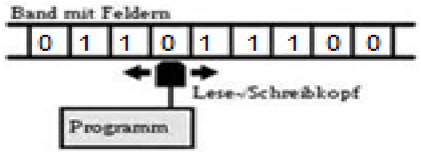
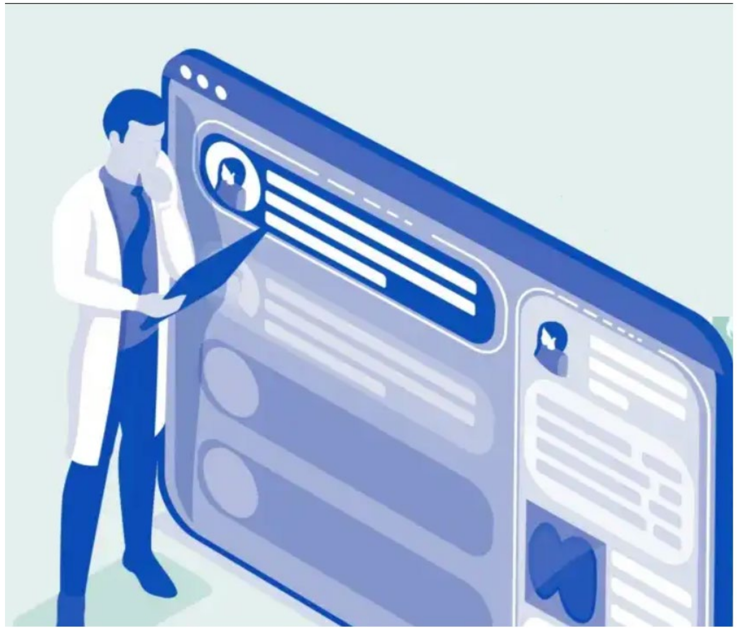
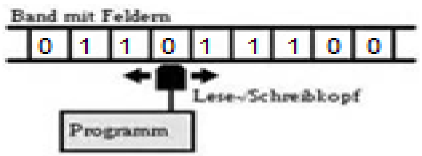


Data Storage and Interpretation



Number Systems: Roman Numbers



I	1	X	10	C	100
II	2	XX	20	CC	200
III	3	XXX	30	CCC	300
IV	4	XL	40	CD	400
V	5	L	50	D	500
VI	6	LX	60	DC	600
VII	7	LXX	70	DCC	700
VIII	8	LXXX	80	DCCC	800
IX	9	XC	90	CM	900
				M	1000

Position systems for natural numbers

A position system is a number system in which a number is divided by the powers the systems base (e.g. 2, 8, 10, 16,…).

A natural number can be represented by the following sum in different positional systems:

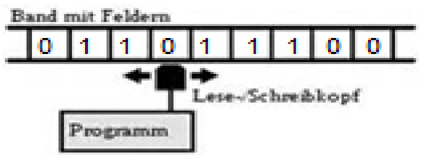
Decimal system

$25647_{10} = 7*10^0 + 4*10^1 + 6*10^2 + 5*10^3 + 2*10^4$

Binary system

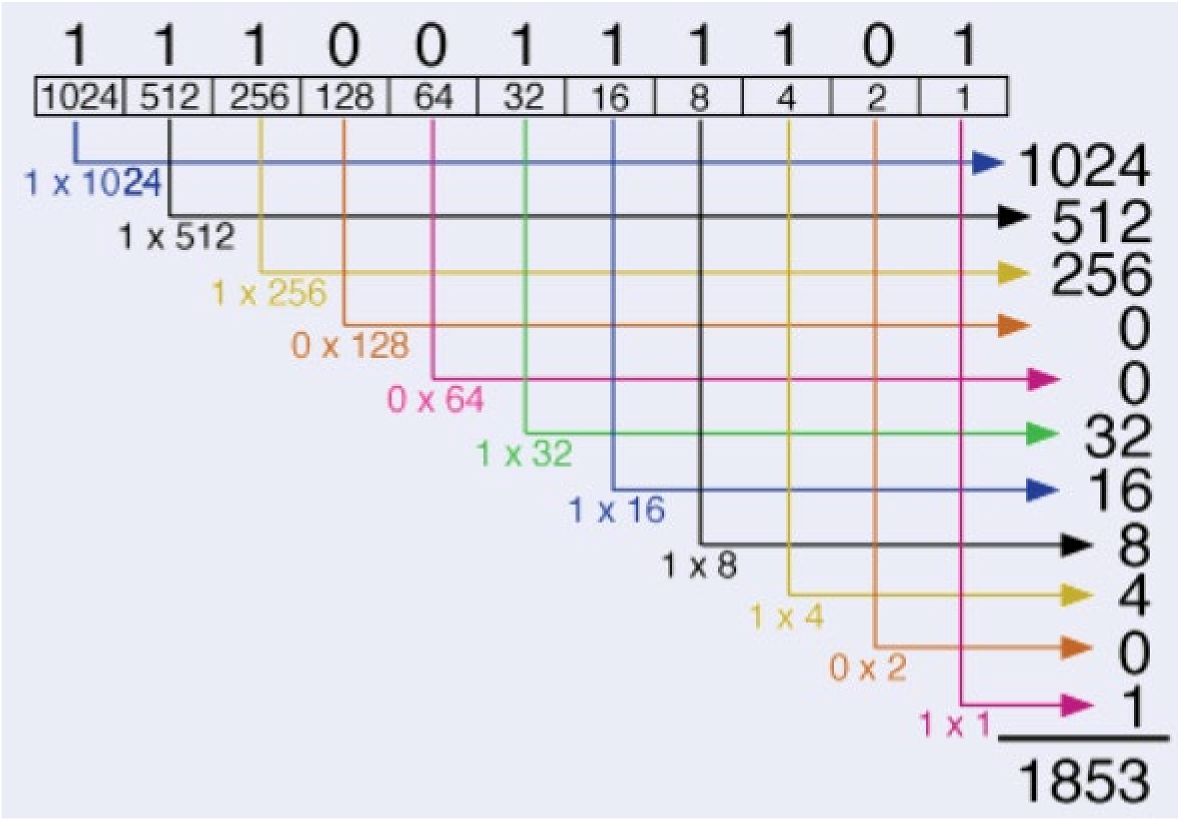
$147_{10} = 10010011 = 1*2^0 + 1*2^1 + 0*2^3 + 1*2^4 + 1*2^7$

Binary System



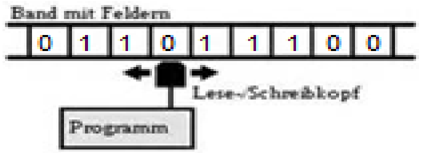
Conversion of a binary number to a digital number

binary number



digital number

The binary system



The decimal system with 10 different digits 0, 1, ... 9 is very difficult to realize technically.

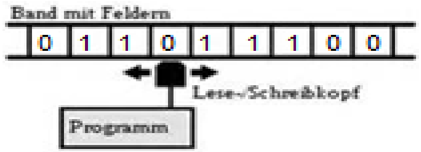
Therefore, the binary system is used in computer science today.

This consists of the digits 0 and 1, which are easy to replicate:

0 ← → no voltage/false

1 ← → voltage/true

The binary system



A single binary digit (0 or 1) is called a bit (binary digit).

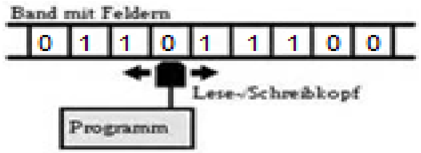
A bit is the smallest unit of information that a computer can process.

The binary system is a position system. Each position corresponds to a power of 2.

$$\begin{aligned} 10011 &= 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\ &= 19 \end{aligned}$$

$$\begin{aligned} (19)_{10} &= (10011)_2 \\ (11)_{10} &= (1011)_2 \\ (214)_{10} &= (11010110)_2 \end{aligned}$$

Addition of binary numbers



Rules for the addition of binary numbers:

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$ + carry bit

$1 + 1 + 1$ with carry bit $= 1$ + carry bit

Example:

17
+ 29

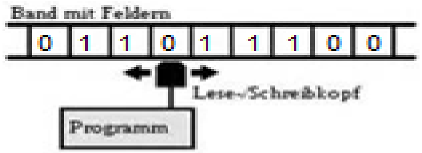
= 46

	1				1	
		1	0	0	0	1
+		1	1	1	0	1
	1	0	1	1	1	0
	32	16	8	4	2	1

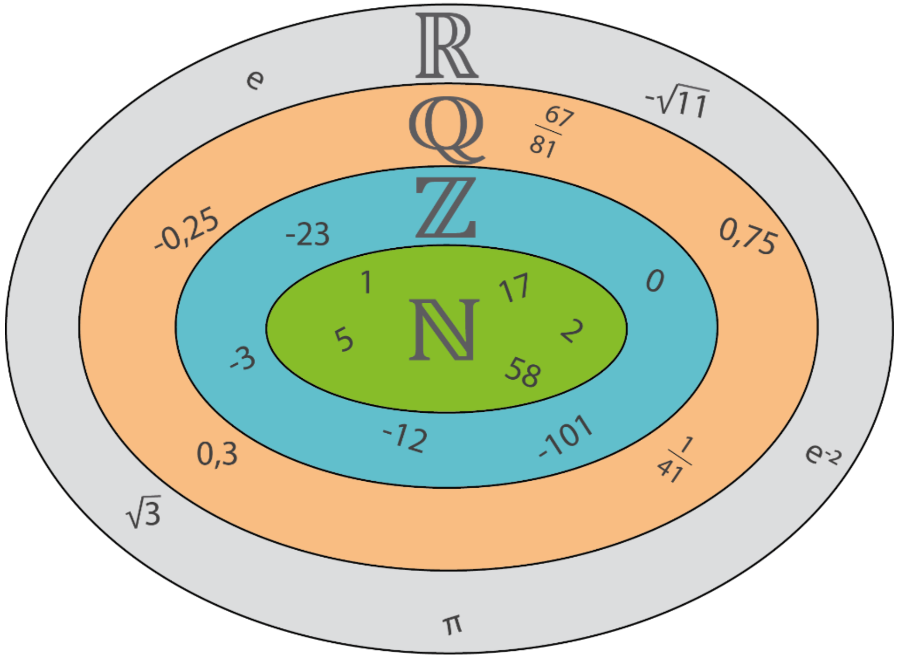
carry bit

Beatrice Amrhein, BFH

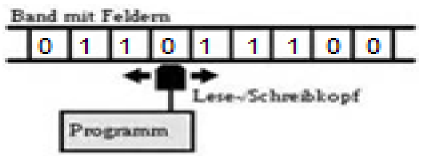
Different number types



The different number types known from mathematics are represented (more or less) similarly in computer science.



Position systems in case of floating-point numbers

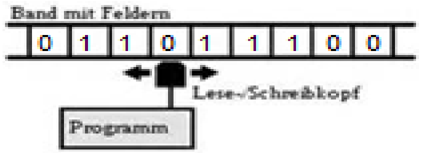


In the case of floating-point numbers, a point separates the integer part from the rational part.

Decimal system:

$$\begin{array}{lllll}
(17.05)_{10} & = 1 \cdot 10^1 & + 7 \cdot 10^0 & + 0 \cdot 10^{-1} & + 5 \cdot 10^{-2} \\
(3758.0)_{10} & = 3 \cdot 10^3 & + 7 \cdot 10^2 & + 5 \cdot 10^1 & + 8 \cdot 10^0 \\
(9.702)_{10} & = 9 \cdot 10^0 & + 7 \cdot 10^{-1} & + 0 \cdot 10^{-2} & + 2 \cdot 10^{-3} \\
(0.503)_{10} & = 0 \cdot 10^0 & + 5 \cdot 10^{-1} & + 0 \cdot 10^{-2} & + 3 \cdot 10^{-3}
\end{array}$$

Convert a broken decimal number to a binary number



The algorithm at the following link can be used to convert a floating-point decimal number x to a (floating point) binary representation:

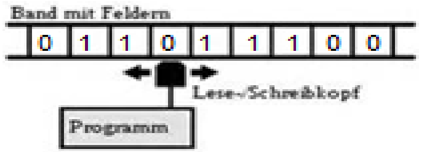
→ www.sps-lehrgang.de/umrechnung-gebrochener-dezimalzahlen

Accuracy Loss

Some floating-point numbers that can be represented exactly in the decimal system cannot be accurately represented as a binary number.

- Examples of this are numbers that can only be represented by a periodic sequence.
- E.g. The conversion of the decimal number 1.1 into a binary number is periodic:
  1.1(10) = 0.0 0011 0011 0011 0011 …(2)
- In the example, the bit pattern 0011 repeats itself.

Accuracy Loss



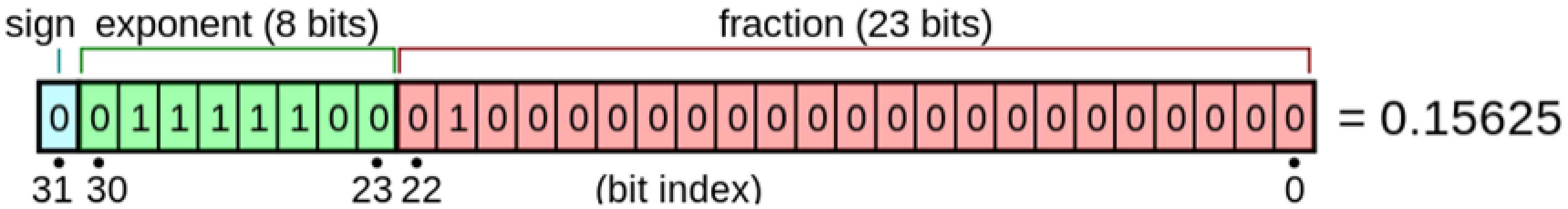
Only the numbers which can be represented as a sum of 2^k and 2^{-k} terms can be converted without accuracy loss.

$$1.1_{dec}$$
$$= 1 + 0.1_{dec}$$
$$= 1 + \frac{1}{16} + 0.0375_{dec}$$
$$= 1 + \frac{1}{16} + \frac{1}{32} + 0.00625_{dec}$$
$$1 + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + 0.00234375_{dec}$$
$$= 1 + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{512} + 0.000390625_{dec}$$
$$= 1 + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{512} + \frac{1}{4096} + 0.000146484375_{dec}$$
$$= 1 + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{512} + \frac{1}{4096} + \frac{1}{8192} + 0.0000244140625_{dec}$$
$$1.0001100110011_{bin} \text{ (with an error of } 0.0000244140625_{dec})$$

→ 1.1 can only be displayed approximately (cf. 1/3 in the decimal system)

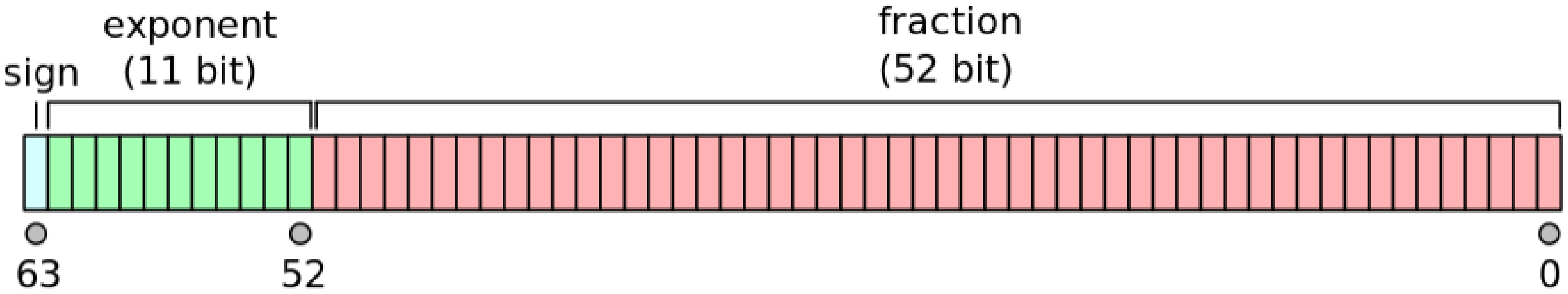
IEEE format for floating-point numbers

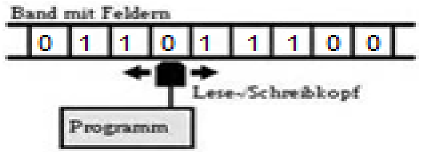
The standardized IEEE format is generally used to represent simple floating-point numbers



→ https://de.wikipedia.org/wiki/Einfache_Genauigkeit

Higher accuracy can be obtained by the double-precision floating-point format (64 instead of 32 bits)



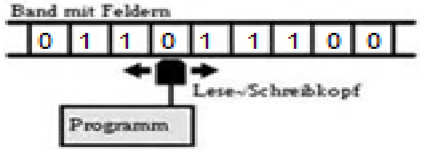


Character-encoding schemes

ASCII Code Chart

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

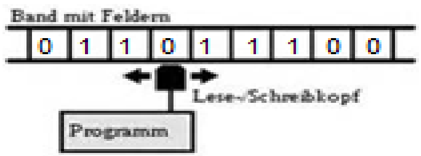
Codes for representing characters



ASCII stands for the American Standard for Coded Information Interchange for binary character encoding.

- The ASCII code includes lower- and upper-case letters of the Latin alphabet, digits, and many special characters.
- The ASCII encoding is in one byte (8 bits) so that 256 different characters can be represented.
- Because the first bit is not used by the standard ASCII code, only 128 characters can be represented.
- Different ASCII code extensions use the first bit to represent another 128 characters.

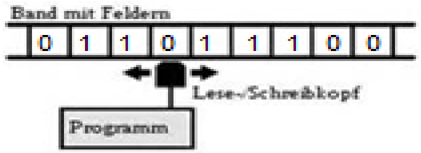
ASCII-Code table (Snippet)



HEX	DEC	CHR	HEX	DEC	CHR	HEX	DEC	CHR
20	32	SP	40	64	@	60	96	`
21	33	!	41	65	A	61	97	a
22	34	”	42	66	B	62	98	b
23	35	#	43	67	C	63	99	c
24	36	$	44	68	D	64	100	d
25	37	%	45	69	E	65	101	e
26	38	&	46	70	F	66	102	f
27	39	’	47	71	G	67	103	g
28	40	(	48	72	H	68	104	h
29	41	)	49	73	I	69	105	I
2A	42	*	4A	74	J	6A	106	j
2B	43	+	4B	75	K	6B	107	k
2C	44	,	4C	76	L	6C	108	l
2D	45	-	4D	77	M	6D	109	m
2E	46	.	4E	78	N	6E	100	n
2F	47	/	4F	79	O	6F	111	o
30	48	0	50	80	P	70	112	p
31	49	1	51	81	Q	71	113	q
32	50	2	52	82	R	72	114	r
33	51	3	53	83	S	73	115	s

https://www.ascii-code.com/

Unicode

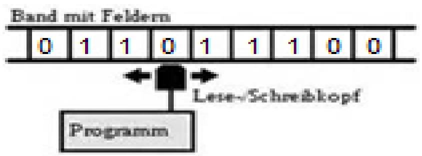


- The ASCII code is limited to 256 characters.
- Unicode introduced different standards in which the characters of almost all known cultures and drawing systems can be mapped.

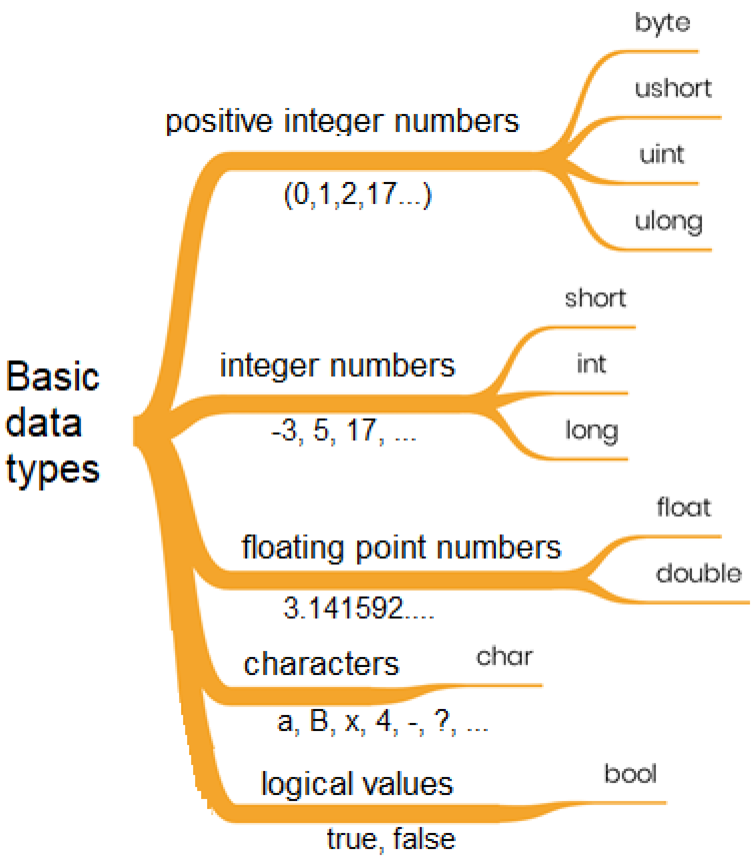
- UTF-8 encoding is usually recommended for standard character values.

  https://de.wikipedia.org/wiki/UTF-8
- For languages with many special characters longer encodings (UTF-16 or UTF-32) are recommended.

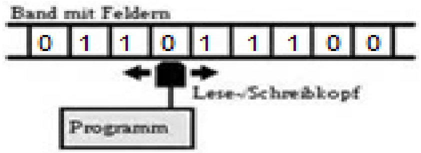
Unicode stands for Universal Coded Character Set


Band mit Feldern
0 1 1 0 1 1 1 0 0
Lese-/Schreibkopf
Programm

Datatypes


Basic data types
positive integer numbers
(0,1,2,17...)
byte
ushort
uint
ulong
integer numbers
-3, 5, 17, ...
short
int
long
floating point numbers
3.141592....
float
double
characters
a, B, x, 4, -, ?, ...
char
logical values
true, false
bool

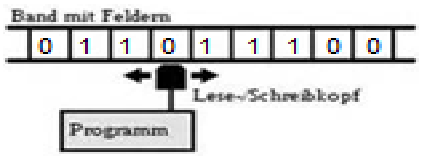
Basic data types

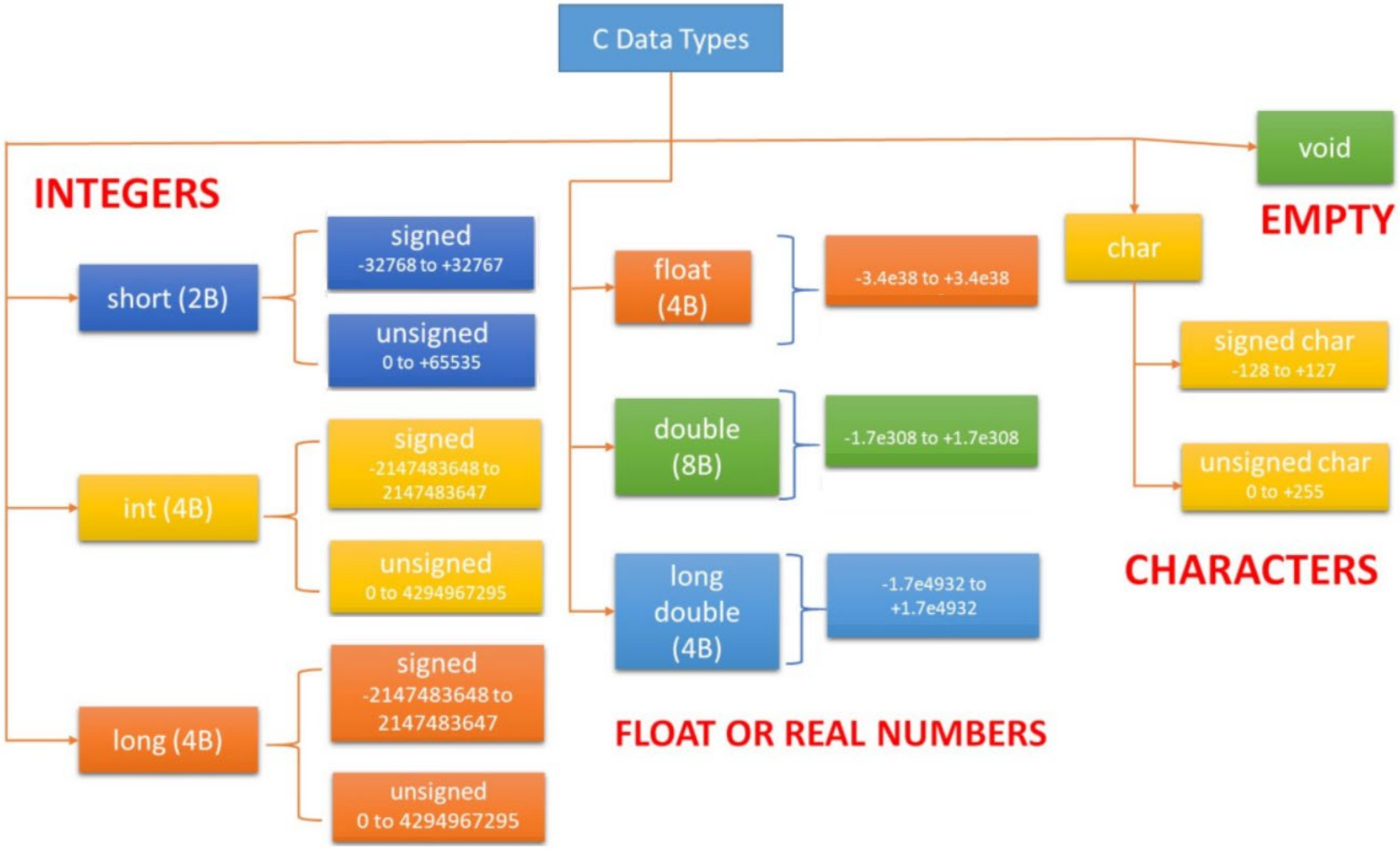


- Depending on the type of data (letters, integers, floating point numbers, ...) data is stored differently.
- A classification of different data is necessary.
- Different memory requirements
- Different representable number precision
- Different characters encodings
- Interpretation of the bit pattern.
- ....

The assignment of the data to specific classes such as characters, integer, single or double precision numbers, ... defines their data type.

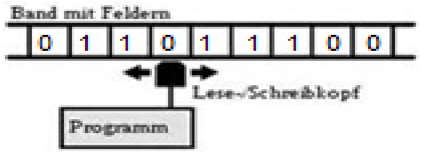
Basic Data Types in C

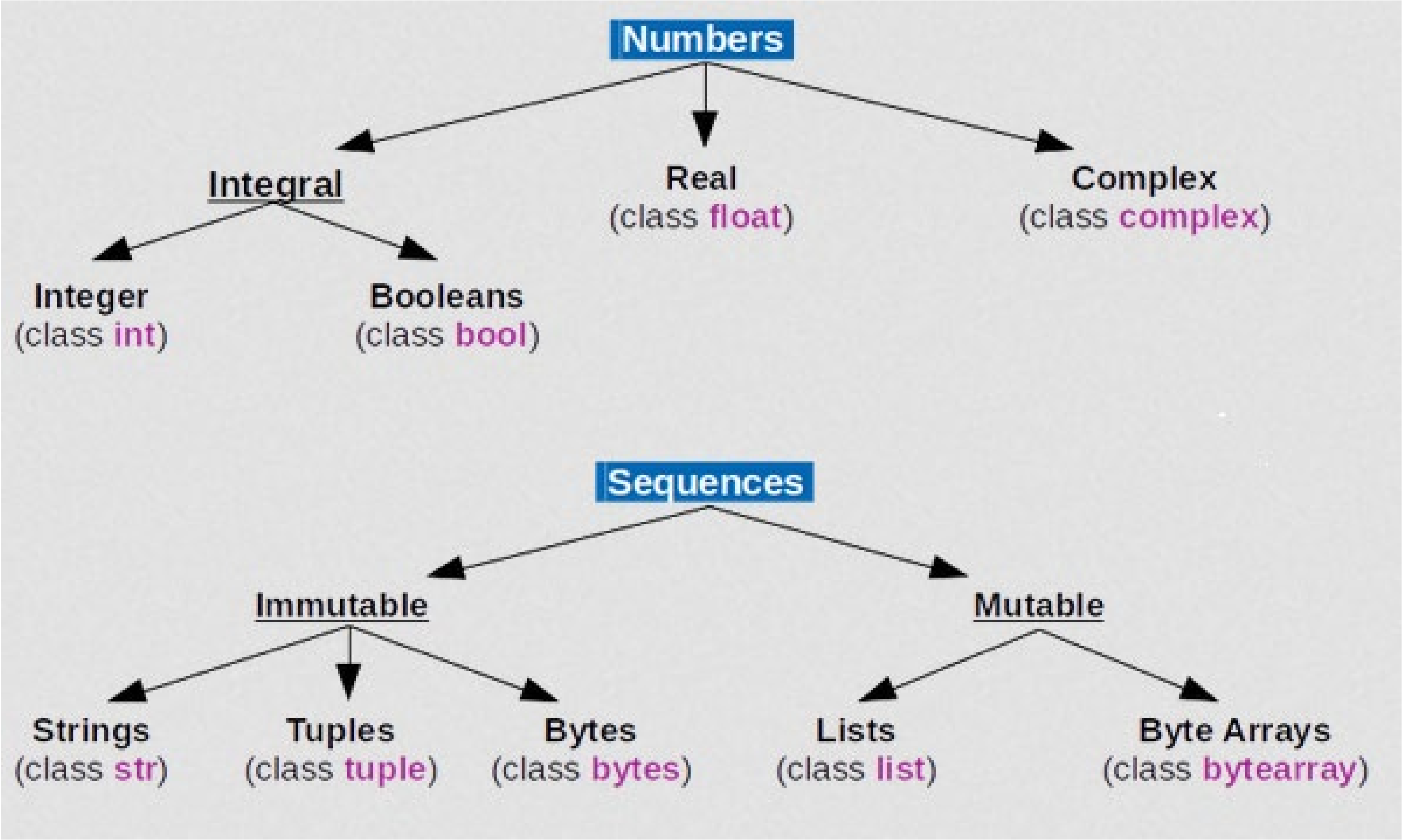




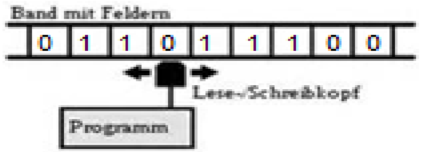
Beatrice Amrhein, BFH

Basic Data Types in Python





Interpretation of data



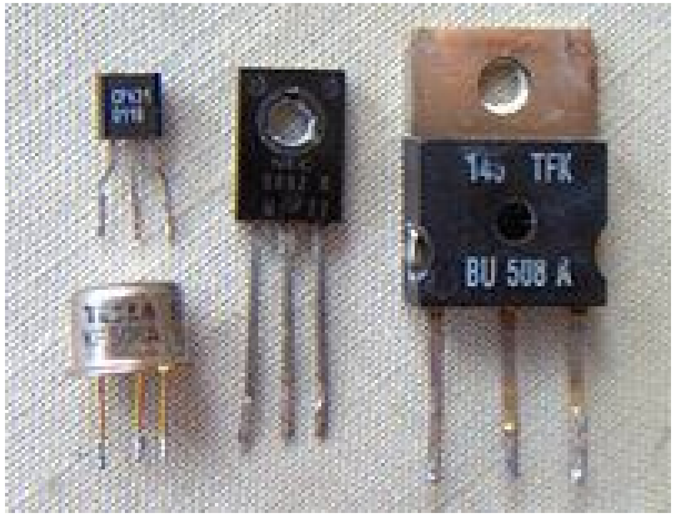
Binary data of a file with different interpretations

```
377 330 377 340
000 020 112 106
111 106 000 001
001 001 000 110
000 110 000 000
377 333 000 103
000 006 004 005
006 005 004 006
006 005 006 007
007 006 010 012
020 012 012 011
011 012 ....
```

Interpretation as an octal sequence

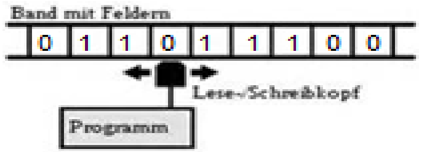
```
377 330 377 340
\0  020 J   F
I   F   \0  001
001 001 \0  H
\0  H   \0  \0
377 333 \0  C
\0  006 004 005
006 005 004 006
006 005 006 \a
\a  006 \b  \n
020 \n  \n  \t
\t  \n  ....
```

interpretation as ASCII byte



Interpretation as image (jpg)

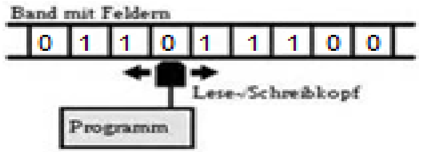
Conclusion



- All numbers, characters are digitally processed and stored.
- The semantics becomes clear only in combination with the used representation (its data type).

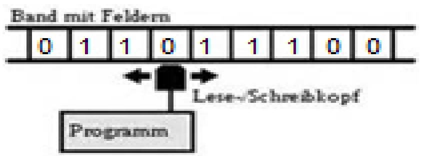
- Since the processing of data is different depending on the type of data, the knowledge of its classification or data type is mandatory.

Conclusion



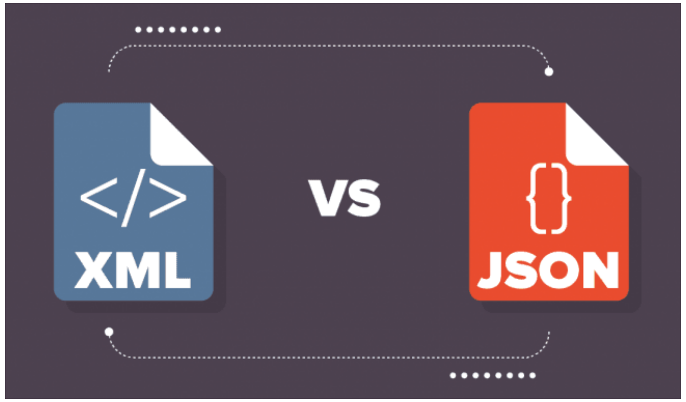
- Specifying, testing and using software requires knowledge of data types.
  - data types and their properties and usage.
  - Value ranges / precision
  - Accuracy loss (explicit, or implicit conversion)
  - Impact on resource demand (e.g. memory or CPU time)

- Incorrect specification may result in unusable results.
  - The requirements for the above points must be known at the time of specification.

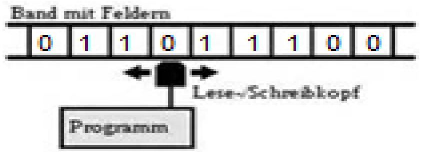


Data Exchange Formats

XML ← → JSON

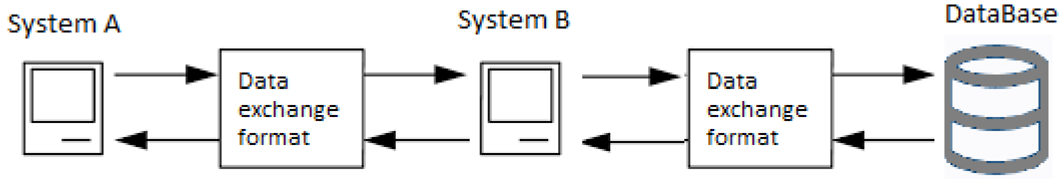


XML and JSON

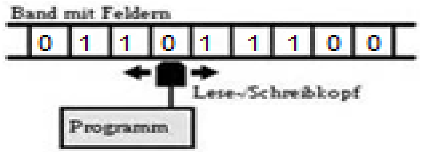


Data interchange format to

- store structured data
- transmit information (data objects) between (local or remote) applications
- send requests over the internet
- receive data object (as a response to a previous request)



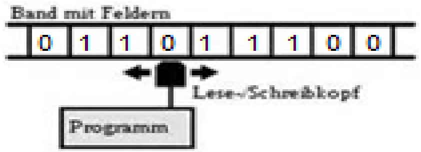
XML Data Format



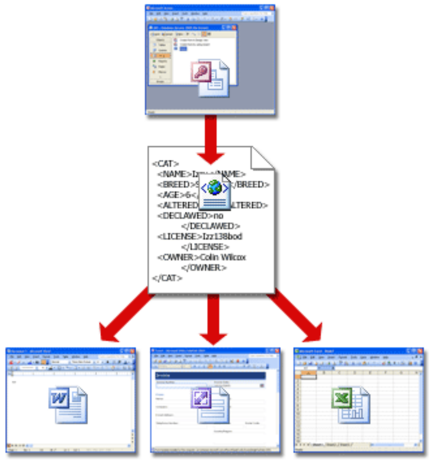
- XML stands for eXtensible Markup Language.
- XML was designed to store and transport data.
- XML was designed to be both human- and machine-readable.

```
<?xml version = "1.0", encoding="UTF-8"?>
<book id="se-xvv-2332">
   <title>A Song of Ice and Fire</title>
   <author>
       <firstName>George R. R.</firstName>
       <lastName>Martin</lastName>
   </author>
   <language>EN</language>
   <release>2010-08-10</release>
   <genre>Epic fantasy</genre>
   <pages>895</pages>
   <price currency="CHF">32.50</price>
   <price currency="USD">30.10</price>
</book>
```

XML Data Format

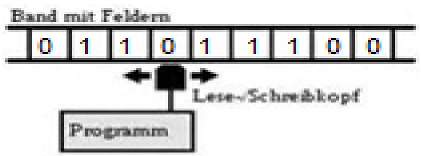


- XML plays an important role in different IT systems.
- XML is often used for distributing data over the Internet.



- Important XML standards and tools
  - XML XPath/ XQuery → access XML elements
  - XML XSLT → transform XML documents (filter, sort, …)
  - XML Schema → Validate XML input

JSON

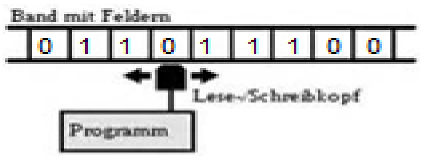


JSON

- stands for JavaScript Object Notation
- is used to send data between different applications over the Internet
- is text based and human readable

```
{
  "name": "Georg",
  "age": 47,
  "children": [
    {
      "name": "Lukas",
      "age": 21,
      "school": "university"
    },
    {
      "name": "Lisa",
      "age": 14,
      "school": "college"
    }
  ]
}
```

JSON



JSON

- is supported by many programming languages
- consists of name/value pairs

```json
{
  "string": "Hi",
  "number": 2.5,
  "boolean": true,
  "object": { "category" : "Kl", "nr": 24 },
  "array": ["Hello", 5, false, null],
  "arrayOfObjects": [
    { "name": "Jerry", "age": 28 },
    { "name": "Sally", "age": 26 }
  ]
}
```

JSON vs. XML

JSON is Like XML because

- They are "self describing
- They are hierarchical (values within values)
- They can be parsed and used by lots of programming languages
- They can be read with an XMLHttpRequest

- JSON code is shorter than XML code
- XML has strong tool support
- XML defines standards for many topics (SVG, OpenStreetMap, SOAP, MathML, …)