# COMP 141

CS1: Programming Fundamentals

Rhodes College



## Announcements

Reminder
- Program 9 due Monday, December 9th by 11:55pm.

Final Exam
- Friday, December 6th
- 1:00-3:30pm
- Briggs 019
- You will be given the same lists & strings functions handout as Midterm 2
- Review packet on Course Website (solutions on Moodle)



## Tic-Tac-Toe with Graphics

Any questions?

Solutions in Box.com folder

## Underlying Data Representation

- Remember back to the beginning of the semester
- We said that all data in a computer is stored in sequences of 0s and 1s
- **Byte**: just enough memory to store letter or small number
  - Divided into eight bits
  - **Bit**: electrical component that can hold positive or negative charge, like on/off switch
  - The on/off pattern of bits in a byte represents data stored in the byte

## Binary Numbers

A Binary Number is made up of only **0**s and **1**s.

Example of a Binary Number

**110100**

There is no 2,3,4,5,6,7,8 or 9 in Binary!



## How do we count using binary?

| Binary | |
|---|---|
| 0 | We start at 0 |
| 1 | Then 1 |
| ??? | But then there is no symbol for 2…what do we do? |



## How do we count in Decimal?

| Decimal | |
|---|---|
| 0 | Start at 0 |
| … | Count 1,2,3,4,5,6,7,8 |
| 9 | This is the last digit in Decimal |
| 10 | So we start back at 0 again, but add 1 on the left |



## Applying to Binary

| | Binary | |
|---|---|---|
| | 0 | We start at 0 |
| * | 1 | Then 1 |
| ** | 10 | Now we start back at 0, and add 1 to the left |
| *** | 11 | 1 more |
| **** | 100 | Start back at 0 again, and add one to the number on the left… … but that number is already at 1 so it also goes back to 0 … … and 1 is added to the *next position* on the left |

## Decimal vs. Binary

| Decimal: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Binary: | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

| Decimal: | 20 | 25 | 30 | 40 | 50 | 100 | 200 | 500 |
|---|---|---|---|---|---|---|---|---|
| Binary: | 10100 | 11001 | 11110 | 101000 | 110010 | 1100100 | 11001000 | 111110100 |

"Binary is as easy as 1, 10, 11."



## Binary Numbers

- Each position in a binary number represents $2^n$

$$10111 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$$

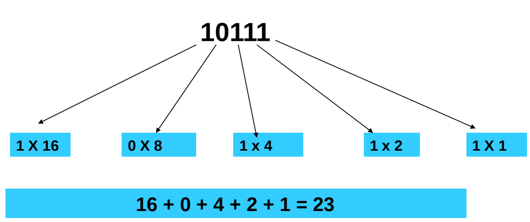
$$101001 = 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0$$

- This is the same as the decimal system:

$$193 = 1 * 10^2 + 9 * 10^1 + 3 * 10^0$$



## Converting Binary to Decimal






## Converting Decimal to Binary

- **47 / 2 = 23 rem 1**
- **23 / 2 = 11 rem 1**
- **11 / 2 = 5 rem 1**
- **5 / 2 = 2 rem 1**
- **2 / 2 = 1 rem 0**
- **1 / 2 = 0 rem 1**
- **Hence 47 in decimal format equals 101111 in binary format.**

## Adding Binary Numbers

**10011**

**+ 1111**

**100010**



## Practice

- Convert $39_{10}$ into binary
- Convert $1010110_2$ into decimal

## Practice

- Write 2 functions:
  - toBinary(decimal) – takes in a decimal number and returns its binary equivalent
  - toDecimal(binary) – takes in a binary number and returns its decimal equivalent
  - **Hints**:
    - In toDecimal, you should convert binary to a string
    - In toBinary, you should create binary as a string, then typecast it to an integer before returning.
- Examples:

```
print(toBinary(1198)) #Prints  10010101110
print(toBinary(5))    #Prints 101

print(toDecimal(10001110)) #Prints 142
print(toDecimal(11))       #Prints 3
```