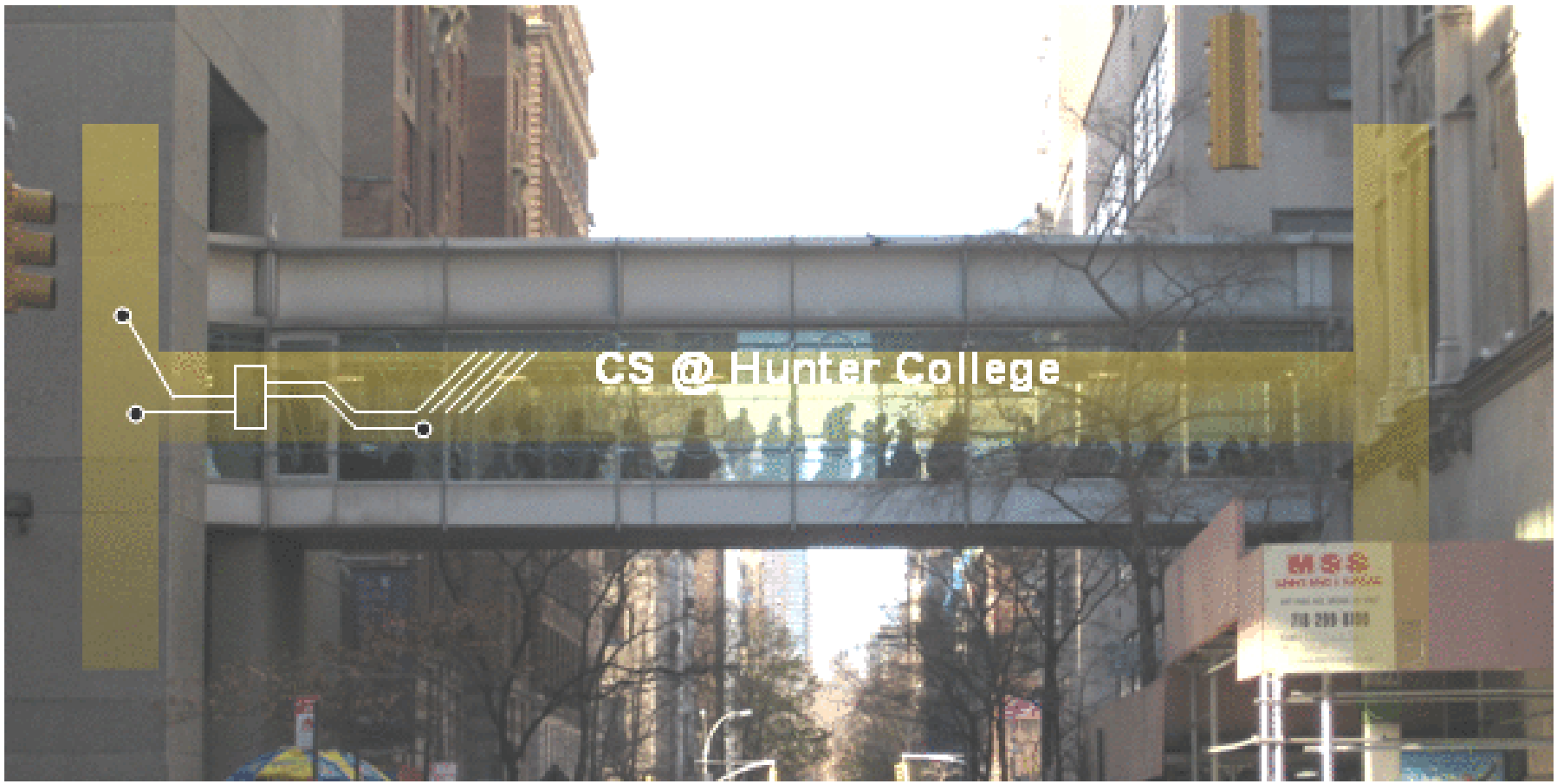
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

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

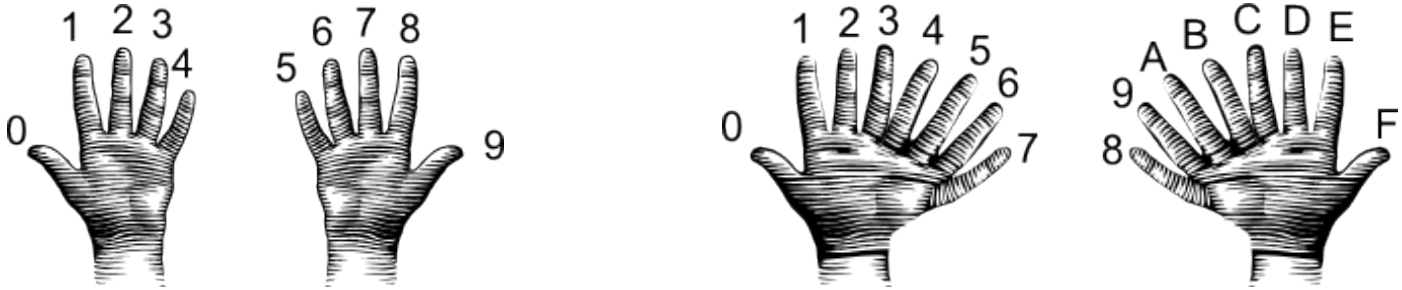
From lecture slips & recitation sections.

- 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

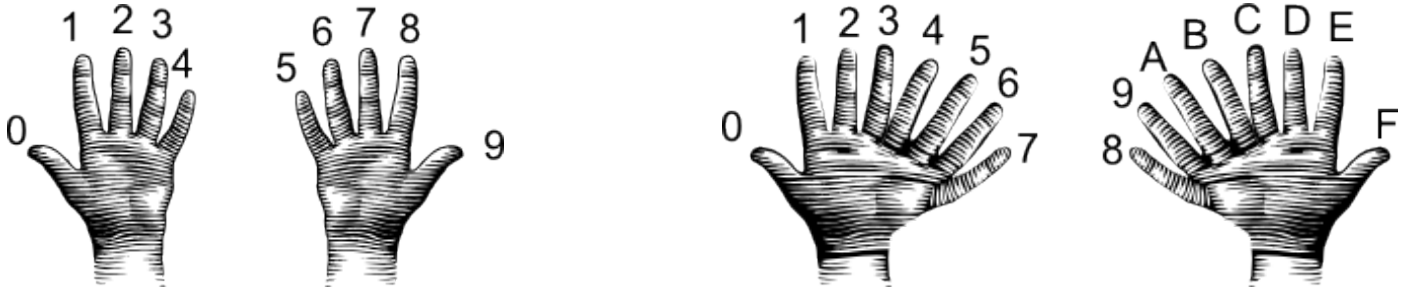
Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:

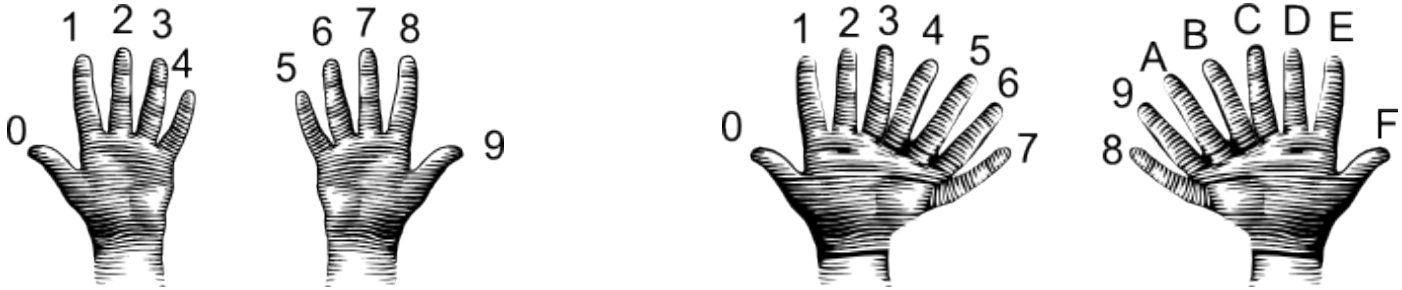
Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:
  - Divide by 16.

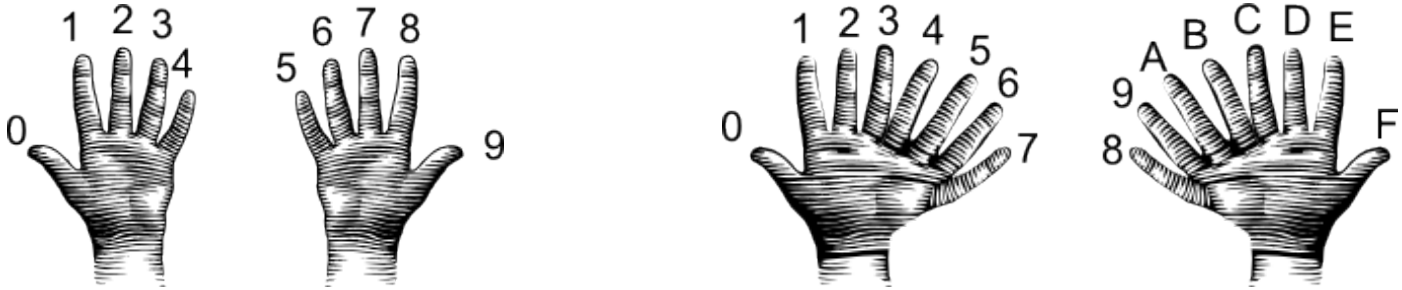
Recap: Converting between bases



(from i-programmer.info)

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

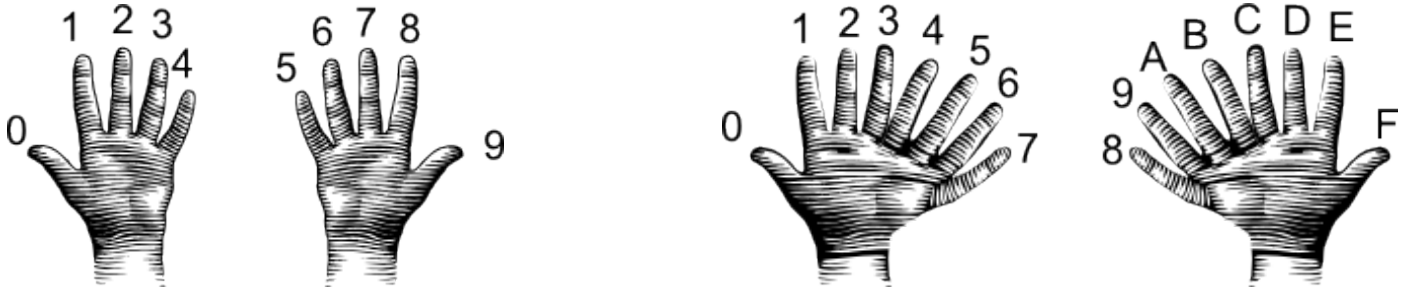
Recap: Converting between bases



(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.

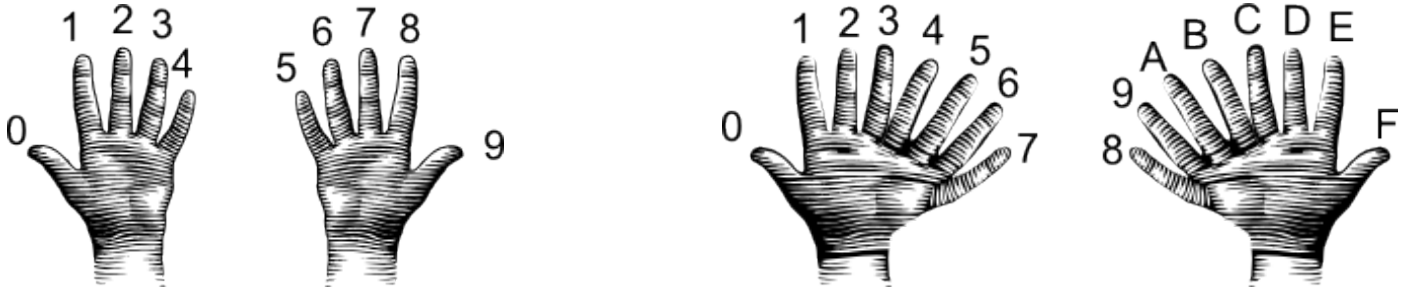
Recap: Converting between bases



(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?

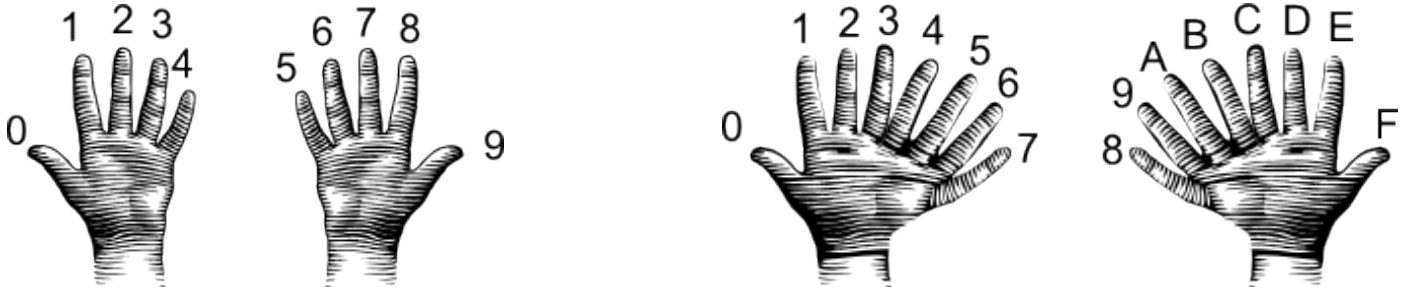
Recap: Converting between bases



(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.`

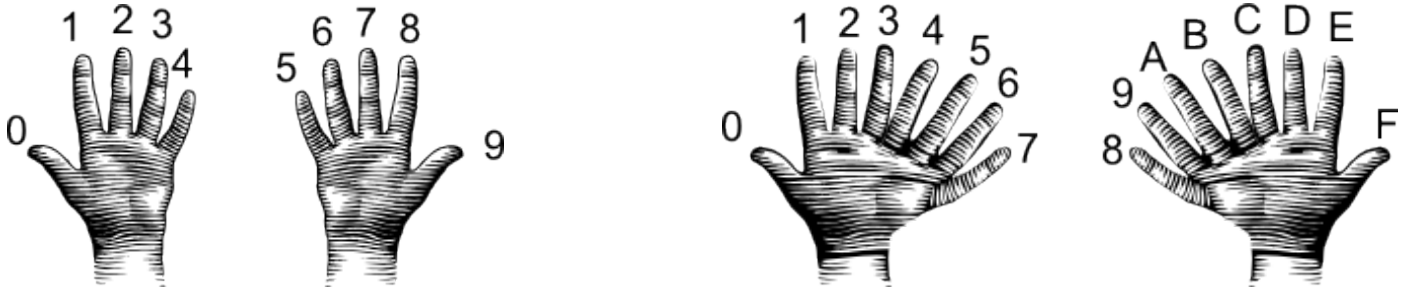
Recap: Converting between bases



(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.`

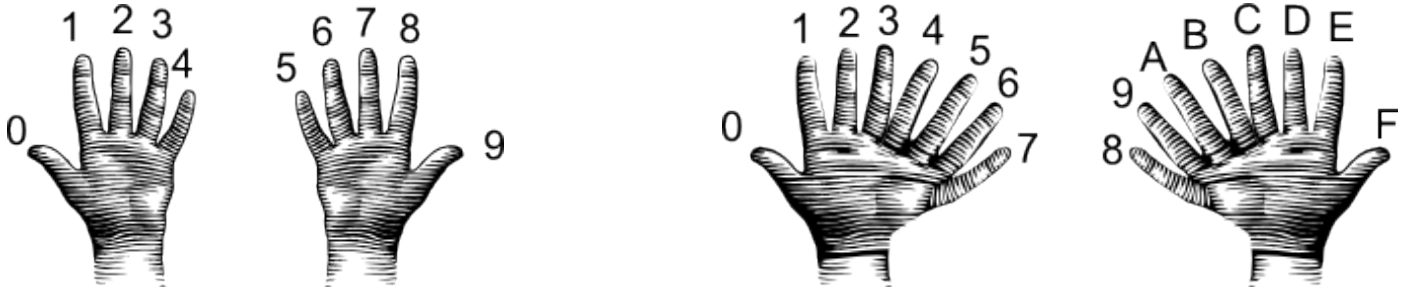
Recap: Converting between bases



(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.`

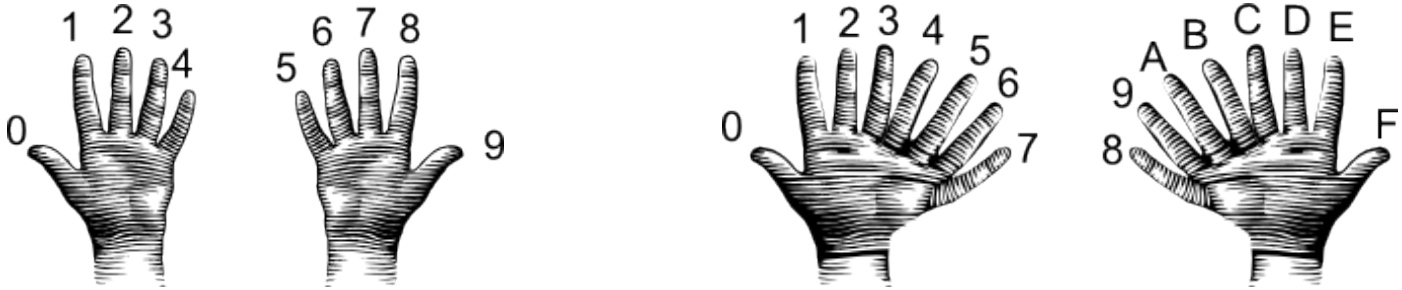
Recap: Converting between bases



(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.

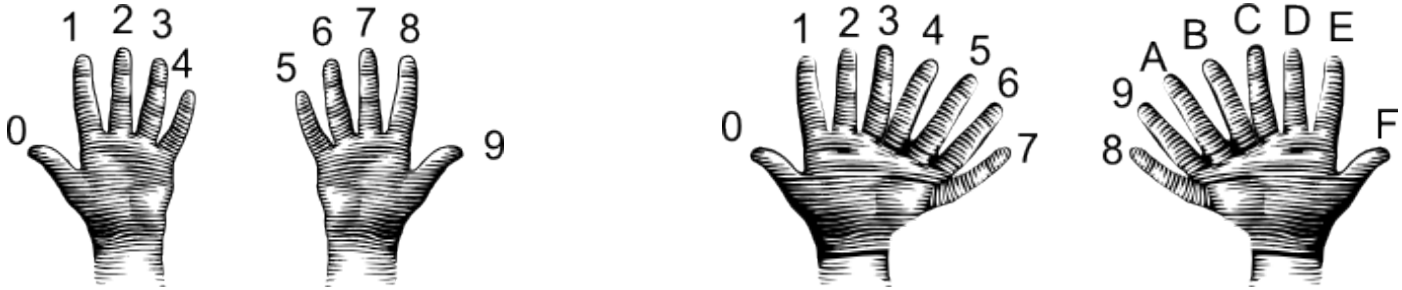
Recap: Converting between bases



(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?

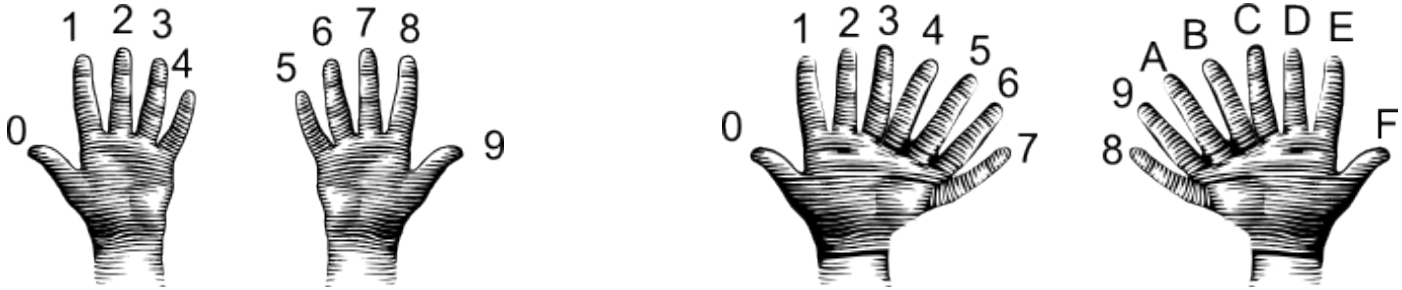
Recap: Converting between bases



(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.`

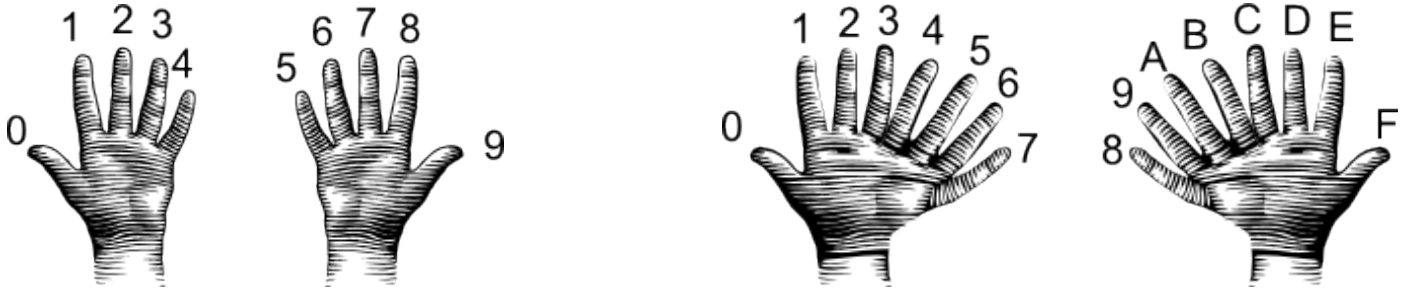
Recap: Converting between bases



(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.`

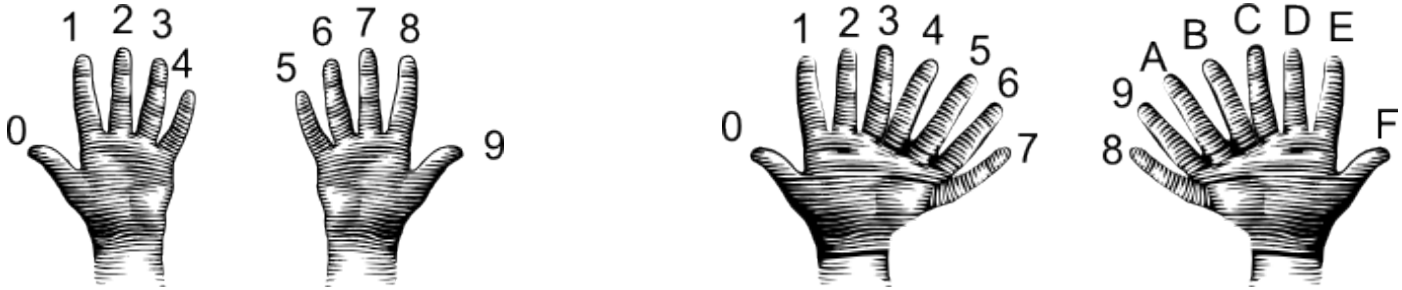
Recap: Converting between bases



(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.`

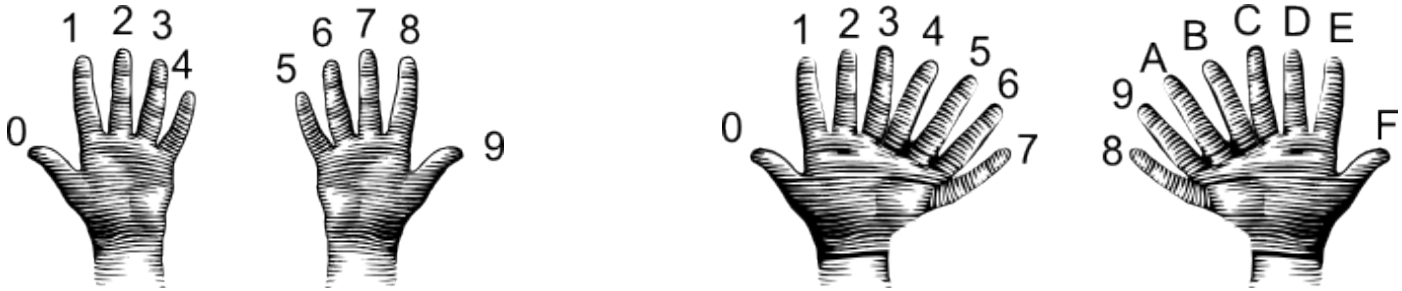
Recap: Converting between bases



(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.

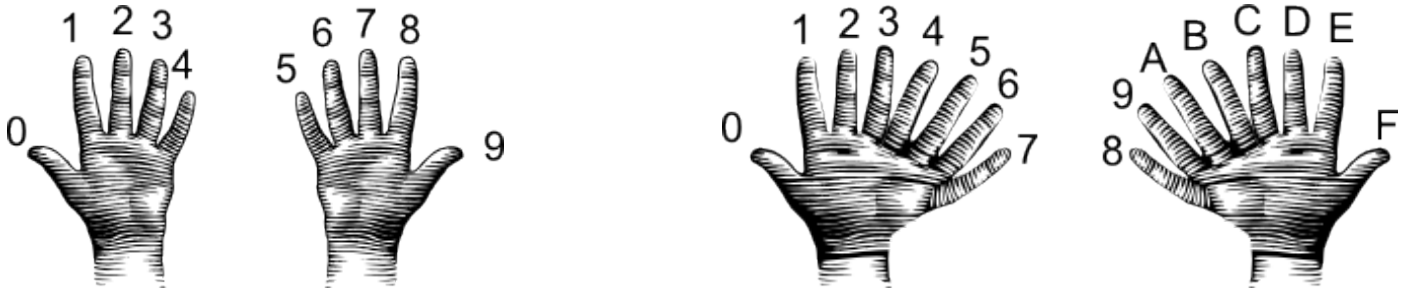
Recap: Converting Between Bases



(from i-programmer.info)

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

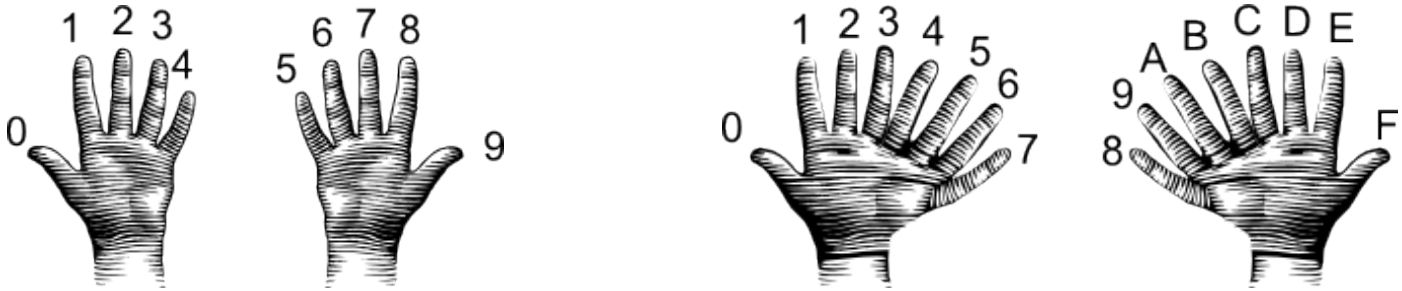
Recap: Converting Between Bases



(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.

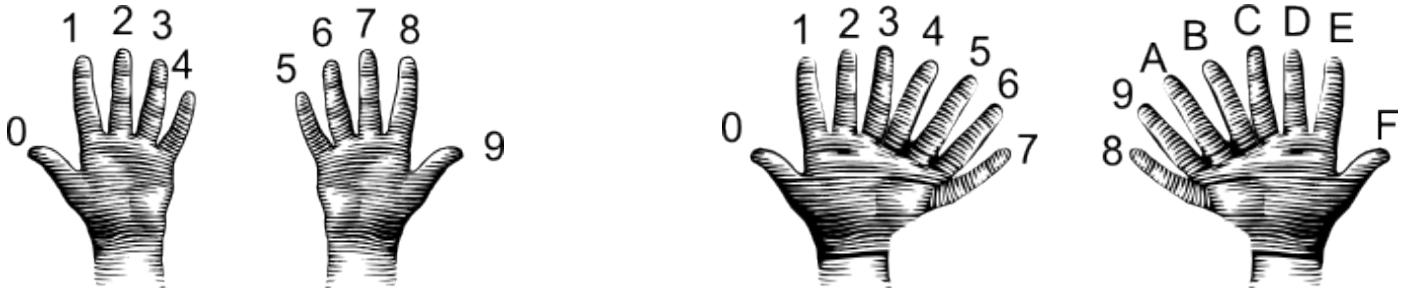
Recap: Converting Between Bases



(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?

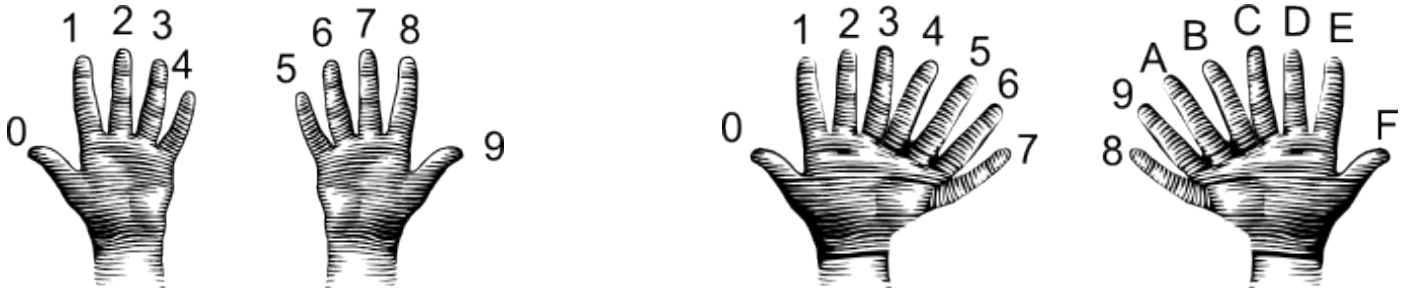
Recap: Converting Between Bases



(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.`

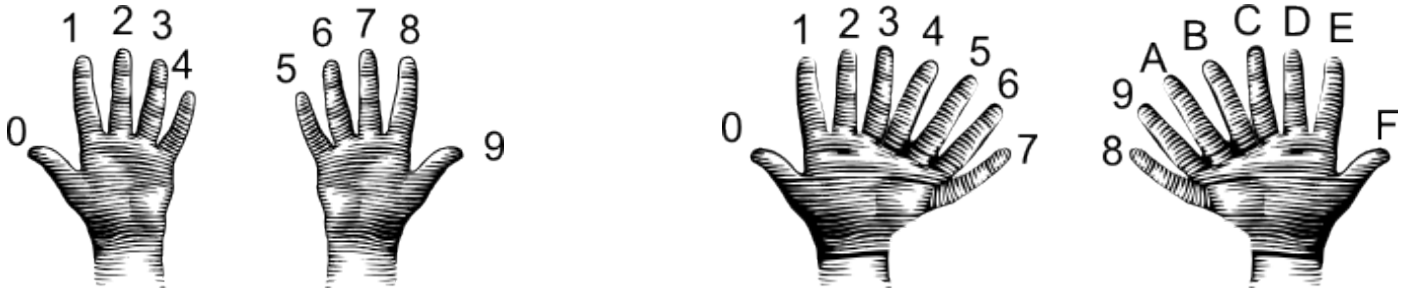
Recap: Converting Between Bases



(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*16 is 32.`

Recap: Converting Between Bases



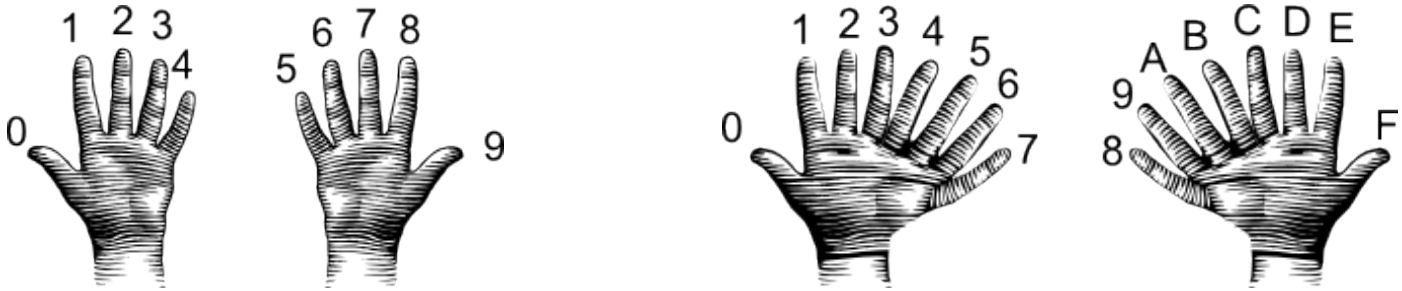
(from i-programmer.info)

- 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*16 is 32.`

    `A in decimal digits is 10.`

Recap: Converting Between Bases



(from i-programmer.info)

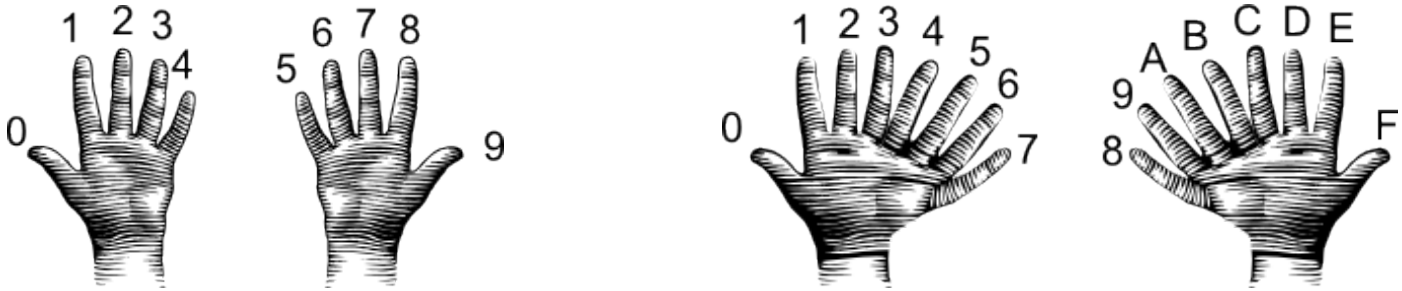
- From hexadecimal to decimal:
  - Convert first digit to decimal and multiple by 16.
  - Convert second digit to decimal and add to total.
  - Example: what is 2A as a decimal number?

    `2 in decimal is 2. 2*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?

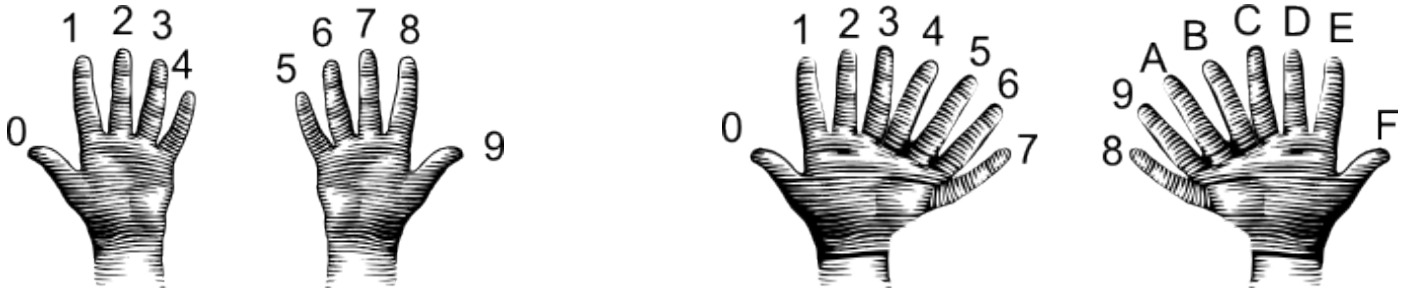
    `2 in decimal is 2. 2*16 is 32.`

    `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*16 is 32.`

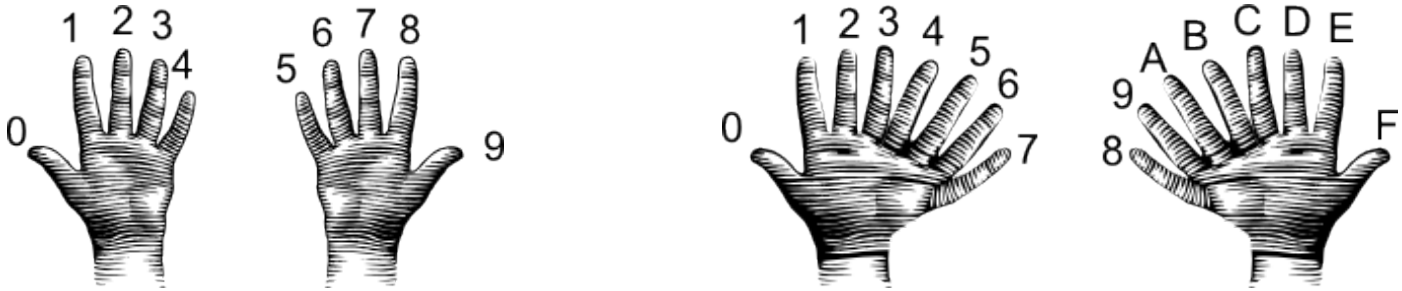
    `A in decimal digits is 10.`

    `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*16 is 32.`

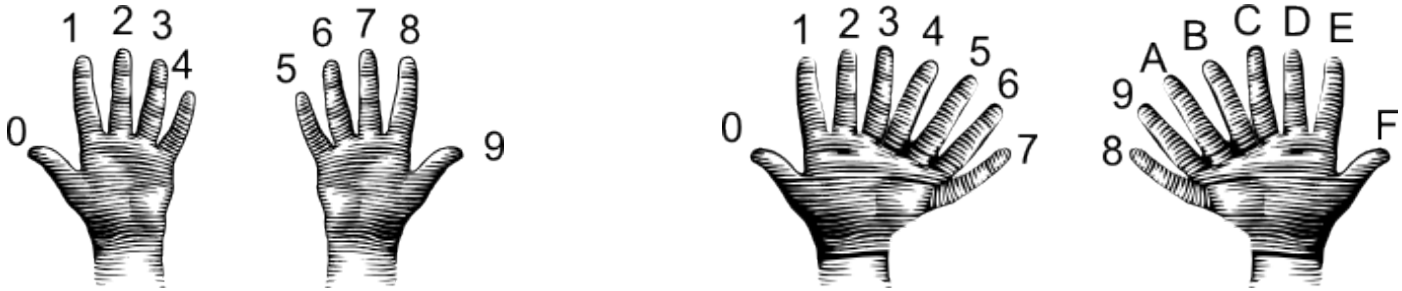
    `A in decimal digits is 10.`

    `32 + 10 is 42.`

    Answer is 42.
  - Example: what is 99 as a decimal number?

    `9 in decimal is 9. 9*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*16 is 32.`

    `A in decimal digits is 10.`

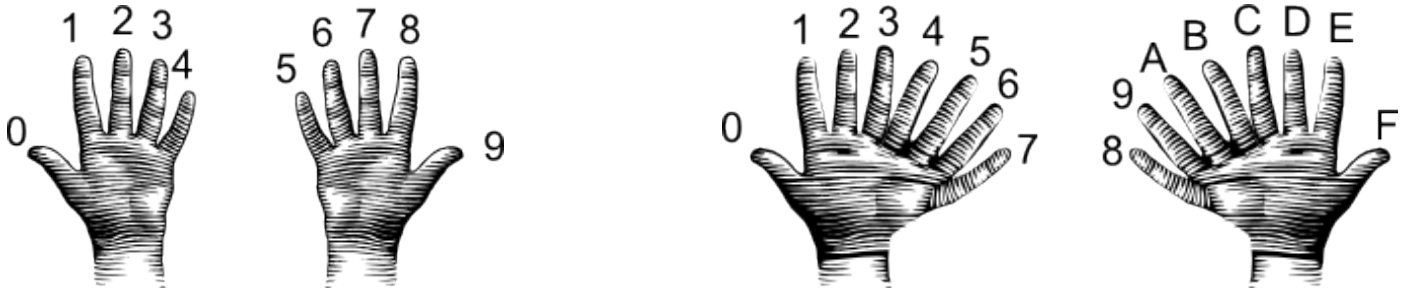
    `32 + 10 is 42.`

    Answer is 42.
  - Example: what is 99 as a decimal number?

    `9 in decimal is 9. 9*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*16 is 32.`

    `A in decimal digits is 10.`

    `32 + 10 is 42.`

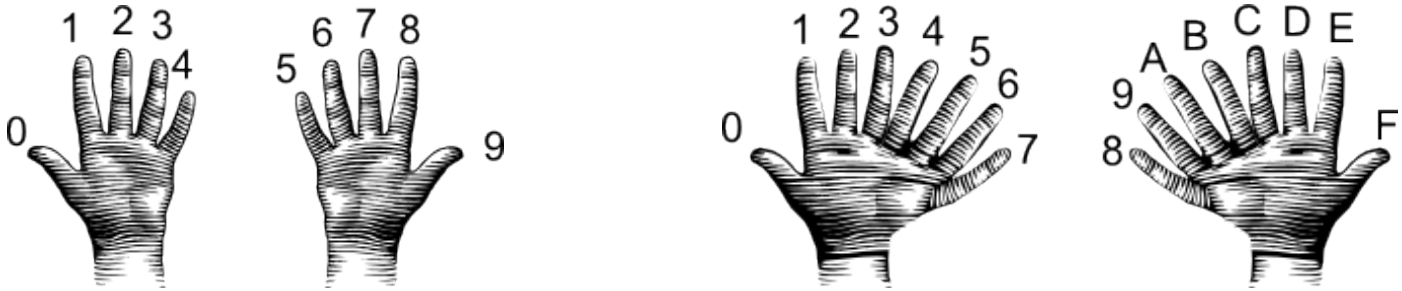
    Answer is 42.
  - Example: what is 99 as a decimal number?

    `9 in decimal is 9. 9*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*16 is 32.`

    `A in decimal digits is 10.`

    `32 + 10 is 42.`

    Answer is 42.
  - Example: what is 99 as a decimal number?

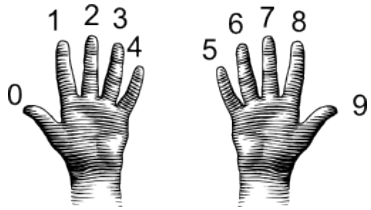
    `9 in decimal is 9. 9*16 is 144.`

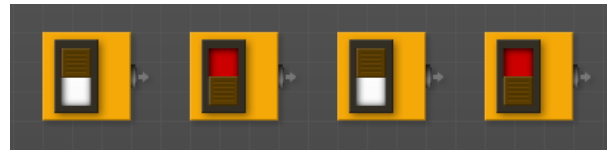
    `9 in decimal digits is 9`

    `144 + 9 is 153.`

    Answer is 153.

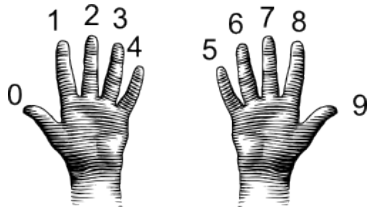
Recap: Converting Between Bases

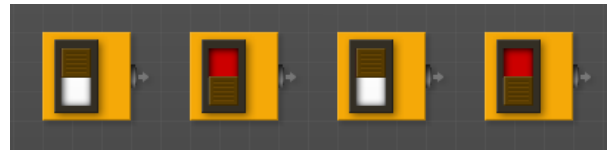




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.

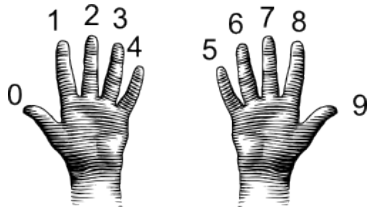
Recap: Converting Between Bases

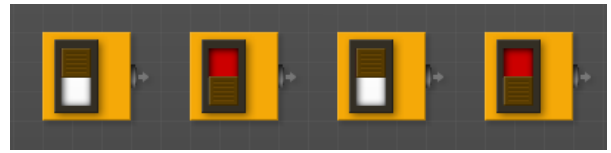




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.

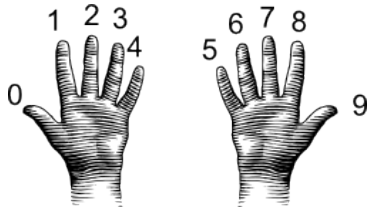
Recap: Converting Between Bases

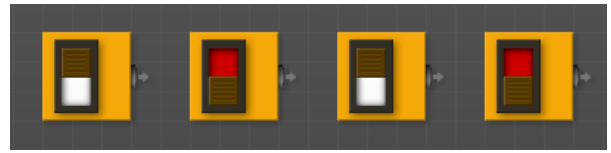




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.

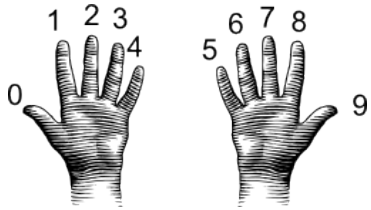
Recap: Converting Between Bases

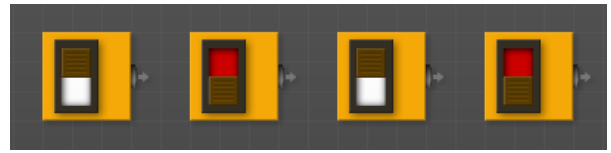




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.

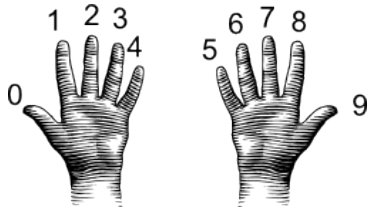
Recap: Converting Between Bases

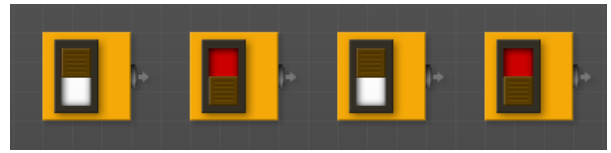




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

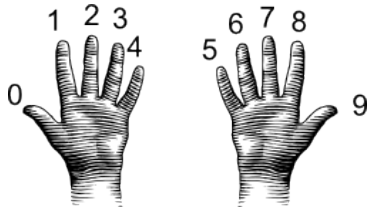
Recap: Converting Between Bases

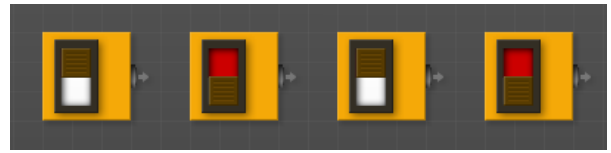




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

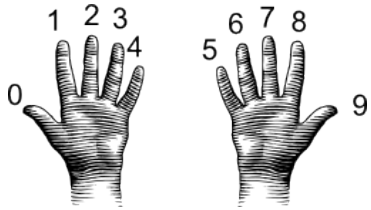
Recap: Converting Between Bases

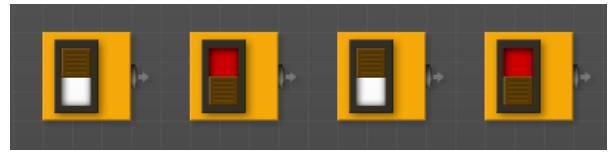




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

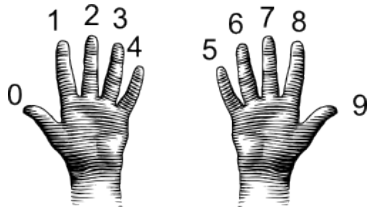
Recap: Converting Between Bases

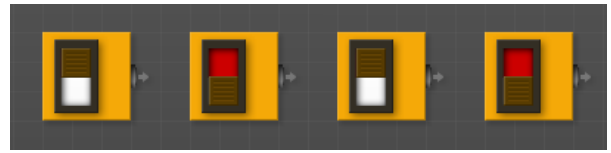




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

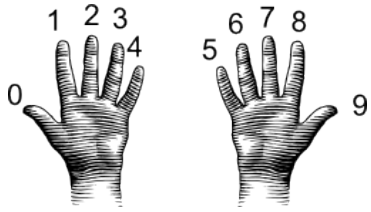
Recap: Converting Between Bases

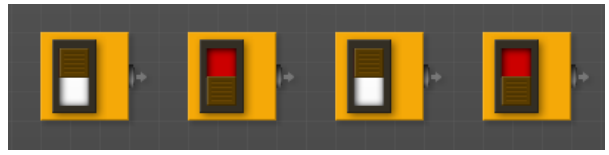




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

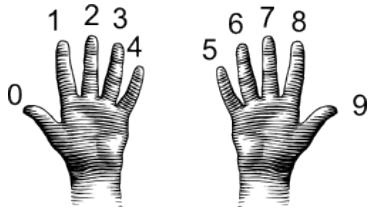


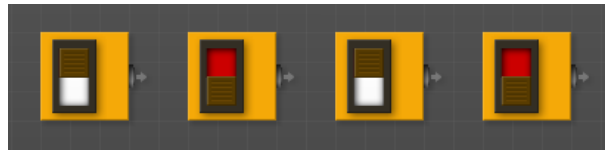


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

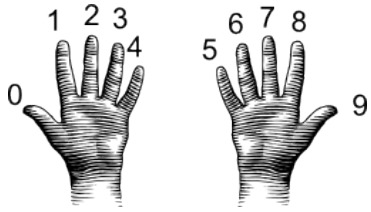


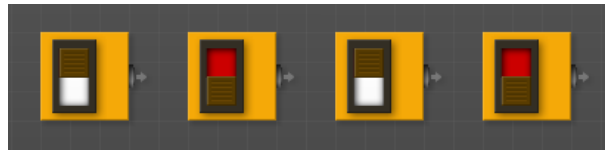


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

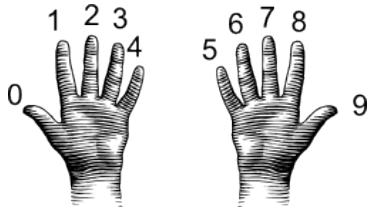


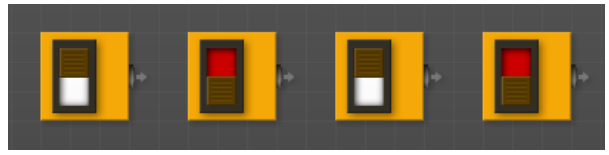


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

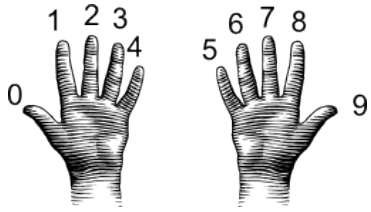


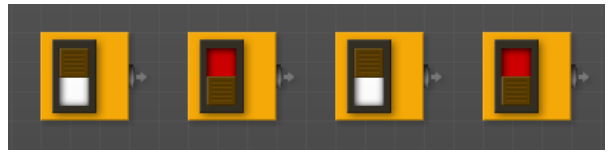


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

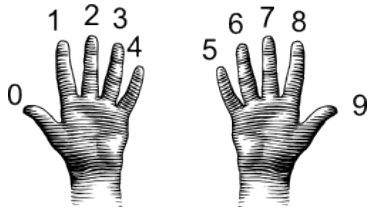


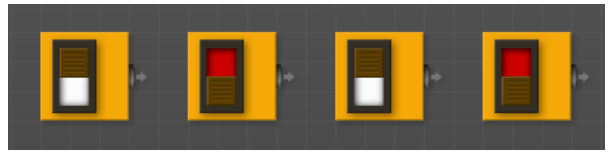


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

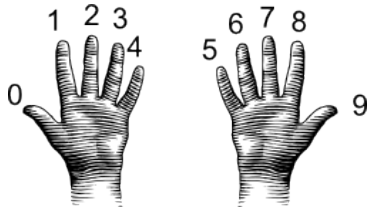


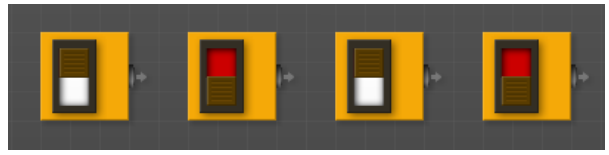


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

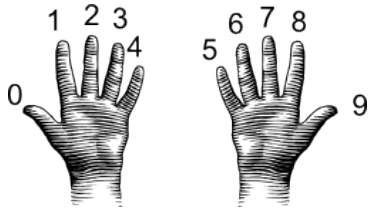


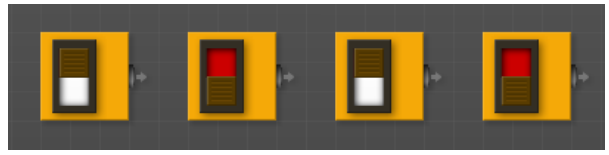


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

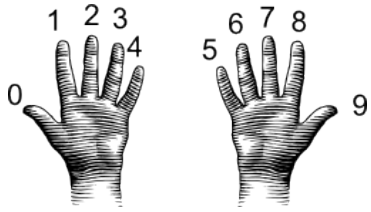


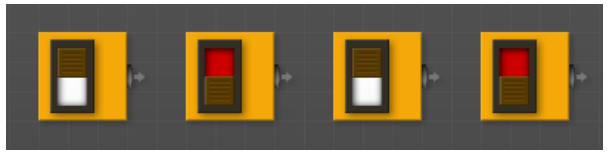


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

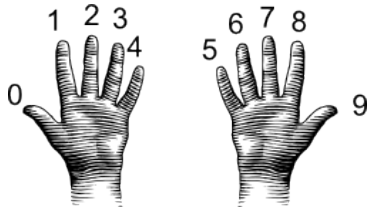


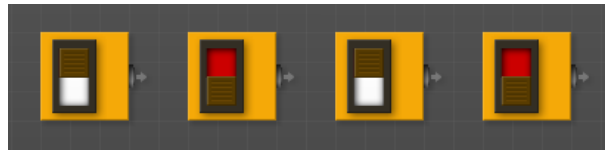


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

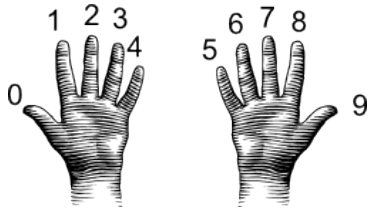


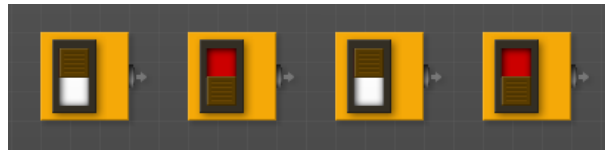


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

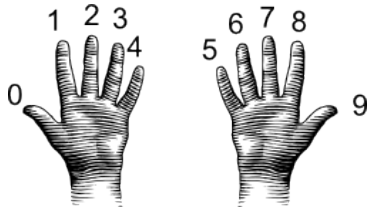


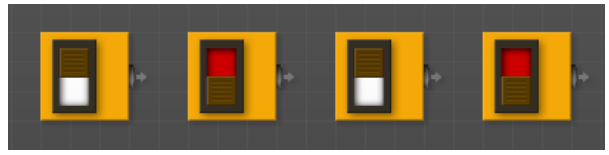


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

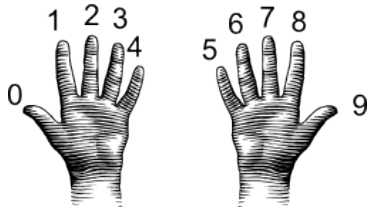


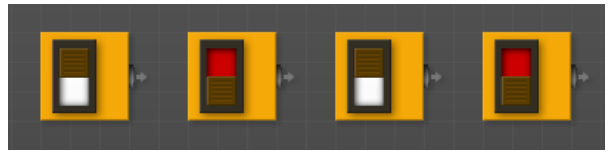


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

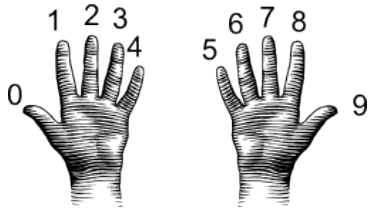


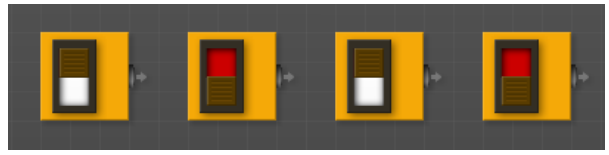


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

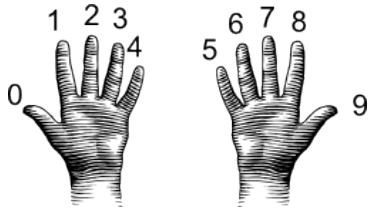


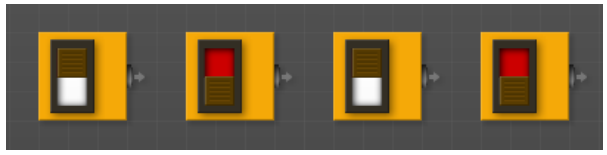


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

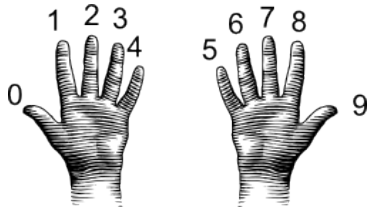


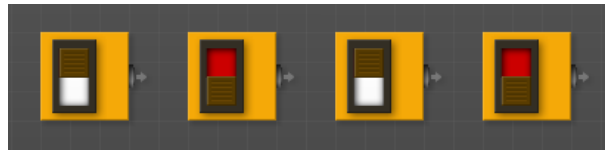


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

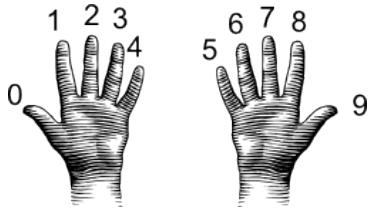


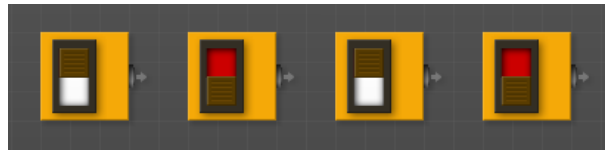


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

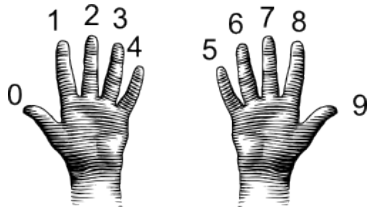


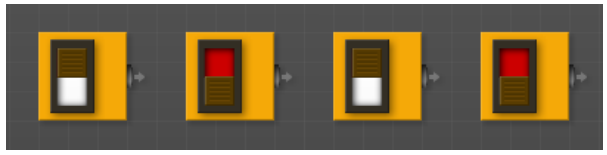


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

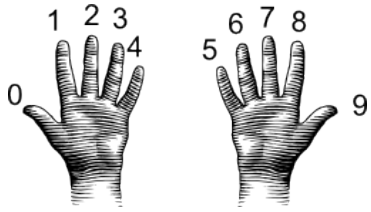


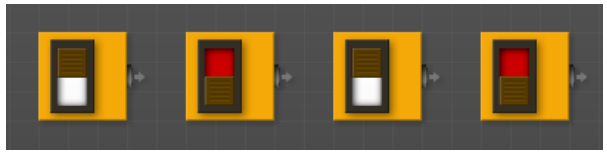


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

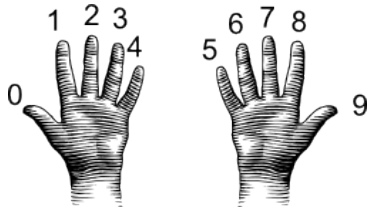


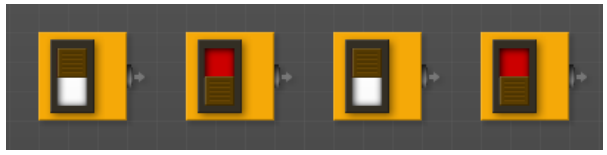


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

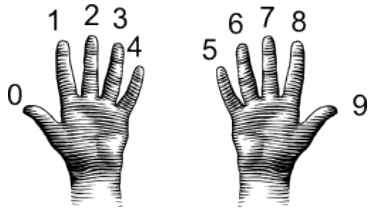


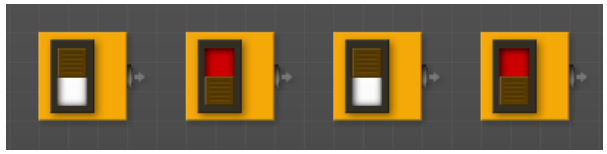


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

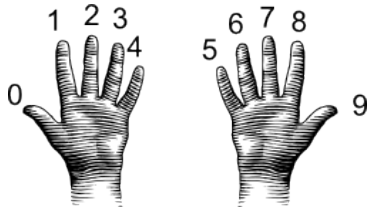


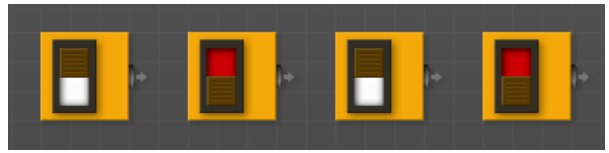


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

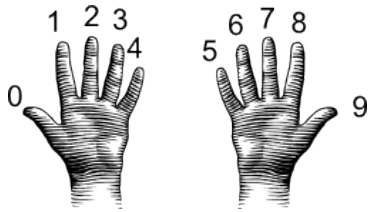


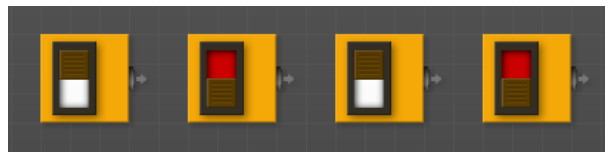


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder 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
```

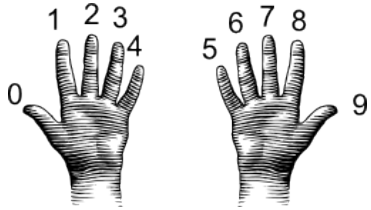
Recap: Converting Between Bases

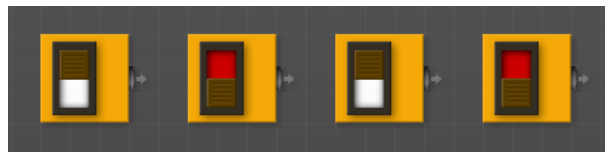




- Example: what is 99 in binary notation?

Recap: Converting Between Bases

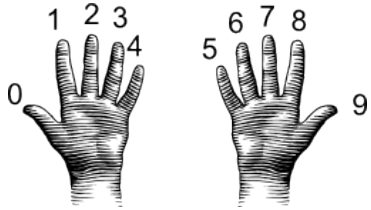


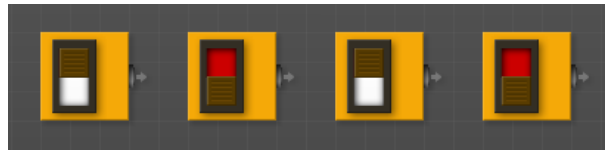


- Example: what is 99 in binary notation?

```
99/128 is 0 rem 99.
```

Recap: Converting Between Bases

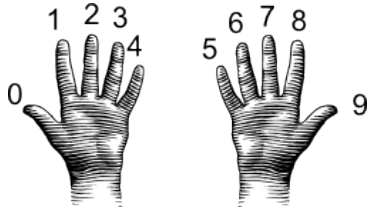


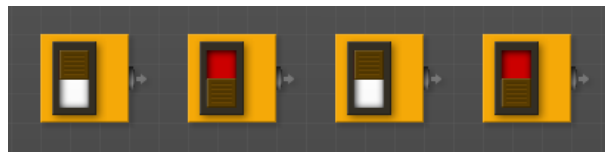


- Example: what is 99 in binary notation?

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

Recap: Converting Between Bases

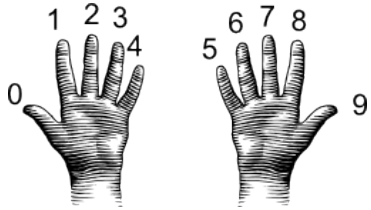


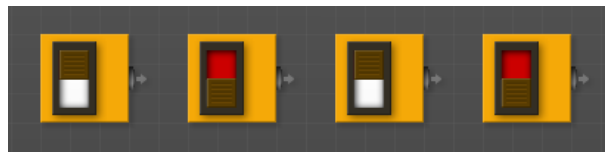


- Example: what is 99 in binary notation?

```
99/128 is 0 rem 99. First digit is 0:    0...
99/64 is 1 rem 35.
```

Recap: Converting Between Bases

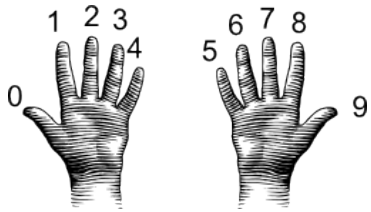


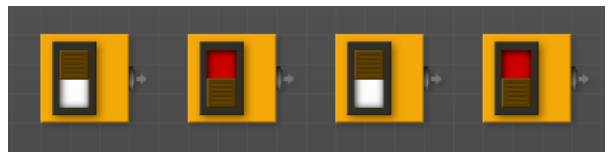


- Example: what is 99 in binary notation?

```
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

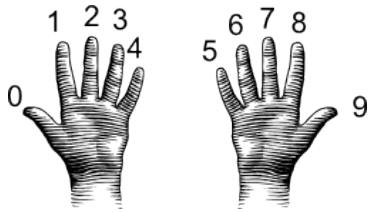


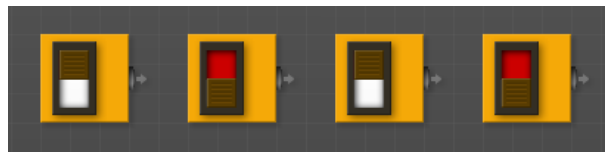


- Example: what is 99 in binary notation?

```
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

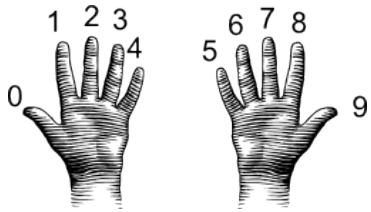


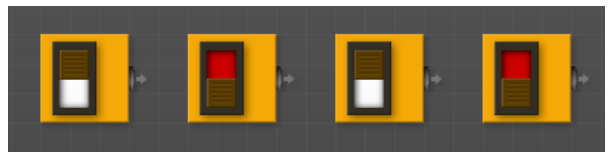


- Example: what is 99 in binary notation?

```
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

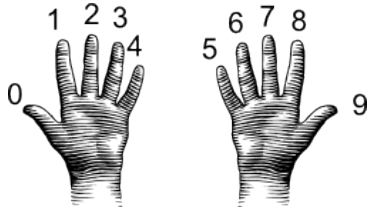


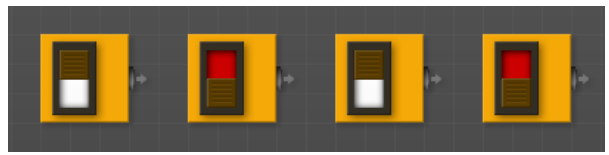


- Example: what is 99 in binary notation?

```
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

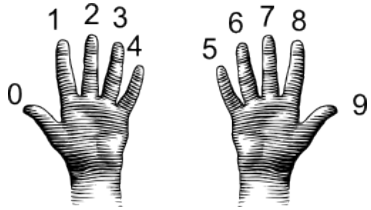


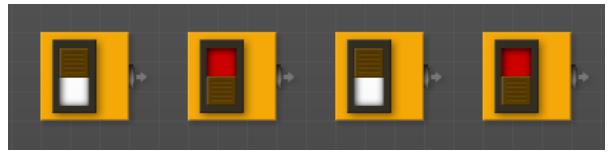


- Example: what is 99 in binary notation?

```
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

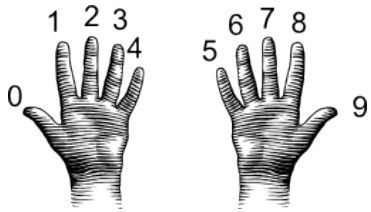


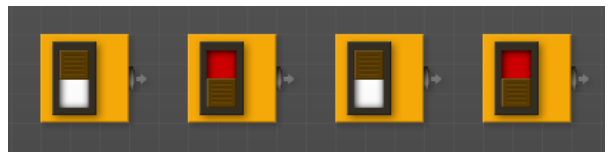


- Example: what is 99 in binary notation?

```
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

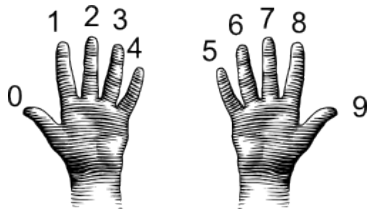


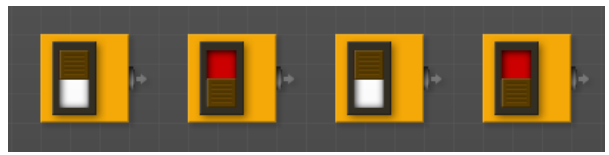


- Example: what is 99 in binary notation?

```
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

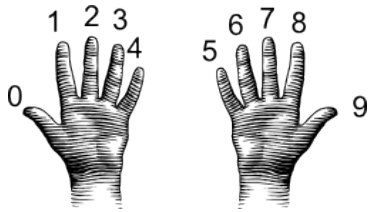


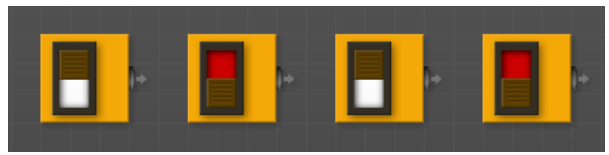


- Example: what is 99 in binary notation?

```
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

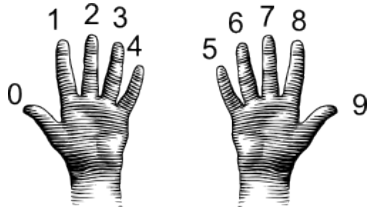


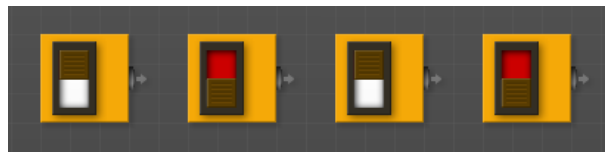


- Example: what is 99 in binary notation?

```
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

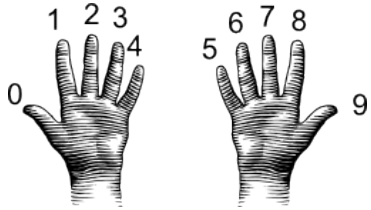


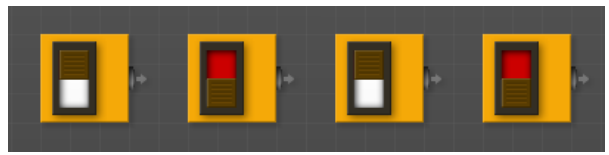


- Example: what is 99 in binary notation?

```
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

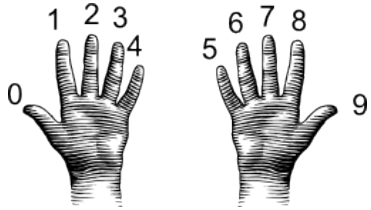


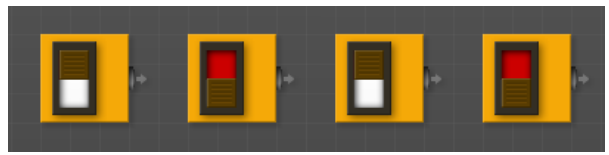


- Example: what is 99 in binary notation?

```
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

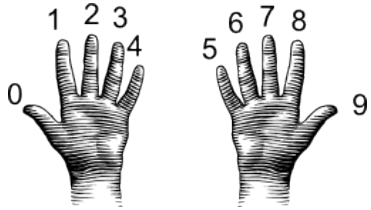


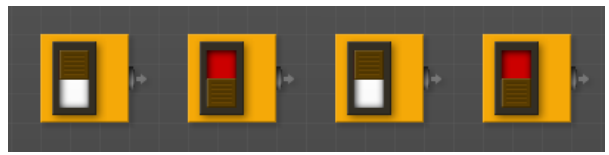


- Example: what is 99 in binary notation?

```
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

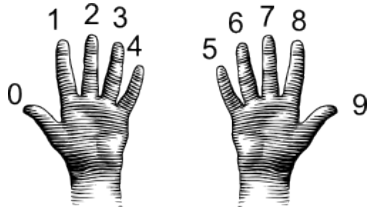


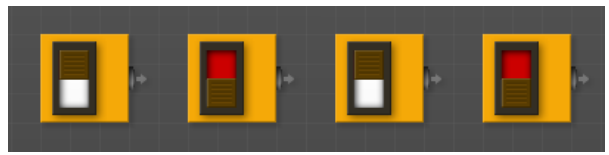


- Example: what is 99 in binary notation?

```
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

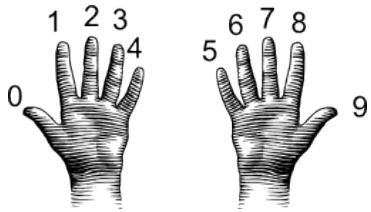


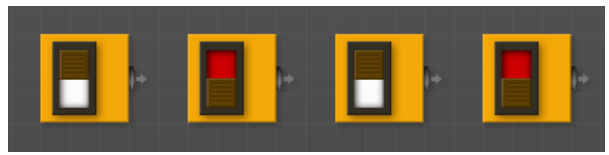


- Example: what is 99 in binary notation?

```
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

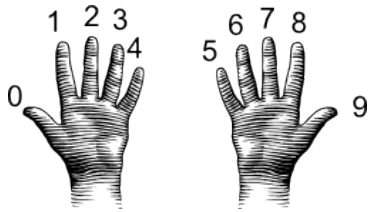


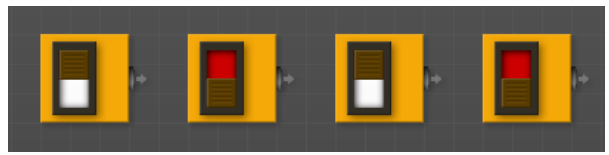


- Example: what is 99 in binary notation?

```
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

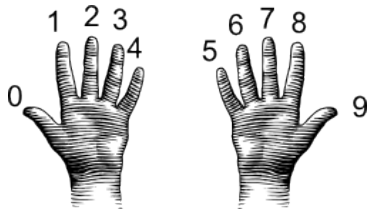


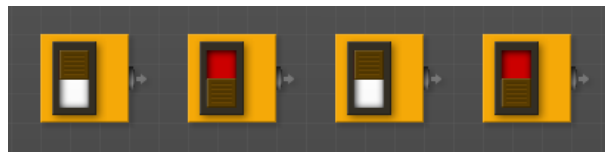


- Example: what is 99 in binary notation?

```
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

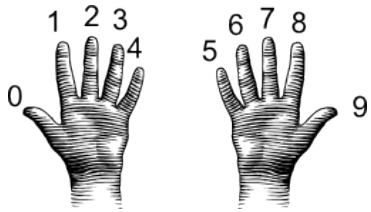


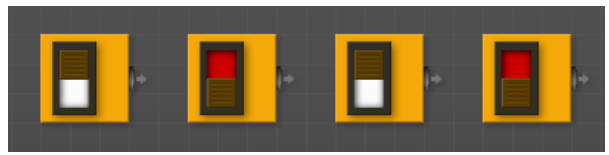


- Example: what is 99 in binary notation?

```
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

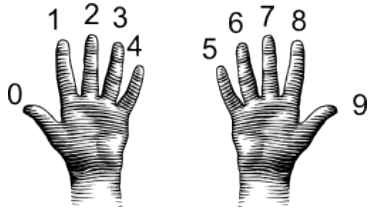


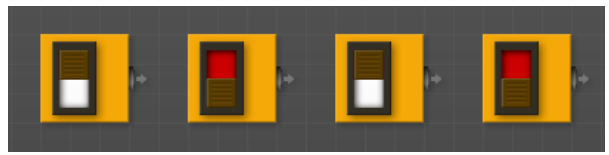


- Example: what is 99 in binary notation?

```
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



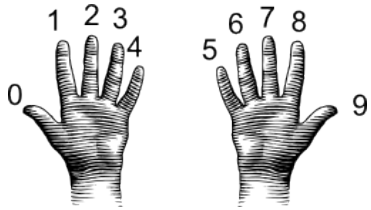


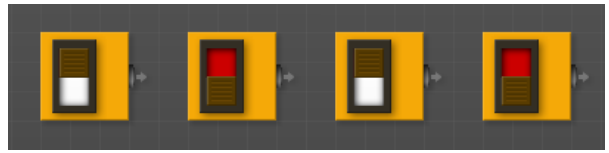
- Example: what is 99 in binary notation?

```
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.

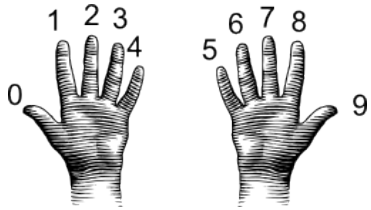
Recap: Converting Between Bases

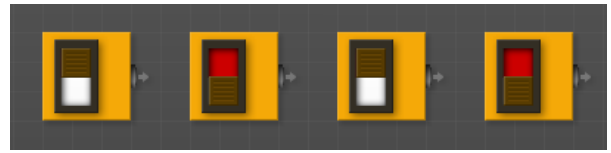




- From binary to decimal:
  - Set sum = last digit.

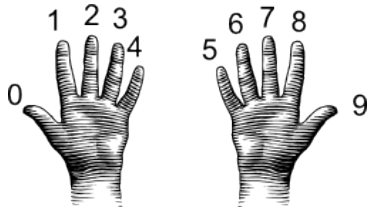
Recap: Converting Between Bases

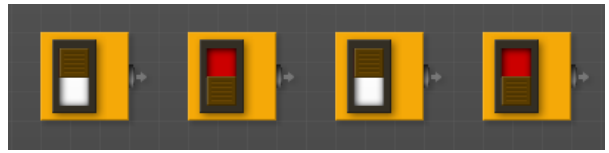




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.

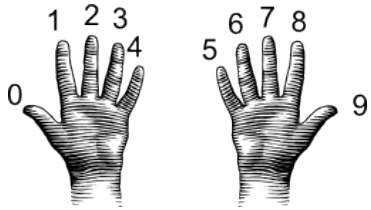
Recap: Converting Between Bases

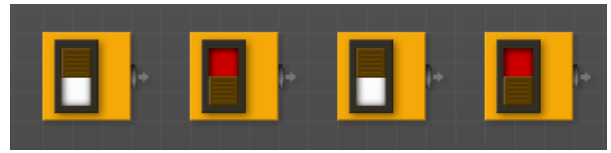




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.

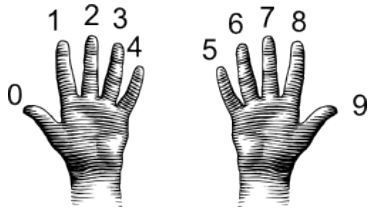
Recap: Converting Between Bases

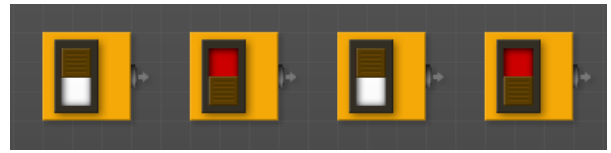




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.

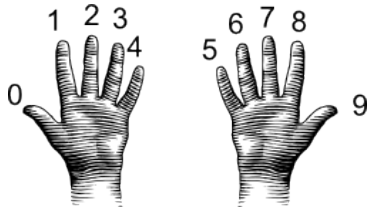
Recap: Converting Between Bases

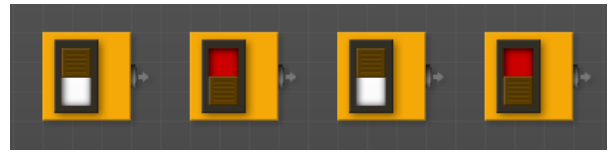




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.

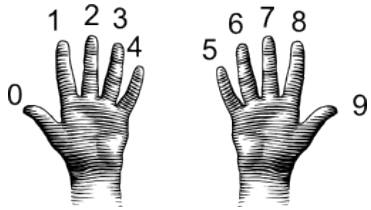
Recap: Converting Between Bases

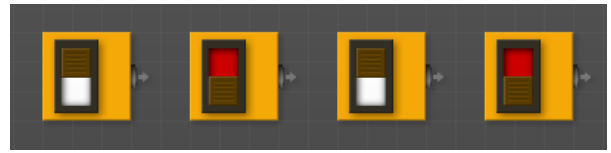




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.

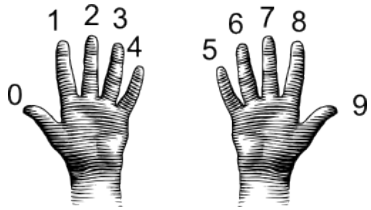
Recap: Converting Between Bases

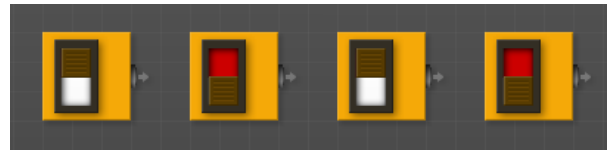




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

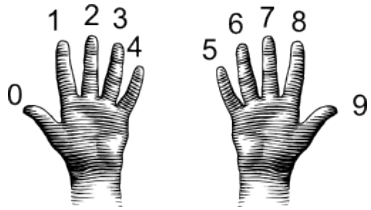
Recap: Converting Between Bases

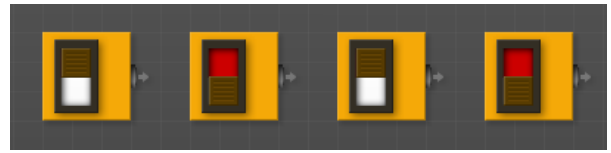




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

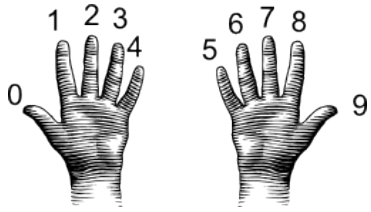
Recap: Converting Between Bases

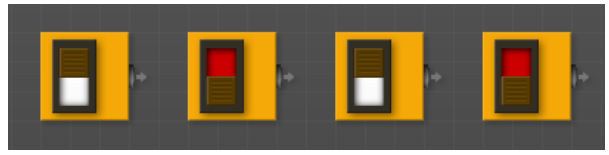




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

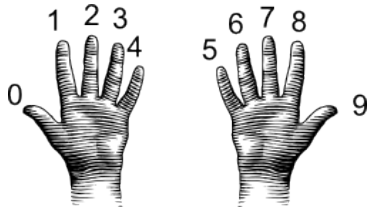


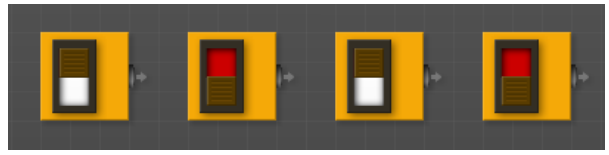


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

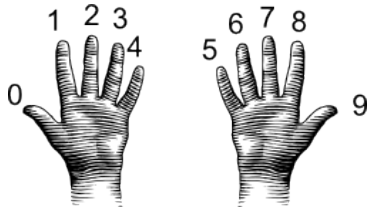


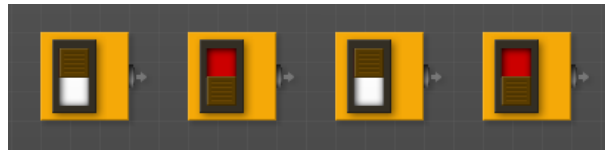


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:
```

Recap: Converting Between Bases

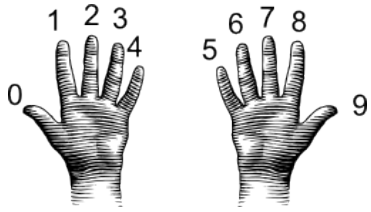


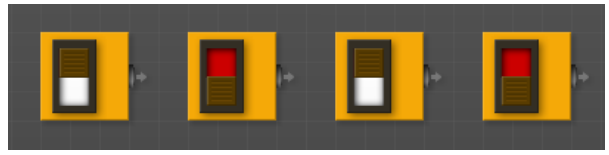


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
```

Recap: Converting Between Bases

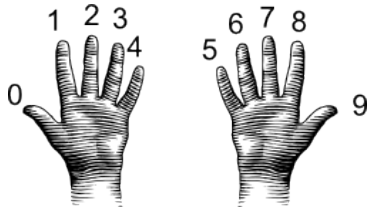


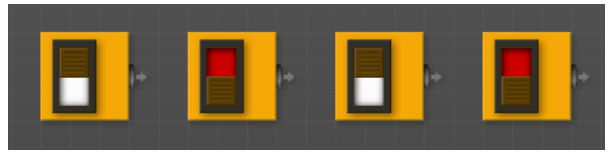


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:
```

Recap: Converting Between Bases

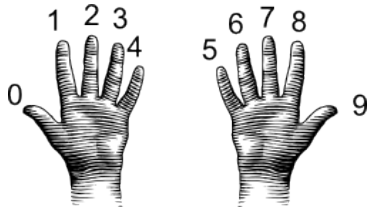


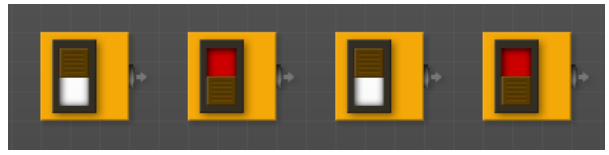


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
```

Recap: Converting Between Bases

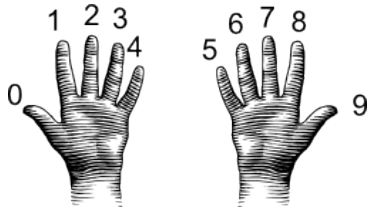


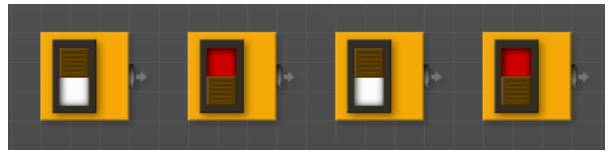


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
1*8 = 8.  Add 8 to sum:
```

Recap: Converting Between Bases

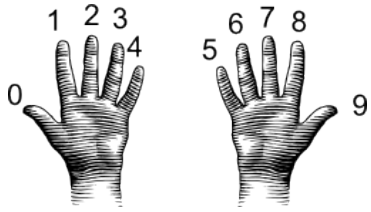


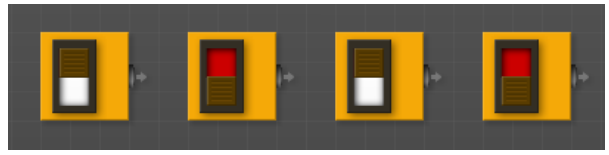


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
1*8 = 8.  Add 8 to sum:       13
```

Recap: Converting Between Bases

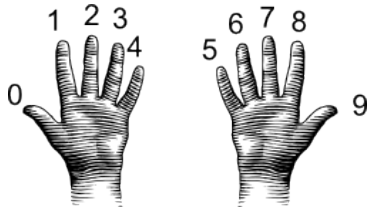


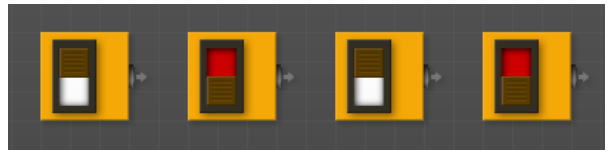


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
1*8 = 8.  Add 8 to sum:       13
1*16 = 16.  Add 16 to sum:
```

Recap: Converting Between Bases

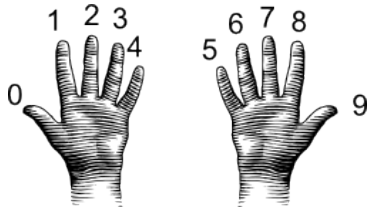


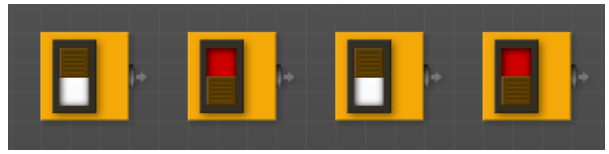


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
1*8 = 8.  Add 8 to sum:       13
1*16 = 16.  Add 16 to sum:    29
```

Recap: Converting Between Bases

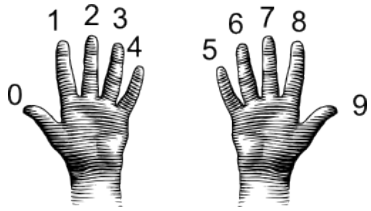


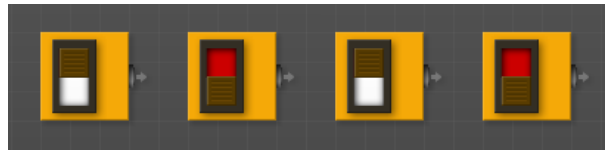


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
1*8 = 8.  Add 8 to sum:       13
1*16 = 16.  Add 16 to sum:    29
1*32 = 32.  Add 32 to sum:
```

Recap: Converting Between Bases

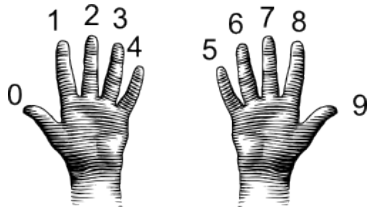


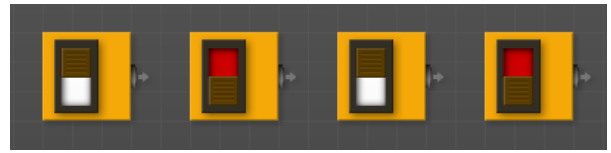


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
1*8 = 8.  Add 8 to sum:       13
1*16 = 16.  Add 16 to sum:    29
1*32 = 32.  Add 32 to sum:    61
```

Recap: Converting Between Bases

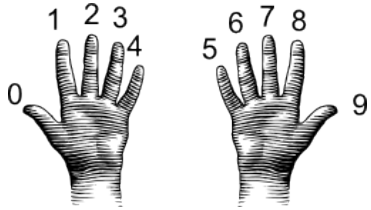


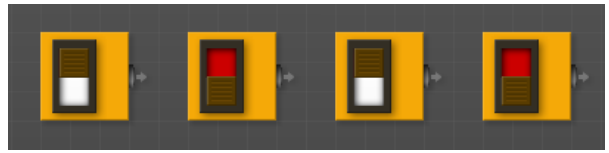


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.
  - 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*2 = 0.  Add 0 to sum:       1
1*4 = 4.  Add 4 to sum:       5
1*8 = 8.  Add 8 to sum:       13
1*16 = 16.  Add 16 to sum:    29
1*32 = 32.  Add 32 to sum:    61
```

Recap: Converting Between Bases

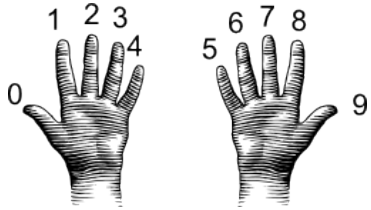


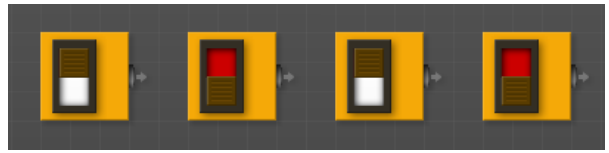


- Example: What is 10100100 in decimal?

  `Sum starts with:`

Recap: Converting Between Bases

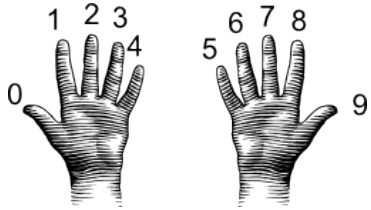


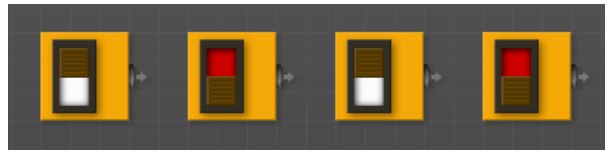


- Example: What is 10100100 in decimal?

```
Sum starts with:                    0
0*2 = 0.  Add 0 to sum:
```

Recap: Converting Between Bases

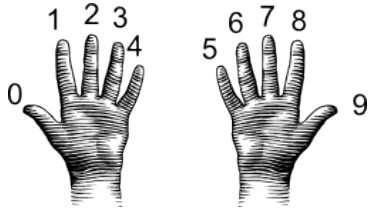


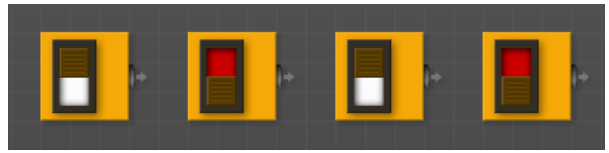


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.  Add 0 to sum:         0
```

Recap: Converting Between Bases

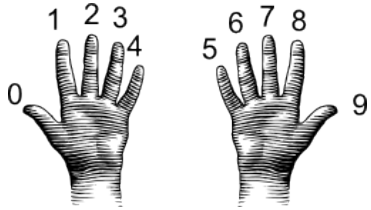


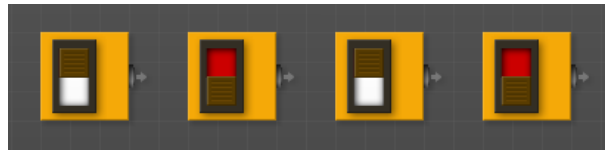


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.  Add 0 to sum:         0
1*4 = 4.  Add 4 to sum:
```

Recap: Converting Between Bases

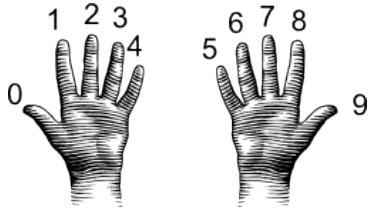


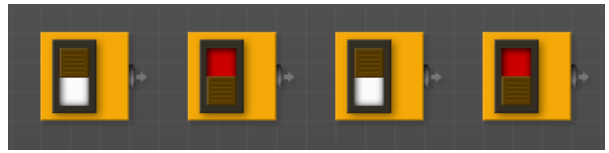


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.  Add 0 to sum:         0
1*4 = 4.  Add 4 to sum:         4
```

Recap: Converting Between Bases

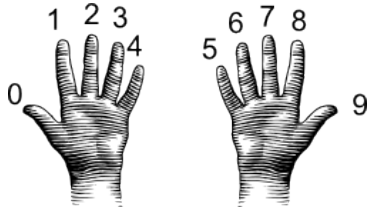


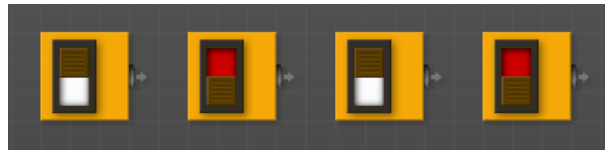


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.  Add 0 to sum:         0
1*4 = 4.  Add 4 to sum:         4
0*8 = 0.  Add 0 to sum:
```

Recap: Converting Between Bases

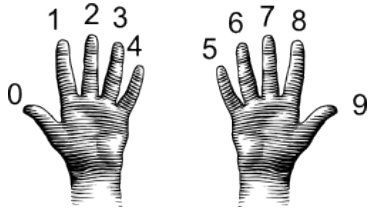


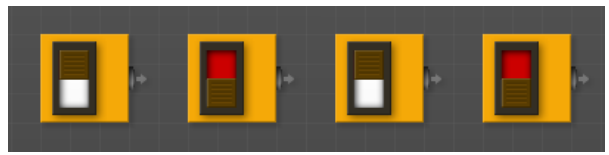


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.  Add 0 to sum:         0
1*4 = 4.  Add 4 to sum:         4
0*8 = 0.  Add 0 to sum:         4
```

Recap: Converting Between Bases

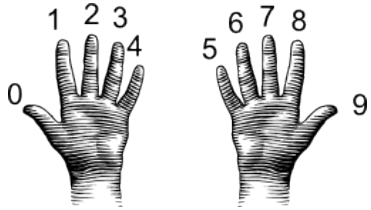


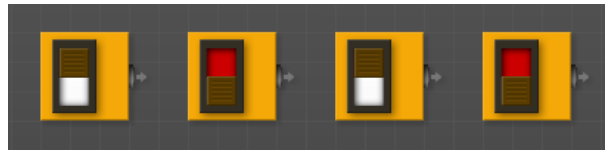


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.  Add 0 to sum:         0
1*4 = 4.  Add 4 to sum:         4
0*8 = 0.  Add 0 to sum:         4
0*16 = 0.  Add 0 to sum:
```

Recap: Converting Between Bases

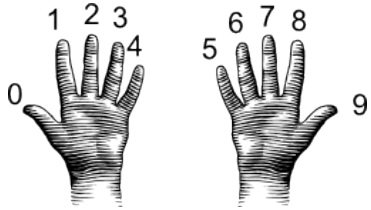


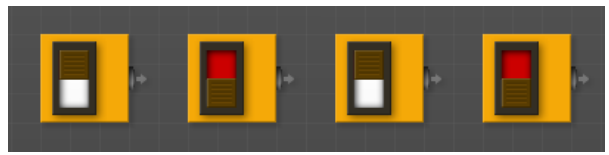


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.   Add 0 to sum:        0
1*4 = 4.   Add 4 to sum:        4
0*8 = 0.   Add 0 to sum:        4
0*16 = 0.   Add 0 to sum:       4
```

Recap: Converting Between Bases

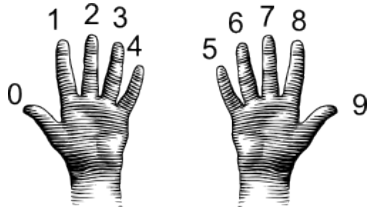


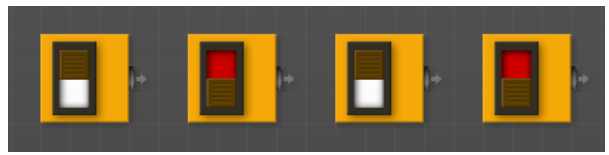


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.  Add 0 to sum:         0
1*4 = 4.  Add 4 to sum:         4
0*8 = 0.  Add 0 to sum:         4
0*16 = 0.  Add 0 to sum:        4
1*32 = 32.  Add 32 to sum:
```

Recap: Converting Between Bases

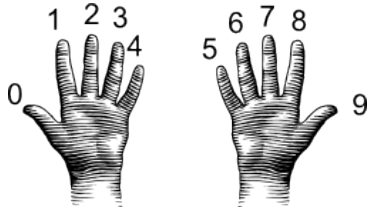


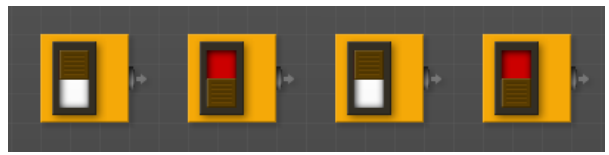


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.   Add 0 to sum:        0
1*4 = 4.   Add 4 to sum:        4
0*8 = 0.   Add 0 to sum:        4
0*16 = 0.   Add 0 to sum:       4
1*32 = 32.   Add 32 to sum:     36
```

Recap: Converting Between Bases

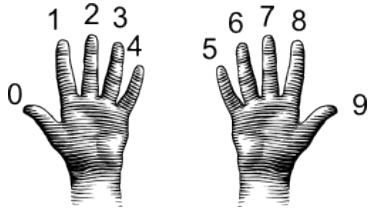


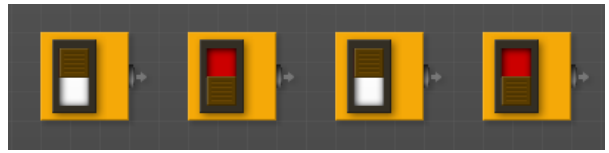


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.   Add 0 to sum:        0
1*4 = 4.   Add 4 to sum:        4
0*8 = 0.   Add 0 to sum:        4
0*16 = 0.   Add 0 to sum:       4
1*32 = 32.   Add 32 to sum:     36
0*64 = 0.   Add 0 to sum:
```

Recap: Converting Between Bases

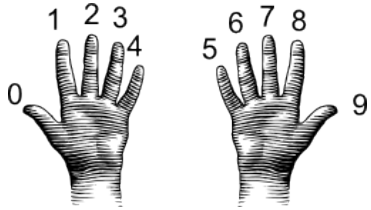


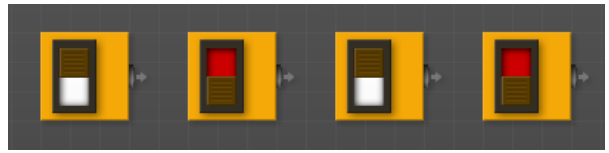


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.   Add 0 to sum:        0
1*4 = 4.   Add 4 to sum:        4
0*8 = 0.   Add 0 to sum:        4
0*16 = 0.   Add 0 to sum:       4
1*32 = 32.   Add 32 to sum:     36
0*64 = 0.   Add 0 to sum:       36
```

Recap: Converting Between Bases

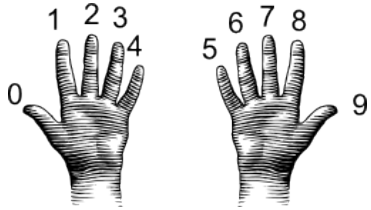


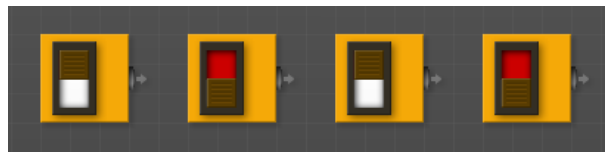


- Example: What is 10100100 in decimal?

```
Sum starts with:                0
0*2 = 0.   Add 0 to sum:        0
1*4 = 4.   Add 4 to sum:        4
0*8 = 0.   Add 0 to sum:        4
0*16 = 0.   Add 0 to sum:       4
1*32 = 32.   Add 32 to sum:     36
0*64 = 0.   Add 0 to sum:       36
1*128 = 0.   Add 128 to sum:
```

Recap: Converting Between Bases

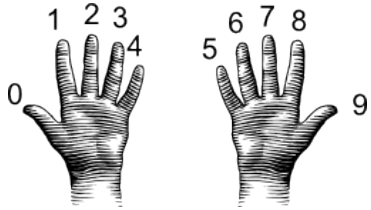


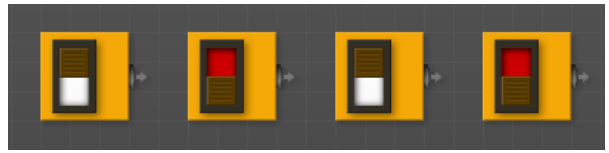


- Example: What is 10100100 in decimal?

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

Recap: Converting Between Bases





- Example: What is 10100100 in decimal?

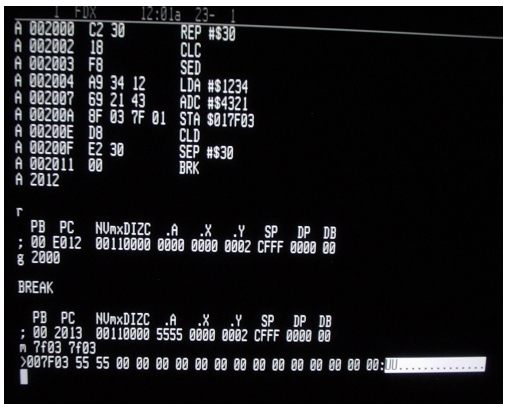
```
Sum starts with:                0
0*2 = 0.   Add 0 to sum:        0
1*4 = 4.   Add 4 to sum:        4
0*8 = 0.   Add 0 to sum:        4
0*16 = 0.   Add 0 to sum:       4
1*32 = 32.   Add 32 to sum:     36
0*64 = 0.   Add 0 to sum:       36
1*128 = 0.   Add 128 to sum:    164
```

The answer is 164.

Today's Topics

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

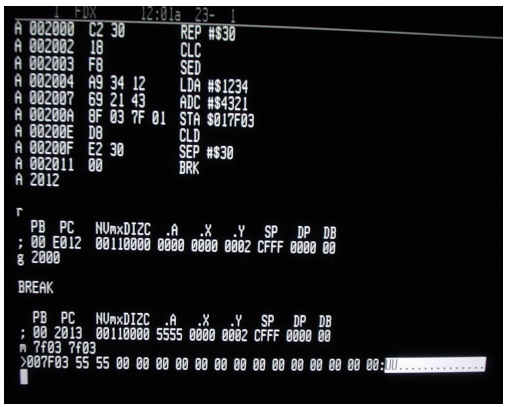
Recap: Machine Language



(wiki)

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

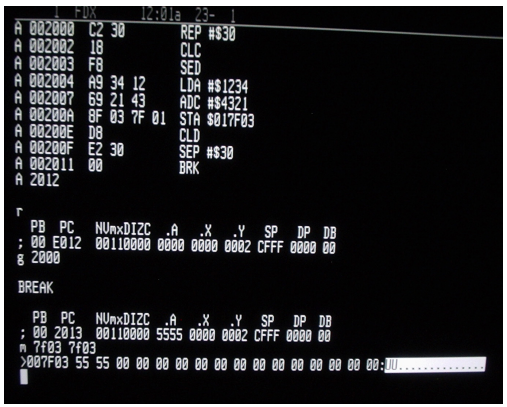
Recap: Machine Language



(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.

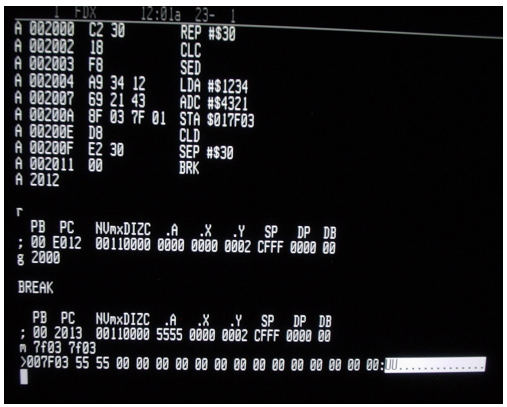
Recap: Machine Language



(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.

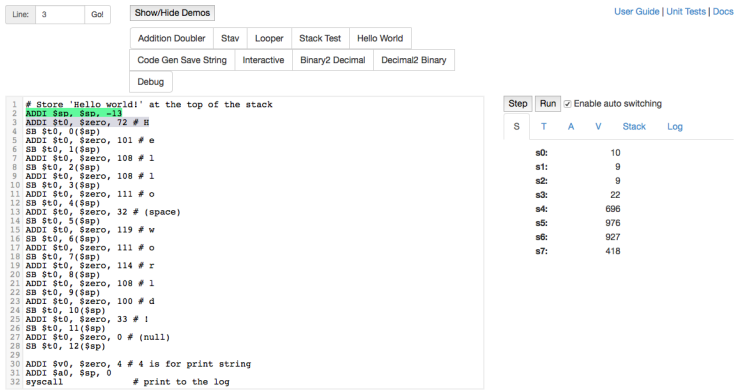
Recap: Machine Language



(wiki)

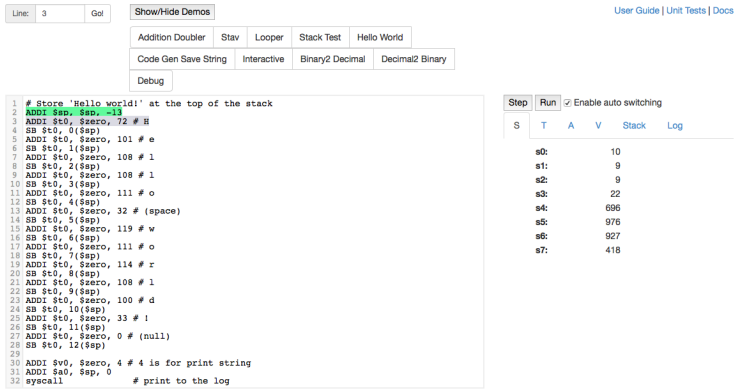
- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.
- Due to its small set of commands, processors can be designed to run those commands very efficiently.
- More in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

Recap: MIPS Commands



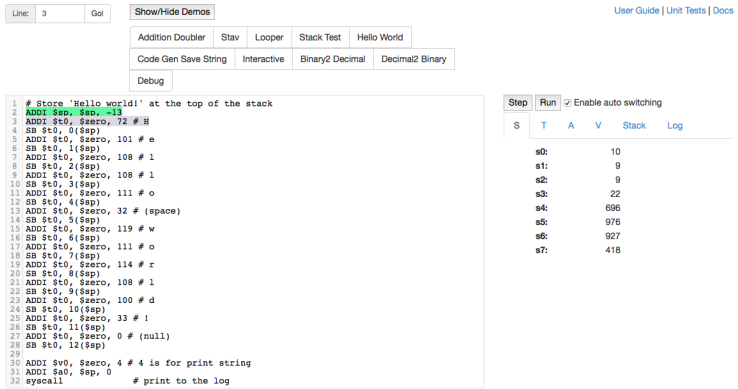
- **Registers:** locations for storing information that can be quickly accessed.

Recap: MIPS Commands



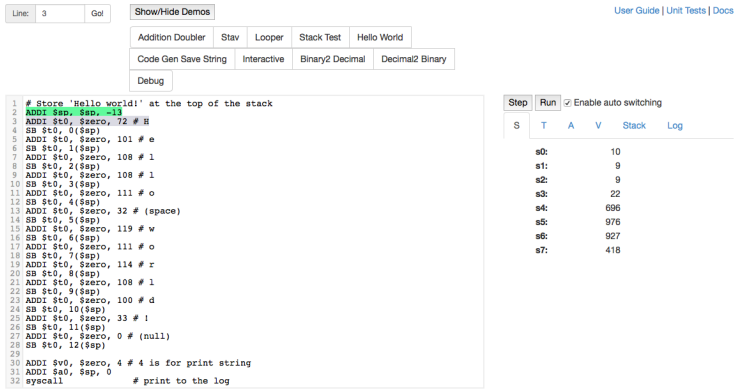
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...

Recap: MIPS Commands



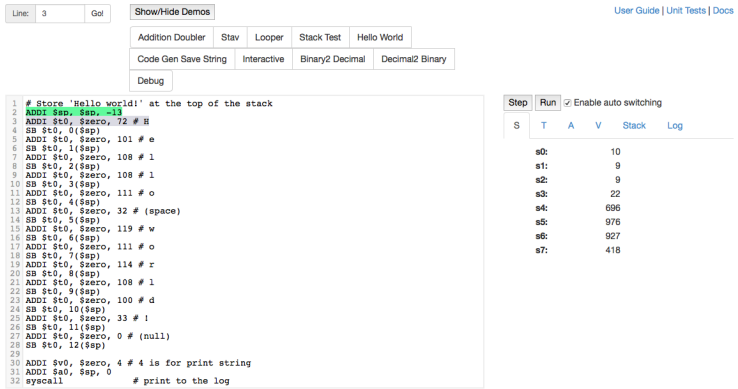
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:

Recap: MIPS Commands



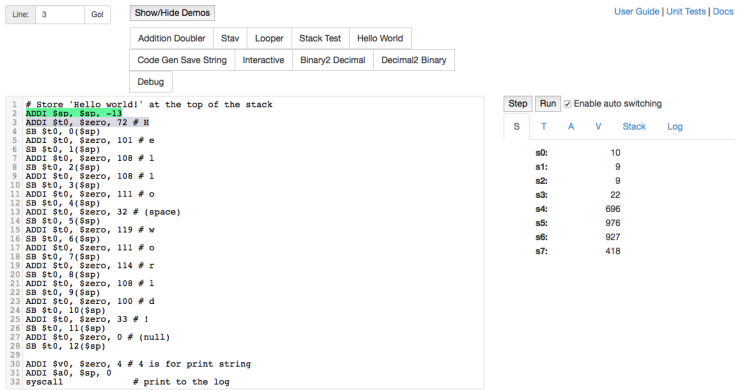
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers: `add $s1, $s2, $s3`

Recap: MIPS Commands



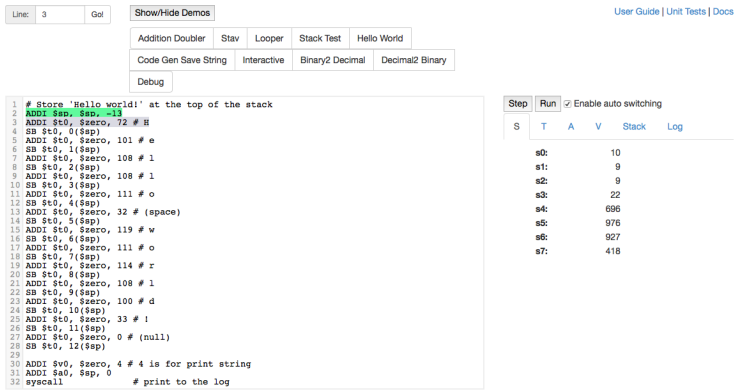
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers: `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

Recap: MIPS Commands



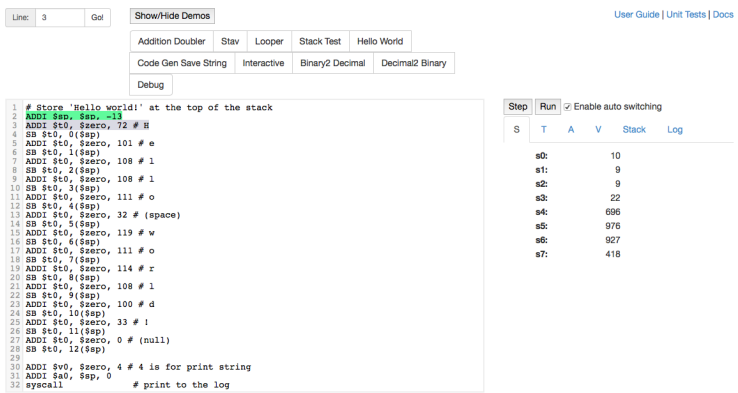
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers: `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

Recap: MIPS Commands



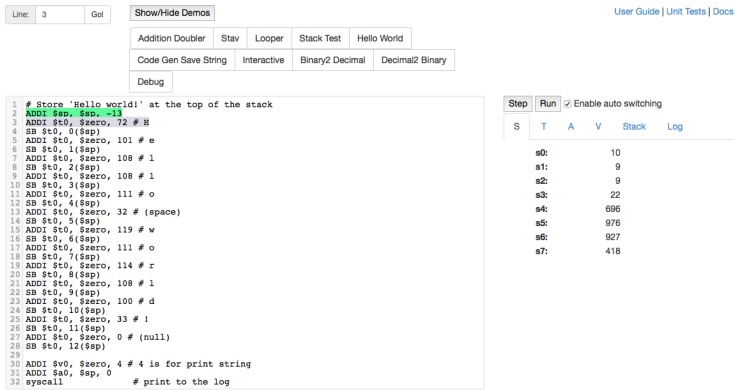
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:
  `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

Recap: MIPS Commands



- **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`

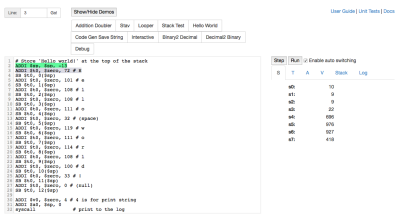
Recap: MIPS Commands



- **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`)

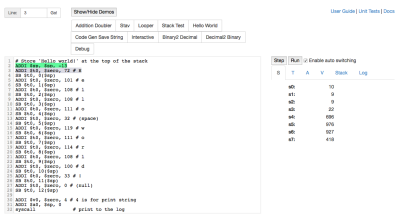
Loops & Jumps in Machine Language

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

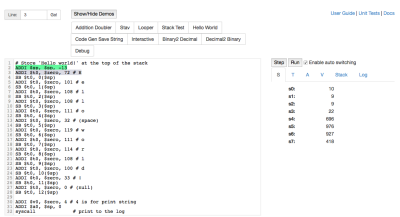


Loops & Jumps in Machine Language

- 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.

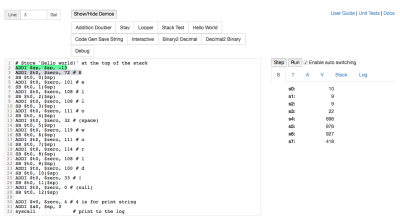


Loops & Jumps in Machine Language



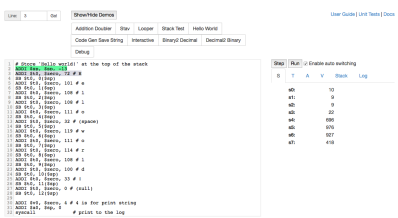
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.

Loops & Jumps in Machine Language



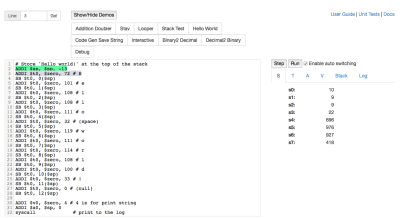
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:

Loops & Jumps in Machine Language



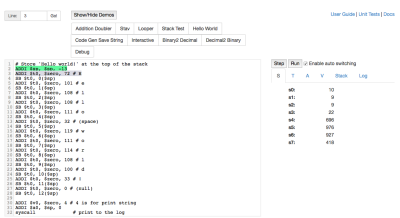
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
  - **Unconditional:**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

Loops & Jumps in Machine Language



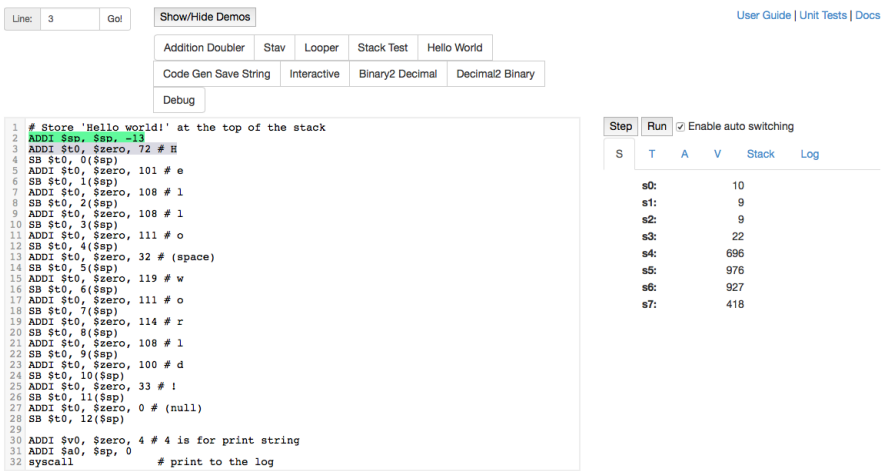
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
  - **Unconditional:**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



(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():

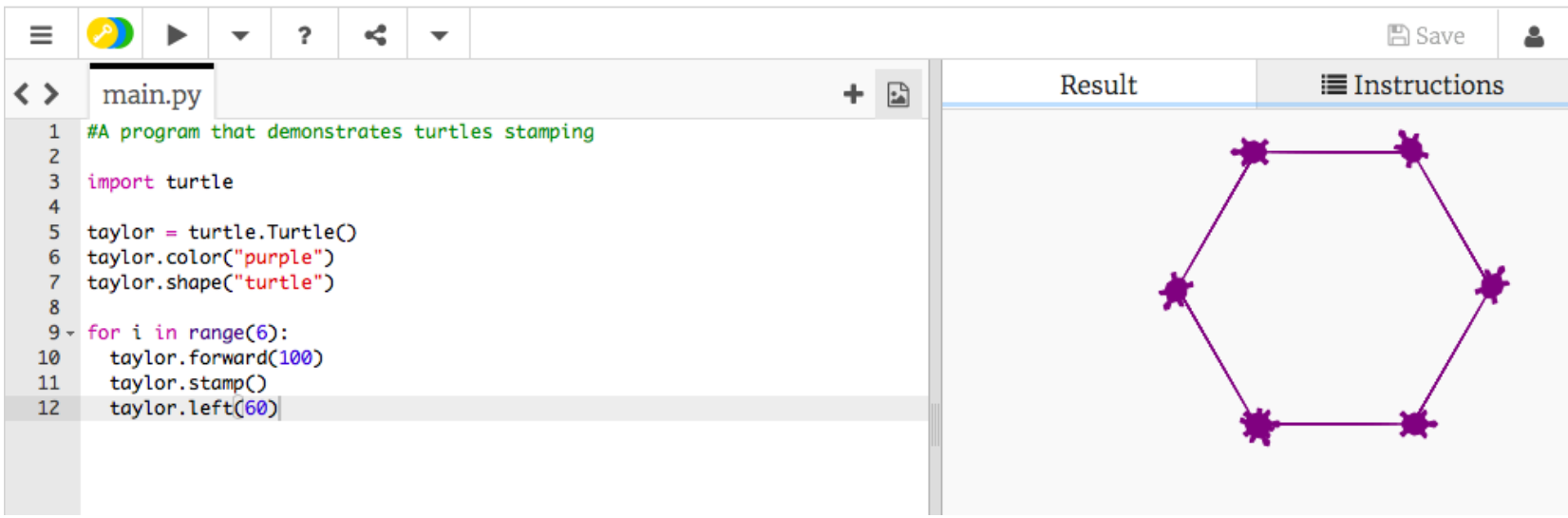
```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)

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.

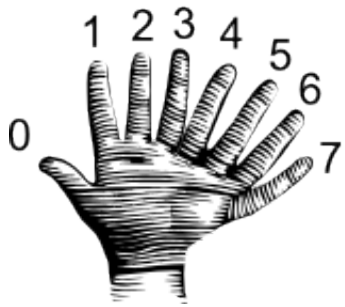
Week 2: variables, data types, more on loops & range()

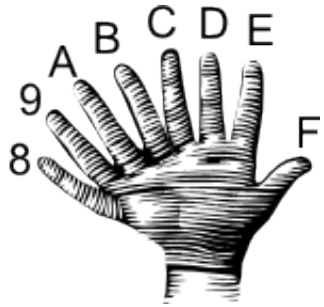
- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
  - **int**: integer or whole numbers
  - **float**: floating point or real numbers
  - **string**: sequence of characters
  - **list**: a sequence of items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

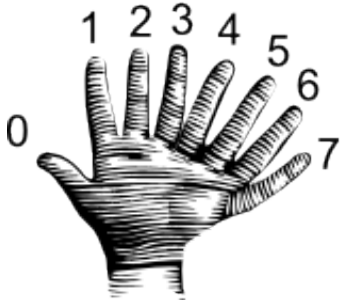
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

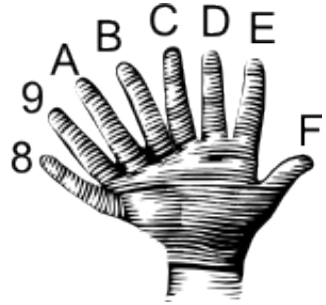


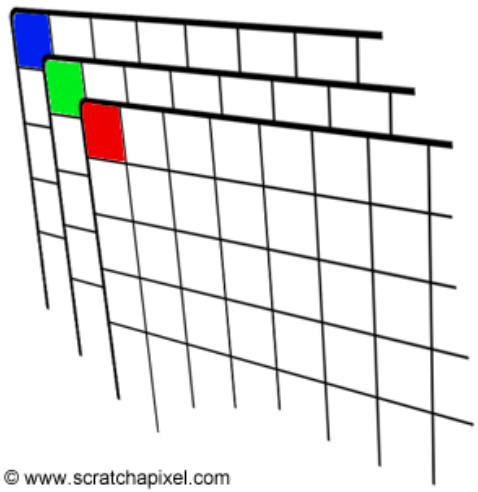


Week 3: colors, hex, slices, numpy & images

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

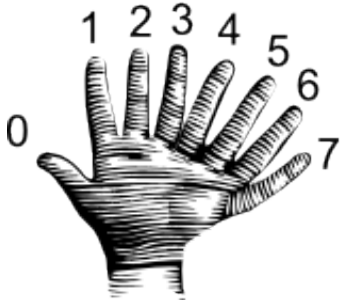


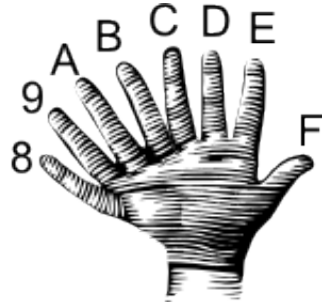


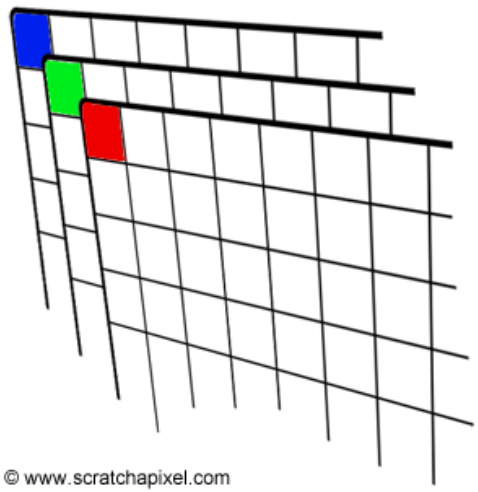


Week 3: colors, hex, slices, numpy & images

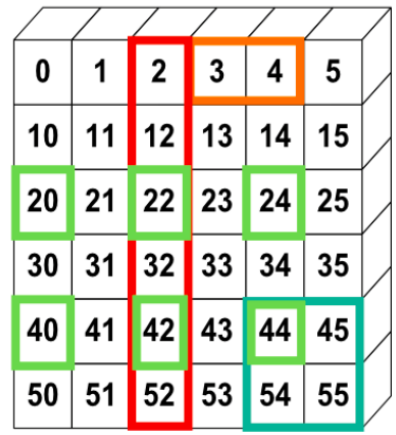
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



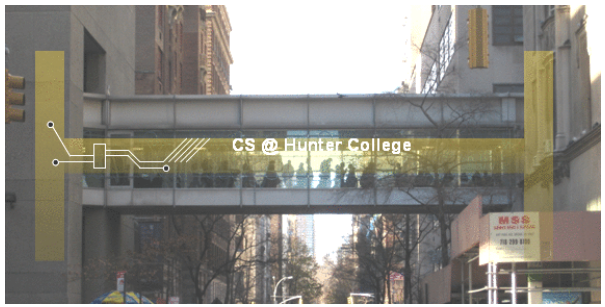


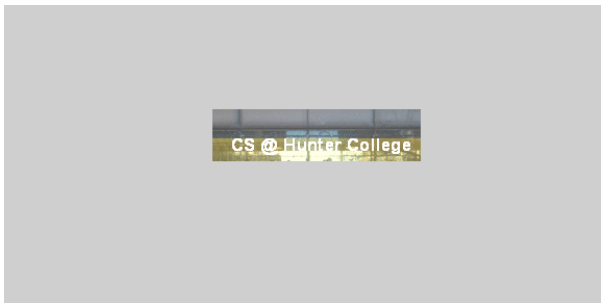



```
>>> a[0,3:5]
array([3,4])

>>> 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]])
```

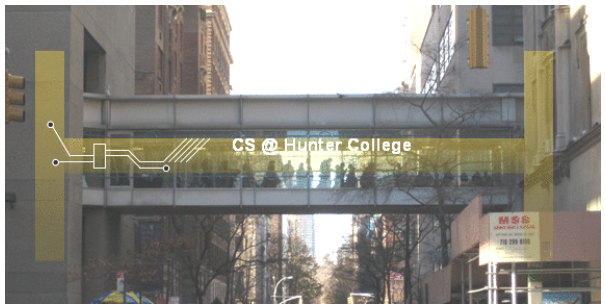


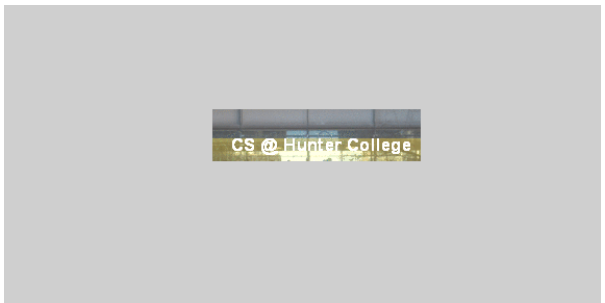
Week 4: design problem (cropping images) & decisions





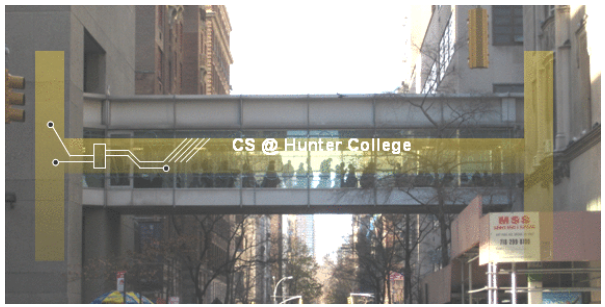
Week 4: design problem (cropping images) & decisions





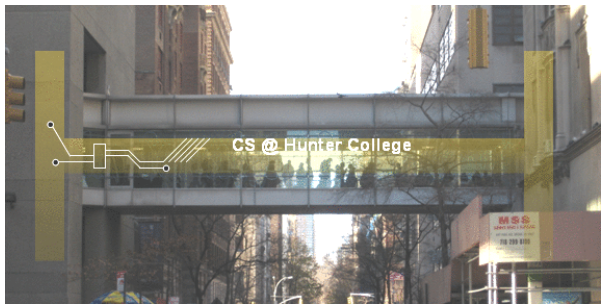
- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*

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')
```

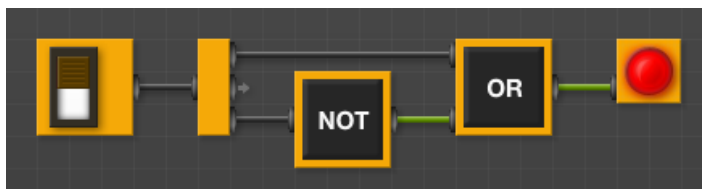
Week 5: logical operators, truth tables & logical circuits

```
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
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

`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
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

`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
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

nycHistPop.csv

In Lab 6

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
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

nycHistPop.csv

In Lab 6

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

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

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

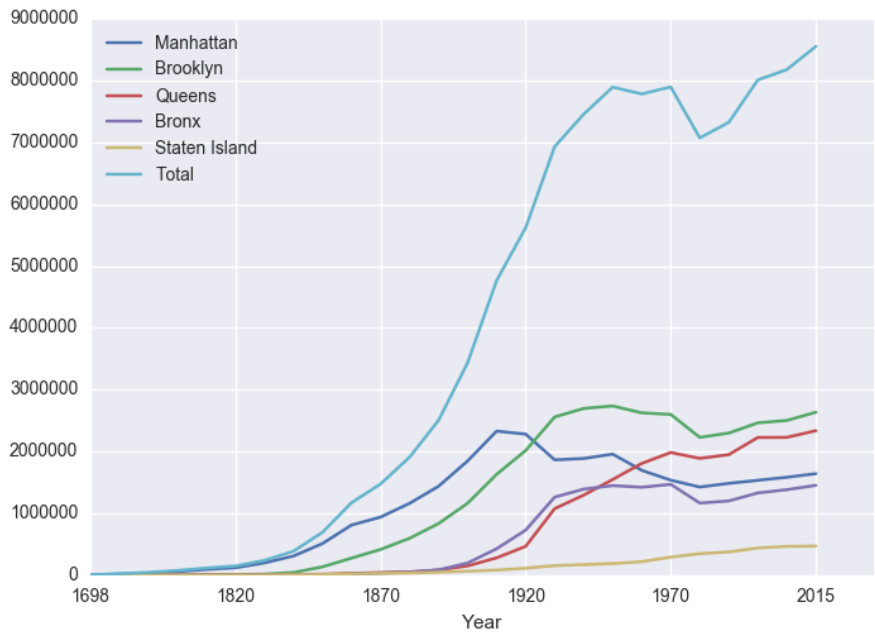
Week 6: structured data, pandas, & more design

```
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
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

nycHistPop.csv

In Lab 6

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

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

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



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.

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

```
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).

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**.

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

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)
```

- 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)
```

- 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)
```

- 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

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.
- 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/hashes 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:

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.

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

Show sorting demo.

Lecture Slip: Design Patterns

In pairs or triples:

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

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).

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)
```