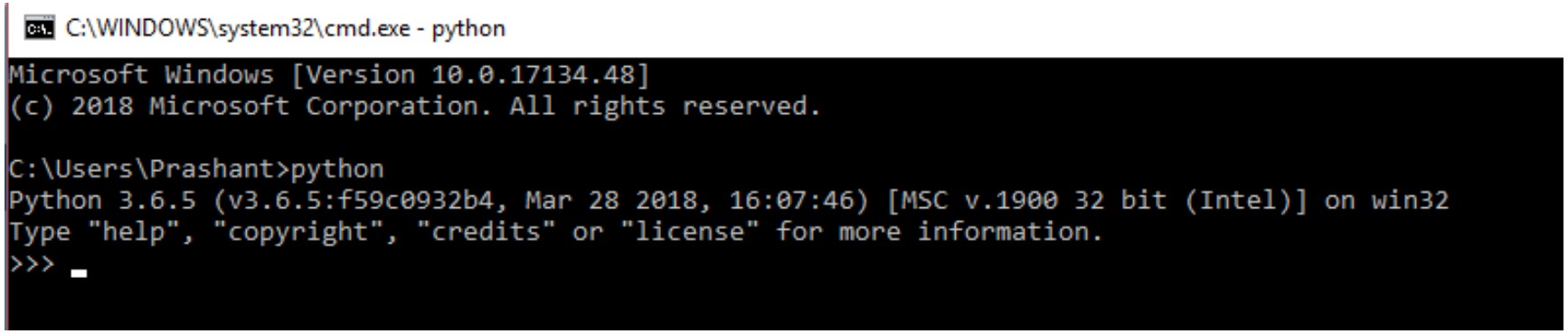
- Assignment 1, Task 2 contains a short guide
- Make sure you're running latest Python 3.x.x
  - https://www.python.org/downloads/
- On Mac: Watch out, "python" is probably python2. You must always run `python3`
- Running `python` / `python3` on the command line should produce something like this:

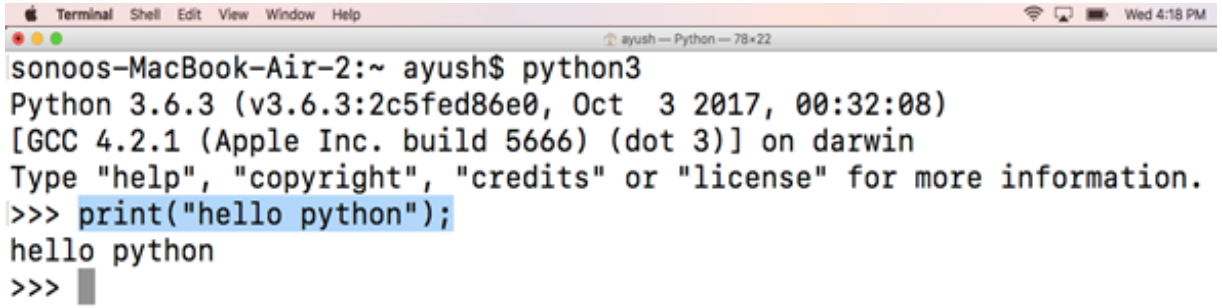
```
finch% python
Python 3.12.4 (main, Jun  7 2024, 06:33:07) [GCC 14.1.1 20240522] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

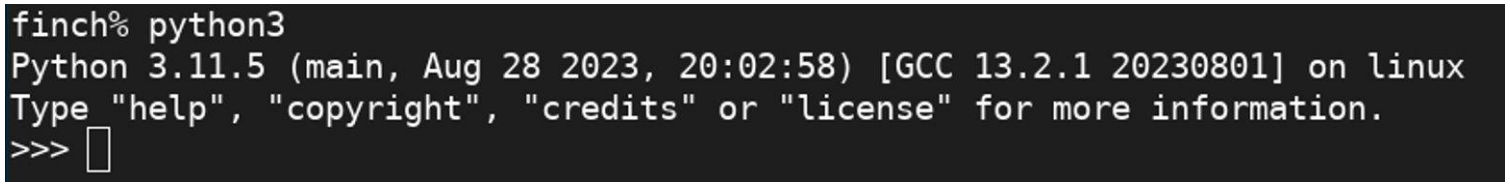
What is "the command line"

- You "talk" to the computer instead of clicking buttons
- After installing python correctly, you should be able to launch it via the terminal. To open a terminal:
  - On Windows: cmd or "command prompt"; Make sure you selected 'add to PATH' when given the option during the Python installation
  - On Mac: Terminal; Watch out, "python" is python2. You must always run python3
  - On Linux: you know what to do

What is "the command line"







Help! I'm lost!!!

- Try to keep up with the weekly material
  - The knowledge debt only increases exponentially if you miss a few weeks
- Try to solve the exercises one by one on your own
  - It starts easy, but it'll get more difficult quickly.
- If you're stuck, attend office hours! There are 3 per week and our 3 tutors' main job is to help you. You may attend multiple sessions.
- Write your questions in OLAT, we try to answer (please don't post solutions, *we can see your code in ACCESS*).
- **Learning to program is like learning an instrument: it primarily just needs a lot of practice.**