

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From email and tutoring.

- When is the final? Is there a review sheet?

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, December 19, 9-11am.

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, December 19, 9-11am.

The early final exam (alternative date) is on Friday, December 16, 8-10am.

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, December 19, 9-11am.

The early final exam (alternative date) is on Friday, December 16, 8-10am.

Instead of a review sheet, we have:

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, December 19, 9-11am.

The early final exam (alternative date) is on Friday, December 16, 8-10am.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, December 19, 9-11am.

The early final exam (alternative date) is on Friday, December 16, 8-10am.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *UTAs in drop-in tutoring happy to review concepts and old exam questions.*

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, December 19, 9-11am.

The early final exam (alternative date) is on Friday, December 16, 8-10am.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *UTAs in drop-in tutoring happy to review concepts and old exam questions.*
- ▶ *There will be opportunity for practice during our last meeting on 13 December.*

Handle Exam Anxiety – courtesy Dr. St. John

- Print out the past exams and do as much as possible in **1 hour**.
- Then grade yourselves, figure out which problems are similar to past problems, keeping all the exams you've done in a 3-hole notebook, 1 problem per page, organized by problem number, reinforces the similarity.
- Make a list of what does not make sense and asking the instructor.
- Attempting to do the exam in half the time means that in the real exam, you will have plenty of time.

Today's Topics



- Design Patterns: Searching
- Python Recap
- Machine Language
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

Today's Topics



- **Design Patterns: Searching**
- Python Recap
- Machine Language
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

Predict what the code will do:

```
1 def search(nums, locate):
2     found = False
3     i = 0
4     while not found and i < len(nums):
5         print(nums[i])
6         if locate == nums[i]:
7             found = True
8         else:
9             i = i+1
10
11 return(found)
```

Predict what the code will do: II

```
11 nums= [1,4,10,6,5,42,9,8,12]
12 target = 6
13 if search(nums, target):
14     print(target, 'is in the list.')
15 else:
16     print(target, 'is not in the list.')
```

Simplified but a little tricky

```
1 def search(nums, locate):
2     i = 0
3     while i < len(nums) and locate!=nums[i] :
4         print(nums[i])
5         i = i+1
6
7     return (i < len(nums))
8 #If locate is in the list,
9 #then for some i < len(nums), we have
10 #locate == nums[i].
11 #If i >= len(nums), this implies that all
12 #items are searched, no match is found.
```

Simplified but a little tricky

```
13 nums= [1,4,10,6,5,42,9,8,12]  
14 target = 6  
15 if search(nums, target):  
16     print(target, 'is in the list.')  
17 else:  
18     print(target, 'is not in the list.')
```

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
target = 6
if search(nums,target):
    print(target, 'is in the list!')
else:
    print(target, 'is not in the list.')
```

- Example of **linear search**.

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
target = 6
if search(nums,target):
    print(target, 'is in the list!')
else:
    print(target, 'is not in the list.')
```

- Example of **linear search**.
- Start at the beginning of the list.

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
target = 6
if search(nums,target):
    print(target, 'is in the list!')
else:
    print(target, 'is not in the list.')
```

- Example of **linear search**.
- Start at the beginning of the list.
- Look at each item, one-by-one.

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
target = 6
if search(nums,target):
    print(target, 'is in the list!')
else:
    print(target, 'is not in the list.')
```

- Example of **linear search**.
- Start at the beginning of the list.
- Look at each item, one-by-one.
- Stop when found, or the end of list reached.

Today's Topics



- Design Patterns: Searching
- **Python Recap**
- Machine Language
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic

Python & Circuits Review: 10 Weeks in 10 Minutes



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

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

```
1 #Texts following # are comments.  
2 #Comments are read by human beings, not  
   computer.  
3 #Name: Thomas Hunter  
4 #Date: September 1, 2017  
5 #This program prints: Hello, World!  
6  
7 print("Hello, World!")
```

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

```
1 import turtle  
2  
3 taylor = turtle.Turtle()  
4 taylor.color("purple")  
5 taylor.shape("turtle")  
6  
7 n = 6  
8 for i in range(n):  
9     taylor.forward(100)  
10    taylor.stamp()  
11    taylor.left(360/n)
```

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()

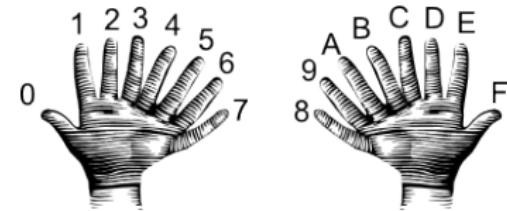
- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items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

Examples on loop and ranges

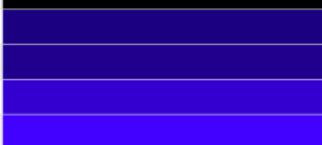
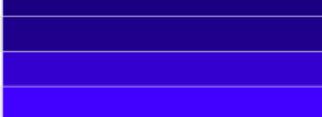
```
1  for num in [2,4,6,8,10]:  
2      print(num)  
  
3  
  
4  sum = 0  
5  for x in range(0,12,2):  
6      print(x)  
7      sum += x  
  
8  
  
9  print(sum)  
  
10  
  
11 for c in 'ABCD':  
12     print(c)
```

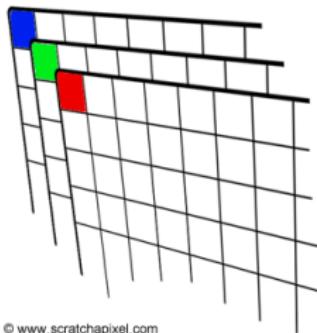
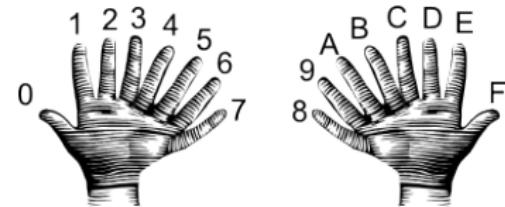
Week 3: colors, hex, slices, numpy & images

| 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 |  |



Two Dimensional Array Slicing

```
1 import numpy as np  
2  
3 numRows = 6  
4 numCols = 6  
5 a = np.zeros((numRows, numCols))  
6 #create a table with 6 rows and 6 columns,  
7 #each element is initialized to be zero.  
8 #Do not forget parentheses around  
9 #numRows, numCols.
```

Two Dimensional Array Slicing: II

```
8  for i in range(numRows):
9      for j in range(numCols):
10         a[i, j] = i*10 + j
11 #range(numRows) returns [0, 1, 2, 3, 4, 5],
12 #where outer loop variable i chooses from.
13 #When i is 0, run
14 #     for j in range(numCols):
15 #         a[i, j] = i*10 + j
16 #When i is 1, run
17 #     for j in range(numCols):
18 #         a[i, j] = i*10 + j
19 #The last round of i is 5.
```

Two Dimensional Array Slicing: III

```
20 for i in range numRows:
21     for j in range numCols:
22         print("%3i"%(a[i, j]), end="")
23         #"%3i"%(a[i, j]) prints a[i, j] --
24         #element of a at ith row and
25         #jth column -- as an 3-digit int.
26         #"%3i" is a place holder and is
27         #filled by a[i, j].
28         #If a[i, j] does not have 3 digits,
29         #pad space(s) to the left.
30         #end="" print w/o a new line.
31
32     print() #print a new line after each row
```

Two Dimensional Array Slicing: III

32

```
print(a[0, 3:5])
```

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

Two Dimensional Array Slicing: III

32

```
print(a[0, 3:5])
```

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

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

print

[3. 4.]

Two Dimensional Array Slicing: IV

33

```
print(a[4:, 4:])
```

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

Two Dimensional Array Slicing: IV

33

```
print(a[4:, 4:])
```

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

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

Print out

```
[[44, 45],  
 [54, 55]]
```

Two Dimensional Array Slicing: V

34

```
print(a[:, 2])
```

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

Two Dimensional Array Slicing: V

34

```
print(a[:, 2])
```

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

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

Print out

```
[ 2. 12. 22. 32. 42. 52.]
```

Two Dimensional Array Slicing: VI

35

```
print(a[2::2, ::2])
```

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

Two Dimensional Array Slicing: VI

35

```
print(a[2::2, ::2])
```

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

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

```
print
```

```
[[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.

Grayed surrounding area of an image

```
1 #Leave middle 1/5 height * 1/2 width  
  unchanged,  
2 #gray the rest area of the image  
3 import matplotlib.pyplot as plt  
4 import numpy as np  
5  
6 fileName = input("Enter a file name: ")  
7 img = plt.imread(fileName)  
8  
9 height = img.shape[0]  
10 width = img.shape[1]
```

Grayed surrounding area of an image: //

```
11 #make the top 2/5 area gray
12 img[:height*2//5, :] = [0.5, 0.5, 0.5, 1]
13 #[0.5, 0.5, 0.5, 1] means
14 #red 0.5, green 0.5, blue 0.5 and opacity 1
15
16 #make the bottom 1/4 area gray
17 img[-height*2//5:, :] = [0.5, 0.5, 0.5, 1]
18 #img[-height*2//5:, :] same as
19 #img[height*3//5:, :]
20 ##height*2//5 from BOTTOM same as
21 #height*3//5 from TOP
22 #img[height*3//5:, :] = [0.5, 0.5, 0.5, 1]
```

Grayed surrounding area of an image: III

```
23 #make the left 1/4 area gray  
24 img[:, :width//4] = [0.5, 0.5, 0.5, 1]  
25  
26 #make the right 1/4 area gray  
27 img[:, width*3//4:] = [0.5, 0.5, 0.5, 1]  
28 #img[:, width*3//4:] same as  
29 #img[:, -width//4:]  
30 #width*3//4 from LEFT same as  
31 #width//4 from RIGHT  
32 #img[:, -width//4:] = [0.5, 0.5, 0.5, 1]  
33  
34 plt.imshow(img)  
35 plt.show()
```

Highlight part of image

```
1 #Leave middle 1/5 height * 1/2 width  
2 #section unchanged,  
3 #dim the rest area of the image  
4 #It is like to highlight the middle part.  
5  
6 import matplotlib.pyplot as plt  
7 import numpy as np  
8  
9 fileName = "csBridge.png"  
10 #fileName = input("Enter a file name: ")  
11 img = plt.imread(fileName)
```

Highlight part of image: II

```
12 height = img.shape[0]
13 width = img.shape[1]
14
15 #make the top 1/4 area gray
16 #img[:height*2//5, :] = [0.5, 0.5, 0.5, 1]
17 #[0.5, 0.5, 0.5, 1] means
18 #red 0.5, green 0.5, blue 0.5 and opacity 1
19 #unlike user created stripe images,
20 #some images have four channels.
21
22 #dim the top 1/4 area
23 img[:height*2//5, :, 3] = 0.5
```

Highlight part of image: III

```
24 #dim the bottom 1/4 area  
25 img[-height*2//5:, :, 3] = 0.5  
26 #img[-height*2//5:, :, 3] same as img[height  
27 *3//5:, :, 3]  
28 #height*2//5 from BOTTOM same as height*3//5  
29 from TOP  
30  
31 #dim the left 1/4 area  
32 img[:, :width//4, 3] = 0.5
```

Highlight part of image: IV

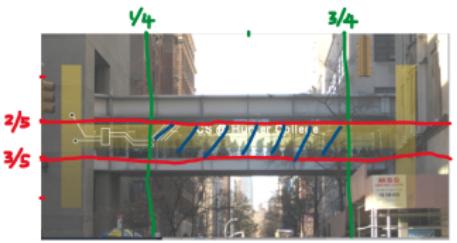
```
31 #dim the right 1/4 area  
32 img[:, width*3//4:, 3] = 0.5  
33 #img[:, width*3//4:, 3] same as  
34 #img[:, -width//4:, 3]  
35 #width*3//4 from LEFT same as  
36 #width//4 from RIGHT  
37  
38 plt.imshow(img)  
39 plt.show()
```

Crop image

```
1 #crop middle 1/5 * height * 1/2 * width area  
2 import matplotlib.pyplot as plt  
3 import numpy as np  
4  
5 fileName = "csBridge.png"  
6 #fileName = input("Enter a file name: ")  
7 img = plt.imread(fileName)  
8  
9 height = img.shape[0]  
10 width = img.shape[1]
```

Crop image: II

```
1 img2 = img[height*2//5 : height*3//5, width  
2      *1//4 : width*3//4] #// cannot be replaced  
3      by /, indices need to be int.  
4  
5  
6 plt.imshow(img2)  
plt.show()  
  
plt.imsave('cropped_image.png', img2)
```



Week 4: design problem (cropping images) & decisions

```
1 yearBorn = int(input("Enter year born: "))
2 if yearBorn < 1946:
3     print("Greatest Generation")
4 elif yearBorn <= 1964:
5     print("Baby Boomer")
6 elif yearBorn <= 1984:
7     print("Generation X")
8 elif yearBorn <= 2004:
9     print("Millennial")
10 else:
11     print("TBD")
```

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

```
1 x = int(input("Enter number: "))
2
3 if x % 2 == 0:
4     print("Even number")
5 else:
6     print("Odd number")
```

Week 5: logical operators, truth tables & logical circuits

```
1 origin = "Indian Ocean"
2 winds = 100
3 if winds >= 74:
4     print("Major storms, called a ", end="")
5     if origin == "Indian Ocean" or origin == "
6         South Pacific":
7         print("cyclone.")
8     elif origin == "North Pacific":
9         print("typhoon.")
10    else:
11        print("hurricane.")
```

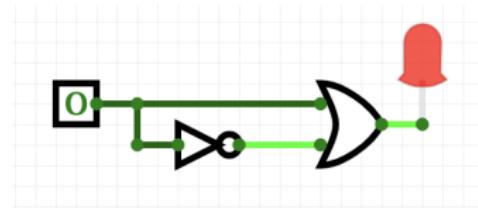
Week 5: logical operators, truth tables & logical circuits: II

```
1 visibility = 0.2
2 winds = 40
3 conditions = "blowing snow"
4 if (winds > 35) and (visibility < 0.25) and
5   (conditions == "blowing snow" or
6     conditions == "heavy snow"):
    print("Blizzard!")
```

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

III

| in1 | and | in2 | <i>returns:</i> |
|-------|-----|-------|-----------------|
| 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
1890,1,230,203,727,784,2,230
1891,1,230,203,727,784,2,230
1892,1,230,203,727,784,2,230
1893,1,230,203,727,784,2,230
1894,1,230,203,727,784,2,230
1895,1,230,203,727,784,2,230
1896,1,230,203,727,784,2,230
1897,1,230,203,727,784,2,230
1898,1,230,203,727,784,2,230
1899,1,230,203,727,784,2,230
1900,1,230,203,727,784,2,230
1901,1,230,203,727,784,2,230
1902,1,230,203,727,784,2,230
1903,1,230,203,727,784,2,230
1904,1,230,203,727,784,2,230
1905,1,230,203,727,784,2,230
1906,1,230,203,727,784,2,230
1907,1,230,203,727,784,2,230
1908,1,230,203,727,784,2,230
1909,1,230,203,727,784,2,230
1910,1,230,203,727,784,2,230
1911,1,230,203,727,784,2,230
1912,1,230,203,727,784,2,230
1913,1,230,203,727,784,2,230
1914,1,230,203,727,784,2,230
1915,1,230,203,727,784,2,230
1916,1,230,203,727,784,2,230
1917,1,230,203,727,784,2,230
1918,1,230,203,727,784,2,230
1919,1,230,203,727,784,2,230
1920,1,230,203,727,784,2,230
1921,1,230,203,727,784,2,230
1922,1,230,203,727,784,2,230
1923,1,230,203,727,784,2,230
1924,1,230,203,727,784,2,230
1925,1,230,203,727,784,2,230
1926,1,230,203,727,784,2,230
1927,1,230,203,727,784,2,230
1928,1,230,203,727,784,2,230
1929,1,230,203,727,784,2,230
1930,1,230,203,727,784,2,230
1931,1,230,203,727,784,2,230
1932,1,230,203,727,784,2,230
1933,1,230,203,727,784,2,230
1934,1,230,203,727,784,2,230
1935,1,230,203,727,784,2,230
1936,1,230,203,727,784,2,230
1937,1,230,203,727,784,2,230
1938,1,230,203,727,784,2,230
1939,1,230,203,727,784,2,230
1940,1,230,203,727,784,2,230
1941,1,230,203,727,784,2,230
1942,1,230,203,727,784,2,230
1943,1,230,203,727,784,2,230
1944,1,230,203,727,784,2,230
1945,1,230,203,727,784,2,230
1946,1,230,203,727,784,2,230
1947,1,230,203,727,784,2,230
1948,1,230,203,727,784,2,230
1949,1,230,203,727,784,2,230
1950,1,230,203,727,784,2,230
1951,1,230,203,727,784,2,230
1952,1,230,203,727,784,2,230
1953,1,230,203,727,784,2,230
1954,1,230,203,727,784,2,230
1955,1,230,203,727,784,2,230
1956,1,230,203,727,784,2,230
1957,1,230,203,727,784,2,230
1958,1,230,203,727,784,2,230
1959,1,230,203,727,784,2,230
1960,1,230,203,727,784,2,230
1961,1,230,203,727,784,2,230
1962,1,230,203,727,784,2,230
1963,1,230,203,727,784,2,230
1964,1,230,203,727,784,2,230
1965,1,230,203,727,784,2,230
1966,1,230,203,727,784,2,230
1967,1,230,203,727,784,2,230
1968,1,230,203,727,784,2,230
1969,1,230,203,727,784,2,230
1970,1,230,203,727,784,2,230
1971,1,230,203,727,784,2,230
1972,1,230,203,727,784,2,230
1973,1,230,203,727,784,2,230
1974,1,230,203,727,784,2,230
1975,1,230,203,727,784,2,230
1976,1,230,203,727,784,2,230
1977,1,230,203,727,784,2,230
1978,1,230,203,727,784,2,230
1979,1,230,203,727,784,2,230
1980,1,230,203,727,784,2,230
1981,1,230,203,727,784,2,230
1982,1,230,203,727,784,2,230
1983,1,230,203,727,784,2,230
1984,1,230,203,727,784,2,230
1985,1,230,203,727,784,2,230
1986,1,230,203,727,784,2,230
1987,1,230,203,727,784,2,230
1988,1,230,203,727,784,2,230
1989,1,230,203,727,784,2,230
1990,1,230,203,727,784,2,230
1991,1,230,203,727,784,2,230
1992,1,230,203,727,784,2,230
1993,1,230,203,727,784,2,230
1994,1,230,203,727,784,2,230
1995,1,230,203,727,784,2,230
1996,1,230,203,727,784,2,230
1997,1,230,203,727,784,2,230
1998,1,230,203,727,784,2,230
1999,1,230,203,727,784,2,230
2000,1,230,203,727,784,2,230
2001,1,230,203,727,784,2,230
2002,1,230,203,727,784,2,230
2003,1,230,203,727,784,2,230
2004,1,230,203,727,784,2,230
2005,1,230,203,727,784,2,230
2006,1,230,203,727,784,2,230
2007,1,230,203,727,784,2,230
2008,1,230,203,727,784,2,230
2009,1,230,203,727,784,2,230
2010,1,230,203,727,784,2,230
2011,1,230,203,727,784,2,230
2012,1,230,203,727,784,2,230
2013,1,230,203,727,784,2,230
2014,1,230,203,727,784,2,230
2015,1,230,203,727,784,2,230

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.....
Five census after the consolidation of the five boroughs.....
.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1890,4937,2037,727,788,1000
1891,21843,36231,2847,28423
1890,33131,4549,6159,1781,3827,49447
1890,60515,5740,6442,1755,4543,7595
1890,112306,11187,8246,2792,6135,152056
1890,123704,11187,8246,2792,6135,152056
1890,202589,20535,9049,3023,7082,242278
1840,31510,12013,14000,5346,10965,39114
1850,35549,12800,14895,5346,10965,39114
1860,613469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59940,5653,51980,33029,1911803
1890,123704,11187,8246,2792,6135,152056
1900,185093,116582,152999,200567,67621,2437202
1910,223342,1634351,284041,430980,8569,4766803
1920,2233103,2018354,446070,446070,73201,11651,50048
1930,1867132,1297634,1394711,1380108,47454995,4930446
1940,1889924,2498285,1297634,1394711,1374441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1660101,2738175,1550949,1451277,191555,7991957
1970,1660101,2738175,1550949,1451277,191555,7991984
1970,1660101,2738175,1550949,1451277,191555,7991984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2803664,1951598,1203789,372977,7322564
2000,1537195,2485326,2223379,1332450,419728,8080879
2010,1583873,2504705,2272722,1385108,4175133,8175133
2010,1444018,2646733,2339150,1459446,474558,8059405

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,Bronx,Brooklyn,Queens,Bronx,Staten Island,Totals  
1690,4937,2037,727,788,111  
1771,21843,3623,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67540,6250,6840,2000,4543,89734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14000,5346,10965,391114  
1850,35549,21800,18891,5346,10965,415115  
1860,813469,279122,32003,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59940,5653,51980,33029,1911801  
1890,1352000,70000,65000,58000,33029,2141534  
1900,1850593,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22101103,2018354,44600,720201,11651,50048  
1930,18671128,1796128,1796128,1796128,1796128,4930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2738175,1550949,1451277,191555,7981984  
1970,1539231,2460701,1471701,1235443,7071646  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,379877,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,4175133,8175133  
2015,1444018,2646733,2339150,1459446,474558,8059405
```

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,Population  
1698,203,2037,...,727,7181  
1771,21843,36241,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,72000,6340,6840,1840,4934,85934  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,142278  
1840,311510,11013,14040,5348,10965,391114  
1850,35549,128901,18951,5815,11515,45115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1385711,71871,6814,5816,41861,215134  
1900,1850993,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22461103,2018354,44601,72021,11651,55048  
1930,26671128,2079128,2554,2554,5582,4930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1696010,2738175,1550949,1451277,191555,7991957  
1970,1539231,2465701,1472121,1472121,135443,7984860  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,379977,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2277722,1385108,474558,8155405  
2015,1444518,2546733,2339150,1459446,474558,8155405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

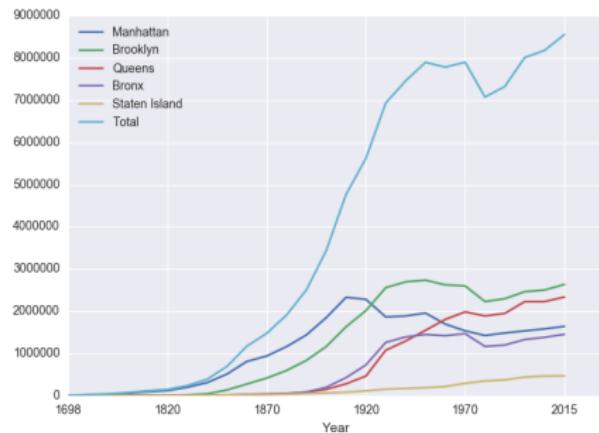
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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,Population  
1699,4937,2017,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,63541,7000,1800,5000,19334  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,12013,14000,5348,10965,391114  
1850,455441,128000,18500,10000,5000,115134  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59940,5653,51980,33001,1911801  
1890,1400000,720000,120000,110000,100000,1911814  
1900,1850093,116582,152999,200567,67921,2437202  
1910,233142,1634351,2841,430980,8569,476683  
1920,2210103,2018354,44601,73201,11651,50048  
1930,2667137,2486128,35254,25545,15821,630446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7991957  
1960,1690000,27319,18000,18000,18000,781984  
1970,1539231,2465701,1472701,1235443,798462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2216722,1385108,419728,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6



Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 8: function parameters, github

- Functions can have **input parameters**.

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

Week 8: function parameters, github

```
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)
print('Actual Parameters')

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 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, max design pattern, random()

```
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, max design pattern, random()

```
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, max design pattern, random()

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

print('The distance entered is', dist)
```

```
import turtle
import random

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

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

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

Week 10: more on loops, max design pattern, random()

```
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.`

Week 10: more on loops, max design pattern, random()

```
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`.
- The max design pattern provides a template for finding maximum value from a list.

Python & Circuits Review: 10 Weeks in 10 Minutes



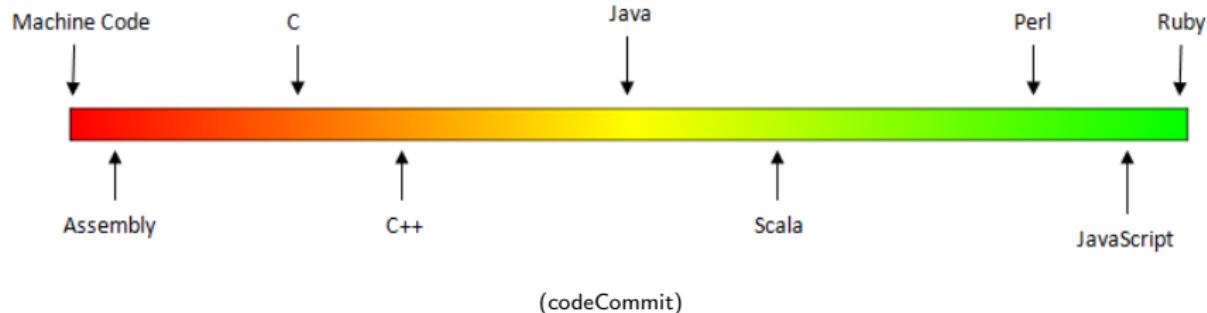
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
 - ▶ Primitive: `int`, `float`, `bool`, `string`;
 - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`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

Today's Topics



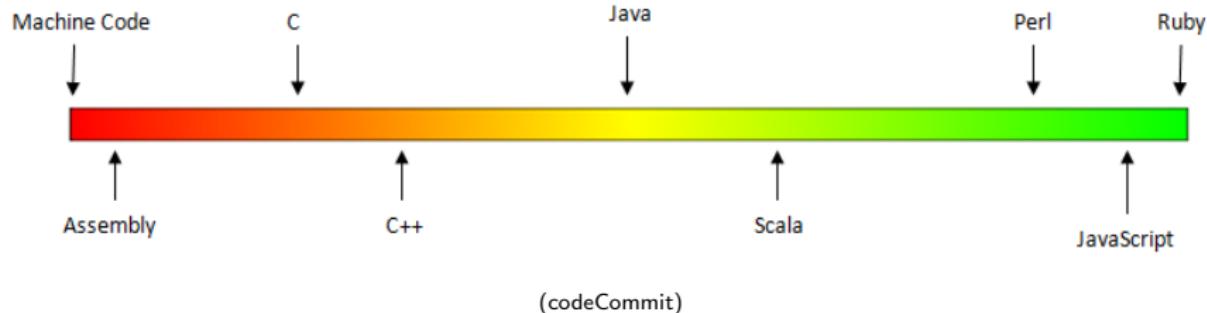
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic

Low-Level vs. High-Level Languages



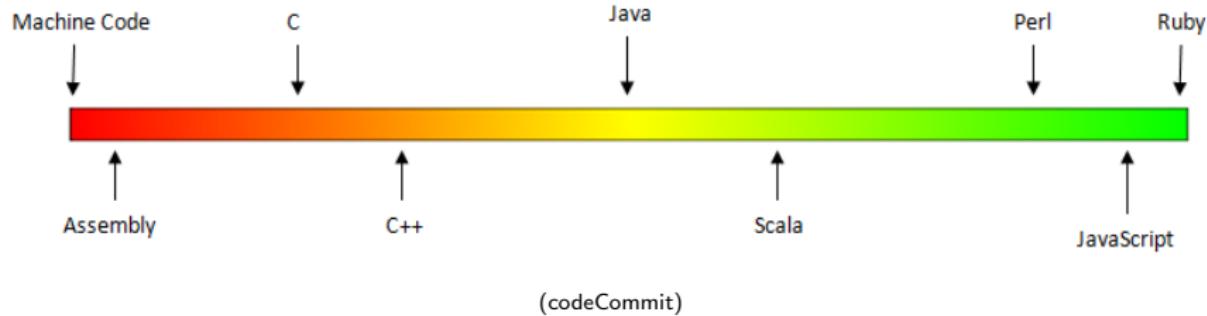
- Can view programming languages on a continuum.

Low-Level vs. High-Level Languages



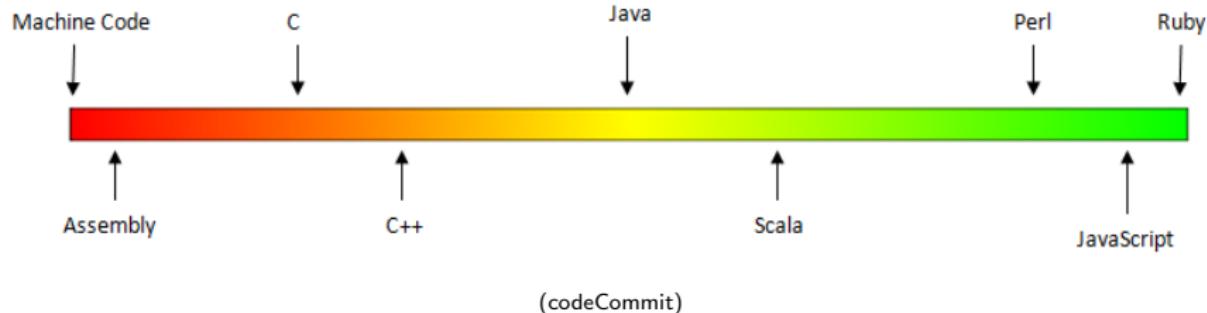
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

Low-Level vs. High-Level Languages



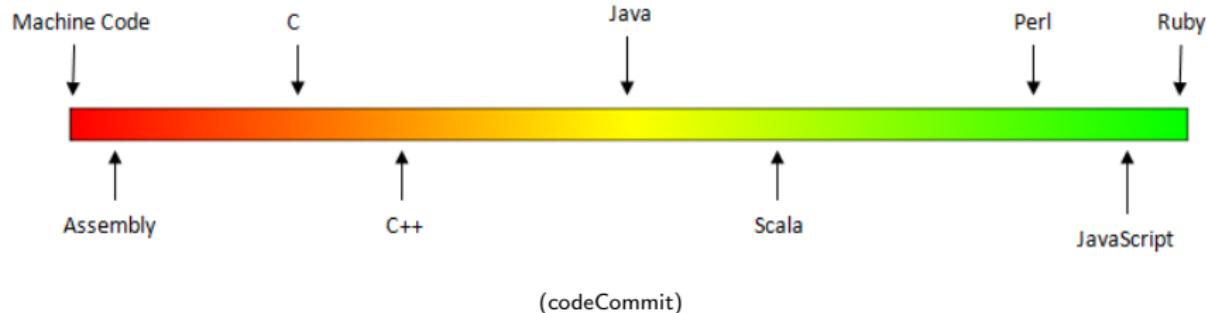
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

Low-Level vs. High-Level Languages



- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine details, such as complex variables, functions and looping that do not translate directly into machine code) are called **high-level languages**.

Low-Level vs. High-Level Languages



- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine details, such as complex variables, functions and looping that do not translate directly into machine code) are called **high-level languages**.
- Some languages, like C, are in between— allowing both low level access and high level data structures.

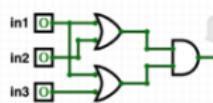
Processing

Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schrifffläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.

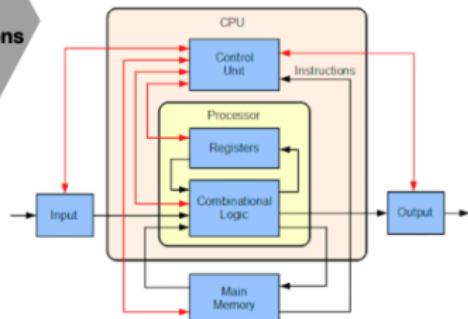
Dies ist ein Blindtext. An ihm lässt sich



Data
&
Instructions

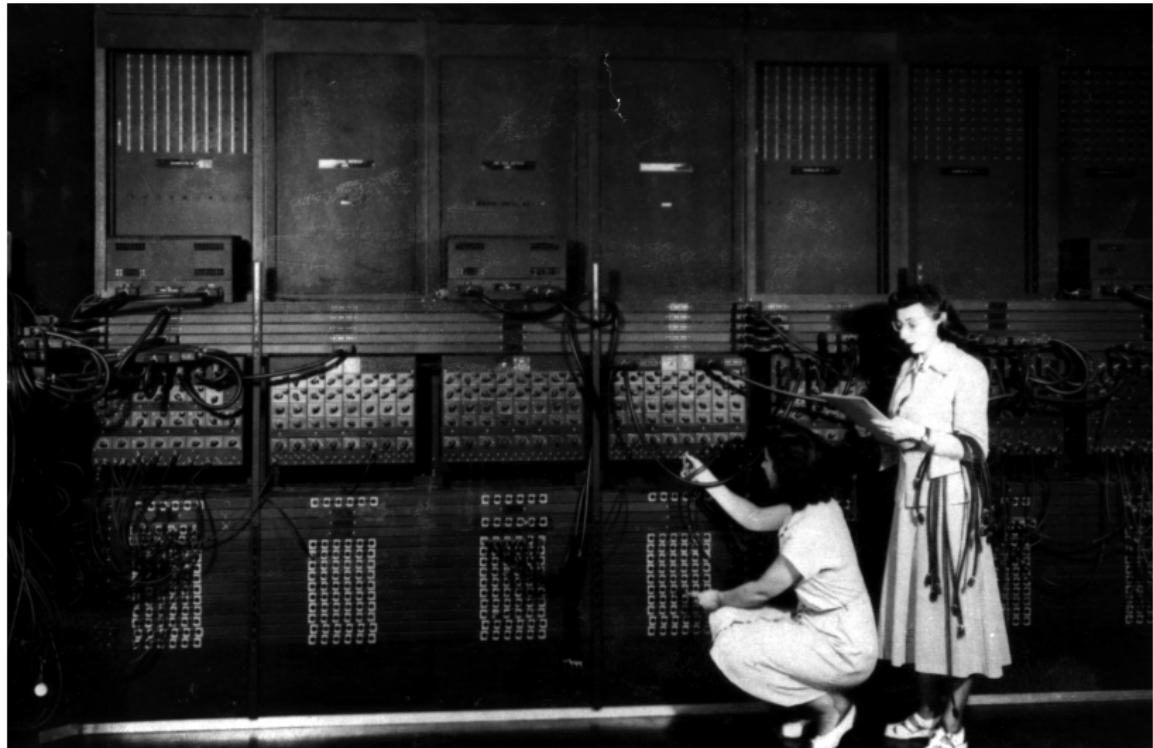


Circuits (switches)
On/Off 1/0 Logic
Billions of switches/bits



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

Machine Language



(Ruth Gordon & Ester Gerston programming the ENIAC, UPenn)

Machine Language

```
I FOX 12:01a 23- 1
A 002000 C2 30      REP #$30
A 002002 18          CLC
A 002003 F8          SED
A 002004 A9 34 12    LDA #$1234
A 002007 69 21 43    ADC #$4321
A 00200A 8F 03 7F 01 STA $017F03
A 00200E D8          CLD
A 00200F E2 30      SEP #$30
A 002011 00          BRK
A 2012

r
PB PC  NUMxDIZC .A .X .Y SP DP DB
; 00 E012 00110000 0000 0000 0002 CFFF 0000 00
g 2000

BREAK

PB PC  NUMxDIZC .A .X .Y SP DP DB
; 00 2013 00110000 5555 0000 0002 CFFF 0000 00
m 7f03 7f03
>007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00:UU .....
```

(wiki)

Machine Language

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

(wiki)

Machine Language

- 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.

(wiki)

Machine Language



The screenshot shows a computer interface for writing assembly code. At the top, there's a status bar with 'File' and 'Edit' tabs. Below that is a menu bar with 'File', 'Edit', 'Run', 'Help', and 'About'. The main window contains two panes: one for assembly code and one for register values.

Assembly Code:

```
    .C2 30    REP $#30
    .B1      CLD
    .SEI
    .LDA #1234
    .LDI 21,43  #0C #4321
    .STB 03,7F,01  #01 #7F03
    .CLD
    .SER #38
    .BRK
    .B11 30    SER #38
    .BRK
    .B12

    PB PC  MU#012C A X Y SP DP R
    : 00 E012 00110000 0000 0000 0002 CFFF 0000 00
    & 2000

BREAK

    PB PC  MU#012C A X Y SP DP R
    : 00 E012 00110000 5555 0000 0002 CFFF 0000 00
    & 2000
```

Registers:

| PB | PC | MU#012C | A | X | Y | SP | DP | R |
|--------|------|----------|------|------|------|------|------|----|
| : 00 | E012 | 00110000 | 5555 | 0000 | 0002 | CFFF | 0000 | 00 |
| & 2000 | | | | | | | | |

(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.

Machine Language

```

A 002000 C2 3B REP #3B
A 002002 SED
A 002003 CLC
A 002004 FS
A 002005 LD #1234
A 002006 LD #123451
A 002007 LD #123452
A 002008 LD #123453
A 002009 LD #123454
A 00200A LD #123455
A 00200B LD #123456
A 00200C LD #123457
A 00200D LD #123458
A 00200E LD #123459
A 00200F LD #123460
A 002011 RR
A 002012 RR

F PB PC Mm#012C .A X Y SP DP IB
; 00 2012 00110000 0000 0000 0002 CF77 0000 00
; 2000

BREAK

PB PC Mm#012C .A X Y SP DP IB
; 00 2013 00110000 0555 0000 0002 CF77 0000 00
; 7703 7703

n 7703 7703

```

(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....

“Hello World!” in Simplified Machine Language

```
1 # Store 'Hello world!' at the top of the  
2 stack  
3 ADDI $sp, $sp, -13  
4 ADDI $t0, $zero, 72 # 72 is ASCII code of 'H'  
5 SB $t0, 0($sp)  
6 ADDI $t0, $zero, 101 # e  
7 SB $t0, 1($sp)  
8 ADDI $t0, $zero, 108 # l  
9 SB $t0, 2($sp)  
10 ADDI $t0, $zero, 108 # l  
11 SB $t0, 3($sp)  
12 ADDI $t0, $zero, 111 # o  
13 SB $t0, 4($sp)
```

“Hello World!” in Simplified Machine Language: II

```
13 ADDI $t0, $zero, 32 # (space)
14 SB $t0, 5($sp)
15 ADDI $t0, $zero, 119 # w
16 SB $t0, 6($sp)
17 ADDI $t0, $zero, 111 # o
18 SB $t0, 7($sp)
19 ADDI $t0, $zero, 114 # r
20 SB $t0, 8($sp)
21 ADDI $t0, $zero, 108 # l
22 SB $t0, 9($sp)
23 ADDI $t0, $zero, 100 # d
24 SB $t0, 10($sp)
```

“Hello World!” in Simplified Machine Language: II

```
25 ADDI $t0, $zero, 33 # !
26 SB $t0, 11($sp)
27 ADDI $t0, $zero, 0 # (null)
28 SB $t0, 12($sp)

29
30 ADDI $v0, $zero, 4 # 4 is for print string
31 ADDI $a0, $sp, 0
32 syscall           # print to the log
```

WeMIPS

(Demo with WeMIPS)

MIPS Commands

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

MIPS Commands

The screenshot shows a software interface for assembly language development. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Help', and tabs for 'ShowFide Demo', 'Addition Counter', 'Itax', 'Looper', 'Stack Test', 'Hello World', 'Code Gen Save String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. Below the tabs is a toolbar with icons for file operations like Open, Save, Print, and Run. The main area contains assembly code and a register dump.

Assembly Code:

```
1 # Shows "Hello world!" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    li $t0, 0x48454c4c # Hello
6    li $t1, 0x6f6f6f6f # world
7    add $t2, $t0, $t1, lsh $t2, 1
8    add $t3, $t2, $t2, lsh $t3, 1
9    add $t4, $t3, $t3, lsh $t4, 1
10   add $t5, $t4, $t4, lsh $t5, 1
11   add $t6, $t5, $t5, lsh $t6, 1
12   add $t7, $t6, $t6, lsh $t7, 1
13   add $t8, $t7, $t7, lsh $t8, 1
14   add $t9, $t8, $t8, lsh $t9, 1
15   add $t10, $t9, $t9, lsh $t10, 1
16   add $t11, $t10, $t10, lsh $t11, 1
17   add $t12, $t11, $t11, lsh $t12, 1
18   add $t13, $t12, $t12, lsh $t13, 1
19   add $t14, $t13, $t13, lsh $t14, 1
20   add $t15, $t14, $t14, lsh $t15, 1
21   add $t16, $t15, $t15, lsh $t16, 1
22   add $t17, $t16, $t16, lsh $t17, 1
23   add $t18, $t17, $t17, lsh $t18, 1
24   add $t19, $t18, $t18, lsh $t19, 1
25   add $t20, $t19, $t19, lsh $t20, 1
26   add $t21, $t20, $t20, lsh $t21, 1
27   add $t22, $t21, $t21, lsh $t22, 1
28   add $t23, $t22, $t22, lsh $t23, 1
29   add $t24, $t23, $t23, lsh $t24, 1
30   add $t25, $t24, $t24, lsh $t25, 1
31   add $t26, $t25, $t25, lsh $t26, 1
32   add $t27, $t26, $t26, lsh $t27, 1
33   add $t28, $t27, $t27, lsh $t28, 1
34   add $t29, $t28, $t28, lsh $t29, 1
35   add $t30, $t29, $t29, lsh $t30, 1
36   add $t31, $t30, $t30, lsh $t31, 1
37   add $t32, $t31, $t31, lsh $t32, 1
38   add $t33, $t32, $t32, lsh $t33, 1
39   add $t34, $t33, $t33, lsh $t34, 1
40   add $t35, $t34, $t34, lsh $t35, 1
41   add $t36, $t35, $t35, lsh $t36, 1
42   add $t37, $t36, $t36, lsh $t37, 1
43   add $t38, $t37, $t37, lsh $t38, 1
44   add $t39, $t38, $t38, lsh $t39, 1
45   add $t40, $t39, $t39, lsh $t40, 1
46   add $t41, $t40, $t40, lsh $t41, 1
47   add $t42, $t41, $t41, lsh $t42, 1
48   add $t43, $t42, $t42, lsh $t43, 1
49   add $t44, $t43, $t43, lsh $t44, 1
50   add $t45, $t44, $t44, lsh $t45, 1
51   add $t46, $t45, $t45, lsh $t46, 1
52   add $t47, $t46, $t46, lsh $t47, 1
53   add $t48, $t47, $t47, lsh $t48, 1
54   add $t49, $t48, $t48, lsh $t49, 1
55   add $t50, $t49, $t49, lsh $t50, 1
56   add $t51, $t50, $t50, lsh $t51, 1
57   add $t52, $t51, $t51, lsh $t52, 1
58   add $t53, $t52, $t52, lsh $t53, 1
59   add $t54, $t53, $t53, lsh $t54, 1
60   add $t55, $t54, $t54, lsh $t55, 1
61   add $t56, $t55, $t55, lsh $t56, 1
62   add $t57, $t56, $t56, lsh $t57, 1
63   add $t58, $t57, $t57, lsh $t58, 1
64   add $t59, $t58, $t58, lsh $t59, 1
65   add $t60, $t59, $t59, lsh $t60, 1
66   add $t61, $t60, $t60, lsh $t61, 1
67   add $t62, $t61, $t61, lsh $t62, 1
68   add $t63, $t62, $t62, lsh $t63, 1
69   add $t64, $t63, $t63, lsh $t64, 1
70   add $t65, $t64, $t64, lsh $t65, 1
71   add $t66, $t65, $t65, lsh $t66, 1
72   add $t67, $t66, $t66, lsh $t67, 1
73   add $t68, $t67, $t67, lsh $t68, 1
74   add $t69, $t68, $t68, lsh $t69, 1
75   add $t70, $t69, $t69, lsh $t70, 1
76   add $t71, $t70, $t70, lsh $t71, 1
77   add $t72, $t71, $t71, lsh $t72, 1
78   add $t73, $t72, $t72, lsh $t73, 1
79   add $t74, $t73, $t73, lsh $t74, 1
80   add $t75, $t74, $t74, lsh $t75, 1
81   add $t76, $t75, $t75, lsh $t76, 1
82   add $t77, $t76, $t76, lsh $t77, 1
83   add $t78, $t77, $t77, lsh $t78, 1
84   add $t79, $t78, $t78, lsh $t79, 1
85   add $t80, $t79, $t79, lsh $t80, 1
86   add $t81, $t80, $t80, lsh $t81, 1
87   add $t82, $t81, $t81, lsh $t82, 1
88   add $t83, $t82, $t82, lsh $t83, 1
89   add $t84, $t83, $t83, lsh $t84, 1
90   add $t85, $t84, $t84, lsh $t85, 1
91   add $t86, $t85, $t85, lsh $t86, 1
92   add $t87, $t86, $t86, lsh $t87, 1
93   add $t88, $t87, $t87, lsh $t88, 1
94   add $t89, $t88, $t88, lsh $t89, 1
95   add $t90, $t89, $t89, lsh $t90, 1
96   add $t91, $t90, $t90, lsh $t91, 1
97   add $t92, $t91, $t91, lsh $t92, 1
98   add $t93, $t92, $t92, lsh $t93, 1
99   add $t94, $t93, $t93, lsh $t94, 1
100  add $t95, $t94, $t94, lsh $t95, 1
101  add $t96, $t95, $t95, lsh $t96, 1
102  add $t97, $t96, $t96, lsh $t97, 1
103  add $t98, $t97, $t97, lsh $t98, 1
104  add $t99, $t98, $t98, lsh $t99, 1
105  add $t100, $t99, $t99, lsh $t100, 1
106  add $t101, $t100, $t100, lsh $t101, 1
107  add $t102, $t101, $t101, lsh $t102, 1
108  add $t103, $t102, $t102, lsh $t103, 1
109  add $t104, $t103, $t103, lsh $t104, 1
110  add $t105, $t104, $t104, lsh $t105, 1
111  add $t106, $t105, $t105, lsh $t106, 1
112  add $t107, $t106, $t106, lsh $t107, 1
113  add $t108, $t107, $t107, lsh $t108, 1
114  add $t109, $t108, $t108, lsh $t109, 1
115  add $t110, $t109, $t109, lsh $t110, 1
116  add $t111, $t110, $t110, lsh $t111, 1
117  add $t112, $t111, $t111, lsh $t112, 1
118  add $t113, $t112, $t112, lsh $t113, 1
119  add $t114, $t113, $t113, lsh $t114, 1
120  add $t115, $t114, $t114, lsh $t115, 1
121  add $t116, $t115, $t115, lsh $t116, 1
122  add $t117, $t116, $t116, lsh $t117, 1
123  add $t118, $t117, $t117, lsh $t118, 1
124  add $t119, $t118, $t118, lsh $t119, 1
125  add $t120, $t119, $t119, lsh $t120, 1
126  add $t121, $t120, $t120, lsh $t121, 1
127  add $t122, $t121, $t121, lsh $t122, 1
128  add $t123, $t122, $t122, lsh $t123, 1
129  add $t124, $t123, $t123, lsh $t124, 1
130  add $t125, $t124, $t124, lsh $t125, 1
131  add $t126, $t125, $t125, lsh $t126, 1
132  add $t127, $t126, $t126, lsh $t127, 1
133  add $t128, $t127, $t127, lsh $t128, 1
134  add $t129, $t128, $t128, lsh $t129, 1
135  add $t130, $t129, $t129, lsh $t130, 1
136  add $t131, $t130, $t130, lsh $t131, 1
137  add $t132, $t131, $t131, lsh $t132, 1
138  add $t133, $t132, $t132, lsh $t133, 1
139  add $t134, $t133, $t133, lsh $t134, 1
140  add $t135, $t134, $t134, lsh $t135, 1
141  add $t136, $t135, $t135, lsh $t136, 1
142  add $t137, $t136, $t136, lsh $t137, 1
143  add $t138, $t137, $t137, lsh $t138, 1
144  add $t139, $t138, $t138, lsh $t139, 1
145  add $t140, $t139, $t139, lsh $t140, 1
146  add $t141, $t140, $t140, lsh $t141, 1
147  add $t142, $t141, $t141, lsh $t142, 1
148  add $t143, $t142, $t142, lsh $t143, 1
149  add $t144, $t143, $t143, lsh $t144, 1
150  add $t145, $t144, $t144, lsh $t145, 1
151  add $t146, $t145, $t145, lsh $t146, 1
152  add $t147, $t146, $t146, lsh $t147, 1
153  add $t148, $t147, $t147, lsh $t148, 1
154  add $t149, $t148, $t148, lsh $t149, 1
155  add $t150, $t149, $t149, lsh $t150, 1
156  add $t151, $t150, $t150, lsh $t151, 1
157  add $t152, $t151, $t151, lsh $t152, 1
158  add $t153, $t152, $t152, lsh $t153, 1
159  add $t154, $t153, $t153, lsh $t154, 1
160  add $t155, $t154, $t154, lsh $t155, 1
161  add $t156, $t155, $t155, lsh $t156, 1
162  add $t157, $t156, $t156, lsh $t157, 1
163  add $t158, $t157, $t157, lsh $t158, 1
164  add $t159, $t158, $t158, lsh $t159, 1
165  add $t160, $t159, $t159, lsh $t160, 1
166  add $t161, $t160, $t160, lsh $t161, 1
167  add $t162, $t161, $t161, lsh $t162, 1
168  add $t163, $t162, $t162, lsh $t163, 1
169  add $t164, $t163, $t163, lsh $t164, 1
170  add $t165, $t164, $t164, lsh $t165, 1
171  add $t166, $t165, $t165, lsh $t166, 1
172  add $t167, $t166, $t166, lsh $t167, 1
173  add $t168, $t167, $t167, lsh $t168, 1
174  add $t169, $t168, $t168, lsh $t169, 1
175  add $t170, $t169, $t169, lsh $t170, 1
176  add $t171, $t170, $t170, lsh $t171, 1
177  add $t172, $t171, $t171, lsh $t172, 1
178  add $t173, $t172, $t172, lsh $t173, 1
179  add $t174, $t173, $t173, lsh $t174, 1
180  add $t175, $t174, $t174, lsh $t175, 1
181  add $t176, $t175, $t175, lsh $t176, 1
182  add $t177, $t176, $t176, lsh $t177, 1
183  add $t178, $t177, $t177, lsh $t178, 1
184  add $t179, $t178, $t178, lsh $t179, 1
185  add $t180, $t179, $t179, lsh $t180, 1
186  add $t181, $t180, $t180, lsh $t181, 1
187  add $t182, $t181, $t181, lsh $t182, 1
188  add $t183, $t182, $t182, lsh $t183, 1
189  add $t184, $t183, $t183, lsh $t184, 1
190  add $t185, $t184, $t184, lsh $t185, 1
191  add $t186, $t185, $t185, lsh $t186, 1
192  add $t187, $t186, $t186, lsh $t187, 1
193  add $t188, $t187, $t187, lsh $t188, 1
194  add $t189, $t188, $t188, lsh $t189, 1
195  add $t190, $t189, $t189, lsh $t190, 1
196  add $t191, $t190, $t190, lsh $t191, 1
197  add $t192, $t191, $t191, lsh $t192, 1
198  add $t193, $t192, $t192, lsh $t193, 1
199  add $t194, $t193, $t193, lsh $t194, 1
200  add $t195, $t194, $t194, lsh $t195, 1
201  add $t196, $t195, $t195, lsh $t196, 1
202  add $t197, $t196, $t196, lsh $t197, 1
203  add $t198, $t197, $t197, lsh $t198, 1
204  add $t199, $t198, $t198, lsh $t199, 1
205  add $t200, $t199, $t199, lsh $t200, 1
206  add $t201, $t200, $t200, lsh $t201, 1
207  add $t202, $t201, $t201, lsh $t202, 1
208  add $t203, $t202, $t202, lsh $t203, 1
209  add $t204, $t203, $t203, lsh $t204, 1
210  add $t205, $t204, $t204, lsh $t205, 1
211  add $t206, $t205, $t205, lsh $t206, 1
212  add $t207, $t206, $t206, lsh $t207, 1
213  add $t208, $t207, $t207, lsh $t208, 1
214  add $t209, $t208, $t208, lsh $t209, 1
215  add $t210, $t209, $t209, lsh $t210, 1
216  add $t211, $t210, $t210, lsh $t211, 1
217  add $t212, $t211, $t211, lsh $t212, 1
218  add $t213, $t212, $t212, lsh $t213, 1
219  add $t214, $t213, $t213, lsh $t214, 1
220  add $t215, $t214, $t214, lsh $t215, 1
221  add $t216, $t215, $t215, lsh $t216, 1
222  add $t217, $t216, $t216, lsh $t217, 1
223  add $t218, $t217, $t217, lsh $t218, 1
224  add $t219, $t218, $t218, lsh $t219, 1
225  add $t220, $t219, $t219, lsh $t220, 1
226  add $t221, $t220, $t220, lsh $t221, 1
227  add $t222, $t221, $t221, lsh $t222, 1
228  add $t223, $t222, $t222, lsh $t223, 1
229  add $t224, $t223, $t223, lsh $t224, 1
230  add $t225, $t224, $t224, lsh $t225, 1
231  add $t226, $t225, $t225, lsh $t226, 1
232  add $t227, $t226, $t226, lsh $t227, 1
233  add $t228, $t227, $t227, lsh $t228, 1
234  add $t229, $t228, $t228, lsh $t229, 1
235  add $t230, $t229, $t229, lsh $t230, 1
236  add $t231, $t230, $t230, lsh $t231, 1
237  add $t232, $t231, $t231, lsh $t232, 1
238  add $t233, $t232, $t232, lsh $t233, 1
239  add $t234, $t233, $t233, lsh $t234, 1
240  add $t235, $t234, $t234, lsh $t235, 1
241  add $t236, $t235, $t235, lsh $t236, 1
242  add $t237, $t236, $t236, lsh $t237, 1
243  add $t238, $t237, $t237, lsh $t238, 1
244  add $t239, $t238, $t238, lsh $t239, 1
245  add $t240, $t239, $t239, lsh $t240, 1
246  add $t241, $t240, $t240, lsh $t241, 1
247  add $t242, $t241, $t241, lsh $t242, 1
248  add $t243, $t242, $t242, lsh $t243, 1
249  add $t244, $t243, $t243, lsh $t244, 1
250  add $t245, $t244, $t244, lsh $t245, 1
251  add $t246, $t245, $t245, lsh $t246, 1
252  add $t247, $t246, $t246, lsh $t247, 1
253  add $t248, $t247, $t247, lsh $t248, 1
254  add $t249, $t248, $t248, lsh $t249, 1
255  add $t250, $t249, $t249, lsh $t250, 1
256  add $t251, $t250, $t250, lsh $t251, 1
257  add $t252, $t251, $t251, lsh $t252, 1
258  add $t253, $t252, $t252, lsh $t253, 1
259  add $t254, $t253, $t253, lsh $t254, 1
260  add $t255, $t254, $t254, lsh $t255, 1
261  add $t256, $t255, $t255, lsh $t256, 1
262  add $t257, $t256, $t256, lsh $t257, 1
263  add $t258, $t257, $t257, lsh $t258, 1
264  add $t259, $t258, $t258, lsh $t259, 1
265  add $t260, $t259, $t259, lsh $t260, 1
266  add $t261, $t260, $t260, lsh $t261, 1
267  add $t262, $t261, $t261, lsh $t262, 1
268  add $t263, $t262, $t262, lsh $t263, 1
269  add $t264, $t263, $t263, lsh $t264, 1
270  add $t265, $t264, $t264, lsh $t265, 1
271  add $t266, $t265, $t265, lsh $t266, 1
272  add $t267, $t266, $t266, lsh $t267, 1
273  add $t268, $t267, $t267, lsh $t268, 1
274  add $t269, $t268, $t268, lsh $t269, 1
275  add $t270, $t269, $t269, lsh $t270, 1
276  add $t271, $t270, $t270, lsh $t271, 1
277  add $t272, $t271, $t271, lsh $t272, 1
278  add $t273, $t272, $t272, lsh $t273, 1
279  add $t274, $t273, $t273, lsh $t274, 1
280  add $t275, $t274, $t274, lsh $t275, 1
281  add $t276, $t275, $t275, lsh $t276, 1
282  add $t277, $t276, $t276, lsh $t277, 1
283  add $t278, $t277, $t277, lsh $t278, 1
284  add $t279, $t278, $t278, lsh $t279, 1
285  add $t280, $t279, $t279, lsh $t280, 1
286  add $t281, $t280, $t280, lsh $t281, 1
287  add $t282, $t281, $t281, lsh $t282, 1
288  add $t283, $t282, $t282, lsh $t283, 1
289  add $t284, $t283, $t283, lsh $t284, 1
290  add $t285, $t284, $t284, lsh $t285, 1
291  add $t286, $t285, $t285, lsh $t286, 1
292  add $t287, $t286, $t286, lsh $t287, 1
293  add $t288, $t287, $t287, lsh $t288, 1
294  add $t289, $t288, $t288, lsh $t289, 1
295  add $t290, $t289, $t289, lsh $t290, 1
296  add $t291, $t290, $t290, lsh $t291, 1
297  add $t292, $t291, $t291, lsh $t292, 1
298  add $t293, $t292, $t292, lsh $t293, 1
299  add $t294, $t293, $t293, lsh $t294, 1
300  add $t295, $t294, $t294, lsh $t295, 1
301  add $t296, $t295, $t295, lsh $t296, 1
302  add $t297, $t296, $t296, lsh $t297, 1
303  add $t298, $t297, $t297, lsh $t298, 1
304  add $t299, $t298, $t298, lsh $t299, 1
305  add $t300, $t299, $t299, lsh $t300, 1
306  add $t301, $t300, $t300, lsh $t301, 1
307  add $t302, $t301, $t301, lsh $t302, 1
308  add $t303, $t302, $t302, lsh $t303, 1
309  add $t304, $t303, $t303, lsh $t304, 1
310  add $t305, $t304, $t304, lsh $t305, 1
311  add $t306, $t305, $t305, lsh $t306, 1
312  add $t307, $t306, $t306, lsh $t307, 1
313  add $t308, $t307, $t307, lsh $t308, 1
314  add $t309, $t308, $t308, lsh $t309, 1
315  add $t310, $t309, $t309, lsh $t310, 1
316  add $t311, $t310, $t310, lsh $t311, 1
317  add $t312, $t311, $t311, lsh $t312, 1
318  add $t313, $t312, $t312, lsh $t313, 1
319  add $t314, $t313, $t313, lsh $t314, 1
320  add $t315, $t314, $t314, lsh $t315, 1
321  add $t316, $t315, $t315, lsh $t316, 1
322  add $t317, $t316, $t316, lsh $t317, 1
323  add $t318, $t317, $t317, lsh $t318, 1
324  add $t319, $t318, $t318, lsh $t319, 1
325  add $t320, $t319, $t319, lsh $t320, 1
326  add $t321, $t320, $t320, lsh $t321, 1
327  add $t322, $t321, $t321, lsh $t322, 1
328  add $t323, $t322, $t322, lsh $t323, 1
329  add $t324, $t323, $t323, lsh $t324, 1
330  add $t325, $t324, $t324, lsh $t325, 1
331  add $t326, $t325, $t325, lsh $t326, 1
332  add $t327, $t326, $t326, lsh $t327, 1
333  add $t328, $t327, $t327, lsh $t328, 1
334  add $t329, $t328, $t328, lsh $t329, 1
335  add $t330, $t329, $t329, lsh $t330, 1
336  add $t331, $t330, $t330, lsh $t331, 1
337  add $t332, $t331, $t331, lsh $t332, 1
338  add $t333, $t332, $t332, lsh $t333, 1
339  add $t334, $t333, $t333, lsh $t334, 1
340  add $t335, $t334, $t334, lsh $t335, 1
341  add $t336, $t335, $t335, lsh $t336, 1
342  add $t337, $t336, $t336, lsh $t337, 1
343  add $t338, $t337, $t337, lsh $t338, 1
344  add $t339, $t338, $t338, lsh $t339, 1
345  add $t340, $t339, $t339, lsh $t340, 1
346  add $t341, $t340, $t340, lsh $t341, 1
347  add $t342, $t341, $t341, lsh $t342, 1
348  add $t343, $t342, $t342, lsh $t343, 1
349  add $t344, $t343, $t343, lsh $t344, 1
350  add $t345, $t344, $t344, lsh $t345, 1
351  add $t346, $t345, $t345, lsh $t346, 1
352  add $t347, $t346, $t346, lsh $t347, 1
353  add $t348, $t347, $t347, lsh $t348, 1
354  add $t349, $t348, $t348, lsh $t349, 1
355  add $t350, $t349, $t349, lsh $t350, 1
356  add $t351, $t350, $t350, lsh $t351, 1
357  add $t352, $t351, $t351, lsh $t352, 1
358  add $t353, $t352, $t352, lsh $t353, 1
359  add $t354, $t353, $t353, lsh $t354, 1
360  add $t355, $t354, $t354, lsh $t355, 1
361  add $t356, $t355, $t355, lsh $t356, 1
362  add $t357, $t356, $t356, lsh $t357, 1
363  add $t358, $t357, $t357, lsh $t358, 1
364  add $t359, $t358, $t358, lsh $t359, 1
365  add $t360, $t359, $t359, lsh $t360, 1
366  add $t361, $t360, $t360, lsh $t361, 1
367  add $t362, $t361, $t361, lsh $t362, 1
368  add $t363, $t362, $t362, lsh $t363, 1
369  add $t364, $t363, $t363, lsh $t364, 1
370  add $t365, $t364, $t364, lsh $t365, 1
371  add $t366, $t365, $t365, lsh $t366, 1
372  add $t367, $t366, $t366, lsh $t367, 1
373  add $t368, $t367, $t367, lsh $t368, 1
374  add $t369, $t368, $t368, lsh $t369, 1
375  add $t370, $t369, $t369, lsh $t370, 1
376  add $t371, $t370, $t370, lsh $t371, 1
377  add $t372, $t371, $t371, lsh $t372, 1
378  add $t
```

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:

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

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.

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

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.

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

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)

Challenge:

Line: 3 Go! Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

```
1 # Store 'Hello world!' at the top of the stack
2 ADDI $sp, $sp, -13
3 ADDI $t0, $zero, 72 # H
4 SB $t0, 0($sp)
5 ADDI $t0, $zero, 101 # e
6 SB $t0, 1($sp)
7 ADDI $t0, $zero, 108 # l
8 SB $t0, 2($sp)
9 ADDI $t0, $zero, 108 # l
10 SB $t0, 3($sp)
11 ADDI $t0, $zero, 111 # o
12 SB $t0, 4($sp)
13 ADDI $t0, $zero, 32 # (space)
14 SB $t0, 5($sp)
15 ADDI $t0, $zero, 119 # w
16 SB $t0, 6($sp)
17 ADDI $t0, $zero, 111 # o
18 SB $t0, 7($sp)
19 ADDI $t0, $zero, 114 # r
20 SB $t0, 8($sp)
21 ADDI $t0, $zero, 108 # l
22 SB $t0, 9($sp)
23 ADDI $t0, $zero, 100 # d
24 SB $t0, 10($sp)
25 ADDI $t0, $zero, 33 # !
26 SB $t0, 11($sp)
27 ADDI $t0, $zero, 0 # (null)
28 SB $t0, 12($sp)
29
30 ADDI $v0, $zero, 4 # 4 is for print string
31 ADDI $a0, $sp, 0      # print to the log
32 syscall
```

Step Run Enable auto switching

| S | T | A | V | Stack | Log |
|-----|-----|---|---|-------|-----|
| s0: | 10 | | | | |
| s1: | 9 | | | | |
| s2: | 9 | | | | |
| s3: | 22 | | | | |
| s4: | 696 | | | | |
| s5: | 976 | | | | |
| s6: | 927 | | | | |
| s7: | 418 | | | | |

Write a program that prints out the alphabet: a b c d ... x y z

WeMIPS

```
# Store 'Hello world!' at the top of the stack
ADDI $t0, zero, 72 #B
LD $t0, 0($sp)
ADD $t0, $t0, 151 #e
#D $t0, 1($sp)
#M $t0, 150 #d
#W $t0, 150 #e
#R $t0, 150 #f
#S $t0, 150 #g
#L $t0, 150 #h
#O $t0, 150 #i
#N $t0, 150 #j
#P $t0, 150 #k
#Q $t0, 150 #l
#U $t0, 150 #m
#V $t0, 150 #n
#W $t0, 150 #o
#X $t0, 150 #p
#Y $t0, 150 #q
#Z $t0, 150 #r
#B $t0, 0($sp)
#C $t0, 0($sp)
#D $t0, 1($sp)
#E $t0, 2($sp)
#F $t0, 3($sp)
#G $t0, 4($sp)
#H $t0, 5($sp)
#I $t0, 6($sp)
#J $t0, 7($sp)
#K $t0, 8($sp)
#L $t0, 9($sp)
#M $t0, 10($sp)
#N $t0, 11($sp)
#O $t0, 12($sp)
#P $t0, 13($sp)
#Q $t0, 14($sp)
#R $t0, 15($sp)
#S $t0, 16($sp)
#T $t0, 17($sp)
#U $t0, 18($sp)
#V $t0, 19($sp)
#W $t0, 20($sp)
#X $t0, 21($sp)
#Y $t0, 22($sp)
#Z $t0, 23($sp)
#B $t0, 0 #null
#C $t0, 12($sp)
#D $t0, 13($sp)
#E $t0, 14($sp)
#F $t0, 15($sp)
#G $t0, 16($sp)
#H $t0, 17($sp)
#I $t0, 18($sp)
#J $t0, 19($sp)
#K $t0, 20($sp)
#L $t0, 21($sp)
#M $t0, 22($sp)
#N $t0, 23($sp)
#O $t0, 24($sp)
#P $t0, 25($sp)
#Q $t0, 26($sp)
#R $t0, 27($sp)
#S $t0, 28($sp)
#T $t0, 29($sp)
#U $t0, 30($sp)
#V $t0, 31($sp)
#W $t0, 32($sp)
#X $t0, 33 ($sp)
#Y $t0, 34 ($sp)
#Z $t0, 35 ($sp)

ADDI $t0, zero, 4 # 4 is for print string
#D $t0, 0($sp)
#M $t0, 0($sp)
#W $t0, 0 # point to the log
#R $t0, 0($sp)
```

(Demo with WeMIPS)

Today's Topics



- Design Patterns: Searching
- Python Recap
- Machine Language
- **Machine Language: Jumps & Loops**
- Binary & Hex Arithmetic

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.



Print alphabet table in Simplified Machine Language: II

```
1 ADDI $sp, $sp, -27 #setup stack, 26 letters +
  1 null
2 ADDI $t0, $zero, 97 #save ASCII of 'a' to $t0
3 ADDI $s2, $zero, 26 #set $s2 to be 26, track
  whether 26 is reached or not
4 SETUP:SB $t0, 0($sp) #save contents of $t0 to
  stack
5 ADDI $sp, $sp, 1 #increment the stack
6 ADDI $s2, $s2, -1 #subtract 1 from $s2
7 ADDI $t0, $t0, 1 #increment the letter
8 BEQ $s2, $zero, DONE
9 J SETUP
```

Print alphabet table in Simplified Machine Language: II

```
10 DONE: ADDI $t0, $zero, 0 #set null  
11 SB $t0, 0($sp)  
12 ADDI $sp, $sp, -26  
13 ADDI $v0, $zero, 4 #$v0 is 4 means to print  
14 ADDI $a0, $sp, 0 #set $a0 to stack pointer  
15 syscall
```

Jump Demo

Line: 18 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

```
1 ADDI $sp, $sp, -27      # Set up stack
2 ADDI $s3, $zero, 1       # Store 1 in a register
3 ADDI $t0, $zero, 97      # Set $t0 at 97 (a)
4 ADDI $s2, $zero, 26      # Use to test when you reach 26
5 SETUP: SB $t0, 0($sp)    # Next letter in $t0
6 ADDI $sp, $sp, 1         # Increment the stack
7 SUB $s2, $s2, $s3        # Decrease the counter by 1
8 ADDI $t0, $t0, 1         # Increment the letter
9 BEQ $s2, $zero, DONE     # Jump to done if $s2 == 0
10 J SETUP
11 J SETUP
12 DONE: ADDI $t0, $zero, 0 # Null (0) to terminate string
13 SB $t0, 0($sp)          # Add null to stack
14 ADDI $sp, $sp, -26      # Set up stack to print
15 ADDI $v0, $zero, 4       # 4 is for print string
16 ADDI $a0, $sp, 0         # Set $a0 to stack pointer
17 syscall                # Print to the log
```

(Demo
with
WeMIPS)

Step Run Enable auto switching

S T A V Stack Log

Clear Log

Emulation complete, returning to line 1

abcdefghijklmnopqrstuvwxyz

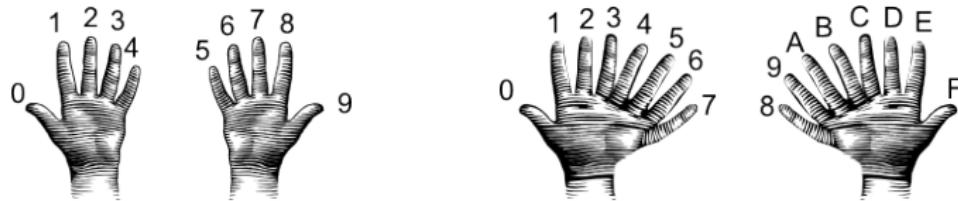


Today's Topics



- Design Patterns: Searching
- Python Recap
- Machine Language
- Machine Language: Jumps & Loops
- **Binary & Hex Arithmetic**

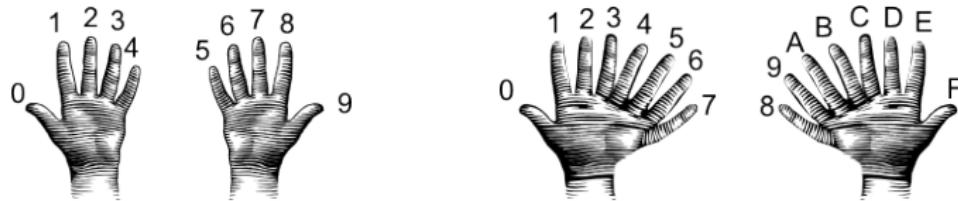
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - Convert first digit to decimal and multiple by 16.

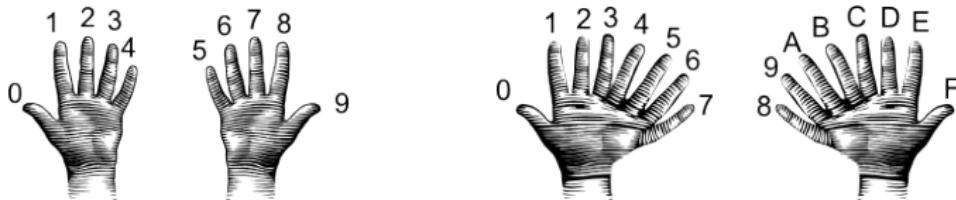
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - Convert first digit to decimal and multiple by 16.
 - Convert second digit to decimal and add to total.

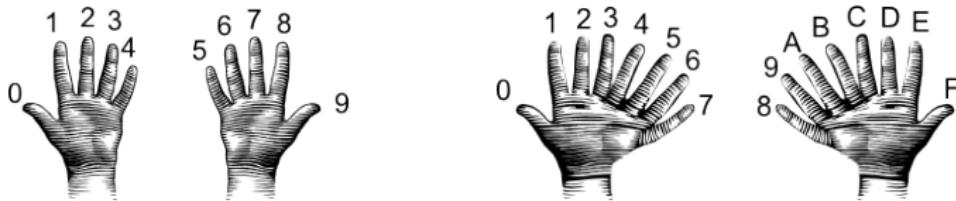
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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?

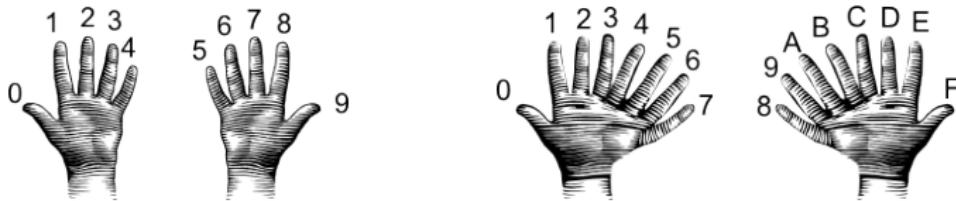
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

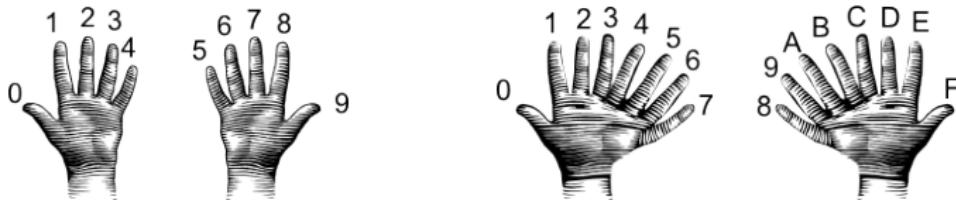
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

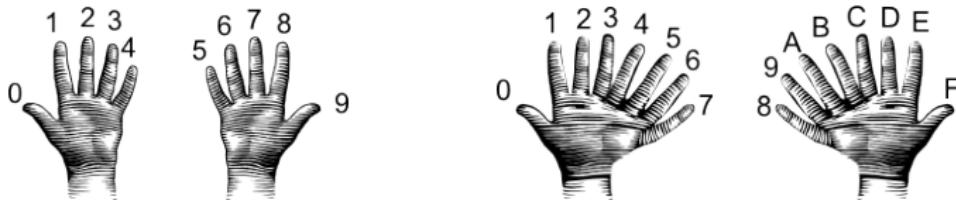
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

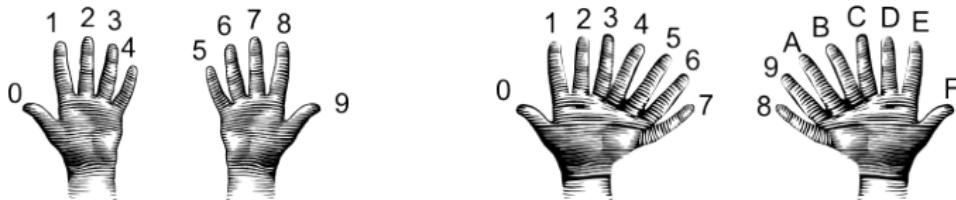
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

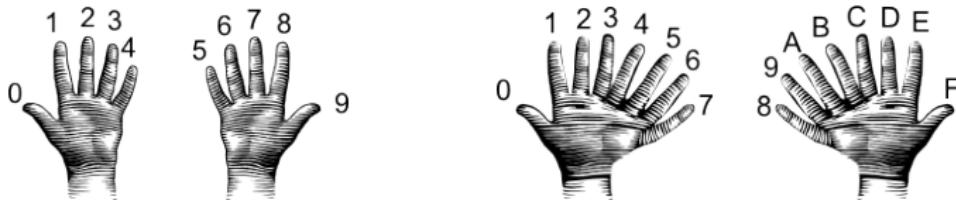
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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?

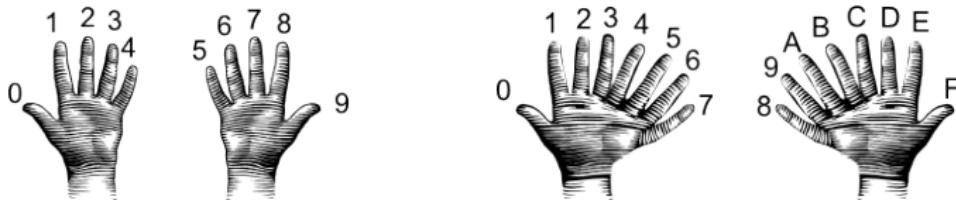
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

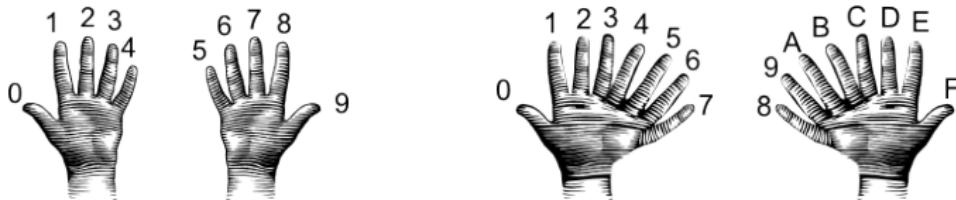
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

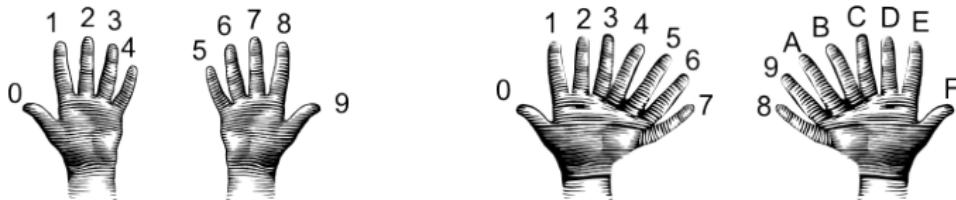
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):

- 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.

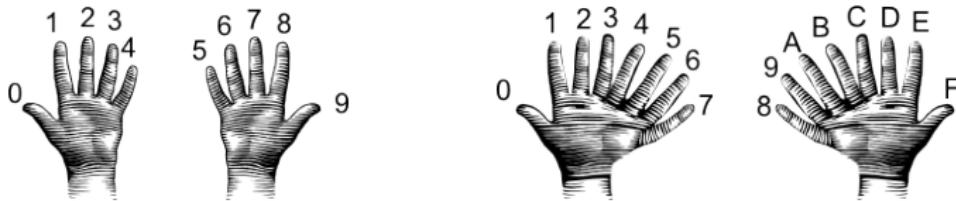
- Example: what is 99 as a decimal number?

9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):

- 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?

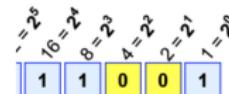
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Answer is 153.

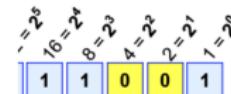
Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

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

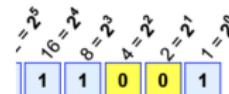
Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

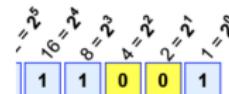
Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

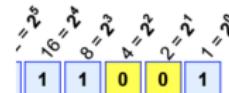


Example: $1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 1 = 16 + 8 + 0 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

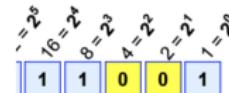


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

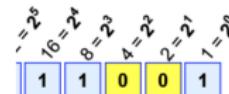


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

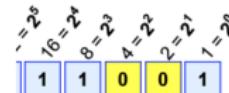


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

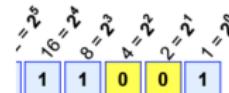


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

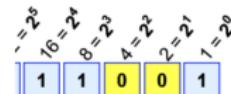


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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?

Decimal to Binary: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

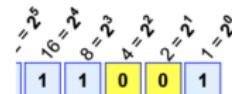
- From decimal to binary:

- 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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

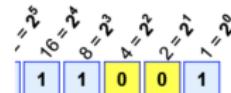
- From decimal to binary:

- 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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

- From decimal to binary:

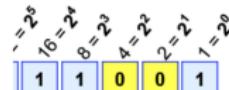
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

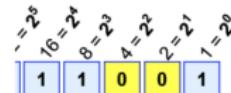
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

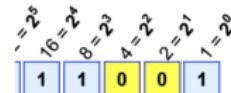
- From decimal to binary:

- 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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

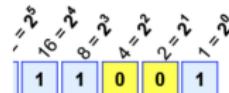
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

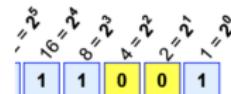
- Example: what is 130 in binary notation?

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

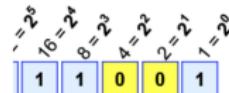
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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?

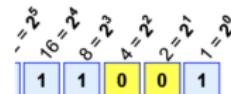
130/128 is 1 rem 2. First digit is 1: 1...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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?

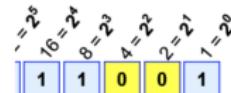
130/128 is 1 rem 2. First digit is 1: 1...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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?

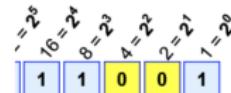
130/128 is 1 rem 2. First digit is 1: 1...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

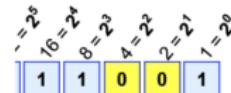
2/64 is 0 rem 2. Next digit is 0: 10...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

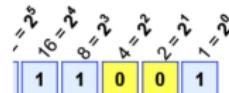
2/64 is 0 rem 2. Next digit is 0: 10...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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...

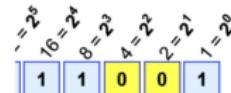
2/64 is 0 rem 2. Next digit is 0: 10...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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...

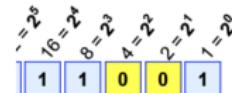
2/32 is 0 rem 2. Next digit is 0: 100...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

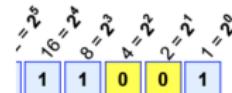
2/32 is 0 rem 2. Next digit is 0: 100...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

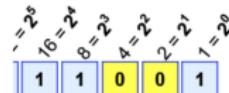
2/32 is 0 rem 2. Next digit is 0: 100...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

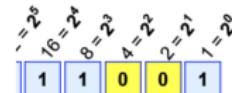
2/16 is 0 rem 2. Next digit is 0: 1000...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

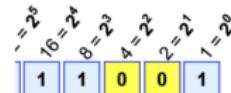
2/16 is 0 rem 2. Next digit is 0: 1000...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

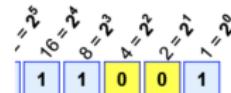
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

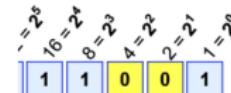
2/8 is 0 rem 2. Next digit is 0: 10000...

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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

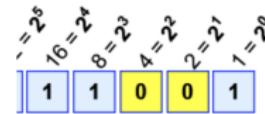
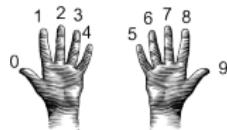
2/4 is 0 remainder 2. Next digit is 0: 100000...

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

Adding the last remainder: 10000010



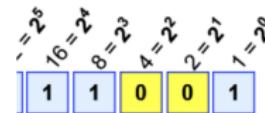
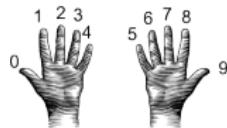
Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

Decimal to Binary: Converting Between Bases

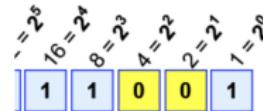
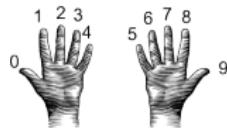


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99/128$ is 0 rem 99.

Decimal to Binary: Converting Between Bases

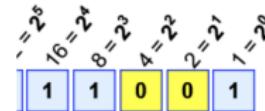
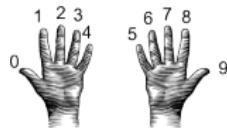


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases



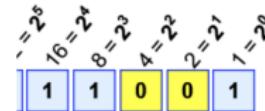
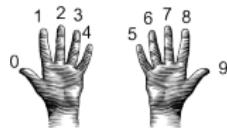
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

Decimal to Binary: Converting Between Bases



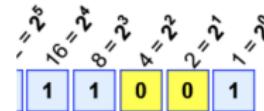
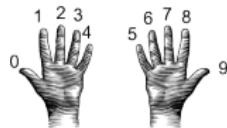
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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:

Decimal to Binary: Converting Between Bases



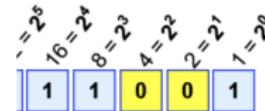
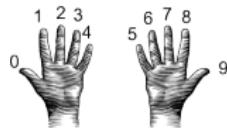
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

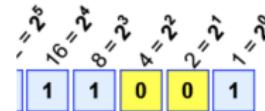
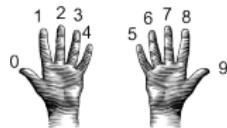
- Example: what is 99 in binary notation?

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

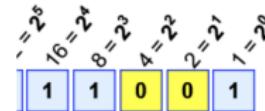
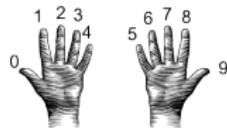
- Example: what is 99 in binary notation?

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

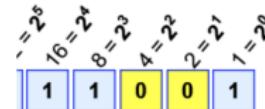
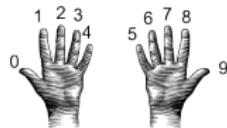
- Example: what is 99 in binary notation?

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: what is 99 in binary notation?

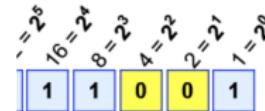
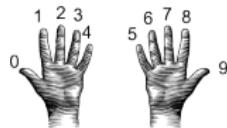
99/128 is 0 rem 99. First digit is 0: 0...

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

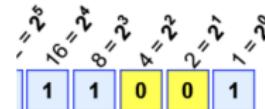
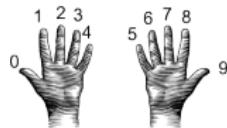
35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3.

Decimal to Binary: Converting Between Bases



Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

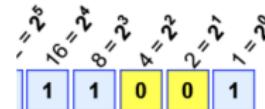
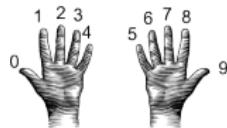
99/128 is 0 rem 99. First digit is 0: 0...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

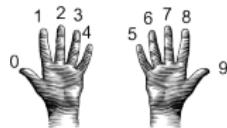
99/64 is 1 rem 35. Next digit is 1: 01...

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

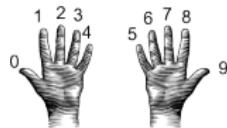
3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3.

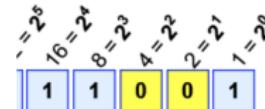
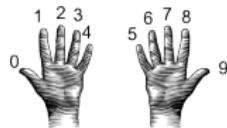
Decimal to Binary: Converting Between Bases



Decimal to Binary: Converting Between Bases



Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

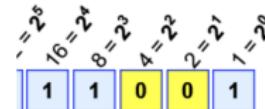
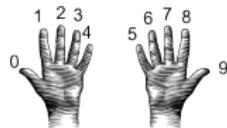
35/32 is 1 rem 3. Next digit is 1: 011...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

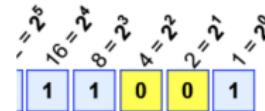
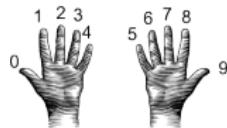
35/32 is 1 rem 3. Next digit is 1: 011...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

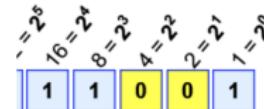
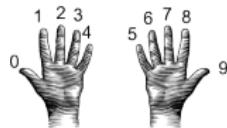
35/32 is 1 rem 3. Next digit is 1: 011...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

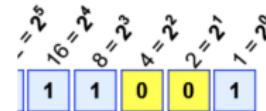
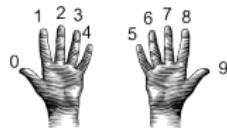
3/16 is 0 rem 3. Next digit is 0: 0110...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 0 + 1 \times 1 = 16 + 8 + 0 + 1 = 25$

- 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...

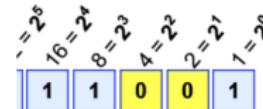
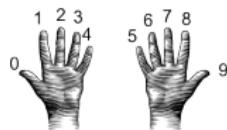
3/16 is 0 rem 3. Next digit is 0: 0110...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- 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...

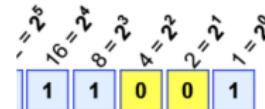
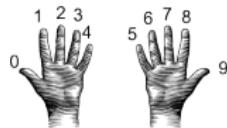
3/16 is 0 rem 3. Next digit is 0: 0110...

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...

Decimal to Binary: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$$

- 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...

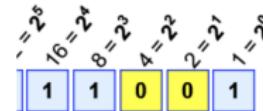
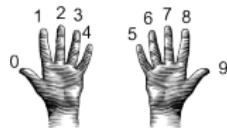
3/8 is 0 rem 3. Next digit is 0: 01100...

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

Decimal to Binary: Converting Between Bases



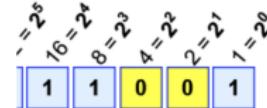
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

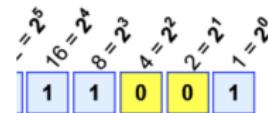
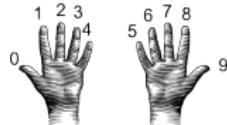
Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

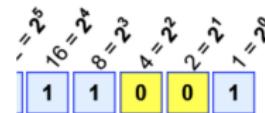
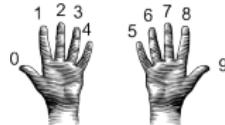
Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

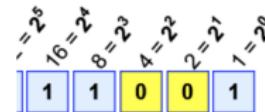
Binary to Decimal: Converting Between Bases



$$\text{Example: } 1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$$

- 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.

Binary to Decimal: Converting Between Bases

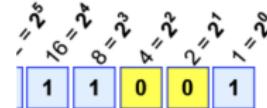
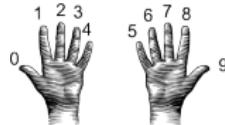


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Binary to Decimal: Converting Between Bases

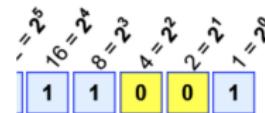


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Binary to Decimal: Converting Between Bases

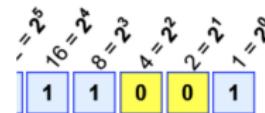
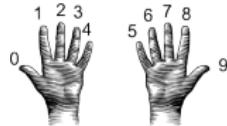


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Binary to Decimal: Converting Between Bases

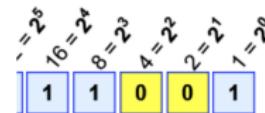
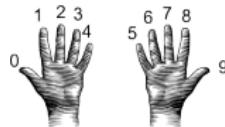


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Binary to Decimal: Converting Between Bases

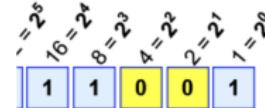
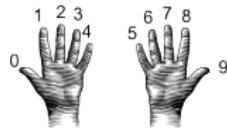


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Binary to Decimal: Converting Between Bases

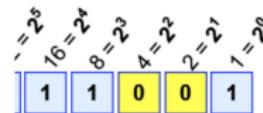
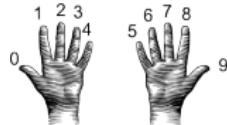


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Binary to Decimal: Converting Between Bases



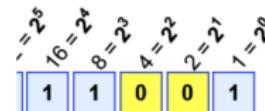
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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:

Binary to Decimal: Converting Between Bases



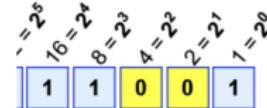
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



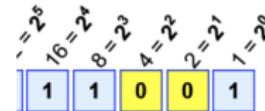
$$\text{Example: } 1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

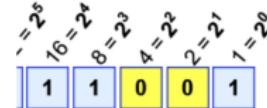
- Set sum = last digit.
- Multiply next digit by 2^1 . Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$. Add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

Binary to Decimal: Converting Between Bases



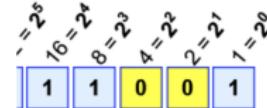
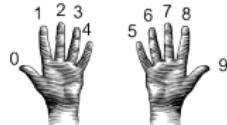
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

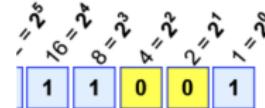
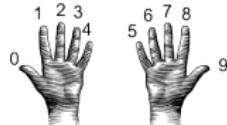
Sum starts with: 1

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

$1 \times 4 = 4$. Add 4 to sum: 5

$1 \times 8 = 8$. Add 8 to sum:

Binary to Decimal: Converting Between Bases



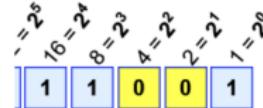
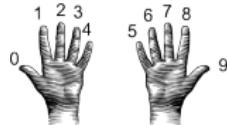
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



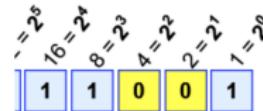
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



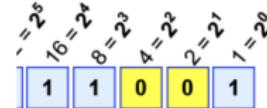
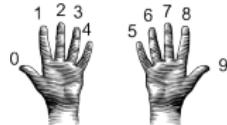
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



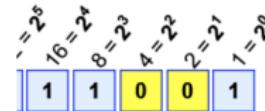
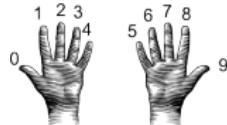
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



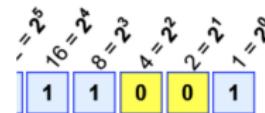
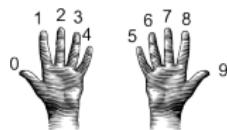
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



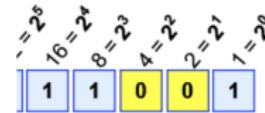
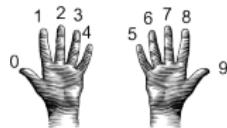
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases

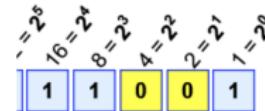
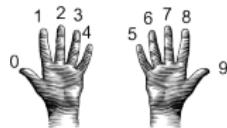


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

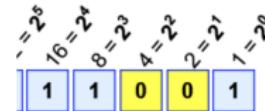
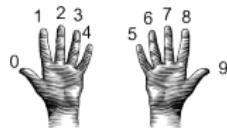
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

Binary to Decimal: Converting Between Bases



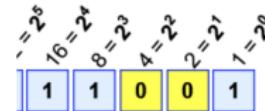
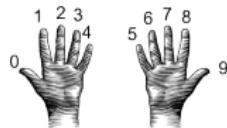
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

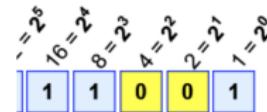
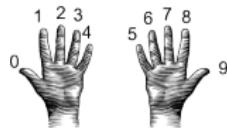
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

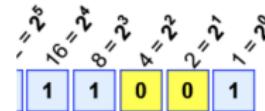
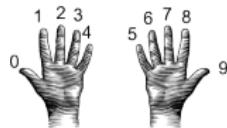
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

- Example: What is 10100100 in decimal?

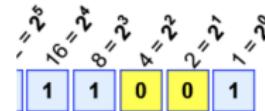
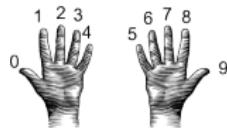
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases

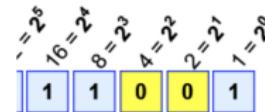
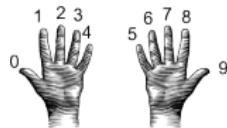


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

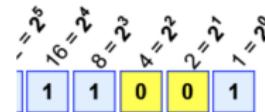
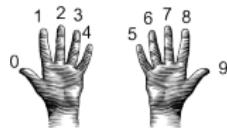
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

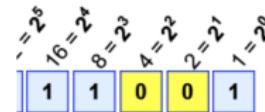
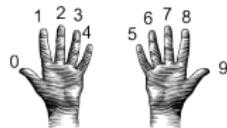
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

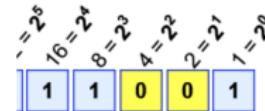
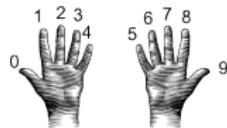
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

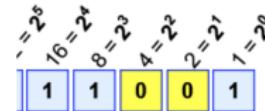
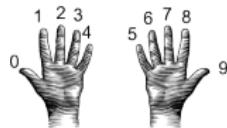
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

$1 \times 32 = 32$. Add 32 to sum: 36

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

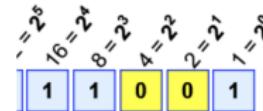
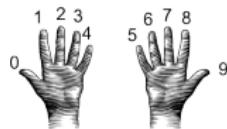
$0 \times 8 = 0$. Add 0 to sum: 4

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

$1 \times 32 = 32$. Add 32 to sum: 36

$0 \times 64 = 0$. Add 0 to sum:

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

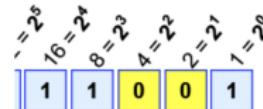
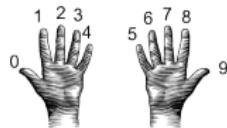
$0 \times 8 = 0$. Add 0 to sum: 4

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

$1 \times 32 = 32$. Add 32 to sum: 36

$0 \times 64 = 0$. Add 0 to sum: 36

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

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

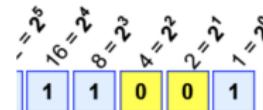
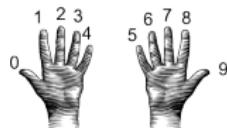
$0 \times 16 = 0$. Add 0 to sum: 4

$1 \times 32 = 32$. Add 32 to sum: 36

$0 \times 64 = 0$. Add 0 to sum: 36

$1 \times 128 = 0$. Add 128 to sum:

Binary to Decimal: Converting Between Bases

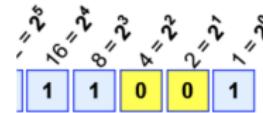
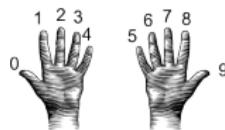


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases



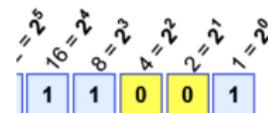
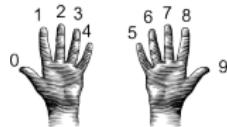
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

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

The answer is 164.

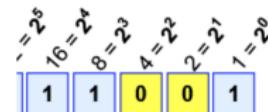
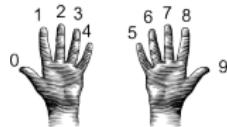
Design Challenge: Incrementers



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Simplest arithmetic: add one ("increment") a variable.

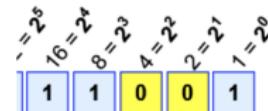
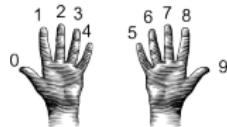
Design Challenge: Incrementers



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

Design Challenge: Incrementers

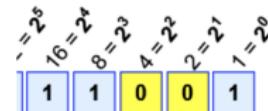
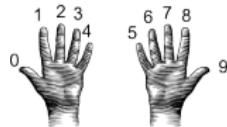


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

Design Challenge: Incrementers



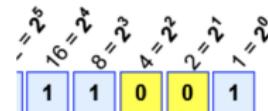
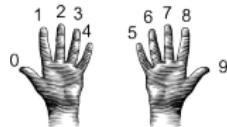
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.

Design Challenge: Incrementers



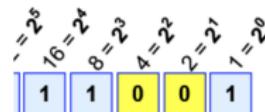
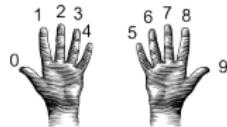
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.
Example: "forty one" → "forty two"

Design Challenge: Incrementers



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

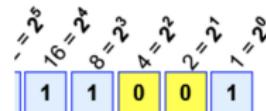
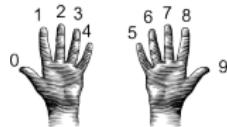
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.
Example: "forty one" → "forty two"

Hint: Convert to numbers, increment, and convert back to strings.

Design Challenge: Incrementers



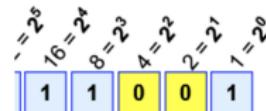
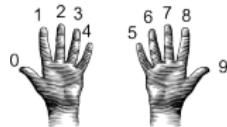
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.
Example: "forty one" → "forty two"
Hint: Convert to numbers, increment, and convert back to strings.
- Challenge: Write an algorithm for incrementing binary numbers.

Design Challenge: Incrementers



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.
Example: "forty one" → "forty two"

Hint: Convert to numbers, increment, and convert back to strings.

- Challenge: Write an algorithm for incrementing binary numbers.

Example: "1001" → "1010"

Recap



- Searching through data is a common task– built-in functions and standard design patterns for this.

Recap



- Searching through data is a common task— built-in functions and standard design patterns for this.
- Programming languages can be classified by the level of abstraction and direct access to data.

Recap



- Searching through data is a common task— built-in functions and standard design patterns for this.
- Programming languages can be classified by the level of abstraction and direct access to data.
- WeMIPS simplified machine language

Recap



- Searching through data is a common task— built-in functions and standard design patterns for this.
- Programming languages can be classified by the level of abstraction and direct access to data.
- WeMIPS simplified machine language
- Converting between Bases

Final Overview: Format

- The exam is 2 hours long.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
 - ▶ More on logistics next lecture.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
 - ▶ More on logistics next lecture.
- Past exams available on webpage (includes answer keys).

Exam Options

Exam Times:

FINAL EXAM, VERSION 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

14 December 2022

Exam Rules

- None of your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper folded in half.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication, sharing unfair advantage, and fabrication of records and official documents) as serious violations of institutional standards. The Hunter College Code of Conduct, CCF Policy on Academic Honesty and self-purge rates of academic dishonesty according to the Hunter College Code of Conduct.

| | |
|--|--|
| Indication that all cases of academic dishonesty will be reported to the Dean of Students and all grades will be deducted. | |
| Name: | |
| Social Security: | |
| Email: | |
| Signature: | |

Exam Options

Exam Times:

- Default Regular Time: Monday, December 19, 9-11am.

FINAL EXAM, VERSION 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

19 December 2022

Exam Rules

- None of your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper folded in half.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication, sharing unfair advantage, and fabrication of records and official documents) as serious violations of student conduct. The Hunter College Code of Conduct and CUNY Policy on Academic Honesty and Self-Possessive Cases of Academic Dishonesty according to the Hunter College website.

| | |
|--|--|
| I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and all records will be retained. | |
| Name: | |
| Sigil: | |
| Email: | |
| Signature: | |

Exam Options

Exam Times:

- Default Regular Time: Monday, December 19, 9-11am.
- Alternate Time: Friday, 16 December, 8am-10am.

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

19 December 2022

Exam Rules

- None of your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper folded in half.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication, sharing unfair advantage, and fabrication of records and official documents) as serious violations of student conduct. The Hunter College Code of Conduct and CUNY Policy on Academic Honesty and self-punitive rates of academic dishonesty according to the Hunter College

| | |
|--|--|
| I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and all records will be retained. | |
| Name: | |
| Sigil: | |
| Email: | |
| Signature: | |

Exam Options

Exam Times:

- Default Regular Time: Monday, December 19, 9-11am.
- Alternate Time: Friday, 16 December, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 5 December. If you have not done so already, email me no later than 5 December.

FINAL EXAM, VERSION 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

14 December 2018

Exam Rules

- None of your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that you can write on.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commutation, obtaining unfair advantage, and fabrication of records and official documents) as serious violations of its educational mission. The Hunter College Academic Dishonesty and CUEP Policy on Academic Integrity and self-purging rates of academic dishonesty according to the Hunter College

| |
|--|
| <small>Indication that all cases of academic dishonesty will be reported to the Dean of Students and self-purging conditions</small> |
| Name: _____ |
| Social Security Number: _____ |
| Email: _____ |
| Signature: _____ |

Exam Options

Exam Times:

- Default Regular Time: Monday, December 19, 9-11am.
- Alternate Time: Friday, 16 December, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 5 December. If you have not done so already, email me no later than 5 December.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on December 19.**

FINAL EXAM, VERSION 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

14 December 2018

Exam Rules

- More of your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that you can write on.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commutation, obtaining unfair advantage, and fabrication of records and official documents) as serious violations of student conduct. The Hunter College Academic Dishonesty Policy can be found at [CUNY Policy on Academic Integrity and Self-Percived Rates of Academic Dishonesty according to the Hunter College](#).

| | |
|---|--|
| I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and will result in discipline. | |
| Name: | |
| Social Security: | |
| Email: | |
| Signature: | |

Exam Options

Exam Times:

- Default Regular Time: Monday, December 19, 9-11am.
- Alternate Time: Friday, 16 December, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 5 December. If you have not done so already, email me no later than 5 December.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on December 19.**
- If you choose to take the early date, **you will not be given access to the exam on December 19 even if you miss the early exam.**

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

14 December 2024

Exam Rules

- None of your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper listing formulas and definitions.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commutation, obtaining unfair advantage, and fabrication of records and official documents) as serious violations of student conduct. The Hunter College Policy on Academic Integrity and the CUNY Policy on Academic Honesty and Self-Percived Cases of Academic Dishonesty according to the Hunter College

| | |
|---|--|
| I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and will result in discipline. | |
| Name: | |
| Social: | |
| Email: | |
| Signature: | |

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (**programs 51-55**)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (**programs 51-55**)
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5pm

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (**programs 51-55**)
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5pm
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)

Lecture Slips & Writing Boards



- Hand your lecture slip to a UTA.
- Return writing boards as you leave.