



Ceng 111 – Fall 2021

Week 7a

Credit: Some slides are from the “Invitation to Computer Science” book by G. M. Schneider, J. L. Gersting and some from the “Digital Design” book by M. M. Mano and M. D. Ciletti.



Previously on CENG111!



- An **interpretive/scripting** PL that:
 - Longs for code readability
 - Ease of use, clear syntax
 - Wide range of applications, libraries, tools
- **Zen of Python** [https://en.wikipedia.org/wiki/Zen_of_Python]
 - Beautiful is better than ugly.
 - Explicit is better than implicit.
 - Simple is better than complex.
 - Complex is better than complicated.
 - Flat is better than nested.
 - Sparse is better than dense.
 - Readability counts.
 - ...



Previously on CENG111!



- Supports multiple paradigms:
 - Functional
 - Imperative
 - Object-oriented



Previously on CENG111!



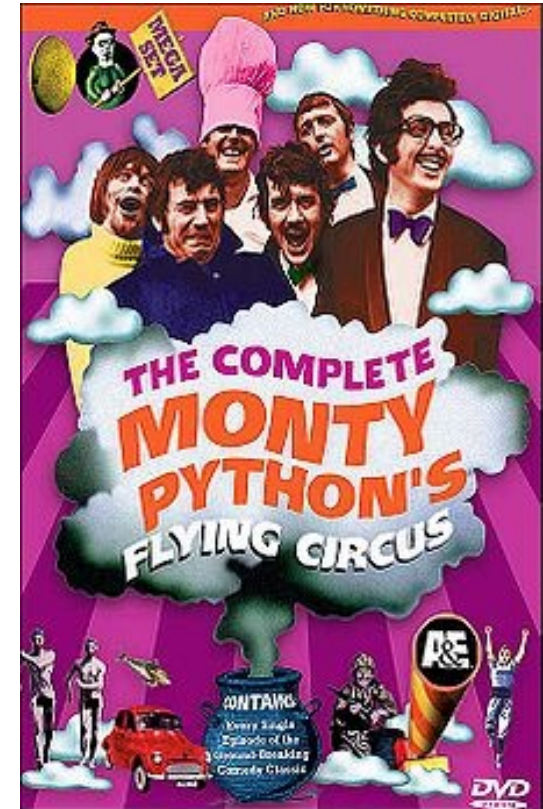
- Started at the end of 1980s.
- V2.0 was released in 2000
 - With a big change in development perspective: Community-based
 - Major changes in the facilities.
- V3.0 was released in 2008
 - Backward-incompatible
 - Some of its features are put into v2.6 and v2.7.



Previously on CENG111!



- Where does the name come from?
 - While van Rossum was developing Python, he read the scripts of Monty Python's Flying Circus and thought 'python' was "short, unique and mysterious" for the new language [1]
- One goal of Python: "fun to use"
 - The origin of the name is the comedy group "Monty Python"
 - This is reflected in sample codes that are written in Python by the original developers.





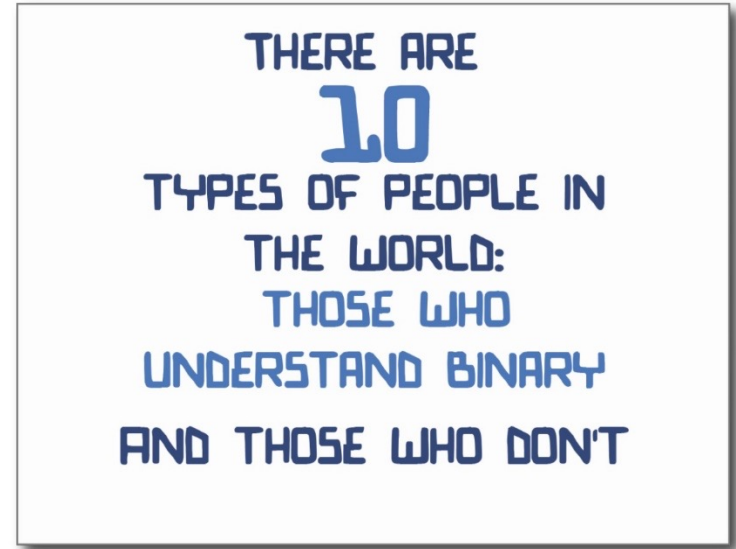
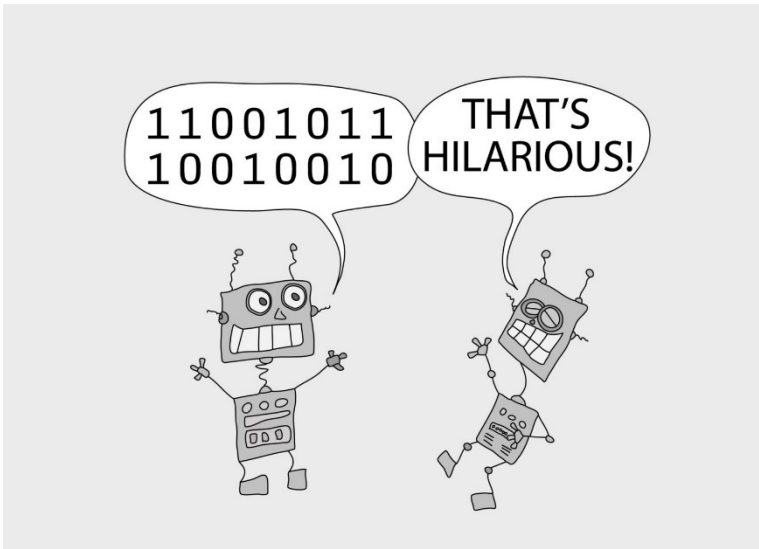
Today

■ Data Representation



Administrative Notes

- Scratch assignment
- Midterm date:
 - 22 December, Wednesday, 18:00



BINARY REPRESENTATION



Data Representation

- Based on 1s and 0s
 - So, everything is represented as a set of binary numbers
- We will now see how we can represent:
 - Integers: 3, 1234435, -12945 etc.
 - Floating point numbers: 4.5, 124.3458, -1334.234 etc.
 - Characters: /, &, +, -, A, a, ^, 1, etc.
 - ...



Binary Representation of Numeric Information

■ Decimal numbering system

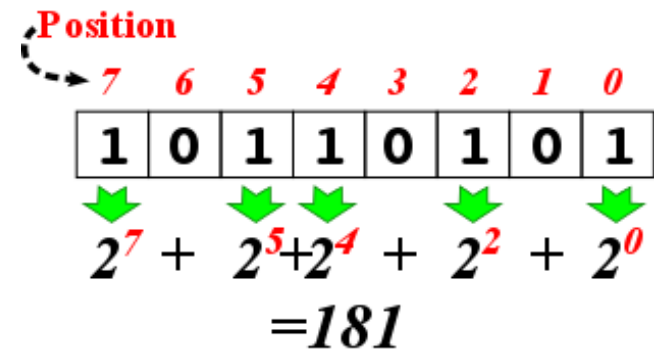
- Base-10
- Each position is a power of 10

$$3052 = 3 \times 10^3 + 0 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$$

■ Binary numbering system

- Base-2
- Uses ones and zeros
- Each position is a power of 2

$$1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$



Decimal-to-binary Conversion

	Dividend		Divisor		Quotient	Remainder
Step 1	19	÷	2	=	9	1
Step 2	9	÷	2	=	4	1
Step 3	4	÷	2	=	2	0
Step 4	2	÷	2	=	1	0
Step 5	1	÷	2	=	0	1

Continue until quotient is zero


The result:

1	0	0	1	1
---	---	---	---	---



Binary Representation of Numeric Information (continued)

- Representing integers
 - Decimal integers are converted to binary integers
 - **Question:** given k bits, what is the value of the largest integer that can be represented?
 - $2^k - 1$
 - Ex: given 4 bits, the largest is $2^4 - 1 = 15$
- Signed integers must also represent the sign (positive or negative) - ***Sign/Magnitude notation***



Binary Representation of Numeric Information (continued)

■ Sign/magnitude notation

$$1\ 101 = -5$$

$$0\ 101 = +5$$

■ Problems:

■ Two different representations for 0:

- $1\ 000 = -0$

- $0\ 000 = +0$

■ Addition & subtraction require a watch for the sign! Otherwise, you get wrong results:

- $0\ 010 (+2) + 1\ 010 (-2) = 1\ 100 (-4)$

Arithmetic in Computers is Modular

Let's add two numbers in binary
(Assume that there is no sign bit)

1	0	1	1
1	1	1	0

+

~~1~~

1	0	0	1
---	---	---	---

→

→

→

(11)₁₀

(14)₁₀

(9)₁₀

In other words:

- Numbers larger than or equal to 16 (2^4) are discarded in a 4-bit representation.
- Therefore, $11 + 14$ yields 9 in this 4-bit representation.
- This is actually modular arithmetic:

$$11 + 14 \bmod 16 \equiv 9 \bmod 16$$

2020

S. Kalkan & G. Ucoluk - CEng 111

14



Binary Representation of Numeric Information (continued)

- **Two's complement** instead of sign-magnitude representation
 - Positive numbers have a leading 0.
 - $5 \Rightarrow 0101$
 - The representation for negative numbers is found by subtracting the absolute value from 2^N for an N-bit system:
 - $-5 \Rightarrow 2^4 - 5 = 16 - 5 = (11)_{10} \Rightarrow (1011)_2$
- Advantages:
 - 0 has a single representation: $+0 = 0000$, $-0 = 0000$
 - Arithmetic works fine without checking the sign bit:
 - $1011 (-5) + 0110 (6) = 0001 (1)$
 - $1011 (-5) + 0011 (3) = 1110 (-2)$



Binary Representation of Numeric Information (continued)

- Shortcut to convert from “two’s complement” :
 - If the leading bit is zero, no need to convert.
 - If the leading bit is one, invert the number and add 1.
- What is our range?
 - With 2’s complement we can represent numbers from -2^{N-1} to $2^{N-1} - 1$ using N bits.
 - 8 bits: -128 to +127.
 - 16 bits: -32,768 to +32,767.
 - 32 bits: -2,147,483,648 to +2,147,483,647.

Binary Number	Decimal Value	Value in Two’s Complement
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1



Binary Representation of Numeric Information (continued)

■ Example:

- We want to compute: $12 - 6$
- $12 \Rightarrow 01100$
- $-6 \Rightarrow -(00110) \Rightarrow (11001)+1 \Rightarrow (11010)$

■ $12 - 6 =$

$$\begin{array}{r} 01100 \\ + 11010 \\ \hline \end{array}$$

$$00110 \Rightarrow 6$$

So, addition and subtraction operations are simpler in the Two's Complement representation



Binary Representation of Numeric Information (continued)

- Due to its advantages, two's complement is the most common way to represent integers on computers.


Why does Two's Complement work?

■ One perspective:

- Inversion and addition of a 1-bit correspond effectively to subtraction from 0 – i.e., negative of a number.
- Negative of a binary number X : $(00...00)_2 - (X)_2$
- Note that $(00...00)_2 = (11...11)_2 + (1)_2$
- In other words:
 - $(00...00)_2 - (X)_2 = (11...11)_2 - (X)_2 + (1)_2$.

(i.e., how we find two's complement)

Inversion



Why does Two's Complement work?

■ Or, equivalently:

- $i - j \bmod 2^N = i + (2^N - j) \bmod 2^N$

- Example:

- Consider X and Y are positive numbers.

- $$\begin{aligned} X + (-Y) &= X + (2^N - Y) \\ &= 2^N - (Y - X) = -(Y - X) = X - Y \end{aligned}$$

Why does Two's Complement work?

- A smart trick used in mechanical calculators
 - To subtract b from a , invert b and add that to a . Then discard the most significant digit.



http://en.wikipedia.org/wiki/Method_of_complements