

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()

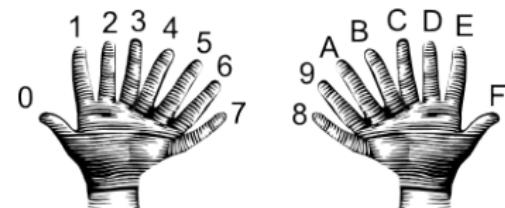
- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items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

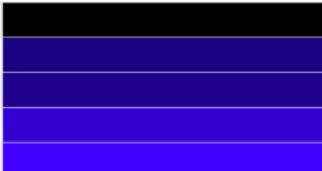
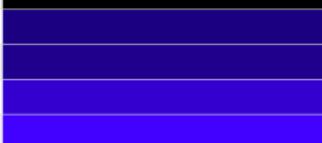
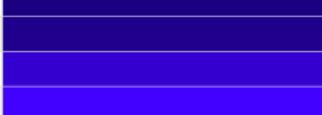
```
1  for num in [2,4,6,8,10]:  
2      print(num)  
  
3  
  
4  sum = 0  
5  for x in range(0,12,2):  
6      print(x)  
7      sum += x  
  
8  
  
9  print(sum)  
  
10  
  
11 for c in 'ABCD':  
12     print(c)
```

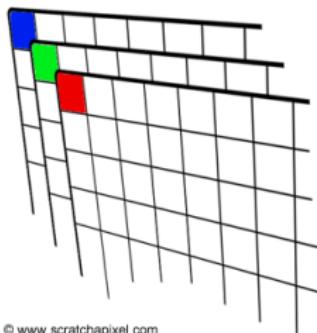
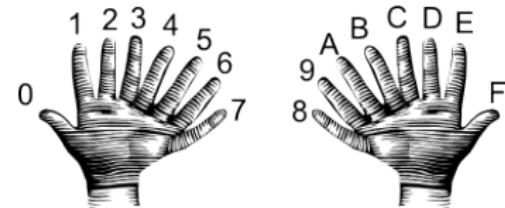
Week 3: colors, hex, slices, numpy & images

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

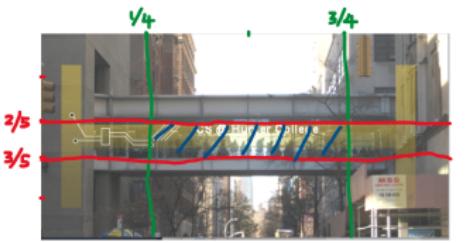
```
23 #make the left 1/4 area gray  
24 img[:, :width//4] = [0.5, 0.5, 0.5, 1]  
25  
26 #make the right 1/4 area gray  
27 img[:, width*3//4:] = [0.5, 0.5, 0.5, 1]  
28 #img[:, width*3//4:] same as  
29 #img[:, -width//4:]  
30 #width*3//4 from LEFT same as  
31 #width//4 from RIGHT  
32 #img[:, -width//4:]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

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.")
```

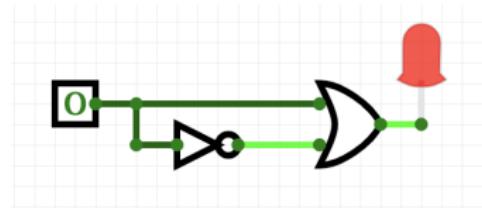
Week 5: logical operators, truth tables & logical circuits: II

```
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
1690,1,037,203,727,784,2,037
1771,21843,3623,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,67541,6250,6840,2155,49734
1820,123704,11487,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,18013,14049,5346,10965,391114
1850,35544,21800,18850,5346,10965,391115
1860,613469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33029,1911801
1890,1367711,707128,67203,51980,33029,2141514
1900,185093,1166582,152999,200567,67921,2437202
1910,223342,1634351,284041,430980,8569,4766803
1920,2210103,2018354,446035,446035,73201,11651,50048
1930,1867112,1578128,1578128,1578128,1578128,4930446
1940,1889924,2498285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7891957
1960,1690311,2000219,1809001,1480000,1480000,781984
1970,1539231,1865070,1747101,1472701,135443,784640
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2485326,2229379,1332650,419782,8080879
2010,1583873,2504705,2216722,1385108,447515,8175133
2015,1444518,2436733,2339150,1454446,474558,8059405

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
1890,4937,2037,727,788,1115
1871,21843,3623,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,7595
1810,63545,5740,6442,1755,4543,7595
1820,123704,11487,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,18013,14049,5346,10965,391114
1850,35549,12890,12890,12890,12890,12890
1860,813469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59945,5653,51980,33029,1911803
1890,1367711,66582,61582,5653,51980,33029,1911803
1900,185093,116582,152999,200567,67621,2437202
1910,223342,1634351,284041,430980,8569,4766803
1920,2210103,2018354,446070,446070,732013,116582,152999
1930,1667113,1667113,1667113,1667113,1667113,1667113,1667113
1940,1889924,2498285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1696211,1696211,1696211,1696211,1696211,1696211,1696211
1970,1359231,1359231,1359231,1359231,1359231,1359231,1359231
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2485326,2229379,1332450,419728,8080879
2010,1583873,2504705,2277722,1385108,4175133,8175133
2015,1444018,2646733,2339150,1459444,474558,8056405

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  
1810,67535,6240,6840,1755,4543,79334  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,19013,14084,5346,10965,391114  
1850,355449,218903,18895,10835,5346,411515  
1860,813449,279122,32093,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911803  
1890,1384473,72012,6254,58100,33029,2347134  
1900,1850593,116582,152999,200567,67921,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44601,44601,73201,11651,50048  
1930,18671128,2091128,15784,15784,5821,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2738175,1550949,1451277,191555,7981984  
1970,1539231,2467071,1472170,1235443,7071646  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,7322564  
2000,1537195,2485326,2229379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,8175133  
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  
1690,203,2037,...,727,7181  
1771,21843,28243,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,60000,60000,1750,4543,75934  
1820,123704,11487,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31510,21013,14000,5346,10965,391114  
1850,35549,21890,14895,5346,10965,391115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33021,1911801  
1890,1385000,720000,68000,51980,33021,1911804  
1900,1850993,116582,152999,200567,67621,2437202  
1910,223342,1634351,2841,430980,8569,476683  
1920,2246103,2018354,44601,44601,73201,11671,50048  
1930,2667128,2079128,2079128,2079128,15821,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690000,2319319,1809000,1451277,191555,7981984  
1970,1539231,2465071,1472701,1472701,135443,7981984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,379977,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,474558,8155405  
2015,1444518,2546733,2339150,1459446,474558,8155405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

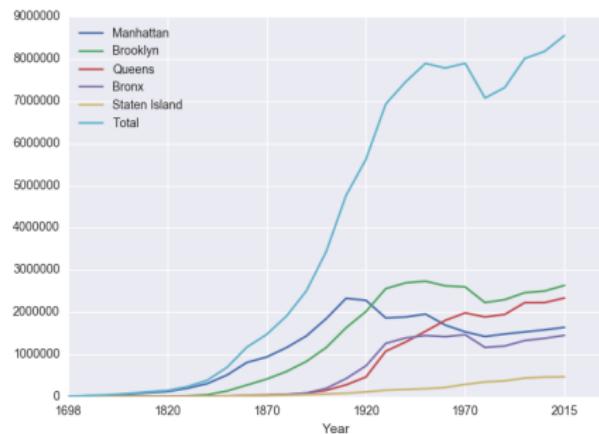
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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,Borough,Population  
1699,Manhattan, Brooklyn, Queens, Bronx, Staten Island, Total  
1771,21843,36231,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75935  
1810,70531,6354,7041,1831,5341,93734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14041,5348,10965,391114  
1850,455441,21800,18851,6835,12515,51154  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33051,1911801  
1890,1400000,720000,680000,518000,380000,215134  
1900,1850093,1165852,152999,200567,67921,2437202  
1910,2331542,1634351,2841,430980,8569,476683  
1920,2281103,2018354,44601,4461,73201,116500,50048  
1930,2667103,2186128,35254,35254,5831,4930446  
1940,1889924,2690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7091957  
1960,1690000,2300000,1800000,1600000,1200000,6781984  
1970,1539231,2465071,2471071,2471071,235443,7084640  
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



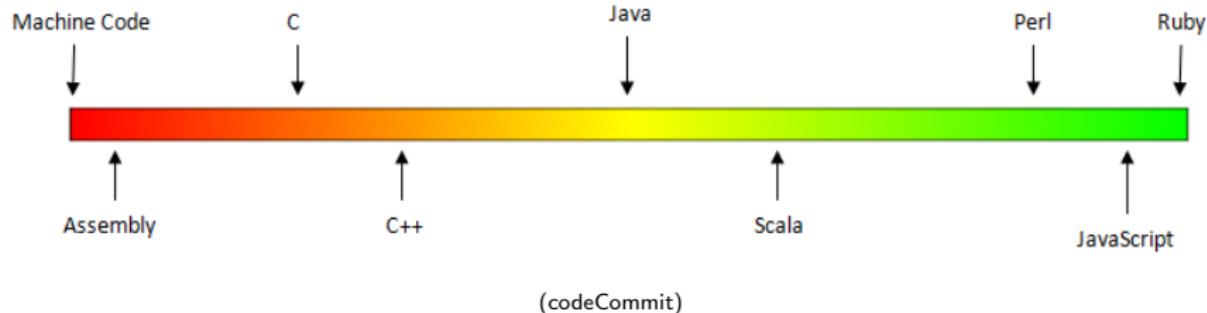
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
 - ▶ Primitive: `int`, `float`, `bool`, `string`;
 - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`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

Today's Topics



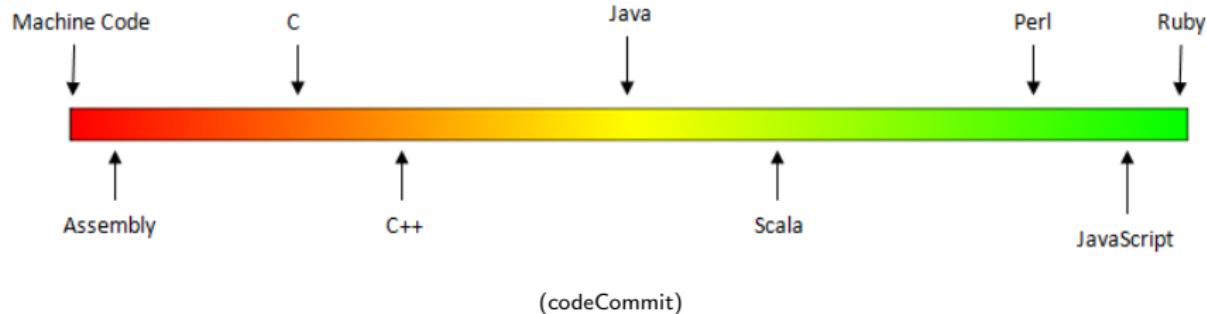
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic

Low-Level vs. High-Level Languages



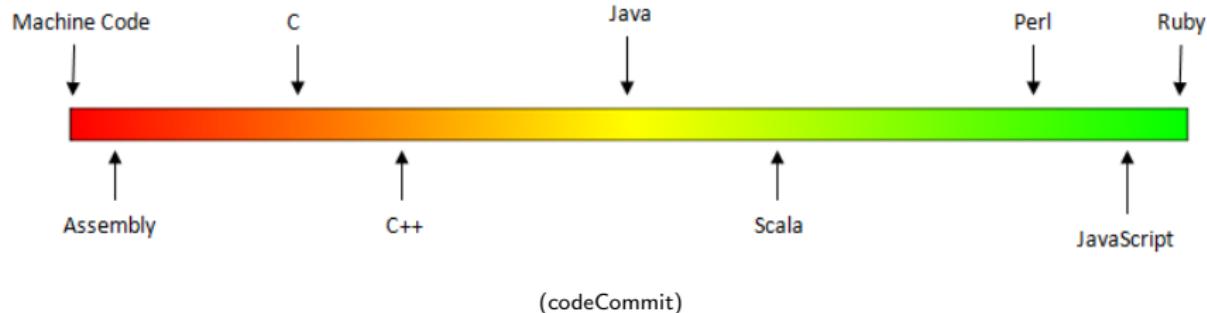
- Can view programming languages on a continuum.

Low-Level vs. High-Level Languages



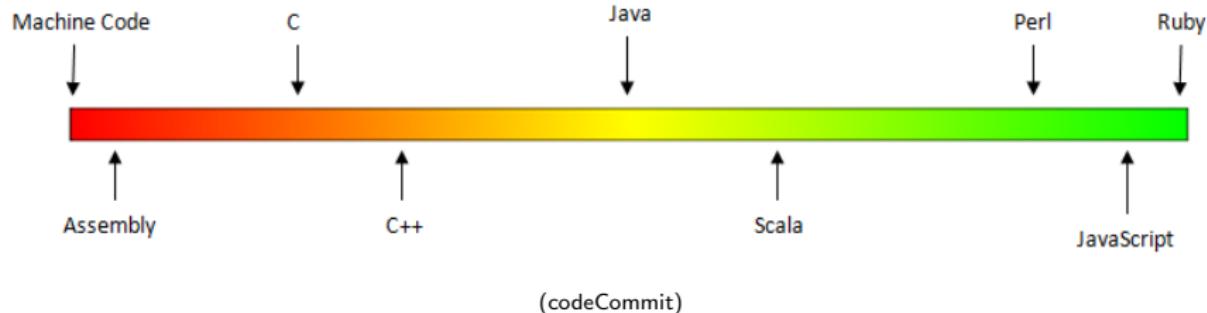
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

Low-Level vs. High-Level Languages



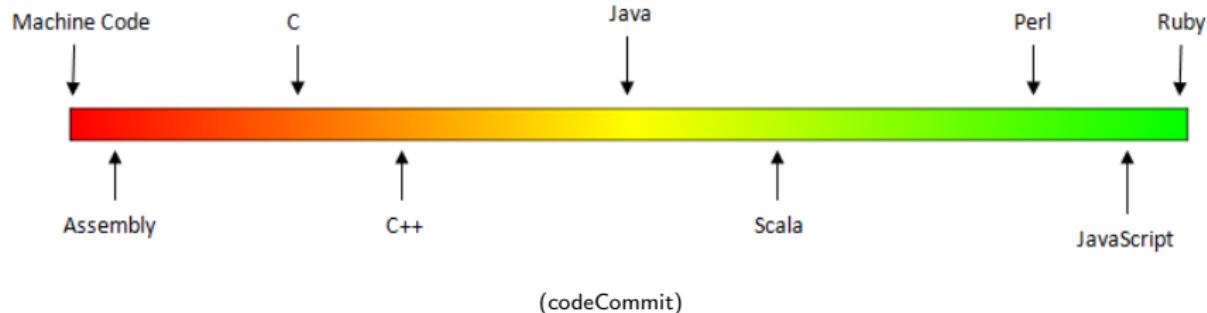
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

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

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

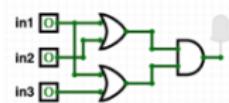


Data
&
Instructions

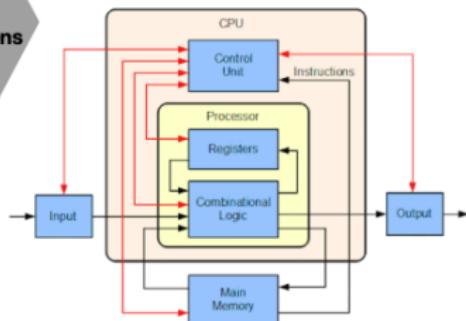


```
0110100011100100110000101  
1100100011011011011011011  
00111100001101000101011  
001011101000100000111111  
11010111101000111011011  
00100101010110011001000  
01001011010101001000001  
1110010000011010011101  
011000101101100011010101  
010000100000001100100000  
011001010110011001011  
100101000000110110011101  
01101101010010000010000  
1110001000000010000101100  
01100011011101101101101  
001101100101110010001101  
001000001101000110111100  
00010...10010001101000100  
111010001101110110111100  
011011010110011010011001  
011100100101110010001101  
001000001101000110111100  
00010...10010001101000100
```

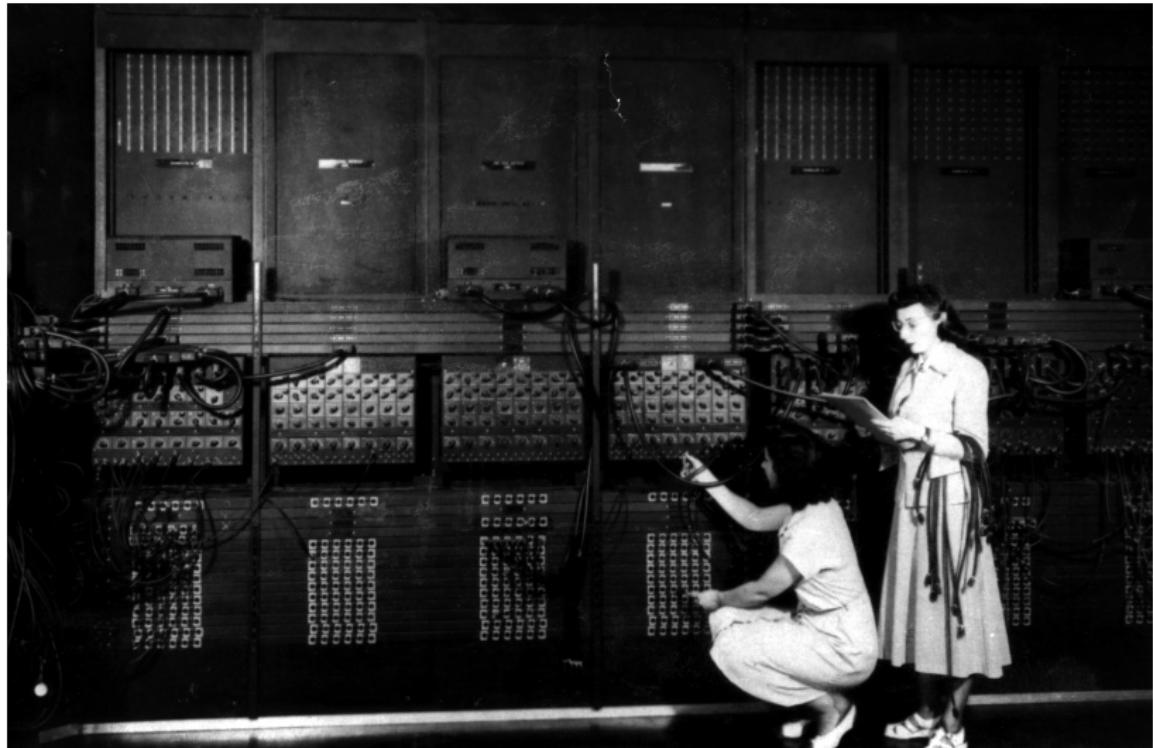
```
def totalWithTax(Food,tip):  
    total = 8  
    tax = 0.0875  
    total = Food + food * tax  
    total = total + tip  
    return(total)
```



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



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.

```

F
PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

BREAK

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

```

(wiki)

Machine Language



(wiki)

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

Machine Language

(wiki)

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

“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

```
# Store 'Hello world!' at the top of the stack
ADDI $t0,$zero,72 #8
LD $t0,1($sp)
ADD $t0,$t0,101 #e
ADD $t0,1($sp),108 #1
ADD $t0,2($sp),108 #1
ADD $t0,3($sp),108 #1
ADD $t0,4($sp),108 #1
ADD $t0,5($sp),108 #1
ADD $t0,6($sp),108 #1
ADD $t0,7($sp),108 #1
ADD $t0,8($sp),108 #1
ADD $t0,9($sp),108 #1
ADD $t0,10($sp),108 #d
ADD $t0,11($sp),33 #1
ADD $t0,12($sp),0 #(null)
ED $t0,12($sp)

ADDI $t0,$zero,4 #4 is for print string
ADDI $t0,$t0,10 #10 is for print null
JAL $t0,syscall #0 points to the log
```

(Demo with WeMIPS)

MIPS Commands

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

MIPS Commands

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

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

The screenshot shows a debugger interface with the following details:

Assembly View:

```
1 # Shows "Hello world" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    addi   $s0,$zero,101 # $s0 = 101
6    addi   $t0,$zero,102 # $t0 = 102
7    addi   $t1,$zero,103 # $t1 = 103
8    addi   $t2,$zero,104 # $t2 = 104
9    addi   $t3,$zero,105 # $t3 = 105
10   addi   $t4,$zero,106 # $t4 = 106
11   addi   $t5,$zero,107 # $t5 = 107
12   addi   $t6,$zero,108 # $t6 = 108
13   addi   $t7,$zero,109 # $t7 = 109
14   addi   $t8,$zero,110 # $t8 = 110
15   addi   $t9,$zero,111 # $t9 = 111
16   addi   $t10,$zero,112 # $t10 = 112
17   addi   $t11,$zero,113 # $t11 = 113
18   addi   $t12,$zero,114 # $t12 = 114
19   addi   $t13,$zero,115 # $t13 = 115
20   addi   $t14,$zero,116 # $t14 = 116
21   addi   $t15,$zero,117 # $t15 = 117
22   addi   $t16,$zero,118 # $t16 = 118
23   addi   $t17,$zero,119 # $t17 = 119
24   addi   $t18,$zero,120 # $t18 = 120
25   addi   $t19,$zero,121 # $t19 = 121
26   addi   $t20,$zero,122 # $t20 = 122
27   addi   $t21,$zero,123 # $t21 = 123
28   addi   $t22,$zero,124 # $t22 = 124
29   addi   $t23,$zero,125 # $t23 = 125
30   addi   $t24,$zero,126 # $t24 = 126
31   addi   $t25,$zero,127 # $t25 = 127
32   addi   $t26,$zero,128 # $t26 = 128
33   addi   $t27,$zero,129 # $t27 = 129
34   addi   $t28,$zero,130 # $t28 = 130
35   addi   $t29,$zero,131 # $t29 = 131
36   addi   $t30,$zero,132 # $t30 = 132
37   addi   $t31,$zero,133 # $t31 = 133
38   addi   $t32,$zero,134 # $t32 = 134
39   addi   $t33,$zero,135 # $t33 = 135
40   addi   $t34,$zero,136 # $t34 = 136
41   addi   $t35,$zero,137 # $t35 = 137
42   addi   $t36,$zero,138 # $t36 = 138
43   addi   $t37,$zero,139 # $t37 = 139
44   addi   $t38,$zero,140 # $t38 = 140
45   addi   $t39,$zero,141 # $t39 = 141
46   addi   $t40,$zero,142 # $t40 = 142
47   addi   $t41,$zero,143 # $t41 = 143
48   addi   $t42,$zero,144 # $t42 = 144
49   addi   $t43,$zero,145 # $t43 = 145
50   addi   $t44,$zero,146 # $t44 = 146
51   addi   $t45,$zero,147 # $t45 = 147
52   addi   $t46,$zero,148 # $t46 = 148
53   addi   $t47,$zero,149 # $t47 = 149
54   addi   $t48,$zero,150 # $t48 = 150
55   addi   $t49,$zero,151 # $t49 = 151
56   addi   $t50,$zero,152 # $t50 = 152
57   addi   $t51,$zero,153 # $t51 = 153
58   addi   $t52,$zero,154 # $t52 = 154
59   addi   $t53,$zero,155 # $t53 = 155
60   addi   $t54,$zero,156 # $t54 = 156
61   addi   $t55,$zero,157 # $t55 = 157
62   addi   $t56,$zero,158 # $t56 = 158
63   addi   $t57,$zero,159 # $t57 = 159
64   addi   $t58,$zero,160 # $t58 = 160
65   addi   $t59,$zero,161 # $t59 = 161
66   addi   $t60,$zero,162 # $t60 = 162
67   addi   $t61,$zero,163 # $t61 = 163
68   addi   $t62,$zero,164 # $t62 = 164
69   addi   $t63,$zero,165 # $t63 = 165
70   addi   $t64,$zero,166 # $t64 = 166
71   addi   $t65,$zero,167 # $t65 = 167
72   addi   $t66,$zero,168 # $t66 = 168
73   addi   $t67,$zero,169 # $t67 = 169
74   addi   $t68,$zero,170 # $t68 = 170
75   addi   $t69,$zero,171 # $t69 = 171
76   addi   $t70,$zero,172 # $t70 = 172
77   addi   $t71,$zero,173 # $t71 = 173
78   addi   $t72,$zero,174 # $t72 = 174
79   addi   $t73,$zero,175 # $t73 = 175
80   addi   $t74,$zero,176 # $t74 = 176
81   addi   $t75,$zero,177 # $t75 = 177
82   addi   $t76,$zero,178 # $t76 = 178
83   addi   $t77,$zero,179 # $t77 = 179
84   addi   $t78,$zero,180 # $t78 = 180
85   addi   $t79,$zero,181 # $t79 = 181
86   addi   $t80,$zero,182 # $t80 = 182
87   addi   $t81,$zero,183 # $t81 = 183
88   addi   $t82,$zero,184 # $t82 = 184
89   addi   $t83,$zero,185 # $t83 = 185
90   addi   $t84,$zero,186 # $t84 = 186
91   addi   $t85,$zero,187 # $t85 = 187
92   addi   $t86,$zero,188 # $t86 = 188
93   addi   $t87,$zero,189 # $t87 = 189
94   addi   $t88,$zero,190 # $t88 = 190
95   addi   $t89,$zero,191 # $t89 = 191
96   addi   $t90,$zero,192 # $t90 = 192
97   addi   $t91,$zero,193 # $t91 = 193
98   addi   $t92,$zero,194 # $t92 = 194
99   addi   $t93,$zero,195 # $t93 = 195
100  addi   $t94,$zero,196 # $t94 = 196
101  addi   $t95,$zero,197 # $t95 = 197
102  addi   $t96,$zero,198 # $t96 = 198
103  addi   $t97,$zero,199 # $t97 = 199
104  addi   $t98,$zero,200 # $t98 = 200
105  addi   $t99,$zero,201 # $t99 = 201
106  addi   $t100,$zero,202 # $t100 = 202
107  addi   $t101,$zero,203 # $t101 = 203
108  addi   $t102,$zero,204 # $t102 = 204
109  addi   $t103,$zero,205 # $t103 = 205
110  addi   $t104,$zero,206 # $t104 = 206
111  addi   $t105,$zero,207 # $t105 = 207
112  addi   $t106,$zero,208 # $t106 = 208
113  addi   $t107,$zero,209 # $t107 = 209
114  addi   $t108,$zero,210 # $t108 = 210
115  addi   $t109,$zero,211 # $t109 = 211
116  addi   $t110,$zero,212 # $t110 = 212
117  addi   $t111,$zero,213 # $t111 = 213
118  addi   $t112,$zero,214 # $t112 = 214
119  addi   $t113,$zero,215 # $t113 = 215
120  addi   $t114,$zero,216 # $t114 = 216
121  addi   $t115,$zero,217 # $t115 = 217
122  addi   $t116,$zero,218 # $t116 = 218
123  addi   $t117,$zero,219 # $t117 = 219
124  addi   $t118,$zero,220 # $t118 = 220
125  addi   $t119,$zero,221 # $t119 = 221
126  addi   $t120,$zero,222 # $t120 = 222
127  addi   $t121,$zero,223 # $t121 = 223
128  addi   $t122,$zero,224 # $t122 = 224
129  addi   $t123,$zero,225 # $t123 = 225
130  addi   $t124,$zero,226 # $t124 = 226
131  addi   $t125,$zero,227 # $t125 = 227
132  addi   $t126,$zero,228 # $t126 = 228
133  addi   $t127,$zero,229 # $t127 = 229
134  addi   $t128,$zero,230 # $t128 = 230
135  addi   $t129,$zero,231 # $t129 = 231
136  addi   $t130,$zero,232 # $t130 = 232
137  addi   $t131,$zero,233 # $t131 = 233
138  addi   $t132,$zero,234 # $t132 = 234
139  addi   $t133,$zero,235 # $t133 = 235
140  addi   $t134,$zero,236 # $t134 = 236
141  addi   $t135,$zero,237 # $t135 = 237
142  addi   $t136,$zero,238 # $t136 = 238
143  addi   $t137,$zero,239 # $t137 = 239
144  addi   $t138,$zero,240 # $t138 = 240
145  addi   $t139,$zero,241 # $t139 = 241
146  addi   $t140,$zero,242 # $t140 = 242
147  addi   $t141,$zero,243 # $t141 = 243
148  addi   $t142,$zero,244 # $t142 = 244
149  addi   $t143,$zero,245 # $t143 = 245
150  addi   $t144,$zero,246 # $t144 = 246
151  addi   $t145,$zero,247 # $t145 = 247
152  addi   $t146,$zero,248 # $t146 = 248
153  addi   $t147,$zero,249 # $t147 = 249
154  addi   $t148,$zero,250 # $t148 = 250
155  addi   $t149,$zero,251 # $t149 = 251
156  addi   $t150,$zero,252 # $t150 = 252
157  addi   $t151,$zero,253 # $t151 = 253
158  addi   $t152,$zero,254 # $t152 = 254
159  addi   $t153,$zero,255 # $t153 = 255
160  addi   $t154,$zero,256 # $t154 = 256
161  addi   $t155,$zero,257 # $t155 = 257
162  addi   $t156,$zero,258 # $t156 = 258
163  addi   $t157,$zero,259 # $t157 = 259
164  addi   $t158,$zero,260 # $t158 = 260
165  addi   $t159,$zero,261 # $t159 = 261
166  addi   $t160,$zero,262 # $t160 = 262
167  addi   $t161,$zero,263 # $t161 = 263
168  addi   $t162,$zero,264 # $t162 = 264
169  addi   $t163,$zero,265 # $t163 = 265
170  addi   $t164,$zero,266 # $t164 = 266
171  addi   $t165,$zero,267 # $t165 = 267
172  addi   $t166,$zero,268 # $t166 = 268
173  addi   $t167,$zero,269 # $t167 = 269
174  addi   $t168,$zero,270 # $t168 = 270
175  addi   $t169,$zero,271 # $t169 = 271
176  addi   $t170,$zero,272 # $t170 = 272
177  addi   $t171,$zero,273 # $t171 = 273
178  addi   $t172,$zero,274 # $t172 = 274
179  addi   $t173,$zero,275 # $t173 = 275
180  addi   $t174,$zero,276 # $t174 = 276
181  addi   $t175,$zero,277 # $t175 = 277
182  addi   $t176,$zero,278 # $t176 = 278
183  addi   $t177,$zero,279 # $t177 = 279
184  addi   $t178,$zero,280 # $t178 = 280
185  addi   $t179,$zero,281 # $t179 = 281
186  addi   $t180,$zero,282 # $t180 = 282
187  addi   $t181,$zero,283 # $t181 = 283
188  addi   $t182,$zero,284 # $t182 = 284
189  addi   $t183,$zero,285 # $t183 = 285
190  addi   $t184,$zero,286 # $t184 = 286
191  addi   $t185,$zero,287 # $t185 = 287
192  addi   $t186,$zero,288 # $t186 = 288
193  addi   $t187,$zero,289 # $t187 = 289
194  addi   $t188,$zero,290 # $t188 = 290
195  addi   $t189,$zero,291 # $t189 = 291
196  addi   $t190,$zero,292 # $t190 = 292
197  addi   $t191,$zero,293 # $t191 = 293
198  addi   $t192,$zero,294 # $t192 = 294
199  addi   $t193,$zero,295 # $t193 = 295
200  addi   $t194,$zero,296 # $t194 = 296
201  addi   $t195,$zero,297 # $t195 = 297
202  addi   $t196,$zero,298 # $t196 = 298
203  addi   $t197,$zero,299 # $t197 = 299
204  addi   $t198,$zero,300 # $t198 = 300
205  addi   $t199,$zero,301 # $t199 = 301
206  addi   $t200,$zero,302 # $t200 = 302
207  addi   $t201,$zero,303 # $t201 = 303
208  addi   $t202,$zero,304 # $t202 = 304
209  addi   $t203,$zero,305 # $t203 = 305
210  addi   $t204,$zero,306 # $t204 = 306
211  addi   $t205,$zero,307 # $t205 = 307
212  addi   $t206,$zero,308 # $t206 = 308
213  addi   $t207,$zero,309 # $t207 = 309
214  addi   $t208,$zero,310 # $t208 = 310
215  addi   $t209,$zero,311 # $t209 = 311
216  addi   $t210,$zero,312 # $t210 = 312
217  addi   $t211,$zero,313 # $t211 = 313
218  addi   $t212,$zero,314 # $t212 = 314
219  addi   $t213,$zero,315 # $t213 = 315
220  addi   $t214,$zero,316 # $t214 = 316
221  addi   $t215,$zero,317 # $t215 = 317
222  addi   $t216,$zero,318 # $t216 = 318
223  addi   $t217,$zero,319 # $t217 = 319
224  addi   $t218,$zero,320 # $t218 = 320
225  addi   $t219,$zero,321 # $t219 = 321
226  addi   $t220,$zero,322 # $t220 = 322
227  addi   $t221,$zero,323 # $t221 = 323
228  addi   $t222,$zero,324 # $t222 = 324
229  addi   $t223,$zero,325 # $t223 = 325
230  addi   $t224,$zero,326 # $t224 = 326
231  addi   $t225,$zero,327 # $t225 = 327
232  addi   $t226,$zero,328 # $t226 = 328
233  addi   $t227,$zero,329 # $t227 = 329
234  addi   $t228,$zero,330 # $t228 = 330
235  addi   $t229,$zero,331 # $t229 = 331
236  addi   $t230,$zero,332 # $t230 = 332
237  addi   $t231,$zero,333 # $t231 = 333
238  addi   $t232,$zero,334 # $t232 = 334
239  addi   $t233,$zero,335 # $t233 = 335
240  addi   $t234,$zero,336 # $t234 = 336
241  addi   $t235,$zero,337 # $t235 = 337
242  addi   $t236,$zero,338 # $t236 = 338
243  addi   $t237,$zero,339 # $t237 = 339
244  addi   $t238,$zero,340 # $t238 = 340
245  addi   $t239,$zero,341 # $t239 = 341
246  addi   $t240,$zero,342 # $t240 = 342
247  addi   $t241,$zero,343 # $t241 = 343
248  addi   $t242,$zero,344 # $t242 = 344
249  addi   $t243,$zero,345 # $t243 = 345
250  addi   $t244,$zero,346 # $t244 = 346
251  addi   $t245,$zero,347 # $t245 = 347
252  addi   $t246,$zero,348 # $t246 = 348
253  addi   $t247,$zero,349 # $t247 = 349
254  addi   $t248,$zero,350 # $t248 = 350
255  addi   $t249,$zero,351 # $t249 = 351
256  addi   $t250,$zero,352 # $t250 = 352
257  addi   $t251,$zero,353 # $t251 = 353
258  addi   $t252,$zero,354 # $t252 = 354
259  addi   $t253,$zero,355 # $t253 = 355
260  addi   $t254,$zero,356 # $t254 = 356
261  addi   $t255,$zero,357 # $t255 = 357
262  addi   $t256,$zero,358 # $t256 = 358
263  addi   $t257,$zero,359 # $t257 = 359
264  addi   $t258,$zero,360 # $t258 = 360
265  addi   $t259,$zero,361 # $t259 = 361
266  addi   $t260,$zero,362 # $t260 = 362
267  addi   $t261,$zero,363 # $t261 = 363
268  addi   $t262,$zero,364 # $t262 = 364
269  addi   $t263,$zero,365 # $t263 = 365
270  addi   $t264,$zero,366 # $t264 = 366
271  addi   $t265,$zero,367 # $t265 = 367
272  addi   $t266,$zero,368 # $t266 = 368
273  addi   $t267,$zero,369 # $t267 = 369
274  addi   $t268,$zero,370 # $t268 = 370
275  addi   $t269,$zero,371 # $t269 = 371
276  addi   $t270,$zero,372 # $t270 = 372
277  addi   $t271,$zero,373 # $t271 = 373
278  addi   $t272,$zero,374 # $t272 = 374
279  addi   $t273,$zero,375 # $t273 = 375
280  addi   $t274,$zero,376 # $t274 = 376
281  addi   $t275,$zero,377 # $t275 = 377
282  addi   $t276,$zero,378 # $t276 = 378
283  addi   $t277,$zero,379 # $t277 = 379
284  addi   $t278,$zero,380 # $t278 = 380
285  addi   $t279,$zero,381 # $t279 = 381
286  addi   $t280,$zero,382 # $t280 = 382
287  addi   $t281,$zero,383 # $t281 = 383
288  addi   $t282,$zero,384 # $t282 = 384
289  addi   $t283,$zero,385 # $t283 = 385
290  addi   $t284,$zero,386 # $t284 = 386
291  addi   $t285,$zero,387 # $t285 = 387
292  addi   $t286,$zero,388 # $t286 = 388
293  addi   $t287,$zero,389 # $t287 = 389
294  addi   $t288,$zero,390 # $t288 = 390
295  addi   $t289,$zero,391 # $t289 = 391
296  addi   $t290,$zero,392 # $t290 = 392
297  addi   $t291,$zero,393 # $t291 = 393
298  addi   $t292,$zero,394 # $t292 = 394
299  addi   $t293,$zero,395 # $t293 = 395
300  addi   $t294,$zero,396 # $t294 = 396
301  addi   $t295,$zero,397 # $t295 = 397
302  addi   $t296,$zero,398 # $t296 = 398
303  addi   $t297,$zero,399 # $t297 = 399
304  addi   $t298,$zero,400 # $t298 = 400
305  addi   $t299,$zero,401 # $t299 = 401
306  addi   $t300,$zero,402 # $t300 = 402
307  addi   $t301,$zero,403 # $t301 = 403
308  addi   $t302,$zero,404 # $t302 = 404
309  addi   $t303,$zero,405 # $t303 = 405
310  addi   $t304,$zero,406 # $t304 = 406
311  addi   $t305,$zero,407 # $t305 = 407
312  addi   $t306,$zero,408 # $t306 = 408
313  addi   $t307,$zero,409 # $t307 = 409
314  addi   $t308,$zero,410 # $t308 = 410
315  addi   $t309,$zero,411 # $t309 = 411
316  addi   $t310,$zero,412 # $t310 = 412
317  addi   $t311,$zero,413 # $t311 = 413
318  addi   $t312,$zero,414 # $t312 = 414
319  addi   $t313,$zero,415 # $t313 = 415
320  addi   $t314,$zero,416 # $t314 = 416
321  addi   $t315,$zero,417 # $t315 = 417
322  addi   $t316,$zero,418 # $t316 = 418
323  addi   $t317,$zero,419 # $t317 = 419
324  addi   $t318,$zero,420 # $t318 = 420
325  addi   $t319,$zero,421 # $t319 = 421
326  addi   $t320,$zero,422 # $t320 = 422
327  addi   $t321,$zero,423 # $t321 = 423
328  addi   $t322,$zero,424 # $t322 = 424
329  addi   $t323,$zero,425 # $t323 = 425
330  addi   $t324,$zero,426 # $t324 = 426
331  addi   $t325,$zero,427 # $t325 = 427
332  addi   $t326,$zero,428 # $t326 = 428
333  addi   $t327,$zero,429 # $t327 = 429
334  addi   $t328,$zero,430 # $t328 = 430
335  addi   $t329,$zero,431 # $t329 = 431
336  addi   $t330,$zero,432 # $t330 = 432
337  addi   $t331,$zero,433 # $t331 = 433
338  addi   $t332,$zero,434 # $t332 = 434
339  addi   $t333,$zero,435 # $t333 = 435
340  addi   $t334,$zero,436 # $t334 = 436
341  addi   $t335,$zero,437 # $t335 = 437
342  addi   $t336,$zero,438 # $t336 = 438
343  addi   $t337,$zero,439 # $t337 = 439
344  addi   $t338,$zero,440 # $t338 = 440
345  addi   $t339,$zero,441 # $t339 = 441
346  addi   $t340,$zero,442 # $t340 = 442
347  addi   $t341,$zero,443 # $t341 = 443
348  addi   $t342,$zero,444 # $t342 = 444
349  addi   $t343,$zero,445 # $t343 = 445
350  addi   $t344,$zero,446 # $t344 = 446
351  addi   $t345,$zero,447 # $t345 = 447
352  addi   $t346,$zero,448 # $t346 = 448
353  addi   $t347,$zero,449 # $t347 = 449
354  addi   $t348,$zero,450 # $t348 = 450
355  addi   $t349,$zero,451 # $t349 = 451
356  addi   $t350,$zero,452 # $t350 = 452
357  addi   $t351,$zero,453 # $t351 = 453
358  addi   $t352,$zero,454 # $t352 = 454
359  addi   $t353,$zero,455 # $t353 = 455
360  addi   $t354,$zero,456 # $t354 = 456
361  addi   $t355,$zero,457 # $t355 = 457
362  addi   $t356,$zero,458 # $t356 = 458
363  addi   $t357,$zero,459 # $t357 = 459
364  addi   $t358,$zero,460 # $t358 = 460
365  addi   $t359,$zero,461 # $t359 = 461
366  addi   $t360,$zero,462 # $t360 = 462
367  addi   $t361,$zero,463 # $t361 = 463
368  addi   $t362,$zero,464 # $t362 = 464
369  addi   $t363,$zero,465 # $t363 = 465
370  addi   $t364,$zero,466 # $t364 = 466
371  addi   $t365,$zero,467 # $t365 = 467
372  addi   $t366,$zero,468 # $t366 = 468
373  addi   $t367,$zero,469 # $t367 = 469
374  addi   $t368,$zero,470 # $t368 = 470
375  addi   $t369,$zero,471 # $t369 = 471
376  addi   $t370,$zero,472 # $t370 = 472
377  addi   $t371,$zero,473 # $t371 = 473
378  addi   $t372,$zero,474 # $t372 = 474
379  addi   $t373,$zero,475 # $t373 = 475
380  addi   $t374,$zero,476 # $t374 = 476
381  addi   $t375,$zero,477 # $t375 = 477
382  addi   $t376,$zero,478 # $t376 = 478
383  addi   $t377,$zero,479 # $t377 = 479
384  addi   $t378,$zero,480 # $t378 = 480
385  addi   $t379,$zero,481 # $t379 = 481
386  addi   $t380,$zero,482 # $t380 = 482
387  addi   $t381,$zero,483 # $t381 = 483
388  addi   $t382,$zero,484 # $t382 = 484
389  addi   $t383,$zero,485 # $t383 = 485
390  addi   $t384,$zero,486 # $t384 = 486
391  addi   $t385,$zero,487 # $t385 = 487
392  addi   $t386,$zero,488 # $t386 = 488
393  addi   $t387,$zero,489 # $t387 = 489
394  addi   $t388,$zero,490 # $t388 = 490
395  addi   $t389,$zero,491 # $t389 = 491
396  addi   $t390,$zero,492 # $t390 = 492
397  addi   $t391,$zero,493 # $t391 = 493
398  addi   $t392,$zero,494 # $t392 = 494
399  addi   $t393,$zero,495 # $t393 = 495
400  addi   $t394,$zero,496 # $t394 = 496
401  addi   $t395,$zero,497 # $t395 = 497
402  addi   $t396,$zero,498 # $t396 = 498
403  addi   $t397,$zero,499 # $t397 = 499
404  addi   $t398,$zero,500 # $t398 = 500
405  addi   $t399,$zero,501 # $t399 = 501
406  addi   $t400,$zero,502 # $t400 = 502
407  addi   $t401,$zero,503 # $t401 = 503
408  addi   $t402,$zero,504 # $t402 = 504
409  addi   $t403,$zero,505 # $t403 = 505
410  addi   $t404,$zero,506 # $t404 = 506
411  addi   $t405,$zero,507 # $t405 = 507
412  addi   $t406,$zero,508 # $t406 = 508
413  addi   $t407,$zero,509 # $t407 = 509
414  addi   $t408,$zero,510 # $t408 = 510
415  addi   $t409,$zero,511 # $t409 = 511
416  addi   $t410,$zero,512 # $t410 = 512
417  addi   $t411,$zero,513 # $t411 = 513
418  addi   $t412,$zero,514 # $t412 = 514
419  addi   $t413,$zero,515 # $t413 = 515
420  addi   $t414,$zero,516 # $t414 = 516
421  addi   $t415,$zero,517 # $t415 = 517
422  addi   $t416,$zero,518 # $t416 = 518
423  addi   $t417,$zero,519 # $t417 = 519

```

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 #8
LD $t0,1($sp)
ADD $t0,$t0,101 #e
ADD $t0,1($sp),108 #1
ADD $t0,2($sp),108 #1
ADD $t0,3($sp),108 #1
ADD $t0,4($sp),108 #1
ADD $t0,5($sp),108 #1
ADD $t0,6($sp),108 #1
ADD $t0,7($sp),108 #1
ADD $t0,8($sp),108 #1
ADD $t0,9($sp),108 #1
ADD $t0,10($sp),108 #d
ADD $t0,11($sp),33 #1
ADD $t0,12($sp),0 #(null)
ED $t0,12($sp)

ADDI $t0,$zero,4 #4 is for print string
ADDI $t0,$t0,10 #10 is for print null
JAL $t0,syscall #0 points to the log
```

(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.



A screenshot of a hex editor application. The left pane shows a list of memory pages, each containing a series of memory addresses and their corresponding byte values. The right pane is a detailed view of a specific page, showing memory addresses from 0x00000000 to 0x0000000F. The bytes displayed are: 48 45 4C 4C 4D 4E 4F 4F 4A 4B 4C 4D 4E 4F 4F 4A 4B 4C. Below the address column, there is a column labeled 'Label' which contains the labels 'start' and 'loop'. The bottom of the window has a status bar with the text 'File Edit View Insert Tools Options Help New Page, Set Hex View'.

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

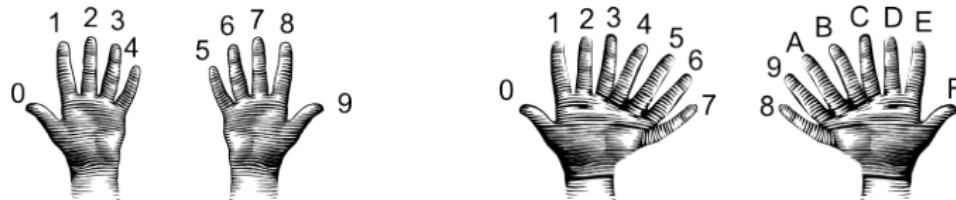


Today's Topics



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

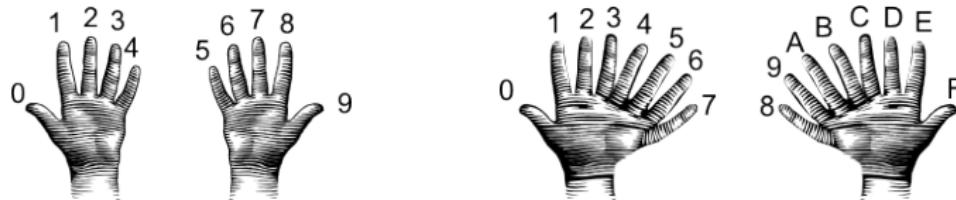
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

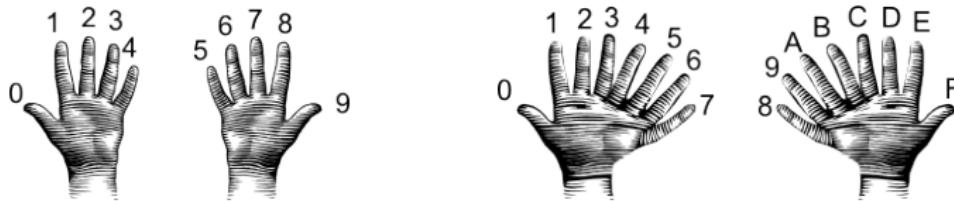
Hexadecimal to Decimal: Converting Between Bases



(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.

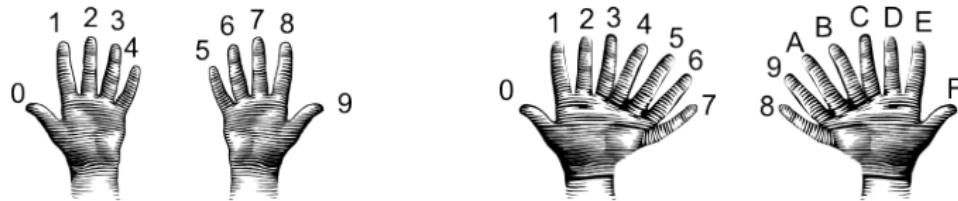
Hexadecimal to Decimal: Converting Between Bases



(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?

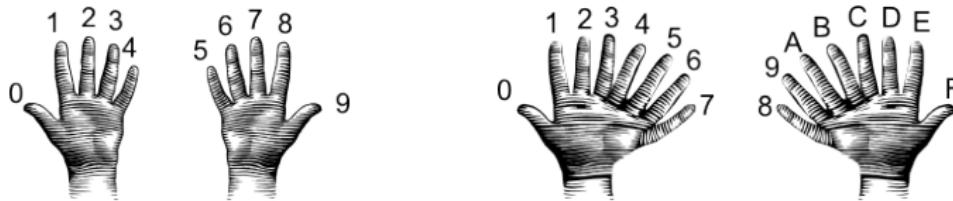
Hexadecimal to Decimal: Converting Between Bases



(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.

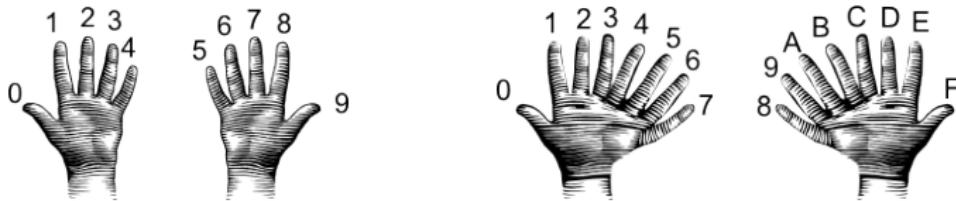
Hexadecimal to Decimal: Converting Between Bases



(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.

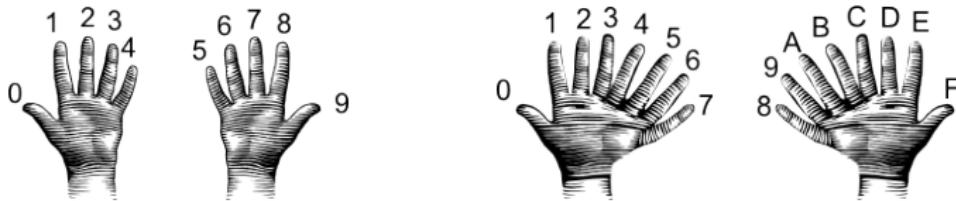
Hexadecimal to Decimal: Converting Between Bases



(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.

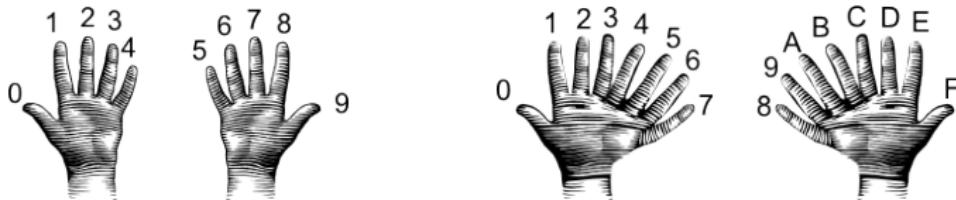
Hexadecimal to Decimal: Converting Between Bases



(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.

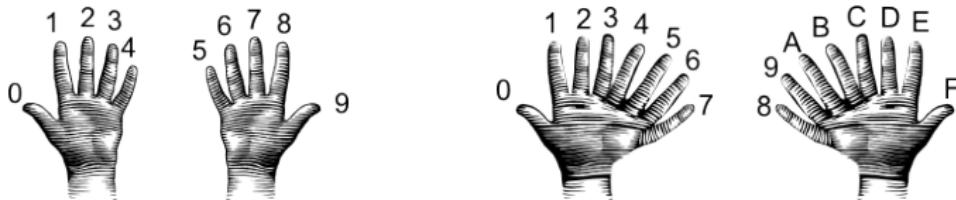
Hexadecimal to Decimal: Converting Between Bases



(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?

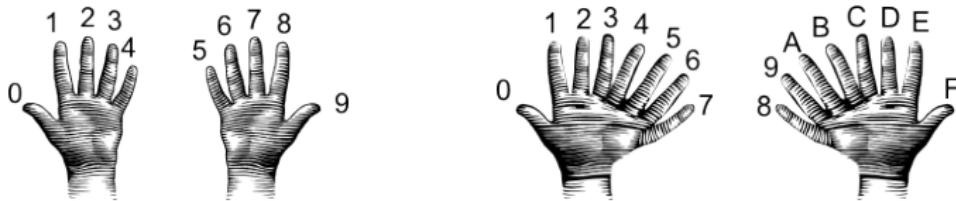
Hexadecimal to Decimal: Converting Between Bases



(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.

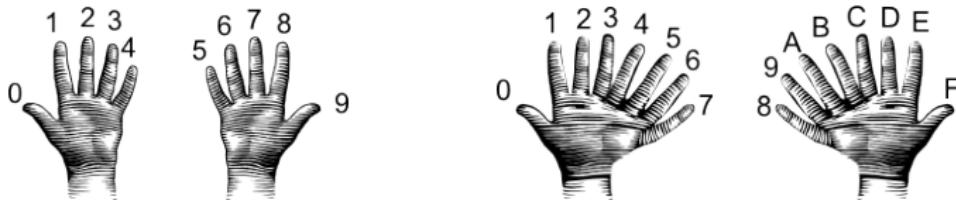
Hexadecimal to Decimal: Converting Between Bases



(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.

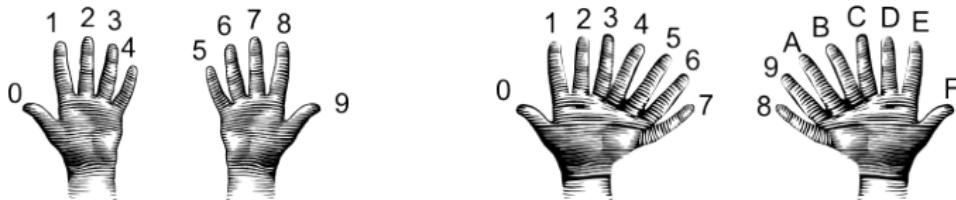
Hexadecimal to Decimal: Converting Between Bases



(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.

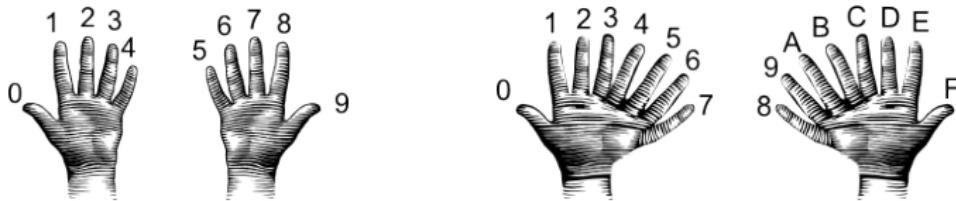
- Example: what is 99 as a decimal number?

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?

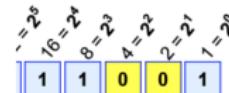
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Answer is 153.

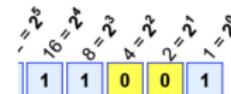
Decimal to Binary: Converting Between Bases



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

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

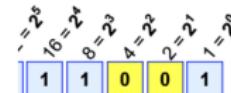
Decimal to Binary: Converting Between Bases



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

- 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.

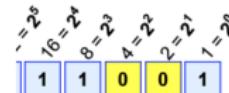
Decimal to Binary: Converting Between Bases



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.

Decimal to Binary: Converting Between Bases

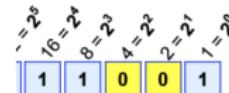


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

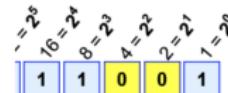


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

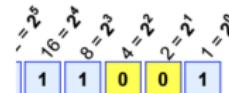


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

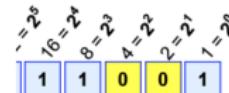


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Decimal to Binary: Converting Between Bases

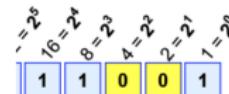


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

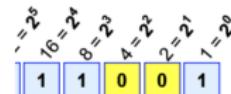


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



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

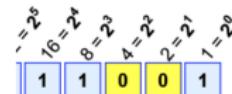
- From decimal to binary:

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

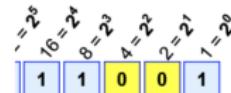
- From decimal to binary:

- 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



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

- From decimal to binary:

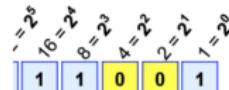
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

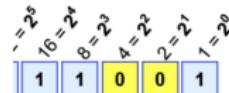
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next 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: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

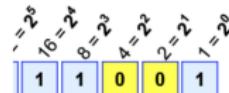
- From decimal to binary:

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

- From decimal to binary:

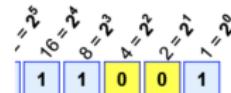
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

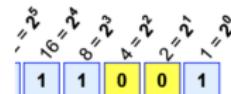
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

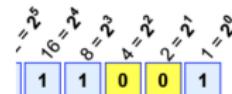
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

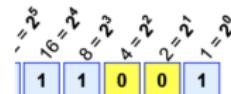
130/128 is 1 rem 2. First digit is 1: 1...

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?

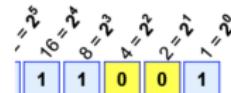
130/128 is 1 rem 2. First digit is 1: 1...

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?

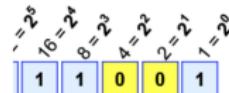
130/128 is 1 rem 2. First digit is 1: 1...

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

- 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...

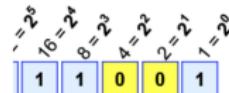
2/64 is 0 rem 2. Next digit is 0: 10...

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.

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...

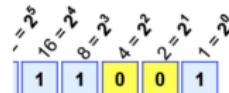
2/64 is 0 rem 2. Next digit is 0: 10...

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

- 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...

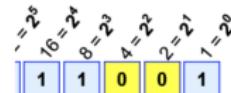
2/64 is 0 rem 2. Next digit is 0: 10...

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...

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...

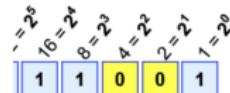
2/32 is 0 rem 2. Next digit is 0: 100...

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.

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...

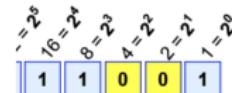
2/32 is 0 rem 2. Next digit is 0: 100...

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. 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...

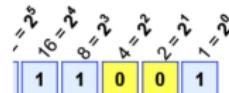
2/32 is 0 rem 2. Next digit is 0: 100...

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...

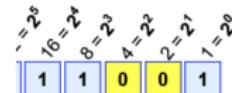
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

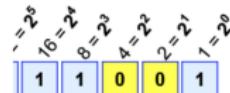
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

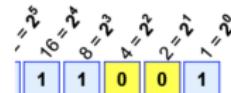
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

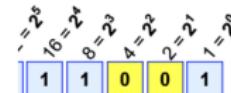
2/8 is 0 rem 2. Next digit is 0: 10000...

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...

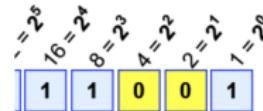
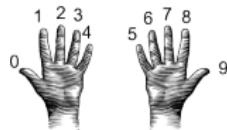
2/4 is 0 remainder 2. Next digit is 0: 100000...

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

Adding the last remainder: 10000010



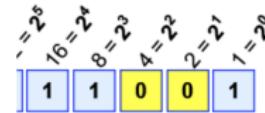
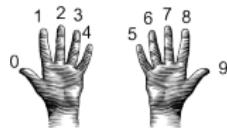
Decimal to Binary: Converting Between Bases



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

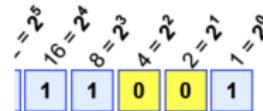
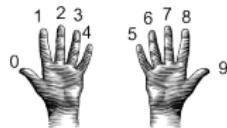


Example: $1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99/128$ is 0 rem 99.

Decimal to Binary: Converting Between Bases

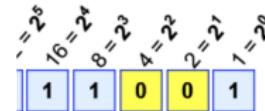
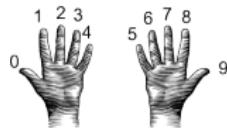


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases



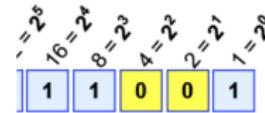
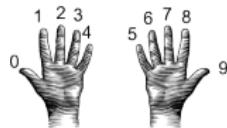
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

Decimal to Binary: Converting Between Bases



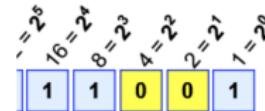
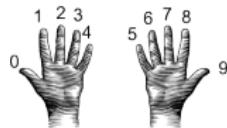
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



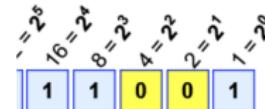
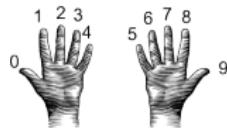
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

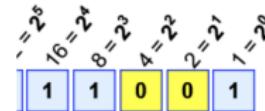
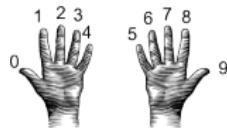
- Example: what is 99 in binary notation?

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

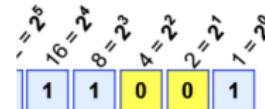
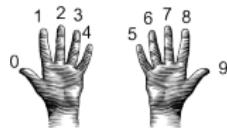
- Example: what is 99 in binary notation?

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:

Decimal to Binary: Converting Between Bases



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

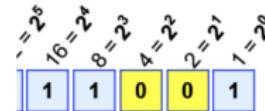
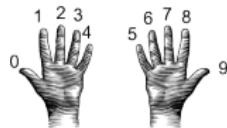
- Example: what is 99 in binary notation?

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

- Example: what is 99 in binary notation?

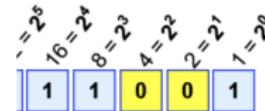
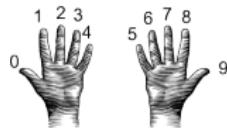
99/128 is 0 rem 99. First digit is 0: 0...

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.

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?

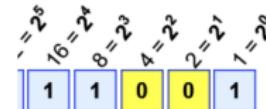
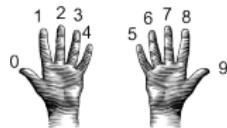
99/128 is 0 rem 99. First digit is 0: 0...

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:

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?

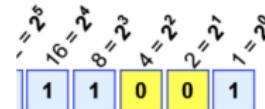
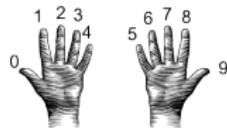
99/128 is 0 rem 99. First digit is 0: 0...

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...

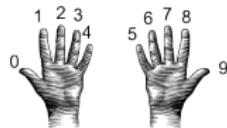
99/64 is 1 rem 35. Next digit is 1: 01...

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

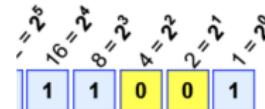
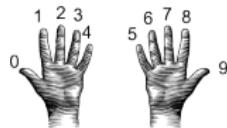
3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3.

Decimal to Binary: Converting Between Bases



Decimal to Binary: Converting Between Bases



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...

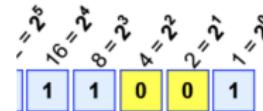
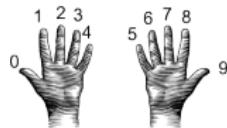
99/64 is 1 rem 35. Next digit is 1: 01...

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...

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...

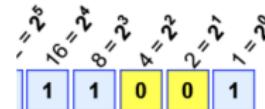
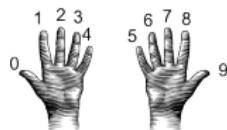
35/32 is 1 rem 3. Next digit is 1: 011...

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.

Decimal to Binary: 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 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...

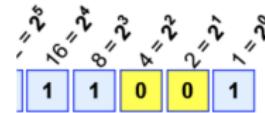
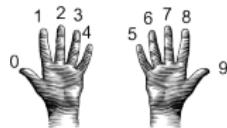
35/32 is 1 rem 3. Next digit is 1: 011...

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...

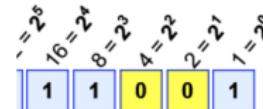
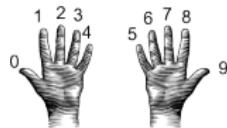
35/32 is 1 rem 3. Next digit is 1: 011...

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...

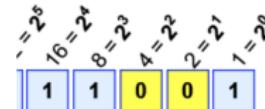
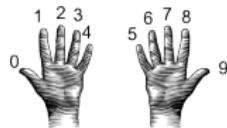
3/16 is 0 rem 3. Next digit is 0: 0110...

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

- 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...

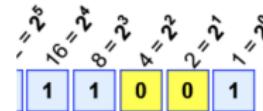
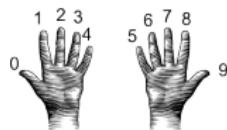
3/16 is 0 rem 3. Next digit is 0: 0110...

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:

Decimal to Binary: 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 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...

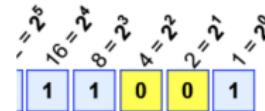
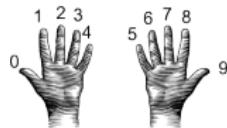
3/16 is 0 rem 3. Next digit is 0: 0110...

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

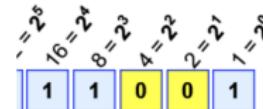
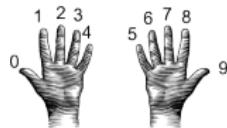


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

Decimal to Binary: Converting Between Bases



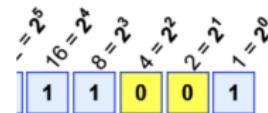
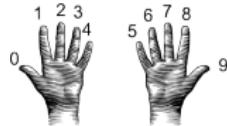
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

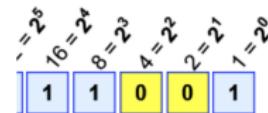
Binary to Decimal: Converting Between Bases



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

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

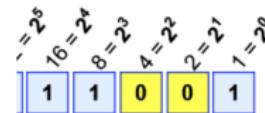
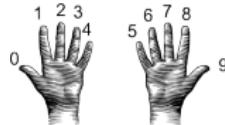
Binary to Decimal: Converting Between Bases



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

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

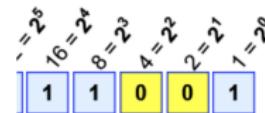
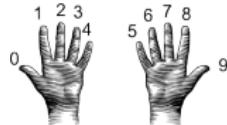
Binary to Decimal: Converting Between Bases



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.

Binary to Decimal: Converting Between Bases

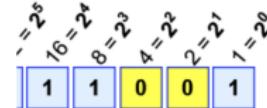
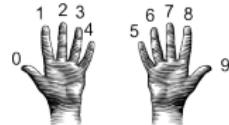


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Binary to Decimal: Converting Between Bases

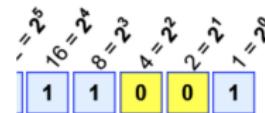
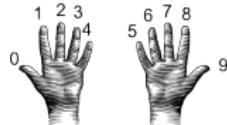


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Binary to Decimal: Converting Between Bases

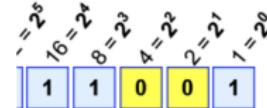


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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

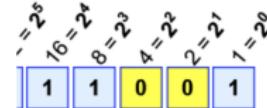


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

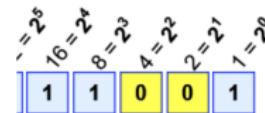
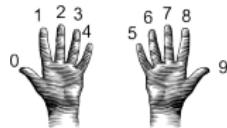


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 2^1 . 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

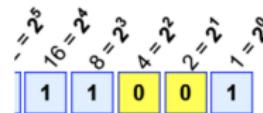


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



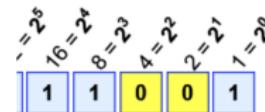
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



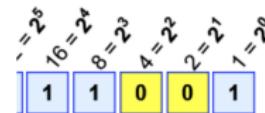
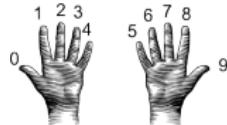
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



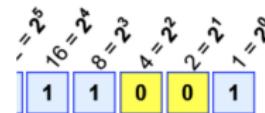
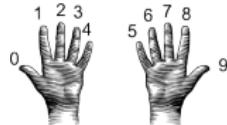
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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

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:

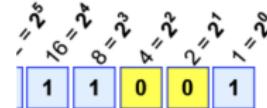
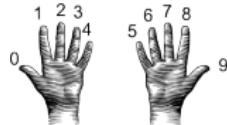
- Set sum = last digit.
- Multiply next digit by $2 = 2^1$. Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$.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

Binary to Decimal: Converting Between Bases



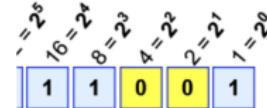
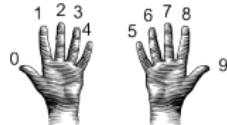
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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?

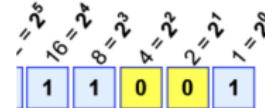
Sum starts with: 1

$0 \times 2 = 0$. 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



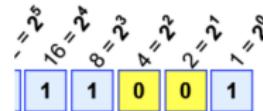
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



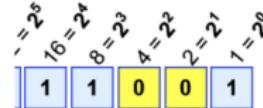
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



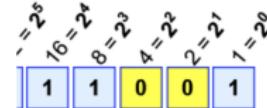
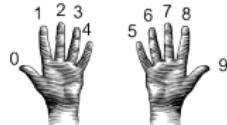
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



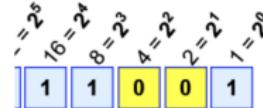
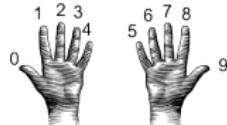
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



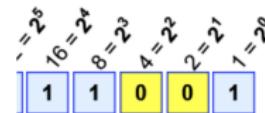
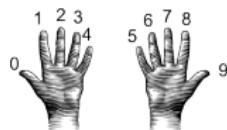
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



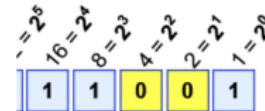
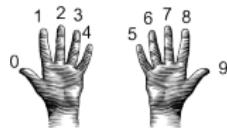
$$\text{Example: } 1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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
$1 \times 4 = 4$. 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

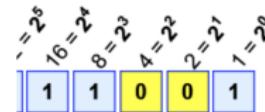
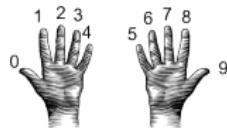


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

Binary to Decimal: Converting Between Bases



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

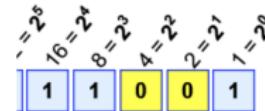
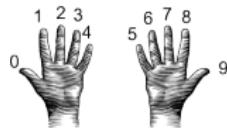
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

Binary to Decimal: Converting Between Bases



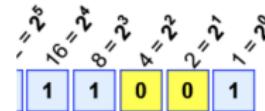
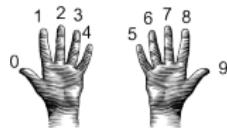
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

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

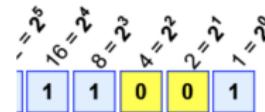
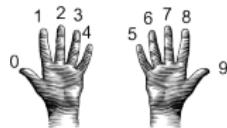
- Example: What is 10100100 in decimal?

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

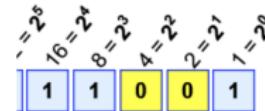
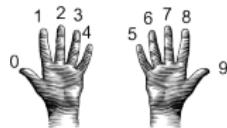
- Example: What is 10100100 in decimal?

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

- Example: What is 10100100 in decimal?

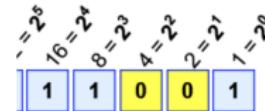
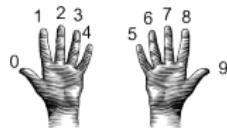
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases

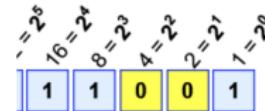
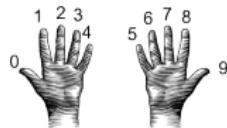


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

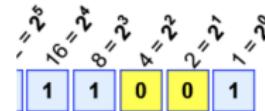
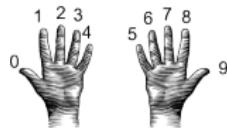
$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. 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

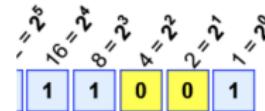
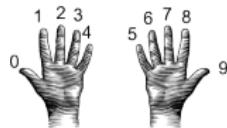
$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. 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: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

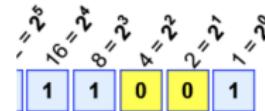
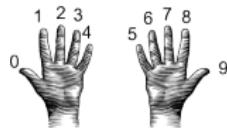
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

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

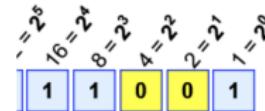
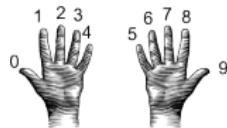
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

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

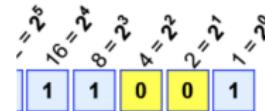
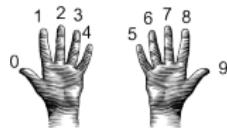
$0 \times 8 = 0$. Add 0 to sum: 4

$0 \times 16 = 0$. 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

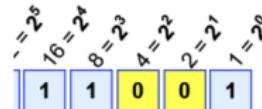
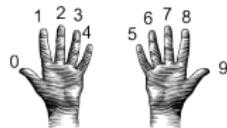
$0 \times 8 = 0$. Add 0 to sum: 4

$0 \times 16 = 0$. 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

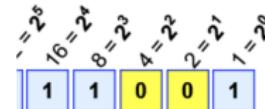
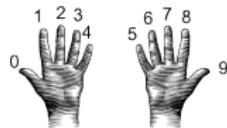


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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

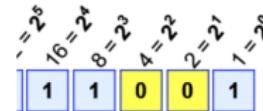
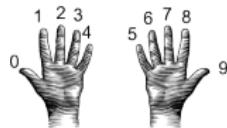


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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



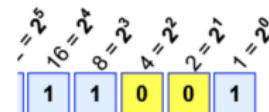
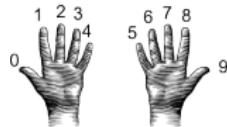
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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
$1 \times 128 = 128.$ Add 128 to sum:	164

The answer is 164.

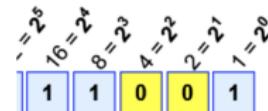
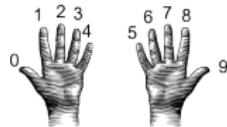
Design Challenge: Incrementers



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

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

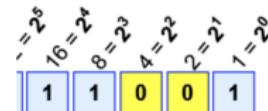
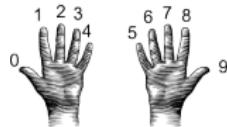
Design Challenge: Incrementers



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

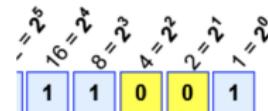
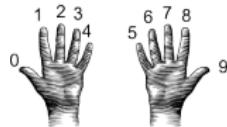


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

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

Design Challenge: Incrementers



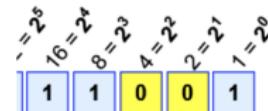
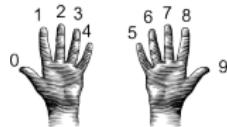
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



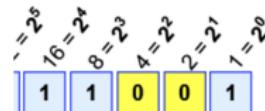
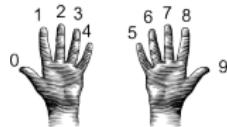
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

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

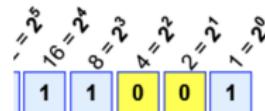
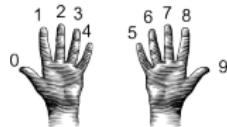
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



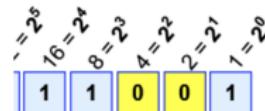
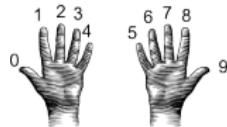
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

<small>Indicate that all cases of academic dishonesty will be reported to the Dean of Students and all cases in question.</small>
Name:
Sigil:
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

- 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 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 its educational mission. The Hunter College Code of Conduct, CUNY Policy on Academic Honesty and self-pervasive rules of academic dishonesty according to the Hunter College Code of Conduct.

I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and all records will be maintained.	
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 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 its educational mission. The Hunter College Code of Conduct and CUNY Policy on Academic Honesty and self-purge 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:	
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.

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-purge 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-purge in condition</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 2022

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 its educational mission. The Hunter College Academic Dishonesty Policy can be found at [http://www.hunter.cuny.edu/policies/academic-dishonesty.html](#). The CUNY Policy on Academic Honesty and Self-Percived 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 all records will be retained</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.**
- 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 containing notes of your own choosing. The notes must be handwritten and must conform to CUNY Policy on Academic Integrity and will forfeit rights of academic dishonesty according to the Hunter College Academic Integrity Policy.
- 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 committment, obtaining unfair advantage, and fabrication of records and official documents) as serious violations of its educational mission. The CUNY Policy on Academic Integrity and the CUNY Policy on Academic Honesty and Self-Percived Cases of Academic Dishonesty according to the Hunter College Academic Integrity Policy.

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.