

# CSci 127: Introduction to Computer Science



[hunter.cuny.edu/csci](http://hunter.cuny.edu/csci)

# Announcements



- No classes Thursday or Friday due to the Thanksgiving holiday.

# Announcements



- No classes Thursday or Friday due to the Thanksgiving holiday.
- Lab will close early today (7pm) due to the holiday.

# Announcements



- No classes Thursday or Friday due to the Thanksgiving holiday.
- Lab will close early today (7pm) due to the holiday.
- Have a great Thanksgiving!

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.*



# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*
  - ▶ *All previous final exams (and answer keys) on the website.*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries– we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*
  - ▶ *All previous final exams (and answer keys) on the website.*
  - ▶ *We'll have a mock exam during the last lecture (12 December).*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries— we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*
  - ▶ *All previous final exams (and answer keys) on the website.*
  - ▶ *We'll have a mock exam during the last lecture (12 December).*
    - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries— we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*
  - ▶ *All previous final exams (and answer keys) on the website.*
  - ▶ *We'll have a mock exam during the last lecture (12 December).*
    - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
    - ★ *We will use the sign-out sheets to give credit for the lecture slip.*



# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries— we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*
  - ▶ *All previous final exams (and answer keys) on the website.*
  - ▶ *We'll have a mock exam during the last lecture (12 December).*
    - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
    - ★ *We will use the sign-out sheets to give credit for the lecture slip.*
    - ★ *Answer key will be available after lecture.*

# Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!  
*No worries— we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*
  - ▶ *All previous final exams (and answer keys) on the website.*
  - ▶ *We'll have a mock exam during the last lecture (12 December).*
    - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
    - ★ *We will use the sign-out sheets to give credit for the lecture slip.*
    - ★ *Answer key will be available after lecture.*
- Can we do more on design patterns?

# Frequently Asked Questions

From lecture slips & recitation sections.

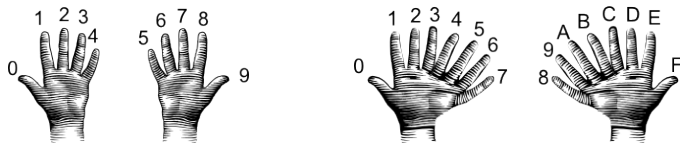
- Help! Binary & hexadecimal numbers make no sense!  
*No worries— we'll start off with those in today's lecture.*
- Why can't in-class quizzes taken at midnight count towards my grade?  
*The in-class quizzes are in place of midterm exams.  
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?  
*The final is Wednesday, 19 December, 9-11am.  
Instead of a review sheet, we have:*
  - ▶ *All previous final exams (and answer keys) on the website.*
  - ▶ *We'll have a mock exam during the last lecture (12 December).*
    - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
    - ★ *We will use the sign-out sheets to give credit for the lecture slip.*
    - ★ *Answer key will be available after lecture.*
- Can we do more on design patterns?  
*Yes, but we're going to transition to C++ after Thanksgiving.*

# Today's Topics



- Data Representation
- Machine Language: Jumps & Loops
- Recap of Python & Circuits
- Design Patterns: Sorting

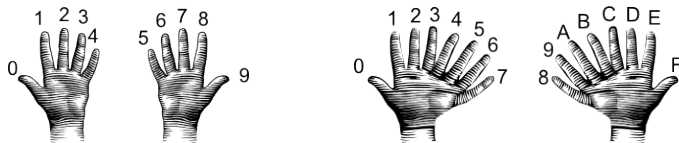
# Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:

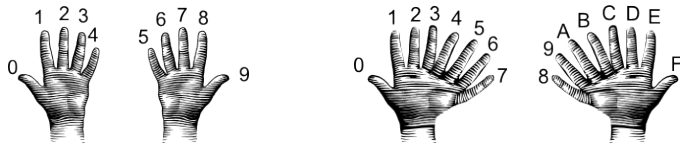
# Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:
  - ▶ Divide by 16.

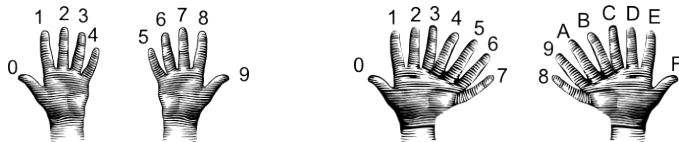
# Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:
  - ▶ Divide by 16.
  - ▶ Convert quotient and remainder into hex digits.

# Recap: Converting between bases



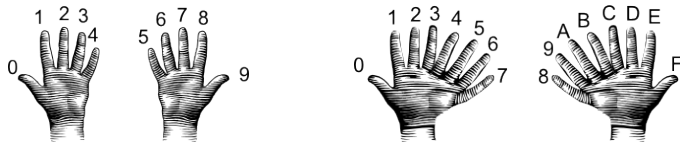
(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.



# Recap: Converting between bases

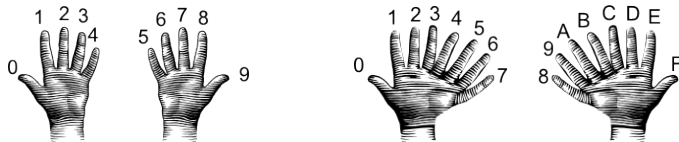


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?

# Recap: Converting between bases

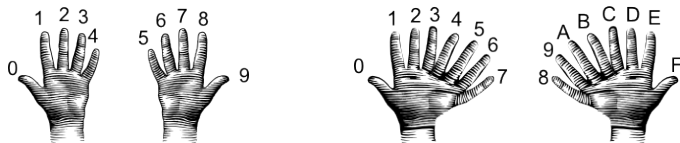


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.

# Recap: Converting between bases

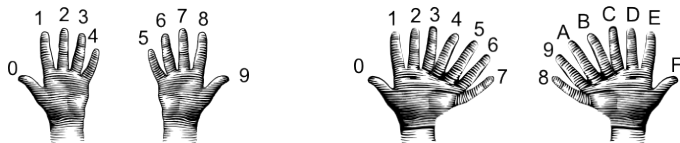


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C.

# Recap: Converting between bases

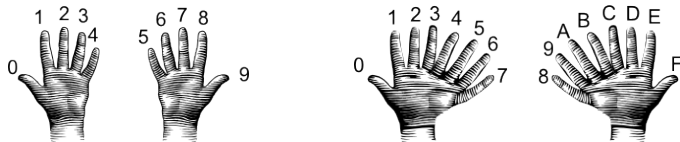


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C. 8 in hex digits is 8.

# Recap: Converting between bases

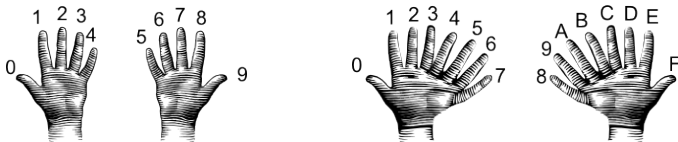


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C. 8 in hex digits is 8.  
Answer is C8.

# Recap: Converting between bases

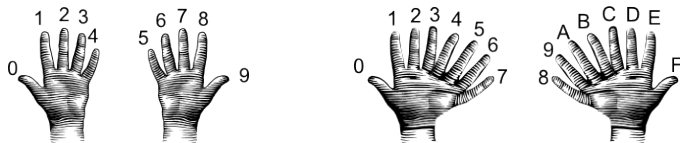


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C. 8 in hex digits is 8.  
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?

# Recap: Converting between bases

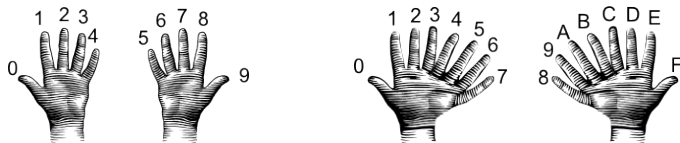


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C. 8 in hex digits is 8.  
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?  
31/16 is 1 remainder 15.

# Recap: Converting between bases



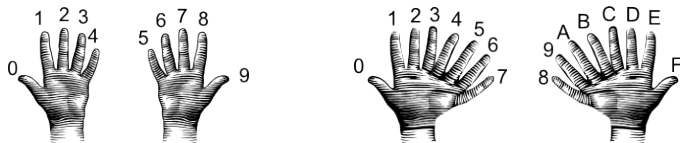
(from i-programmer.info)

## ● From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C. 8 in hex digits is 8.  
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?  
31/16 is 1 remainder 15.  
1 in hex digits is 1.



# Recap: Converting between bases

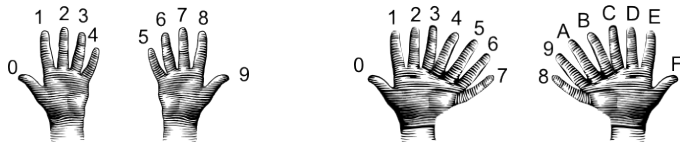


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C. 8 in hex digits is 8.  
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?  
31/16 is 1 remainder 15.  
1 in hex digits is 1. 15 in hex digits is F.

# Recap: Converting between bases

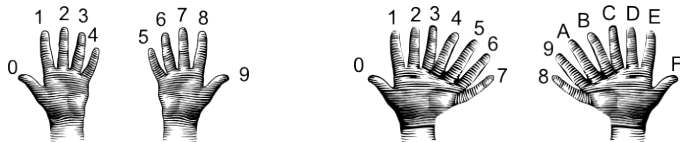


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?  
200/16 is 12 remainder 8.  
12 in hex digits is C. 8 in hex digits is 8.  
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?  
31/16 is 1 remainder 15.  
1 in hex digits is 1. 15 in hex digits is F.  
Answer is 1F.

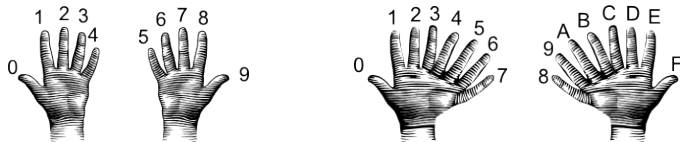
# Recap: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:
  - ▶ Convert first digit to decimal and multiple by 16.

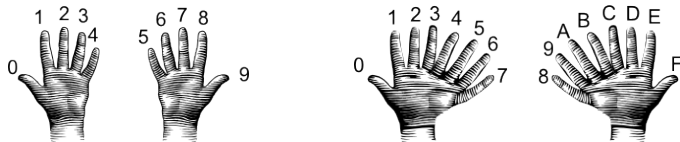
# Recap: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:
  - ▶ Convert first digit to decimal and multiple by 16.
  - ▶ Convert second digit to decimal and add to total.

# Recap: Converting Between Bases

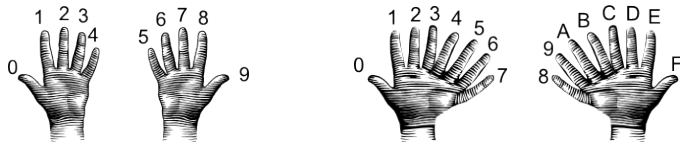


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?

# Recap: Converting Between Bases

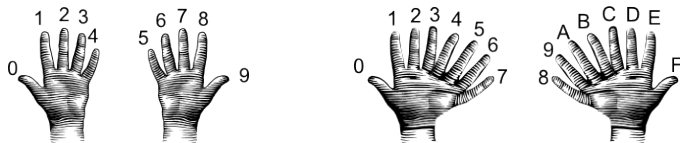


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?  
2 in decimal is 2.

# Recap: Converting Between Bases

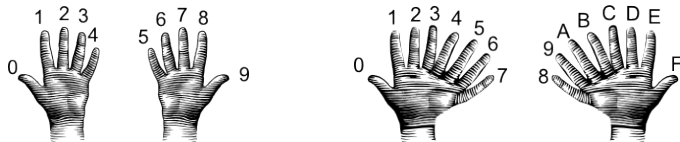


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.

# Recap: Converting Between Bases



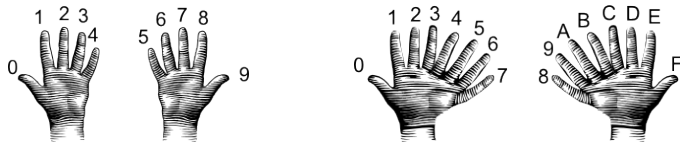
(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
  - 2 in decimal is 2.  $2 \times 16$  is 32.
  - A in decimal digits is 10.



# Recap: Converting Between Bases

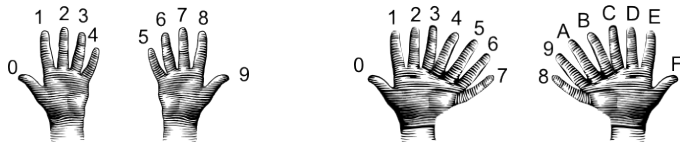


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
  - 2 in decimal is 2.  $2 \times 16$  is 32.
  - A in decimal digits is 10.
  - $32 + 10$  is 42.

# Recap: Converting Between Bases

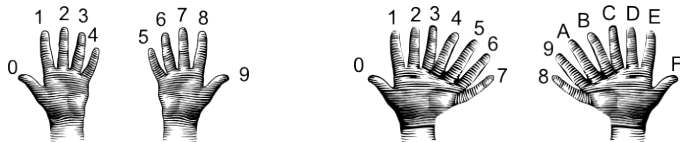


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
  - 2 in decimal is 2.  $2 \times 16$  is 32.
  - A in decimal digits is 10.
  - $32 + 10$  is 42.
  - Answer is 42.
- ▶ Example: what is 99 as a decimal number?

# Recap: Converting Between Bases

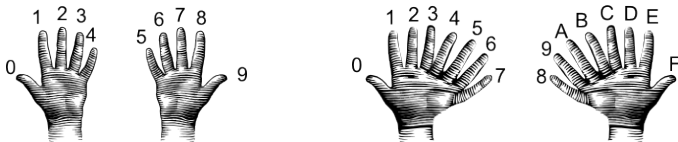


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.  
Answer is 42.
- ▶ Example: what is 99 as a decimal number?  
9 in decimal is 9.

# Recap: Converting Between Bases

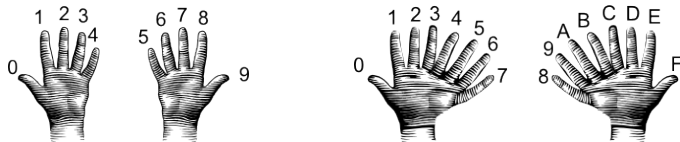


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
  - 2 in decimal is 2.  $2 \times 16$  is 32.
  - A in decimal digits is 10.
  - $32 + 10$  is 42.
  - Answer is 42.
- ▶ Example: what is 99 as a decimal number?
  - 9 in decimal is 9.  $9 \times 16$  is 144.

# Recap: Converting Between Bases

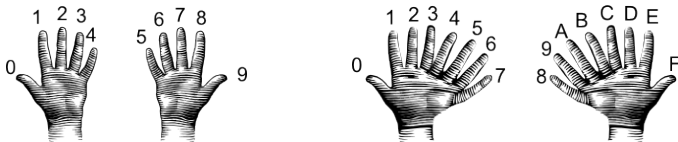


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
  - 2 in decimal is 2.  $2 \times 16$  is 32.
  - A in decimal digits is 10.
  - $32 + 10$  is 42.
  - Answer is 42.
- ▶ Example: what is 99 as a decimal number?
  - 9 in decimal is 9.  $9 \times 16$  is 144.
  - 9 in decimal digits is 9

# Recap: Converting Between Bases

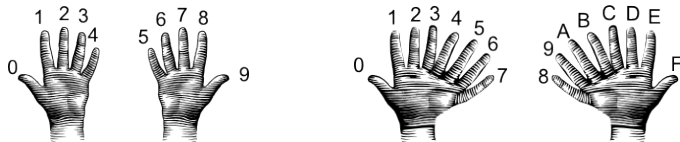


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
  - 2 in decimal is 2.  $2 \times 16$  is 32.
  - A in decimal digits is 10.
  - $32 + 10$  is 42.
  - Answer is 42.
- ▶ Example: what is 99 as a decimal number?
  - 9 in decimal is 9.  $9 \times 16$  is 144.
  - 9 in decimal digits is 9
  - $144 + 9$  is 153.

# Recap: Converting Between Bases

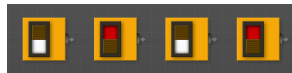
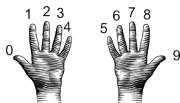


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.  
Answer is 42.
- ▶ Example: what is 99 as a decimal number?  
9 in decimal is 9.  $9 \times 16$  is 144.  
9 in decimal digits is 9  
 $144 + 9$  is 153.  
Answer is 153.

# Recap: Converting Between Bases

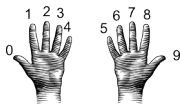


- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.



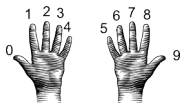
# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.

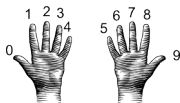
# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.

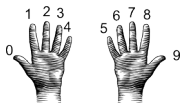
# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.

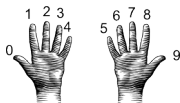
# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.

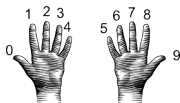
# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.

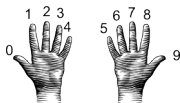
# Recap: Converting Between Bases



## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.

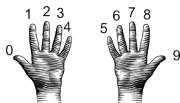
# Recap: Converting Between Bases



## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.

# Recap: Converting Between Bases

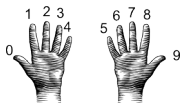


## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?



# Recap: Converting Between Bases

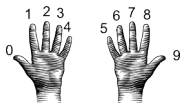


## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2.

# Recap: Converting Between Bases

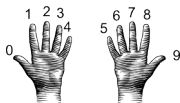


## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1:

# Recap: Converting Between Bases



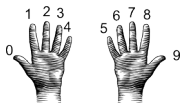
- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2.

# Recap: Converting Between Bases



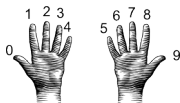
## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0:

# Recap: Converting Between Bases



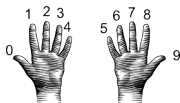
## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

# Recap: Converting Between Bases



## ● From decimal to binary:

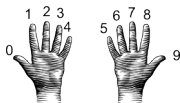
- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2.

# Recap: Converting Between Bases



## ● From decimal to binary:

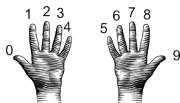
- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0:

# Recap: Converting Between Bases



## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

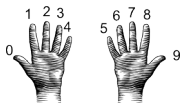
130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...



# Recap: Converting Between Bases



## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

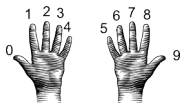
130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2.

# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

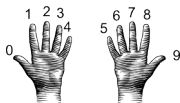
130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2. Next digit is 0:

# Recap: Converting Between Bases

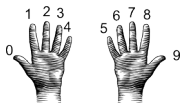


## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...

# Recap: Converting Between Bases

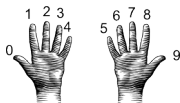


## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...  
2/8 is 0 rem 2.

# Recap: Converting Between Bases

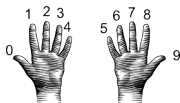


- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...  
2/8 is 0 rem 2. Next digit is 0:

# Recap: Converting Between Bases

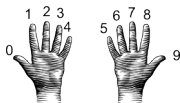


- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...  
2/8 is 0 rem 2. Next digit is 0: 10000...

# Recap: Converting Between Bases

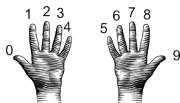


- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...  
2/8 is 0 rem 2. Next digit is 0: 10000...  
2/4 is 0 remainder 2.

# Recap: Converting Between Bases



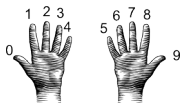
- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...  
2/8 is 0 rem 2. Next digit is 0: 10000...  
2/4 is 0 remainder 2. Next digit is 0:



# Recap: Converting Between Bases

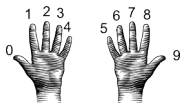


- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...  
2/8 is 0 rem 2. Next digit is 0: 10000...  
2/4 is 0 remainder 2. Next digit is 0: 100000...

# Recap: Converting Between Bases

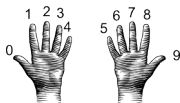


- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

```
130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0:    10...
2/32 is 0 rem 2. Next digit is 0:    100...
2/16 is 0 rem 2. Next digit is 0:    1000...
2/8 is 0 rem 2. Next digit is 0:     10000...
2/4 is 0 remainder 2. Next digit is 0: 100000...
2/2 is 1 rem 0.
```

# Recap: Converting Between Bases

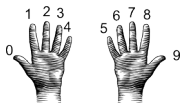


## ● From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2. Next digit is 0: 10...  
2/32 is 0 rem 2. Next digit is 0: 100...  
2/16 is 0 rem 2. Next digit is 0: 1000...  
2/8 is 0 rem 2. Next digit is 0: 10000...  
2/4 is 0 remainder 2. Next digit is 0: 100000...  
2/2 is 1 rem 0. Next digit is 1:

# Recap: Converting Between Bases

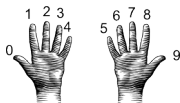


- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

```
130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0:    10...
2/32 is 0 rem 2. Next digit is 0:    100...
2/16 is 0 rem 2. Next digit is 0:    1000...
2/8 is 0 rem 2. Next digit is 0:    10000...
2/4 is 0 remainder 2. Next digit is 0: 100000...
2/2 is 1 rem 0. Next digit is 1:    1000001...
```

# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2. Next digit is 0: 1000...

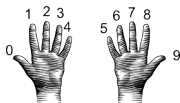
2/8 is 0 rem 2. Next digit is 0: 10000...

2/4 is 0 remainder 2. Next digit is 0: 100000...

2/2 is 1 rem 0. Next digit is 1: 1000001...

Adding the last remainder: 10000010

# Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ( $= 2^7$ ). Quotient is the first digit.
- ▶ Divide remainder by 64 ( $= 2^6$ ). Quotient is the next digit.
- ▶ Divide remainder by 32 ( $= 2^5$ ). Quotient is the next digit.
- ▶ Divide remainder by 16 ( $= 2^4$ ). Quotient is the next digit.
- ▶ Divide remainder by 8 ( $= 2^3$ ). Quotient is the next digit.
- ▶ Divide remainder by 4 ( $= 2^2$ ). Quotient is the next digit.
- ▶ Divide remainder by 2 ( $= 2^1$ ). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2. Next digit is 0: 1000...

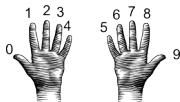
2/8 is 0 rem 2. Next digit is 0: 10000...

2/4 is 0 remainder 2. Next digit is 0: 100000...

2/2 is 1 rem 0. Next digit is 1: 1000001...

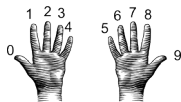
Adding the last remainder: 10000010

# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

# Recap: Converting Between Bases

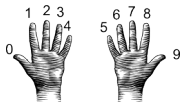


- Example: what is 99 in binary notation?

99/128 is 0 rem 99.

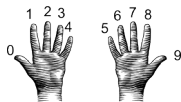


# Recap: Converting Between Bases



- Example: what is 99 in binary notation?  
99/128 is 0 rem 99. First digit is 0:

# Recap: Converting Between Bases

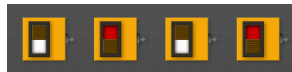
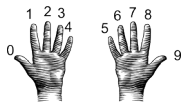


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35.

# Recap: Converting Between Bases

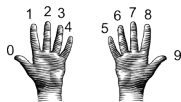


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1:

# Recap: Converting Between Bases

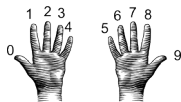


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

# Recap: Converting Between Bases



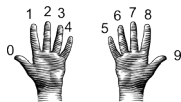
- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3.

# Recap: Converting Between Bases



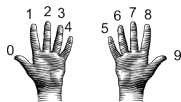
- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1:

# Recap: Converting Between Bases



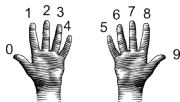
- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

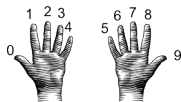
99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3.



# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

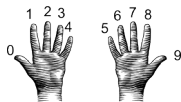
99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3. Next digit is 0:

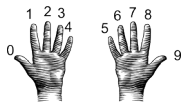
# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...  
99/64 is 1 rem 35. Next digit is 1: 01...  
35/32 is 1 rem 3. Next digit is 1: 011...  
3/16 is 0 rem 3. Next digit is 0: 0110...

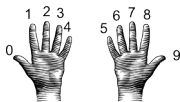
# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...  
99/64 is 1 rem 35. Next digit is 1: 01...  
35/32 is 1 rem 3. Next digit is 1: 011...  
3/16 is 0 rem 3. Next digit is 0: 0110...  
3/8 is 0 rem 3.

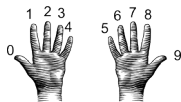
# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...  
99/64 is 1 rem 35. Next digit is 1: 01...  
35/32 is 1 rem 3. Next digit is 1: 011...  
3/16 is 0 rem 3. Next digit is 0: 0110...  
3/8 is 0 rem 3. Next digit is 0:

# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

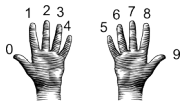
99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

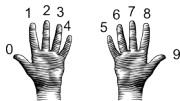
35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

3/4 is 0 remainder 3.

# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

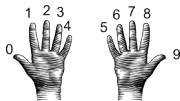
35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

3/4 is 0 remainder 3. Next digit is 0:

# Recap: Converting Between Bases

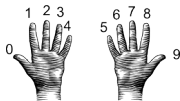


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...  
99/64 is 1 rem 35. Next digit is 1: 01...  
35/32 is 1 rem 3. Next digit is 1: 011...  
3/16 is 0 rem 3. Next digit is 0: 0110...  
3/8 is 0 rem 3. Next digit is 0: 01100...  
3/4 is 0 remainder 3. Next digit is 0: 011000...



# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

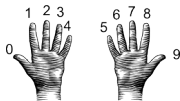
3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

3/4 is 0 remainder 3. Next digit is 0: 011000...

3/2 is 1 rem 1.

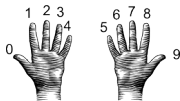
# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...  
99/64 is 1 rem 35. Next digit is 1: 01...  
35/32 is 1 rem 3. Next digit is 1: 011...  
3/16 is 0 rem 3. Next digit is 0: 0110...  
3/8 is 0 rem 3. Next digit is 0: 01100...  
3/4 is 0 remainder 3. Next digit is 0: 011000...  
3/2 is 1 rem 1. Next digit is 1:

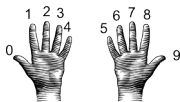
# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0:	0...
99/64 is 1 rem 35. Next digit is 1:	01...
35/32 is 1 rem 3. Next digit is 1:	011...
3/16 is 0 rem 3. Next digit is 0:	0110...
3/8 is 0 rem 3. Next digit is 0:	01100...
3/4 is 0 remainder 3. Next digit is 0:	011000...
3/2 is 1 rem 1. Next digit is 1:	0110001...

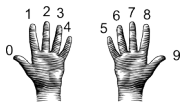
# Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0:	0...
99/64 is 1 rem 35. Next digit is 1:	01...
35/32 is 1 rem 3. Next digit is 1:	011...
3/16 is 0 rem 3. Next digit is 0:	0110...
3/8 is 0 rem 3. Next digit is 0:	01100...
3/4 is 0 remainder 3. Next digit is 0:	011000...
3/2 is 1 rem 1. Next digit is 1:	0110001...
Adding the last remainder:	01100011

# Recap: Converting Between Bases

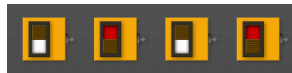
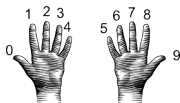


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0:	0...
99/64 is 1 rem 35. Next digit is 1:	01...
35/32 is 1 rem 3. Next digit is 1:	011...
3/16 is 0 rem 3. Next digit is 0:	0110...
3/8 is 0 rem 3. Next digit is 0:	01100...
3/4 is 0 remainder 3. Next digit is 0:	011000...
3/2 is 1 rem 1. Next digit is 1:	0110001...
Adding the last remainder:	01100011

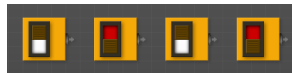
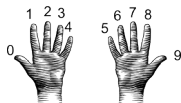
Answer is 1100011.

# Recap: Converting Between Bases



- From binary to decimal:
  - ▶ Set sum = last digit.

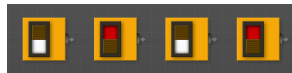
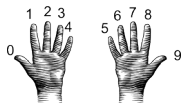
# Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.

# Recap: Converting Between Bases

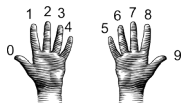


- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.



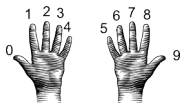
# Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.

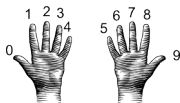
# Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.

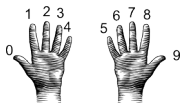
# Recap: Converting Between Bases



## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.

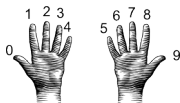
# Recap: Converting Between Bases



## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.

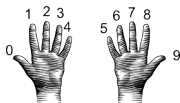
# Recap: Converting Between Bases



## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.

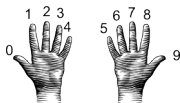
# Recap: Converting Between Bases



## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.

# Recap: Converting Between Bases

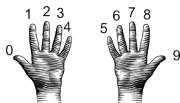


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:

# Recap: Converting Between Bases



## ● From binary to decimal:

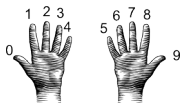
- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1

$0 \times 2 = 0$ . Add 0 to sum:



# Recap: Converting Between Bases



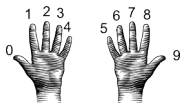
## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1

$0 \times 2 = 0$ . Add 0 to sum: 1

# Recap: Converting Between Bases

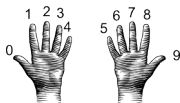


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1  
 $0 \times 2 = 0$ . Add 0 to sum: 1  
 $1 \times 4 = 4$ . Add 4 to sum:

# Recap: Converting Between Bases

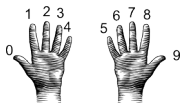


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5

# Recap: Converting Between Bases

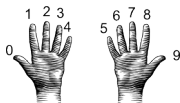


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5
$1 \times 8 = 8$ . Add 8 to sum:	

# Recap: Converting Between Bases

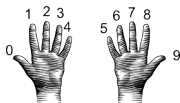


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5
$1 \times 8 = 8$ . Add 8 to sum:	13

# Recap: Converting Between Bases

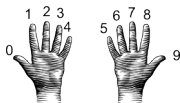


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1  
 $0 \times 2 = 0$ . Add 0 to sum: 1  
 $1 \times 4 = 4$ . Add 4 to sum: 5  
 $1 \times 8 = 8$ . Add 8 to sum: 13  
 $1 \times 16 = 16$ . Add 16 to sum:

# Recap: Converting Between Bases

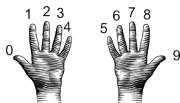


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5
$1 \times 8 = 8$ . Add 8 to sum:	13
$1 \times 16 = 16$ . Add 16 to sum:	29

# Recap: Converting Between Bases



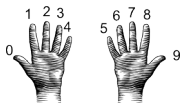
## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5
$1 \times 8 = 8$ . Add 8 to sum:	13
$1 \times 16 = 16$ . Add 16 to sum:	29
$1 \times 32 = 32$ . Add 32 to sum:	



# Recap: Converting Between Bases

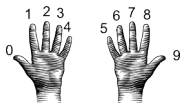


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5
$1 \times 8 = 8$ . Add 8 to sum:	13
$1 \times 16 = 16$ . Add 16 to sum:	29
$1 \times 32 = 32$ . Add 32 to sum:	61

# Recap: Converting Between Bases

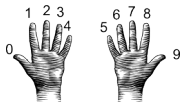


## ● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by  $2 = 2^1$ . Add to sum.
- ▶ Multiply next digit by  $4 = 2^2$ . Add to sum.
- ▶ Multiply next digit by  $8 = 2^3$ . Add to sum.
- ▶ Multiply next digit by  $16 = 2^4$ . Add to sum.
- ▶ Multiply next digit by  $32 = 2^5$ . Add to sum.
- ▶ Multiply next digit by  $64 = 2^6$ . Add to sum.
- ▶ Multiply next digit by  $128 = 2^7$ . Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5
$1 \times 8 = 8$ . Add 8 to sum:	13
$1 \times 16 = 16$ . Add 16 to sum:	29
$1 \times 32 = 32$ . Add 32 to sum:	61

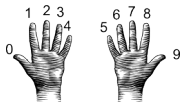
# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:

# Recap: Converting Between Bases

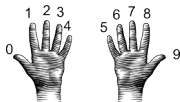


- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 * 2 = 0$ . Add 0 to sum:

# Recap: Converting Between Bases

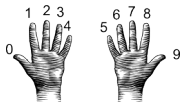


- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 * 2 = 0$ . Add 0 to sum: 0

# Recap: Converting Between Bases



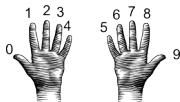
- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . Add 4 to sum:

# Recap: Converting Between Bases



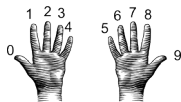
- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . Add 4 to sum: 4

# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

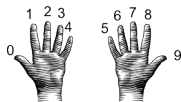
$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . Add 4 to sum: 4

$0 \times 8 = 0$ . Add 0 to sum:



# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

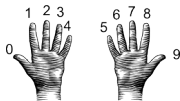
Sum starts with: 0

$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . Add 4 to sum: 4

$0 \times 8 = 0$ . Add 0 to sum: 4

# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

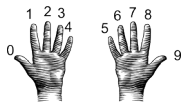
$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . Add 4 to sum: 4

$0 \times 8 = 0$ . Add 0 to sum: 4

$0 \times 16 = 0$ . Add 0 to sum:

# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

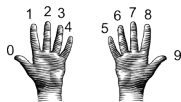
$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . Add 4 to sum: 4

$0 \times 8 = 0$ . Add 0 to sum: 4

$0 \times 16 = 0$ . Add 0 to sum: 4

# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 \times 2 = 0$ . Add 0 to sum: 0

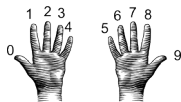
$1 \times 4 = 4$ . Add 4 to sum: 4

$0 \times 8 = 0$ . Add 0 to sum: 4

$0 \times 16 = 0$ . Add 0 to sum: 4

$1 \times 32 = 32$ . Add 32 to sum:

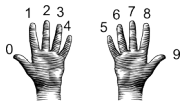
# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$ . Add 0 to sum:	0
$1 \times 4 = 4$ . Add 4 to sum:	4
$0 \times 8 = 0$ . Add 0 to sum:	4
$0 \times 16 = 0$ . Add 0 to sum:	4
$1 \times 32 = 32$ . Add 32 to sum:	36

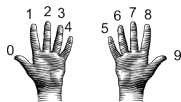
# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$ . Add 0 to sum:	0
$1 \times 4 = 4$ . Add 4 to sum:	4
$0 \times 8 = 0$ . Add 0 to sum:	4
$0 \times 16 = 0$ . Add 0 to sum:	4
$1 \times 32 = 32$ . Add 32 to sum:	36
$0 \times 64 = 0$ . Add 0 to sum:	

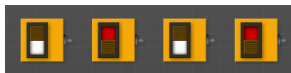
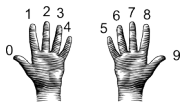
# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$ . Add 0 to sum:	0
$1 \times 4 = 4$ . Add 4 to sum:	4
$0 \times 8 = 0$ . Add 0 to sum:	4
$0 \times 16 = 0$ . Add 0 to sum:	4
$1 \times 32 = 32$ . Add 32 to sum:	36
$0 \times 64 = 0$ . Add 0 to sum:	36

# Recap: Converting Between Bases

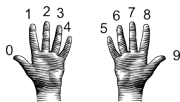


- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$ . Add 0 to sum:	0
$1 \times 4 = 4$ . Add 4 to sum:	4
$0 \times 8 = 0$ . Add 0 to sum:	4
$0 \times 16 = 0$ . Add 0 to sum:	4
$1 \times 32 = 32$ . Add 32 to sum:	36
$0 \times 64 = 0$ . Add 0 to sum:	36
$1 \times 128 = 128$ . Add 128 to sum:	



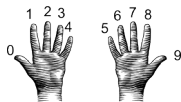
# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$ . Add 0 to sum:	0
$1 \times 4 = 4$ . Add 4 to sum:	4
$0 \times 8 = 0$ . Add 0 to sum:	4
$0 \times 16 = 0$ . Add 0 to sum:	4
$1 \times 32 = 32$ . Add 32 to sum:	36
$0 \times 64 = 0$ . Add 0 to sum:	36
$1 \times 128 = 128$ . Add 128 to sum:	164

# Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$ . Add 0 to sum:	0
$1 \times 4 = 4$ . Add 4 to sum:	4
$0 \times 8 = 0$ . Add 0 to sum:	4
$0 \times 16 = 0$ . Add 0 to sum:	4
$1 \times 32 = 32$ . Add 32 to sum:	36
$0 \times 64 = 0$ . Add 0 to sum:	36
$1 \times 128 = 128$ . Add 128 to sum:	164

The answer is 164.

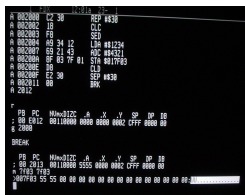
# Today's Topics



- Data Representation
- **Machine Language: Jumps & Loops**
- Recap of Python & Circuits
- Design Patterns: Sorting

# Recap: Machine Language

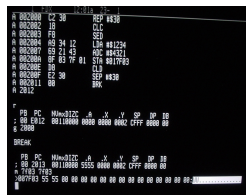
- We will be writing programs in a simplified machine language, WeMIPS.



The screenshot shows a MIPS assembly program in a debugger. The program consists of several instructions: `REP #30`, `CLC`, `SED`, `LDR #1234`, `ROR #4321`, `STW #017F83`, `CLD`, `SEP #30`, and `BRK`. The debugger shows the program counter (PC) at 002000 and the instruction register (IR) at 002000. The program is running on a MIPS architecture. The debugger interface includes a list of instructions, a register window, and a memory window.

(wiki)

# Recap: Machine Language



```
002000 c2 30      REP #030
002002 10          CLC
002003 f0          SED
002004 40 34 12    L3H #01234
002007 60 21 43    R0C #04321
002008 0f 03 7f 01 STN #017f03
00200c 00          CLJ
00200f e2 30      SEP #030
002011 00          BRX
002012

P0 PC Mem32C A X Y SP BP BB
: 00 2012 00110000 0000 0000 0002 c7ff 0000 00
$ 2000

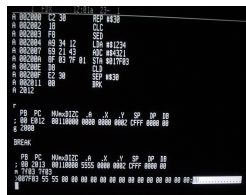
BREAK

P0 PC Mem32C A X Y V SP BP BB
: 00 2013 00110000 5555 0000 0002 c7ff 0000 00
n 1103 7f03
007f03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.

# Recap: Machine Language



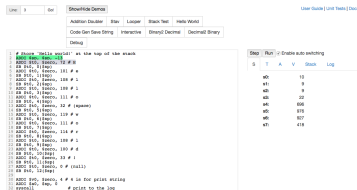
```
002000 c2 30 REP #430  
002002 10 CLC  
002003 F0 SED  
002004 40 34 12 LSH #1224  
002007 60 21 43 RLC #4321  
00200A 0F 03 7F 01 STA #017F03  
00200C 00 CLJ  
00200F E2 30 SEP #430  
002011 00 BRX  
002012  
  
PC PC Mem32C A X Y SP BP BB  
: 00 2012 00110000 0000 0000 0002 C7FF 0000 00  
0 2000  
BREAK  
  
PC PC Mem32C A X Y SP BP BB  
: 00 2013 00110000 5555 0000 0002 C7FF 0000 00  
0 1103 7403  
007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.
- Due to its small set of commands, processors can be designed to run those commands very efficiently.



# Recap: MIPS Commands



The screenshot shows a MIPS simulator interface. On the left, there's a text area with assembly code. On the right, there's a 'Registers' window showing the state of various registers.

```
1 # Store "hello world" at the top of the stack
2 ADDI $0, $zero, 128
3 SW $0, 128($0)
4 ADDI $0, $zero, 191 # n
5 SW $0, 191($0)
6 ADDI $0, $zero, 100 # i
7 SW $0, 100($0)
8 ADDI $0, $zero, 131 # n
9 SW $0, 131($0)
10 ADDI $0, $zero, 12 # (space)
11 SW $0, 12($0)
12 ADDI $0, $zero, 119 # n
13 SW $0, 119($0)
14 ADDI $0, $zero, 131 # n
15 SW $0, 131($0)
16 ADDI $0, $zero, 114 # n
17 SW $0, 114($0)
18 ADDI $0, $zero, 100 # i
19 SW $0, 100($0)
20 ADDI $0, $zero, 100 # n
21 SW $0, 100($0)
22 ADDI $0, $zero, 33 # i
23 SW $0, 33($0)
24 ADDI $0, $zero, 0 # (null)
25 SW $0, 0($0)
26 ADDI $0, $zero, 4 # n is for print ending
27 ADDI $0, $zero, 0 # print to the log
28 syscall
```

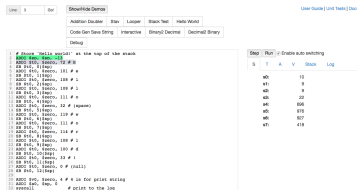
The 'Registers' window on the right shows the following values:

Register	Value
\$0	10
\$1	9
\$2	9
\$3	22
\$4	856
\$5	817
\$6	418

- **Registers:** locations for storing information that can be quickly accessed.



# Recap: MIPS Commands



```
1 # Store "hello world" in the top of the stack
2 ADDI $s0, $zero, 12 # 12
3 SB $s0, 0($zero)
4 ADDI $s1, $zero, 191 # 191
5 SB $s1, 12($zero)
6 ADDI $s0, $zero, 100 # 100
7 SB $s0, 16($zero)
8 ADDI $s0, $zero, 111 # 111
9 SB $s0, 20($zero)
10 ADDI $s0, $zero, 32 # (space)
11 SB $s0, 24($zero)
12 ADDI $s0, $zero, 111 # 111
13 SB $s0, 28($zero)
14 ADDI $s0, $zero, 114 # 114
15 SB $s0, 32($zero)
16 ADDI $s0, $zero, 100 # 100
17 SB $s0, 36($zero)
18 ADDI $s0, $zero, 100 # 100
19 SB $s0, 40($zero)
20 ADDI $s0, $zero, 0 # (null)
21 SB $s0, 44($zero)
22 ADDI $s0, $zero, 4 # 4 is for print ending
23 ADDI $s0, $zero, 0 # print to the log
24 syscall
```

\$s	\$t	\$a	\$v	Stack	Log
\$s0				10	
\$s1				9	
\$s2				8	
\$s3				22	
\$s4				856	
\$s5				876	
\$s6				897	
\$s7				418	

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...

## Recap: MIPS Commands

[illegible]

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:

## Recap: MIPS Commands

[illegible]

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:  
add \$s1, \$s2, \$s3

## Recap: MIPS Commands

[illegible]

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:  
add \$s1, \$s2, \$s3      (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.

## Recap: MIPS Commands

```

1  # Store "Hello world" at the top of the stack
2  ADDI $sp, $zero, 20 #R
3  #R $sp, 10($sp)
4  ADDI $sp, $zero, 181 #R
5  #R $sp, 10($sp)
6  ADDI $sp, $zero, 188 #R
7  #R $sp, 10($sp)
8  ADDI $sp, $zero, 189 #R
9  #R $sp, 10($sp)
10 ADDI $sp, $zero, 191 #R
11 #R $sp, 10($sp)
12 ADDI $sp, $zero, 192 #R
13 #R $sp, 10($sp)
14 ADDI $sp, $zero, 193 #R
15 #R $sp, 10($sp)
16 ADDI $sp, $zero, 194 #R
17 #R $sp, 10($sp)
18 ADDI $sp, $zero, 195 #R
19 #R $sp, 10($sp)
20 ADDI $sp, $zero, 196 #R
21 #R $sp, 10($sp)
22 ADDI $sp, $zero, 197 #R
23 #R $sp, 10($sp)
24 ADDI $sp, $zero, 198 #R
25 #R $sp, 10($sp)
26 ADDI $sp, $zero, 199 #R
27 #R $sp, 10($sp)
28 ADDI $sp, $zero, 200 #R
29 #R $sp, 10($sp)
30 ADDI $sp, $zero, 201 #R
31 #R $sp, 10($sp)
32 ADDI $sp, $zero, 202 #R
33 #R $sp, 10($sp)
34 ADDI $sp, $zero, 203 #R
35 #R $sp, 10($sp)
36 ADDI $sp, $zero, 204 #R
37 #R $sp, 10($sp)
38 ADDI $sp, $zero, 205 #R
39 #R $sp, 10($sp)
40 ADDI $sp, $zero, 206 #R
41 #R $sp, 10($sp)
42 ADDI $sp, $zero, 207 #R
43 #R $sp, 10($sp)
44 ADDI $sp, $zero, 208 #R
45 #R $sp, 10($sp)
46 ADDI $sp, $zero, 209 #R
47 #R $sp, 10($sp)
48 ADDI $sp, $zero, 210 #R
49 #R $sp, 10($sp)
50 ADDI $sp, $zero, 211 #R
51 #R $sp, 10($sp)
52 ADDI $sp, $zero, 212 #R
53 #R $sp, 10($sp)
54 ADDI $sp, $zero, 213 #R
55 #R $sp, 10($sp)
56 ADDI $sp, $zero, 214 #R
57 #R $sp, 10($sp)
58 ADDI $sp, $zero, 215 #R
59 #R $sp, 10($sp)
60 ADDI $sp, $zero, 216 #R
61 #R $sp, 10($sp)
62 ADDI $sp, $zero, 217 #R
63 #R $sp, 10($sp)
64 ADDI $sp, $zero, 218 #R
65 #R $sp, 10($sp)
66 ADDI $sp, $zero, 219 #R
67 #R $sp, 10($sp)
68 ADDI $sp, $zero, 220 #R
69 #R $sp, 10($sp)
70 ADDI $sp, $zero, 221 #R
71 #R $sp, 10($sp)
72 ADDI $sp, $zero, 222 #R
73 #R $sp, 10($sp)
74 ADDI $sp, $zero, 223 #R
75 #R $sp, 10($sp)
76 ADDI $sp, $zero, 224 #R
77 #R $sp, 10($sp)
78 ADDI $sp, $zero, 225 #R
79 #R $sp, 10($sp)
80 ADDI $sp, $zero, 226 #R
81 #R $sp, 10($sp)
82 ADDI $sp, $zero, 227 #R
83 #R $sp, 10($sp)
84 ADDI $sp, $zero, 228 #R
85 #R $sp, 10($sp)
86 ADDI $sp, $zero, 229 #R
87 #R $sp, 10($sp)
88 ADDI $sp, $zero, 230 #R
89 #R $sp, 10($sp)
90 ADDI $sp, $zero, 231 #R
91 #R $sp, 10($sp)
92 ADDI $sp, $zero, 232 #R
93 #R $sp, 10($sp)
94 ADDI $sp, $zero, 233 #R
95 #R $sp, 10($sp)
96 ADDI $sp, $zero, 234 #R
97 #R $sp, 10($sp)
98 ADDI $sp, $zero, 235 #R
99 #R $sp, 10($sp)
100 ADDI $sp, $zero, 236 #R
101 #R $sp, 10($sp)
102 ADDI $sp, $zero, 237 #R
103 #R $sp, 10($sp)
104 ADDI $sp, $zero, 238 #R
105 #R $sp, 10($sp)
106 ADDI $sp, $zero, 239 #R
107 #R $sp, 10($sp)
108 ADDI $sp, $zero, 240 #R
109 #R $sp, 10($sp)
110 ADDI $sp, $zero, 241 #R
111 #R $sp, 10($sp)
112 ADDI $sp, $zero, 242 #R
113 #R $sp, 10($sp)
114 ADDI $sp, $zero, 243 #R
115 #R $sp, 10($sp)
116 ADDI $sp, $zero, 244 #R
117 #R $sp, 10($sp)
118 ADDI $sp, $zero, 245 #R
119 #R $sp, 10($sp)
120 ADDI $sp, $zero, 246 #R
121 #R $sp, 10($sp)
122 ADDI $sp, $zero, 247 #R
123 #R $sp, 10($sp)
124 ADDI $sp, $zero, 248 #R
125 #R $sp, 10($sp)
126 ADDI $sp, $zero, 249 #R
127 #R $sp, 10($sp)
128 ADDI $sp, $zero, 250 #R
129 #R $sp, 10($sp)
130 ADDI $sp, $zero, 251 #R
131 #R $sp, 10($sp)
132 ADDI $sp, $zero, 252 #R
133 #R $sp, 10($sp)
134 ADDI $sp, $zero, 253 #R
135 #R $sp, 10($sp)
136 ADDI $sp, $zero, 254 #R
137 #R $sp, 10($sp)
138 ADDI $sp, $zero, 255 #R
139 #R $sp, 10($sp)
140 ADDI $sp, $zero, 256 #R
141 #R $sp, 10($sp)
142 ADDI $sp, $zero, 257 #R
143 #R $sp, 10($sp)
144 ADDI $sp, $zero, 258 #R
145 #R $sp, 10($sp)
146 ADDI $sp, $zero, 259 #R
147 #R $sp, 10($sp)
148 ADDI $sp, $zero, 260 #R
149 #R $sp, 10($sp)
150 ADDI $sp, $zero, 261 #R
151 #R $sp, 10($sp)
152 ADDI $sp, $zero, 262 #R
153 #R $sp, 10($sp)
154 ADDI $sp, $zero, 263 #R
155 #R $sp, 10($sp)
156 ADDI $sp, $zero, 264 #R
157 #R $sp, 10($sp)
158 ADDI $sp, $zero, 265 #R
159 #R $sp, 10($sp)
160 ADDI $sp, $zero, 266 #R
161 #R $sp, 10($sp)
162 ADDI $sp, $zero, 267 #R
163 #R $sp, 10($sp)
164 ADDI $sp, $zero, 268 #R
165 #R $sp, 10($sp)
166 ADDI $sp, $zero, 269 #R
167 #R $sp, 10($sp)
168 ADDI $sp, $zero, 270 #R
169 #R $sp, 10($sp)
170 ADDI $sp, $zero, 271 #R
171 #R $sp, 10($sp)
172 ADDI $sp, $zero, 272 #R
173 #R $sp, 10($sp)
174 ADDI $sp, $zero, 273 #R
175 #R $sp, 10($sp)
176 ADDI $sp, $zero, 274 #R
177 #R $sp, 10($sp)
178 ADDI $sp, $zero, 275 #R
179 #R $sp, 10($sp)
180 ADDI $sp, $zero, 276 #R
181 #R $sp, 10($sp)
182 ADDI $sp, $zero, 277 #R
183 #R $sp, 10($sp)
184 ADDI $sp, $zero, 278 #R
185 #R $sp, 10($sp)
186 ADDI $sp, $zero, 279 #R
187 #R $sp, 10($sp)
188 ADDI $sp, $zero, 280 #R
189 #R $sp, 10($sp)
190 ADDI $sp, $zero, 281 #R
191 #R $sp, 10($sp)
192 ADDI $sp, $zero, 282 #R
193 #R $sp, 10($sp)
194 ADDI $sp, $zero, 283 #R
195 #R $sp, 10($sp)
196 ADDI $sp, $zero, 284 #R
197 #R $sp, 10($sp)
198 ADDI $sp, $zero, 285 #R
199 #R $sp, 10($sp)
200 ADDI $sp, $zero, 286 #R
201 #R $sp, 10($sp)
202 ADDI $sp, $zero, 287 #R
203 #R $sp, 10($sp)
204 ADDI $sp, $zero, 288 #R
205 #R $sp, 10($sp)
206 ADDI $sp, $zero, 289 #R
207 #R $sp, 10($sp)
208 ADDI $sp, $zero, 290 #R
209 #R $sp, 10($sp)
210 ADDI $sp, $zero, 291 #R
211 #R $sp, 10($sp)
212 ADDI $sp, $zero, 292 #R
213 #R $sp, 10($sp)
214 ADDI $sp, $zero, 293 #R
215 #R $sp, 10($sp)
216 ADDI $sp, $zero, 294 #R
217 #R $sp, 10($sp)
218 ADDI $sp, $zero, 295 #R
219 #R $sp, 10($sp)
220 ADDI $sp, $zero, 296 #R
221 #R $sp, 10($sp)
222 ADDI $sp, $zero, 297 #R
223 #R $sp, 10($sp)
224 ADDI $sp, $zero, 298 #R
225 #R $sp, 10($sp)
226 ADDI $sp, $zero, 299 #R
227 #R $sp, 10($sp)
228 ADDI $sp, $zero, 300 #R
229 #R $sp, 10($sp)
230 ADDI $sp, $zero, 301 #R
231 #R $sp, 10($sp)
232 ADDI $sp, $zero, 302 #R
233 #R $sp, 10($sp)
234 ADDI $sp, $zero, 303 #R
235 #R $sp, 10($sp)
236 ADDI $sp, $zero, 304 #R
237 #R $sp, 10($sp)
238 ADDI $sp, $zero, 305 #R
239 #R $sp, 10($sp)
240 ADDI $sp, $zero, 306 #R
241 #R $sp, 10($sp)
242 ADDI $sp, $zero, 307 #R
243 #R $sp, 10($sp)
244 ADDI $sp, $zero, 308 #R
245 #R $sp, 10($sp)
246 ADDI $sp, $zero, 309 #R
247 #R $sp, 10($sp)
248 ADDI $sp, $zero, 310 #R
249 #R $sp, 10($sp)
250 ADDI $sp, $zero, 311 #R
251 #R $sp, 10($sp)
252 ADDI $sp, $zero, 312 #R
253 #R
```

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:  
add \$s1, \$s2, \$s3      (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.  
addi \$s1, \$s2, 100

## Recap: MIPS Commands

[illegible]

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:  
add \$s1, \$s2, \$s3      (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.  
addi \$s1, \$s2, 100      (Basic form: OP rd, rs, imm)
- **J Instructions:** instructions that jump to another memory location.



## Recap: MIPS Commands

[illegible]

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:  
add \$s1, \$s2, \$s3      (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.  
addi \$s1, \$s2, 100      (Basic form: OP rd, rs, imm)
- **J Instructions:** instructions that jump to another memory location.  
j done      (Basic form: OP label)



# Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.



The screenshot shows a debugger window with two panes. The left pane displays assembly instructions, including `movl $0, %eax`, `movl $1, %ecx`, `movl $2, %edx`, `movl $3, %ebx`, `movl $4, %esi`, `movl $5, %edi`, `movl $6, %ebp`, `movl $7, %esp`, `movl $8, %eip`, `movl $9, %ebp`, `movl $10, %esp`, `movl $11, %eip`, `movl $12, %ebp`, `movl $13, %esp`, `movl $14, %eip`, `movl $15, %ebp`, `movl $16, %esp`, `movl $17, %eip`, `movl $18, %ebp`, `movl $19, %esp`, `movl $20, %eip`, `movl $21, %ebp`, `movl $22, %esp`, `movl $23, %eip`, `movl $24, %ebp`, `movl $25, %esp`, `movl $26, %eip`, `movl $27, %ebp`, `movl $28, %esp`, `movl $29, %eip`, `movl $30, %ebp`, `movl $31, %esp`. The right pane shows the state of registers: `EAX: 00000000`, `ECX: 00000001`, `EDX: 00000002`, `EBX: 00000003`, `ESI: 00000004`, `EDI: 00000005`, `EBP: 00000006`, `ESP: 00000007`, `EIP: 00000008`, `EBP: 00000009`, `ESP: 0000000A`, `EIP: 0000000B`, `EBP: 0000000C`, `ESP: 0000000D`, `EIP: 0000000E`, `EBP: 0000000F`, `ESP: 00000010`.

# Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.



# Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.



# Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:



# Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
  - ▶ **Unconditional:** `j Done` will jump to the address with label `Done`.



# Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
  - ▶ **Unconditional:** `j Done` will jump to the address with label `Done`.
  - ▶ **Branch if Equal:** `beq $s0 $s1 DoAgain` will jump to the address with label `DoAgain` if the registers `$s0` and `$s1` contain the same value.



# Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
  - ▶ **Unconditional:** `j Done` will jump to the address with label `Done`.
  - ▶ **Branch if Equal:** `beq $s0 $s1 DoAgain` will jump to the address with label `DoAgain` if the registers `$s0` and `$s1` contain the same value.
  - ▶ See reading for more variations.



# WeMIPS

Line: 3    dis

Show/Hide Demos

Addition Doubler    Stop    Looper    Stack Test    Hello World

Code Gen Save String    Interactive    Binary2 Decimal    Decimal2 Binary

Debug

```
1 # Store "hello world!" at the top of the stack
2 ADDUI $a0, $zero, 32 # 0
3 ROR $t0, $t0
4 ADDUI $t0, $zero, 101 # e
5 DD $t0, 1($t0)
6 ADDUI $t0, $zero, 108 # l
7 DD $t0, 2($t0)
8 ADDUI $t0, $zero, 108 # l
9 DD $t0, 3($t0)
10 ADDUI $t0, $zero, 111 # o
11 DD $t0, 4($t0)
12 ADDUI $t0, $zero, 32 # (space)
13 DD $t0, 5($t0)
14 ADDUI $t0, $zero, 119 # w
15 DD $t0, 6($t0)
16 ADDUI $t0, $zero, 114 # u
17 DD $t0, 7($t0)
18 ADDUI $t0, $zero, 108 # d
19 DD $t0, 8($t0)
20 ADDUI $t0, $zero, 108 # l
21 DD $t0, 9($t0)
22 ADDUI $t0, $zero, 103 # d
23 DD $t0, 10($t0)
24 ADDUI $t0, $zero, 33 # !
25 DD $t0, 11($t0)
26 DD $t0, 12($t0)
27 ADDUI $t0, $zero, 0 # (null)
28 DD $t0, 13($t0)
29
30 ADDUI $v0, $zero, 6 # 4 in for print string
31 ADDUI $a0, $a0, 0
32 syscall # print to the log
```

Step    Run    ☐ Enable auto-switching

S	T	A	V	Stack	Log
				a0:	10
				t0:	9
				a0:	9
				a0:	22
				a0:	695
				a0:	970
				a0:	927
				a0:	418

(Demo with WeMIPS)



# Today's Topics



- Data Representation
- Machine Language: Jumps & Loops
- **Recap of Python & Circuits**
- Design Patterns: Sorting

# Python & Circuits Review: 10 Weeks in 10 Minutes



A whirlwind tour of the semester, so far...

# Week 1: print(), loops, comments, & turtles

# Week 1: print(), loops, comments, & turtles

- Introduced comments & print():

```
#Name:  Thomas Hunter
```

← *These lines are comments*

```
#Date:  September 1, 2017
```

← *(for us, not computer to read)*

```
#This program prints:  Hello, World!
```

← *(this one also)*

```
print("Hello, World!")
```

← *Prints the string "Hello, World!" to the screen*

# Week 1: print(), loops, comments, & turtles

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

```
#Date: September 1, 2017
```

← (for us, not computer to read)

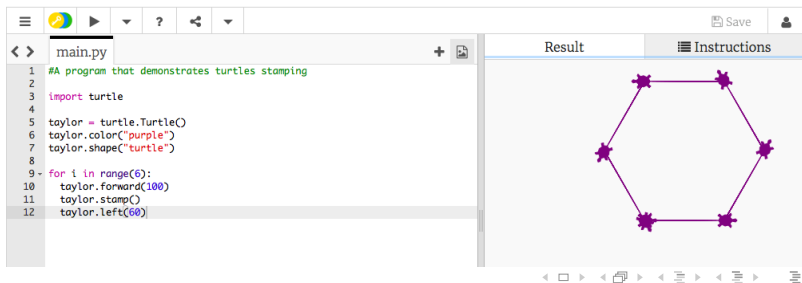
```
#This program prints: Hello, World!
```

← (this one also)

```
print("Hello, World!")
```

← Prints the string "Hello, World!" to the screen

- As well as definite loops & the turtle package:



## Week 2: variables, data types, more on loops & range()

## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.

## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - ▶ **int**: integer or whole numbers



## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - ▶ **int**: integer or whole numbers
  - ▶ **float**: floating point or real numbers

## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - ▶ **int**: integer or whole numbers
  - ▶ **float**: floating point or real numbers
  - ▶ **string**: sequence of characters

## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - ▶ **int**: integer or whole numbers
  - ▶ **float**: floating point or real numbers
  - ▶ **string**: sequence of characters
  - ▶ **list**: a sequence of items

## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - ▶ **int**: integer or whole numbers
  - ▶ **float**: floating point or real numbers
  - ▶ **string**: sequence of characters
  - ▶ **list**: a sequence of items  
e.g. `[3, 1, 4, 5, 9]` or `['violet', 'purple', 'indigo']`

## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - ▶ **int**: integer or whole numbers
  - ▶ **float**: floating point or real numbers
  - ▶ **string**: sequence of characters
  - ▶ **list**: a sequence of items  
e.g. [3, 1, 4, 5, 9] or ['violet', 'purple', 'indigo']
  - ▶ **class variables**: for complex objects, like turtles.

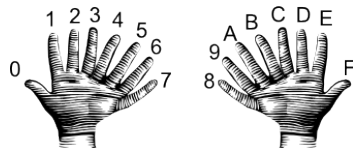
## Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - ▶ **int**: integer or whole numbers
  - ▶ **float**: floating point or real numbers
  - ▶ **string**: sequence of characters
  - ▶ **list**: a sequence of items  
e.g. [3, 1, 4, 5, 9] or ['violet', 'purple', 'indigo']
  - ▶ **class variables**: for complex objects, like turtles.
- More on loops & ranges:

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(x)
12
13 for c in "ABCD":
14     print(c)
```

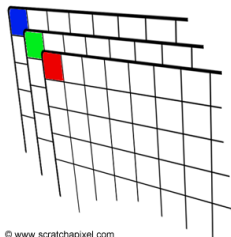
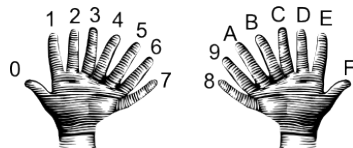
# Week 3: colors, hex, slices, numpy & images

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



# Week 3: colors, hex, slices, numpy & images

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

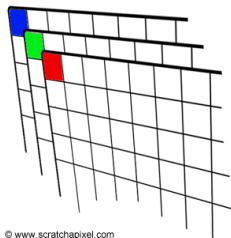
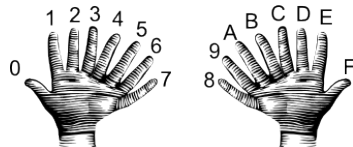


© www.scratchapixel.com



# Week 3: colors, hex, slices, numpy & images

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



© www.scratchapixel.com

```
>>> a[0,3:5]  
array([3,4])
```

```
>>> a[4:,4:]  
array([[44, 45],  
       [54, 55]])
```

```
>>> a[:,2]  
array([2,12,22,32,42,52])
```

```
>>> a[2::2,::2]  
array([[20,22,24]  
       [40,42,44]])
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

# Week 4: design problem (cropping images) & decisions



## Week 4: design problem (cropping images) & decisions



- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*

# Week 4: design problem (cropping images) & decisions



- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*
- Next: write pseudocode.
  - 1 Import numpy and pyplot.
  - 2 Ask user for file names and dimensions for cropping.
  - 3 Save input file to an array.
  - 4 Copy the cropped portion to a new array.
  - 5 Save the new array to the output file.

# Week 4: design problem (cropping images) & decisions



- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*
- Next: write pseudocode.
  - ① Import numpy and pyplot.
  - ② Ask user for file names and dimensions for cropping.
  - ③ Save input file to an array.
  - ④ Copy the cropped portion to a new array.
  - ⑤ Save the new array to the output file.
- Next: translate to Python.

## Week 4: design problem (cropping images) & decisions

```
yearBorn = int(input('Enter year born: '))
if yearBorn < 1946:
    print("Greatest Generation")
elif yearBorn <= 1964:
    print("Baby Boomer")
elif yearBorn <= 1984:
    print("Generation X")
elif yearBorn <= 2004:
    print("Millennial")
else:
    print("TBD")

x = int(input('Enter number: '))
if x % 2 == 0:
    print('Even number')
else:
    print('Odd number')
```

# Week 5: logical operators, truth tables & logical circuits

```
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

# Week 5: logical operators, truth tables & logical circuits

```
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

in1		in2	returns:
False	and	False	False
False	and	True	False
True	and	False	False
True	and	True	True





# Week 6: structured data, pandas, & more design

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21883,3623,,,2847,28423
1790,,30131,45049,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,40203,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419801,45468,37393,33829,1470103
1880,1164673,599495,56559,51980,38991,1911690
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,2437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018296,469042,732016,116511,5620048
1930,1867312,2560461,1079129,1265258,159346,6590446
1940,1889924,2698295,1297634,1394711,174441,7454995
1950,1940101,2738075,1550849,1452177,191555,78991957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7094862
1980,1428285,2230936,1801325,1168872,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1494873,2504790,2230722,1385108,448730,8175123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

# Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21883,3623,,,2847,28423
1790,,30131,45049,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,40203,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419921,45468,37393,33829,1470193
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,2437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018296,469042,732016,116511,5620048
1930,1867312,2560461,1079129,1265258,159346,4590446
1940,1889924,2698295,1297634,1394711,174441,7454995
1950,1940101,2738275,1550849,1452177,191555,78991957
1960,1698281,2627319,1809578,1424815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8006278
2010,1494873,2504790,2230722,1385108,448730,8175123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

# Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv', skiprows=5)
```

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4548,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,8020,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419921,45468,37393,33829,1470193
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,2437202
1910,2331542,1634351,284041,430980,85989,4766883
1920,2284123,2018256,469042,732016,116511,4620048
1930,1867312,2560461,1079129,1565258,159346,4590446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1940101,2738275,1550849,1451277,291555,78991957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7094862
1980,1428285,2230936,1891325,1168972,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8006278
2010,1494873,2504790,2230722,1385108,448730,81751123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

# Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv', skiprows=5)
```

```
pop.plot(x="Year")
plt.show()
```

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4548,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,8023,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419921,45468,37393,33829,1470193
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,2437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284123,2018256,469042,732016,116511,4620048
1930,1867312,2580461,1079129,1265258,158784,4590446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1940101,2738275,1500849,1451277,291555,7893957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1494873,2504790,2230722,1385108,448730,8175123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

# Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv', skiprows=5)
```

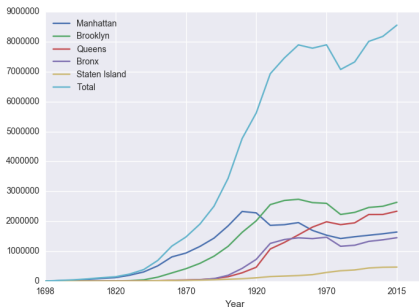
```
pop.plot(x="Year")
plt.show()
```

Source: [https://en.wikipedia.org/wiki/Demographics\\_of\\_New\\_York\\_City](https://en.wikipedia.org/wiki/Demographics_of_New_York_City),  
All population figures are consistent with present-day boundaries.  
First census after the consolidation of the five boroughs.

```
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,727,7681
1771,21863,3623,,2847,28423
1790,33131,4548,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,40203,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419801,45468,37393,33829,1470183
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,3437202
1910,2331542,1634351,284041,430989,85969,4766883
1920,2284103,2018256,469042,732016,116531,5620048
1930,1867312,2560451,1078129,1265598,159346,4906446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1940101,2738275,1500469,1452177,291505,7892957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7094862
1980,1428285,2210936,1801325,1164872,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1484873,2504760,2230722,1385108,468730,8175123
2015,1644518,2636735,2339155,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6



# Week 7: functions

- Functions are a way to break code into pieces, that can be easily reused.

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

# Week 7: functions

```
#Name:  your name here  
#Date:  October 2017  
#This program, uses functions,  
#      says hello to the world!
```

```
def main():  
    print("Hello, World!")  
  
if __name__ == "__main__":  
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.

# Week 7: functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#      says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`



# Week 7: functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

# Week 7: functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#      says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:  
Example: `print("Hello", "World")`

# Week 7: functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

# Week 7: functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

# Week 8: function parameters, github

- Functions can have **input parameters**.

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: ' ))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: ' ))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

# Week 8: function parameters, github

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: ' ))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTIP = float(input('Enter dinner tip: ' ))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

# Week 8: function parameters, github

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: '))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: '))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

# Week 8: function parameters, github

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: '))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: '))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**



# Week 8: function parameters, github

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip: '))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip: '))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**
- Functions can also **return values** to where it was called.

# Week 8: function parameters, github

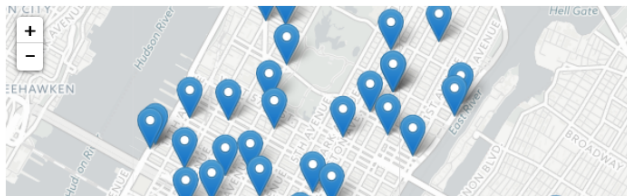
```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: '))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: '))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**.
- Functions can also **return values** to where it was called.

## Week 9: top-down design, folium, loops, and random()



```
def main():
    dataF = getData()
    latColName, lonColName = getColumnNames()
    lat, lon = getLocale()
    cityMap = folium.Map(location = [lat,lon], tiles = 'cartodbpositron', zoom_start=11)
    dotAllPoints(cityMap,dataF,latColName,lonColName)
    markAndFindClosest(cityMap,dataF,latColName,lonColName,lat,lon)
    writeMap(cityMap)
```

## Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.

```
import turtle
import random

trex = turtle.Turtle()
trex.speed(10)

for i in range(100):
    trex.forward(10)
    a = random.randrange(0,360,90)
    trex.right(a)
```

# Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
    trey.forward(10)
    a = random.randrange(0,360,90)
    trey.right(a)
```

# Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
    trey.forward(10)
    a = random.randrange(0,360,90)
    trey.right(a)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.
- Python's built-in random package has useful methods for generating random whole numbers and real numbers.

# Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

```
import turtle
import random

trex = turtle.Turtle()
trex.speed(10)

for i in range(100):
    trex.forward(10)
    a = random.randrange(0,360,90)
    trex.right(a)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.
- Python's built-in random package has useful methods for generating random whole numbers and real numbers.
- To use, must include:  
import random.

# Python & Circuits Review: 10 Weeks in 10 Minutes



- Input/Output (I/O): `input()` and `print()`;  
pandas for CSV files
- Types:
  - ▶ Primitive: `int`, `float`, `bool`, `string`;
  - ▶ Container: lists (but not dictionaries/hashtes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`
- Logical Expressions & Circuits
- Functions: parameters & returns
- Packages:
  - ▶ Built-in: `turtle`, `math`, `random`
  - ▶ Popular: `numpy`, `matplotlib`, `pandas`, `folium`
- Simplified Machine Language




# Today's Topics



- Data Representation
- Machine Language: Jumps & Loops
- Recap of Python & Circuits
- **Design Patterns: Sorting**






# Sorting Demo

- ① Let `num` be the number of items in the list.


# Sorting Demo

- 1 Let `num` be the number of items in the list.
- 2 Repeat `num` times:


	 Insertion	 Selection	 Bubble	 Shell	 Merge
 Random					
 Nearly Sorted					
 Reversed					
 Few Unique					

# Sorting Demo

- ① Let `num` be the number of items in the list.
- ② Repeat `num` times:
  - If the first in line is taller than the second, switch places.

	 Insertion	 Selection	 Bubble	 Shell	 Merge
 Random					
 Nearly Sorted					
 Reversed					
 Few Unique					

# Sorting Demo

- ① Let `num` be the number of items in the list.
- ② Repeat `num` times:
  - ▶ If the first in line is taller than the second, switch places.
  - ▶ Next check if the current second in line is taller than the third.

# Sorting Demo

① Let `num` be the number of items in the list.

② Repeat `num` times:

- ▶ If the first in line is taller than the second, switch places.
- ▶ Next check if the current second in line is taller than the third.
- ▶ If so, switch places.

# Sorting Demo

① Let `num` be the number of items in the list.

② Repeat `num` times:

- ▶ If the first in line is taller than the second, switch places.
- ▶ Next check if the current second in line is taller than the third.
- ▶ If so, switch places.
- ▶ Repeat until you reach the end of the list.


# Sorting Demo

	Insertion	Selection	Bubble	Shell	Merge
Random					
Nearly Sorted					
Reversed					
Few Unique					

Show sorting demo.



# Lecture Slip: Design Patterns

	 Insertion	 Selection	 Bubble	 Shell	 Merge
 Random					
 Nearly Sorted					
 Reversed					
 Few Unique					

In pairs or triples:

- Fill in the UTAs' name at the top of the sheet.
- What does the code do?

# Recap: Python, Languages, & Design

- On lecture slip, write down a topic you wish we had spent more time (and why).

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

# Recap: Python, Languages, & Design

```
#Name: your name here  
#Date: October 2017  
#This program, uses functions,  
#    says hello to the world!
```

```
def main():  
    print("Hello, World!")
```

```
if __name__ == "__main__":  
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language

# Recap: Python, Languages, & Design

```
#Name: your name here  
#Date: October 2017  
#This program, uses functions,  
#    says hello to the world!
```

```
def main():  
    print("Hello, World!")
```

```
if __name__ == "__main__":  
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits

# Recap: Python, Languages, & Design

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits
- Simplified Machine Language

# Recap: Python, Languages, & Design

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits
- Simplified Machine Language
- Design: from written description ('specs') to function inputs & outputs ('APIs')

# Recap: Python, Languages, & Design

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits
- Simplified Machine Language
- Design: from written description ('specs') to function inputs & outputs ('APIs')
- Pass your lecture slips to the aisles for the UTAs to collect.

# Final Overview: Top-Down Design & APIs

For each question, write **only the function header (name & inputs) and return values** (often called the Application Programming Interface (API)):



# Final Overview: Top-Down Design & APIs

For each question, write **only the function header (name & inputs) and return values** (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.
- Write a function that takes a string and returns its length.
- Write a function that, given a DataFrame, returns the minimal value in the first column.
- Write a function that takes a whole number and returns the corresponding binary number as a string.
- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

# Final Overview: Top-Down Design & APIs

For each question, write **only the function header (name & inputs) and return values** (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.
- Write a function that takes a string and returns its length.
- Write a function that, given a DataFrame, returns the minimal value in the first column.
- Write a function that takes a whole number and returns the corresponding binary number as a string.
- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

*(Hint: highlight key words, make list of inputs, list of outputs, then put together.)*

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

```
def kg2lbs(kg):  
    ...  
    return(lbs)
```

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

```
def kg2lbs(kg)
    lbs = kg * 2.2
    return(lbs)
```

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

```
def sLength(str):  
    ...  
    return(length)
```

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

```
def sLength(str):  
    length = len(str)  
    return(length)
```



# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

```
def getMin(df):  
    ...  
    return(min)
```

# Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

```
def getMin(df):  
    min = df['Manhattan'].min()  
    return(min)
```

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

```
def num2bin(num):  
    ...  
    return(bin)
```

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

```
def num2bin(num):  
    binStr = ""  
    while (num > 0):  
        #Divide by 2, and add the remainder to the string  
        r = num %2  
        binString = str(r) + binStr  
        num = num / 2  
    return(binStr)
```

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

# Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

```
def computePayment(loan,rate,year):  
    ....  
    return(payment)
```



# Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

```
def computePayment(loan,rate,year):  
    (Some formula for payment)  
    return(payment)
```