



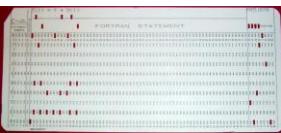
rawpixel: CC 0.0

Data



CC 3.0: Bjørn Tørrissen [Wikicommons](#)

- Plural of "datum"
 - Etymological origins in the Latin "(thing) given"
- "Thing" is an information representation (fact) of an entity that could have qualitative or quantitative features
- Mass plural noun, like rain in that it is treated in the singular,
 - we tend to say the data is good, not the data are good



Fortran Card CC 2.5, [Wikimedia commons](#)



pxhere: CC 0.0



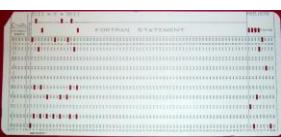
rawpixel: CC 0.0



CC 3.0: Bjørn Tørrissen [Wikicommons](#)

Data in the Empirical Sciences

- Result of observations and measurements
- **Qualitative Data**
 - descriptive features (rocky, sandy, wet, dry)
 - Amenable to Boolean Algebra
- **Quantitative (Numerical) Data**
 - Two Types
 - **Counted** - has a number and entity (two moths)
 - Exact number with no uncertainty
 - **Measured** - has a number, unit and entity (2.21 grams of moth)
 - Inexact number with uncertainty
 - Amenable to Arithmetic & Boolean Algebra



Fortran Card CC 2.5, [Wikimedia commons](#)



pxhere: CC 0.0



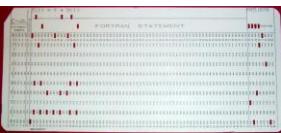
rawpixel: CC 0.0

Data in the Empirical Sciences



CC 3.0: Bjørn Tørrissen [Wikicommons](#)

- Can describe functional (causal) relationships of two or more variables
 - $y=f(x)$
 - y=dependent variable, x=independent variable
(linear, exponential, power, polynomial,...complete scatter)
 - Stored in Files
 - CSV, TSV, XML...
- Used to validate scientific theories
- Historically shared through printed (Gutenberg era) publications
- Digital Data
 - Legacy data extracted from printed journals
 - Assay data acquired through automated techniques
 - IOT data deposited to online databases in real time



Fortran Card CC 2.5, [Wikimedia commons](#)



pxhere: CC 0.0



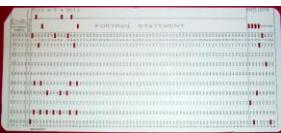
rawpixel: CC 0.0

Metadata in the Empirical Sciences



CC 3.0: Bjørn Tørrissen [Wikicommons](#)

- Metadata is data about data
 - It can contain other variables that are held constant, like the temperature, GPS coordinates...
 - It can contain instrumental parameters like the type of equipment used, solvents, sampling rate,
 - It can contain experimental parameters, like the instrument used, time stamps, operators,



Fortran Card CC 2.5, [Wikimedia commons](#)



pxhere: CC 0.0

Fair Data

F

indable

A

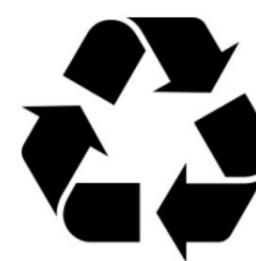
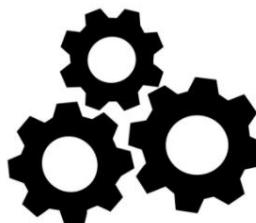
ccessible

I

nteroperable

R

eusable



Why is FAIR Data important?

The advancement of digital science thrives on the timely sharing and accessibility of digital data. Accordingly, the need for development of infrastructures and services that enable a systemic change of science practices to Open Science is now strongly advocated by both research and funding organizations. The FAIR principles strengthen these developments.



www.libereurope.eu

Ligue des Bibliothèques Européennes de Recherche
Association of European Research Libraries

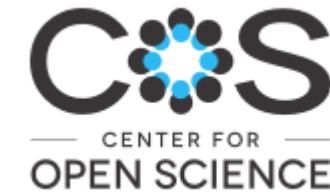
Fair Data, Open & Citizen Science

(All images are linked to resources)



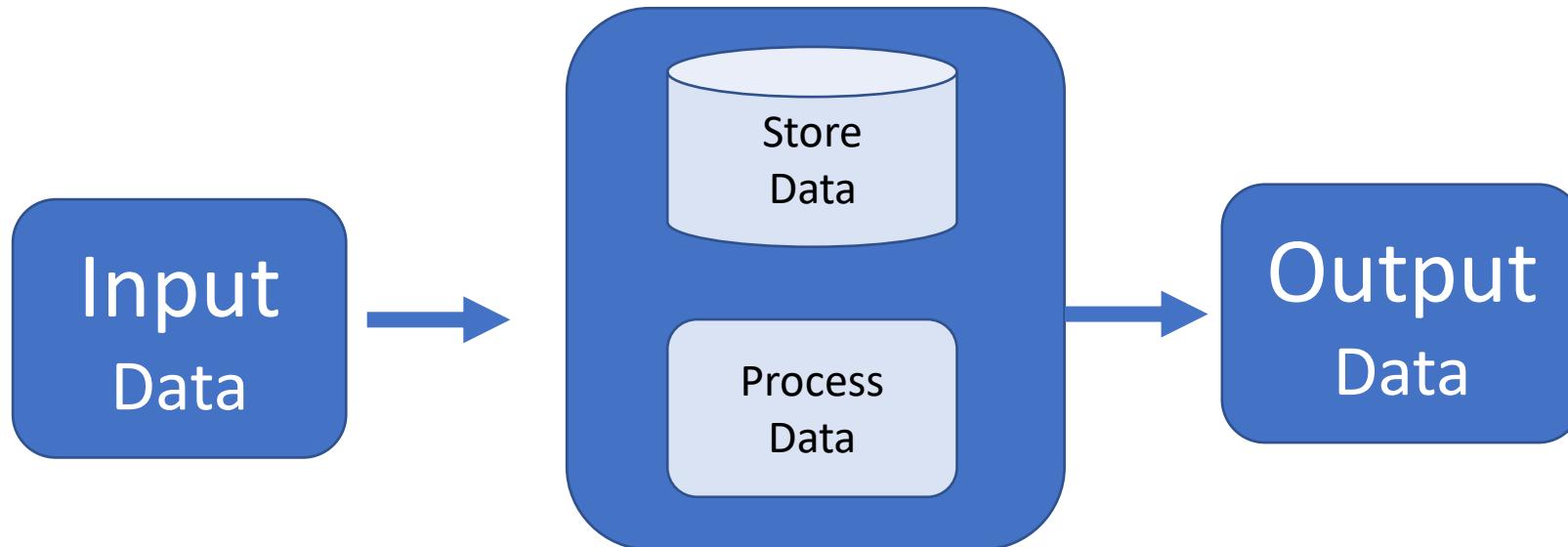
(Global Map)

(4th Paradigm Example)

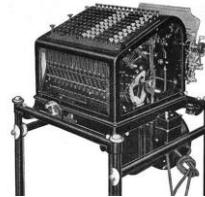


Data in the Computer Science

Computers do two things:



It's all Data



Digital Representations and Manipulation of Information

Digital Data

- Data on a computer is in binary bits with values of 0 or 1
 - Depending on data type each pattern is defined a meaning
 - bit: smallest binary value (0 or 1)
 - 1 bit has 2^1 or 2 possible combinations
 - 2 bits have 2^2 or 4 possible combinations
 - 3 bits have 2^3 or 8 possible combinations
 - 8 bits have 2^8 or 256 possible combinations
 - n bits have 2^n possible combinations
 - byte = 8 bits
 - 2 bytes have 2^{16} or 65,536 possible combinations
 - 4 bytes have 2^{32} or 42,94,967,296 possible combinations
 - 8 bytes have 2^{64} or 1.844×10^{19} possible combinations

01000100	01000101	01010011	01010000	01001001
01001111	01010011	00100000	01000101	01010011
01001110	01001001	01010000	01010101	01001100
01010011	01010101	00100000	01000001	01001110
01001100	00100000	01001110	01010111	01001111
01000001	00100000	01001100	01000001	00100000
00100000	01000100	01000101	00100000	01010100
01000101	01010011	01010100	01010010	01000001
01010100	01000001	01000100	01000101	01010011
01001111	01001110	01010100	01010010	01001111
01001001	01001111	01010011	01001111	00100000
00100000	01001100	01000001	00100000	01010000
10010011	01001110	00100000	01000001	01010011
00100000	01010101	01001110	01000001	00100000
11000011	10010011	01001110	00100000	01000100
01000001	00100000	01000100	01000101	00100000
01000001	01000011	01001001	11000011	10010011
01000000	01000000	01000000	01000000	01000000
01000000	01000000	01000000	01000000	01000000

Digital Data

- Data is Stored in Memory
 - Has an Address
 - Short Term Memory
 - RAM (Random Access Memory)
 - Persistent Memory
 - Hard Drive
 - SD card

Python Basic Data Types

Name	Python type	Description
Integers	int	Whole numbers
Floating Point	float	Decimal point numbers
Booleans	bool	True or False logical values
Strings*	str	Ordered immutable sequence of characters
Data Structures		
List	list	Ordered mutable sequence of objects [10,"10",10.0]
Dictionary	dict	Unordered Key:Value pair {'k1':'V1','K2':'V2'}
Tuple	tup	Ordered immutable sequence of objects (10,'10',10.0)
Set	set	Unordered group of unique objects {'a','b'}

*Strings are encoded as Unicode or ASCII characters and Python treats a character as a string of length=1. So strings are treated as both a data type and a data structure.

Integer (int) Data Type

4 bit binary table

$2^3 \quad 2^2 \quad 2^1 \quad 2^0$

8's	4's	2's	1's	Decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

32 bit (4 byte)

-2^{31} to $2^{31}-1$

(largest is 2,147,483,648)

64 bit (8 byte)

-2^{63} to $2^{63}-1$

(largest is 9,223,372,036,854,775,807)

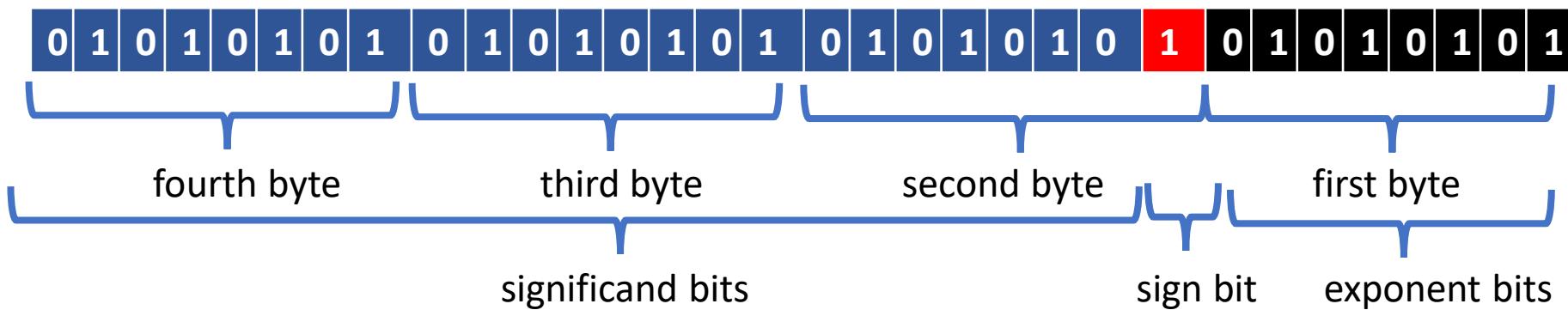
Floating (float) Data Type

-3,493.472
is stored as
 -3.493472×10^4

sign exponent

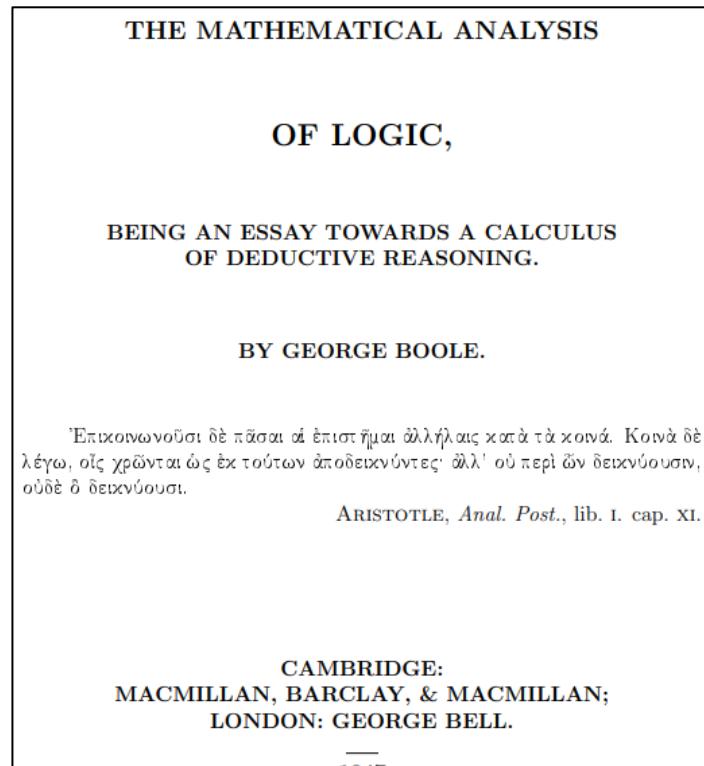
significand (mantissa)

Significant Digits Displayed are Determined by Formatting



Format	Total bits	Significand bits	Exponent bits	Smallest Number	Largest Number
Single Precision	32	23+sign	8	ca. 1.2×10^{-38}	ca. 3.4×10^{38}
Double Precision	64	52+sign	11	ca. 2.2×10^{-308}	ca. 1.8×10^{308}

Bollean (bool) Data Type



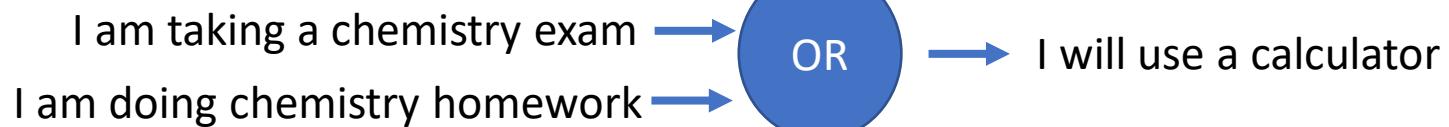
Public Domain: [Internet Archive](#)

Associating logic with binary code

0 = False

1 = True

- Boolean Operators
 - AND
 - NOT
 - OR
- Comparison Operators
 - Equal
 - Not Equal



Truth Table for OR operator

take exam	do homework	need calculator
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

take exam	do homework	need calculator
1	1	1
1	0	1
0	1	1
0	0	0

String (str) Data Type

- encoding symbols to bytes
- an ordered list of 1 or more characters (symbols)
- ASCII – American Standard Code (1963)
- UTF – Unicode Transformation Formats (1991)

Dec	BIN	Symbol	Description
0	00000000		Null Char
10	00001010		Line Feed
48	00110000	0	Zero
68	01000100	D	Uppercase D
100	01100100	d	Lowercase d
128	10000000	€	Euro
197	11000101	Å	Angstrom symbol

Extended ASCII (original was 7 bit)
8 bit representation
(<https://www.ascii-code.com/>)

ASCII Codes

- codes 0-31: unprintable control codes
- codes 31-127: printable characters
 - covers characters on keyboard and more
- codes 128-255: extended ASCII II codes
 - several variations

String (str) Data Type (Structure)

- **Strings use single, double or triple quotations**
 - represent characters not numbers ("1 proton")
 - strings can be concatenated ("1"+"1" becomes "11")
- **Strings are Ordered**
 - Strings can be indexed, even reverse indexed
- **Strings are immutable**
 - Can not change a string (does not support item reassignment)
 - Can be reassigned by slicing and concatenation
- **(Print) Formatting of Strings**
 - Modulo (%) character method (oldest)
 - .format () method
 - f-string method

List(list) Data Structure

```
molar_Mass=['hydrogen',1.00784,'helium',4.002602,'lithium', 6.941, 'beryllium', 9.012182]
```

- **Use Brackets**
 - Can have multiple objects
 - commas separate objects
- **Lists are Ordered**
 - Support Indexing and Slicing
- **Lists can be nested**
- **Lists are mutable**
- **Lists are an object class with methods that can be called of the list**
 - `list.append()` #adds item to back of list

Dictionary(dict) Data Structure

```
z_alkali_dict={"Lithium":3,"Sodium":11,"Potassium":19, "Rubidium":37, "Caesium":55,"Francium":87}
```

- **Use Curly Brackets**
- **Are Mutable**
- **Unordered**
 - Use Key:Value pairs separated by commas
- **Values can be**
 - lists
 - other dictionaries
 - strings, numbers
- **Are an object class with methods**

Tuple (tup) Data Structure

```
[('fluorine', 9, 18.99840316), ('chlorine', 17, 35.45), ('bromine', 35, 379.9),  
('iodine', 53, 126.9045), ('astatine', 85, 209.98715), ('tennessine', 117, 294.211)]
```

(Tuples nested in a **list**)

- **Use Parenthesis**
 - objects separated by commas
- **Are Immutable and ordered**
 - Can not change values or append
 - maintain data integrity
 - can be indexed
- **Can be**
 - Concatenated
 - Nested
 - Sliced
- **Are an object class with methods**

Set (set) Data Structure

- **Use Curly Brackets like Dictionaries**
 - objects separated by commas
- **Are unordered (by default)**
- **Items are unchangeable, but can be removed and added**
- **Sets show unique elements other objects like a list**

Functions

- **Block of code you can call**
- **Built-in functions**
 - Included and loaded when Python starts
 - `input()`, `print()`, `type()`
- **Library functions**
 - Need to be "imported" to use
 - May be included in Python Install
 - May be added through external packages
- **User Defined Functions**
- **Syntax**
 - Names use snake-case `lower_case_with_underscore`
 - require `()` for arguments

Arithmetic Operators

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Exponentiation	**
Division	/
Floor Division	//
Modulus	%

- Floor Division returns the largest integer $10//3 = 3$
- Modulus returns the remainder $10\%3 = 1$

Assignment Operators Act on Variables

Operator	Example	Result
=	v=4	4
+=	v+=1	5
-=	v-=1	3
=	v=3	12
/=	v/=2	2
%=	v%=/3	1
//=	v//=3	1

NOTE: = can also assign a label to a variable, like for a variable name; name="john"

Comparison (Boolean) Operators

Operation	Symbol
Equal	<code>==</code>
Not equal	<code>!=</code>
Greater than	<code>></code>
Less than	<code><</code>
Greater than or equal	<code>>=</code>
Less than or equal	<code><=</code>

NOTE: `=` assigns a value to a variable
`==` checks an equality

Logical (Boolean) Operators

Opeation	Symbol
and	True if both statements are true
or	True if one statement is true
not	True if statement not true

Identity (Boolean) Operators

Opeation	Symbol
is	true if both variables are same object
is not	true if both variables are not same object

Membership (Boolean) Operators

Opeation	Symbol
in	True if something is in an object
not in	True if something is not in an object