

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements

- Thanksgiving Break starts in 9 days.



Announcements



- Thanksgiving Break starts in 9 days.
- No CUNY classes:
Thursday-Saturday, 28-31 November.

Announcements



- Thanksgiving Break starts in 9 days.
- No CUNY classes:
Thursday-Saturday, 28-31 November.
- In response to wrap-up requests, additional challenges today with while loops and binary & hexadecimal numbers.

Today's Topics



- Python Recap
- Design Patterns: Searching
- 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

In Pairs or Triples:

Predict what the code will do:

```
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]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')|
```

Python Tutor

```
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]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

(Demo with pythonTutor)

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]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 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]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 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]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 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]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

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

Today's Topics



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

Python & Circuits Review: 10 Weeks in 10 Minutes



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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

- As well as definite loops & the turtle package:

The screenshot shows a Python code editor with a file named 'main.py' open. The code is as follows:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

To the right of the code editor is a 'Result' window displaying a purple hexagon drawn by the turtle. The turtle has stamped a purple star at each vertex of the hexagon.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

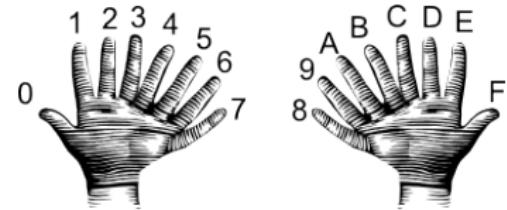
Week 2: variables, data types, more on loops & range()

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

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

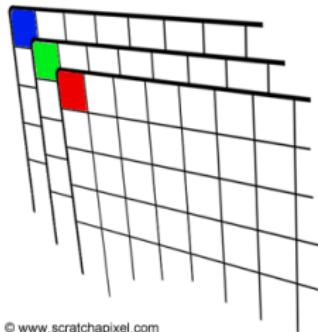
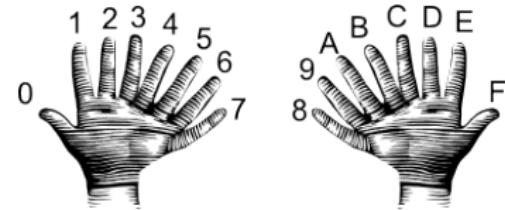
Week 3: colors, hex, slices, numpy & images

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



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

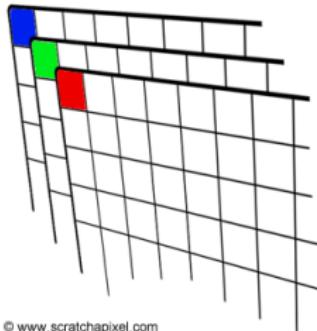
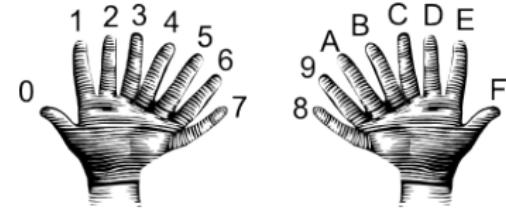
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



© www.scratchapixel.com

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

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



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

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

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

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

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

Week 4: design problem (cropping images) & decisions



Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions

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

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

Week 5: logical operators, truth tables & logical circuits

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

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

Week 5: logical operators, truth tables & logical circuits

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

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

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



Week 6: structured data, pandas, & more design

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
All population figures are consistent with present-day boundaries.....
First census after the consolidation of the five boroughs.....

.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1890,1,203,727,718,1,203,727,718
1891,2,184,3,462,,2,847,28,423
1790,3,313,1,454,6,159,1,781,3,827,4,9447
1800,6,0515,5,740,6,642,1,755,4,543,7,955
1810,6,835,6,280,6,842,2,000,5,349,1,497,34
1820,12,3704,11,187,8,246,2,792,6,135,15,2056
1830,20,2589,20,035,9,049,3,023,7,082,24,2278
1840,31,3110,28,113,14,049,5,348,1,0965,39114
1850,35,544,28,000,14,895,5,348,1,0965,39114
1860,61,3449,279122,32903,23932,25492,174779
1870,94,2292,419921,45468,37393,33029,1479103
1880,11,64473,5,99454,5,653,5,1980,3,0051,1911803
1890,14,66582,11,66582,11,66582,11,66582,11,66582,11,66582
1900,18,505093,11,66582,11,52999,20,05057,67921,24,37202
1910,23,33142,16,34351,26,84041,430980,8569,4766803
1920,22,61103,20,18354,44,60704,7,72021,11,66582,5,0048
1930,19,67112,19,67112,19,67112,19,67112,19,67112,19,67112
1940,18,89924,24,99285,1297634,1394711,174441,7454995
1950,19,66101,1,2738175,1,550849,1451277,191555,7991957
1960,16,96101,1,2738175,1,899049,1451277,191555,7981984
1970,13,539231,1,4657044,1,4727013,1,35443,7981984
1980,14,26285,22,230936,1,1891325,1,168972,2,352121,7071639
1990,14,97536,23,00664,1,1951598,1,203789,3,78977,7322564
2000,15,37195,24,485326,2,2293379,1,332450,4,13728,8080879
2010,15,88373,25,04705,2,271722,1,385108,4,74558,8175405
2015,14,44518,24,947733,2,339150,1,459444,4,74558,8175405

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
All population figures are consistent with present-day boundaries.....
First census after the consolidation of the five boroughs.....

.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1890,4937,2037,,727,7881
1871,21843,3623,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,63545,5740,6442,1755,4543,75934
1820,123704,11187,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,11013,14034,5346,10965,391114
1850,35544,12891,18891,5346,10965,391115
1860,813469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33029,1911801
1890,1367111,66582,116582,116582,116582,116582
1900,185093,116582,152999,200567,67621,2437202
1910,2233142,1634351,284041,430980,8569,4766803
1920,2210103,2018354,446071,446071,732013,116582
1930,1867111,116582,116582,116582,116582,116582
1940,1889924,2469285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1690101,2738175,1550949,1451277,191555,7981984
1970,1539231,2469285,116582,116582,116582,116582
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
2015,1444018,2646733,2339150,1459444,474558,8059405

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,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2037,727,788,102  
1771,21843,3623,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67535,6210,6840,1755,4543,83934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3032,7082,242278  
1840,311510,19013,14087,5346,10965,391114  
1850,355449,218913,18895,6254,12500,451115  
1860,813449,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1364473,72011,6348,57124,35451,2181134  
1900,1850593,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44601,44601,73201,11651,50048  
1930,18671128,1796128,1796128,1796128,1796128,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2319319,1890949,1451277,191555,7981984  
1970,1539231,2465701,1471701,1471701,135443,7981984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2271722,1385108,419782,8175133  
2015,1444018,2646733,2339150,1459444,474558,8059405
```

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.....  
Five census after the consolidation of the five boroughs.....  
.....  
Year,Population  
1690,203,2037,...,727,7181  
1771,21843,28232,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,6350,7000,1800,5300,89734  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31510,21013,14000,5346,10965,391114  
1850,35549,21800,18800,5800,11000,45115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1370000,720000,68000,58000,41000,210000  
1900,1850093,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44600,720201,11673,50048  
1930,26671128,2079128,25000,25000,5800,4930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690000,2319319,1890000,1400000,1300000,781984  
1970,1539231,2465071,1472701,135443,798462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419782,8080879  
2010,1583873,2504705,2216722,1385108,474558,8175133  
2015,1444018,2646733,2339150,1459446,474558,8056405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

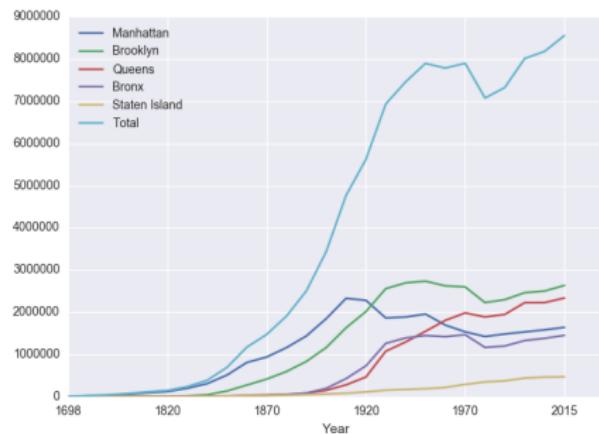
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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,Borough,Population  
1698,Manhattan,2037,727,7181  
1771,21843,36231,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,71031,6354,7041,1811,4973,89734  
1820,123704,11187,8246,2792,6135,152056  
1830,202889,20535,9049,3023,7082,242278  
1840,312110,12013,14031,5348,10965,391114  
1850,435441,12800,14891,5815,13051,491154  
1860,613469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33051,1911801  
1890,1400000,720000,68000,58000,41000,210000  
1900,1850093,116582,152999,200507,67921,2437202  
1910,233142,1634351,2841,430980,8589,476683  
1920,2281103,2018354,44601,44601,73201,11651,50048  
1930,2667103,2079128,45254,45254,5831,45254,45254  
1940,1889924,2690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7091957  
1960,1690000,2300000,1800000,1800000,1800000,1800000  
1970,1539231,2465071,2472101,2472101,2472101,2472101  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1497536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2210722,1385108,419728,8175133  
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)
                                         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



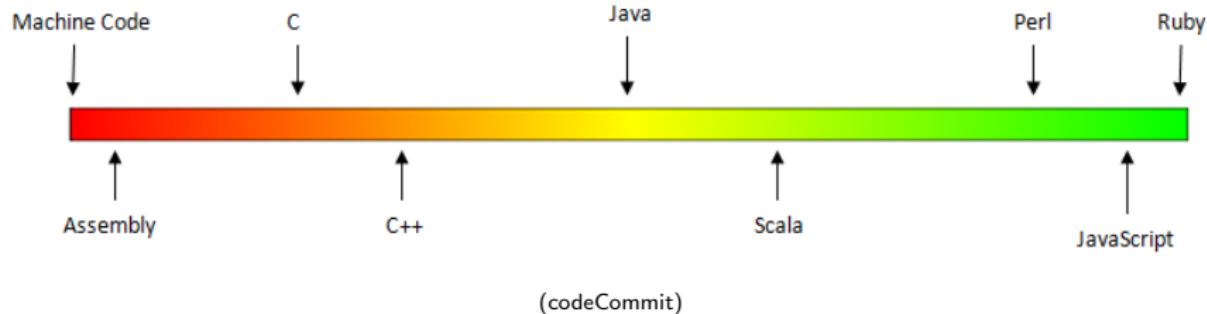
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
 - ▶ Primitive: `int`, `float`, `bool`, `string`;
 - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: `turtles` (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`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

Today's Topics



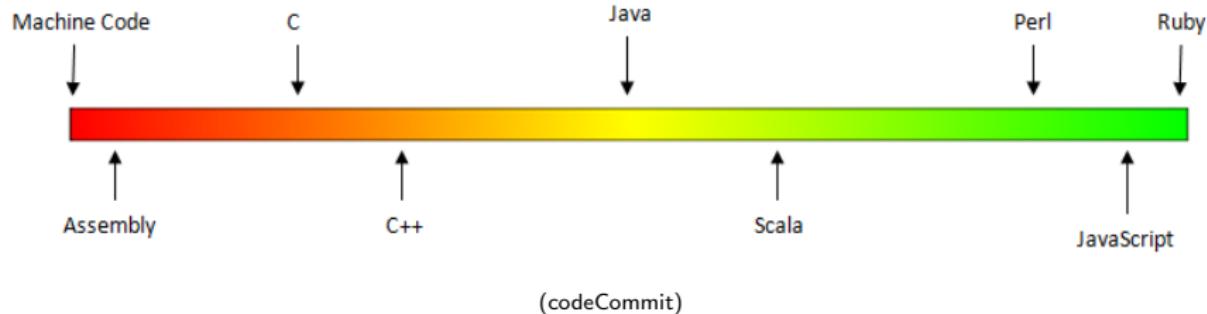
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

Low-Level vs. High-Level Languages



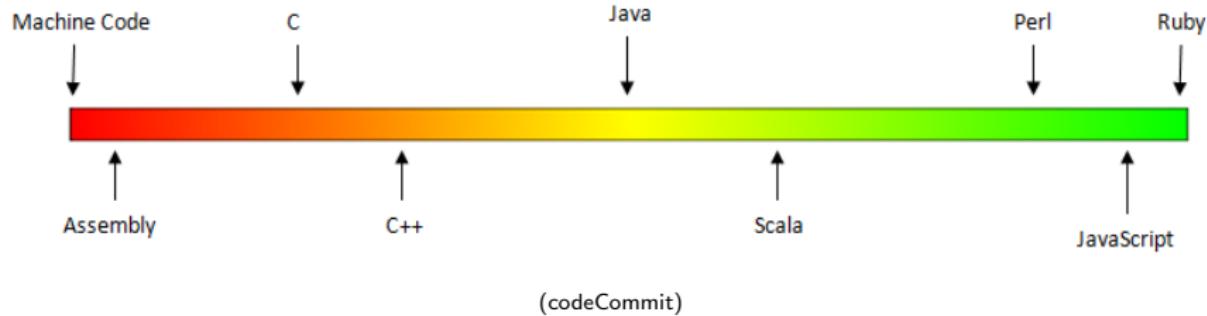
- Can view programming languages on a continuum.

Low-Level vs. High-Level Languages



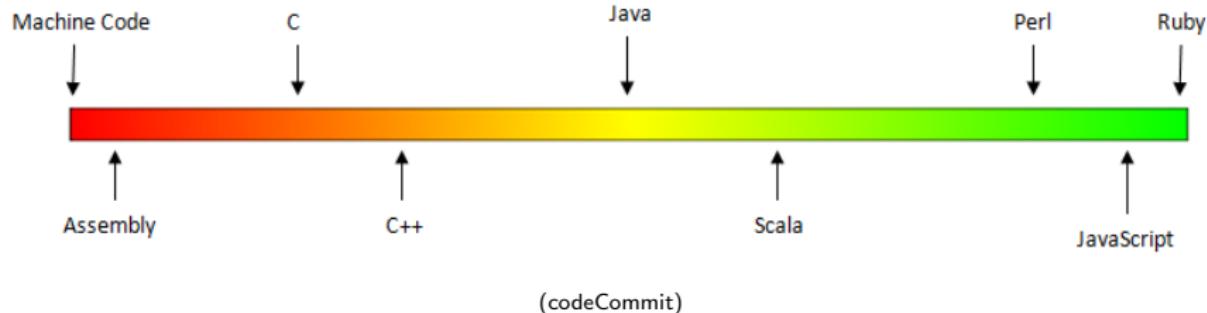
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

Low-Level vs. High-Level Languages



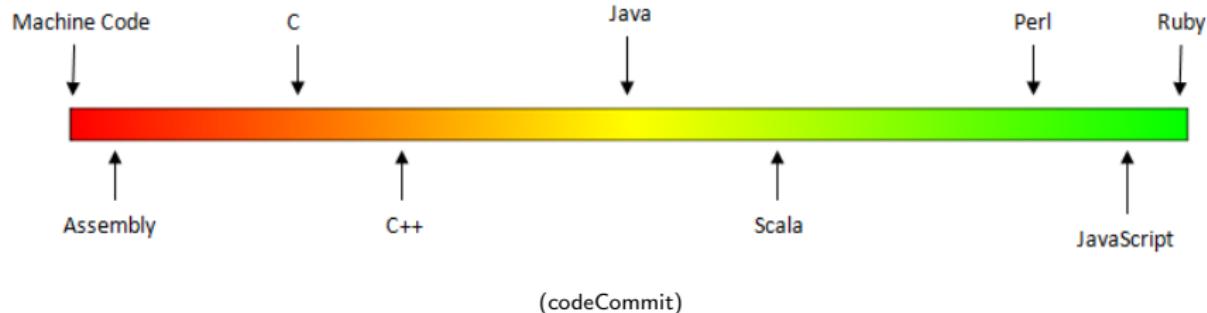
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

Low-Level vs. High-Level Languages



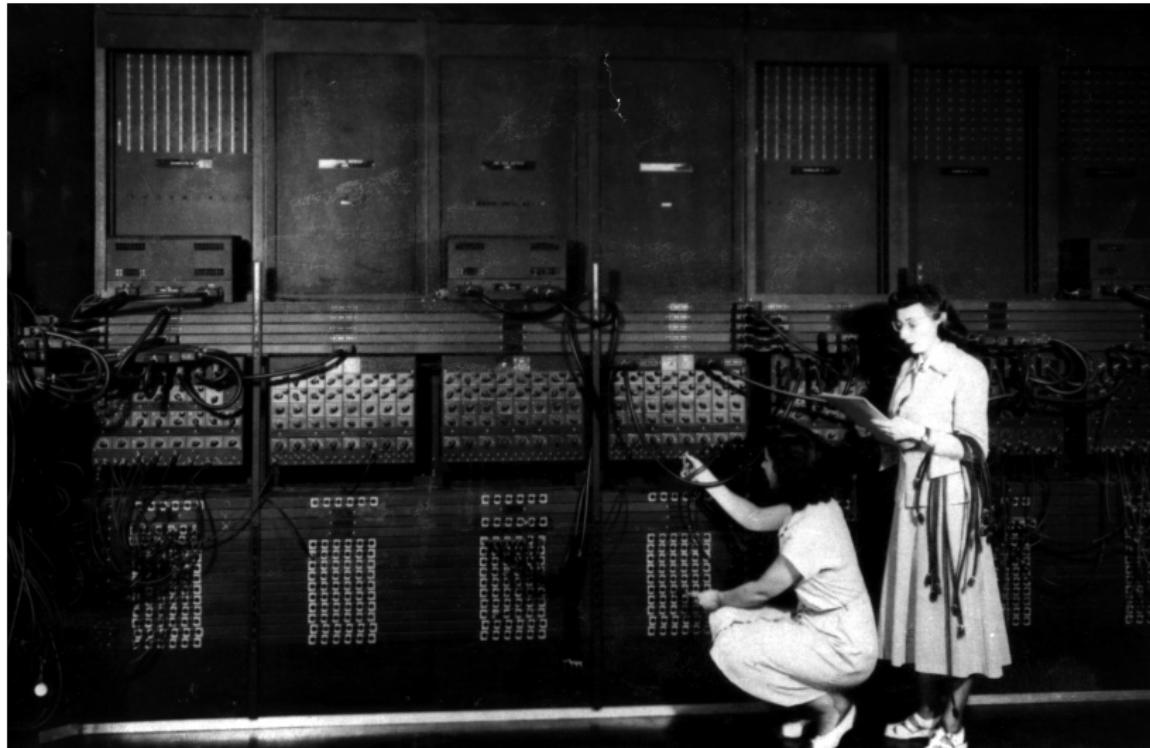
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine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

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

(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.

Machine Language

```

A 002000 C2 3B REP #3B
A 002002 7F CLC
A 002003 FB SED
A 002004 34 32 #1234
A 002007 69 21 43 ADC #43
A 002008 69 03 7F STA #7F
A 00200E D9 CLD
A 00200F E2 3B SEP #3B
A 002011 90 BHK
A 002012

```

F

```

PB PC Min:0x21C A X Y SP DP IR
; 00 2013 00110000 0000 0000 0002 CFFF 0000 00
$ 2800

BREAK

PB PC Min:0x21C A X Y SP DP IR
; 00 2013 00110000 0555 0000 0002 CFFF 0000 00
$ 7193 7193

#T193 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

(wiki)

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

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

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 # i
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
32 syscall           # print to the log
```

Step	Run	<input checked="" type="checkbox"/> 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	

(WeMIPS)



WeMIPS

(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 window titled "ShowMIPS Demo". At the top, there are tabs for "User", "3", "64", "ShowMIPS Demo", "Addition Counter", "Btav", "Looper", "Stack Test", "Hello World", "Code Gen Save String", "Interactive", "Binary Decimal", "Decimal Binary", and "Debug". Below the tabs is a menu bar with "User Guide | Unit Tests | Doc". A "Step" button and a "Run" button with a note "Enable auto switching" are also present.

The main area displays assembly code:

```
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   $s1,$zero,102 # $s1 = 102
7    addi   $s2,$zero,103 # $s2 = 103
8    addi   $s3,$zero,104 # $s3 = 104
9    addi   $s4,$zero,105 # $s4 = 105
10   addi   $s5,$zero,106 # $s5 = 106
11   addi   $s6,$zero,107 # $s6 = 107
12   addi   $s7,$zero,108 # $s7 = 108
13   addi   $s8,$zero,109 # $s8 = 109
14   addi   $s9,$zero,110 # $s9 = 110
15   addi   $s10,$zero,111 # $s10 = 111
16   addi   $s11,$zero,112 # $s11 = 112
17   addi   $s12,$zero,113 # $s12 = 113
18   addi   $s13,$zero,114 # $s13 = 114
19   addi   $s14,$zero,115 # $s14 = 115
20   addi   $s15,$zero,116 # $s15 = 116
21   addi   $s16,$zero,117 # $s16 = 117
22   addi   $s17,$zero,118 # $s17 = 118
23   addi   $s18,$zero,119 # $s18 = 119
24   addi   $s19,$zero,120 # $s19 = 120
25   addi   $s20,$zero,121 # $s20 = 121
26   addi   $s21,$zero,122 # $s21 = 122
27   addi   $s22,$zero,123 # $s22 = 123
28   addi   $s23,$zero,124 # $s23 = 124
29   addi   $s24,$zero,125 # $s24 = 125
30   addi   $s25,$zero,126 # $s25 = 126
31   addi   $s26,$zero,127 # $s26 = 127
32   addi   $s27,$zero,128 # $s27 = 128
33   addi   $s28,$zero,129 # $s28 = 129
34   addi   $s29,$zero,130 # $s29 = 130
35   addi   $s30,$zero,131 # $s30 = 131
36   addi   $s31,$zero,132 # $s31 = 132
37   addi   $s32,$zero,133 # $s32 = 133
38   addi   $s33,$zero,134 # $s33 = 134
39   addi   $s34,$zero,135 # $s34 = 135
40   addi   $s35,$zero,136 # $s35 = 136
41   addi   $s36,$zero,137 # $s36 = 137
42   addi   $s37,$zero,138 # $s37 = 138
43   addi   $s38,$zero,139 # $s38 = 139
44   addi   $s39,$zero,140 # $s39 = 140
45   addi   $s40,$zero,141 # $s40 = 141
46   addi   $s41,$zero,142 # $s41 = 142
47   addi   $s42,$zero,143 # $s42 = 143
48   addi   $s43,$zero,144 # $s43 = 144
49   addi   $s44,$zero,145 # $s44 = 145
50   addi   $s45,$zero,146 # $s45 = 146
51   addi   $s46,$zero,147 # $s46 = 147
52   addi   $s47,$zero,148 # $s47 = 148
53   addi   $s48,$zero,149 # $s48 = 149
54   addi   $s49,$zero,150 # $s49 = 150
55   addi   $s50,$zero,151 # $s50 = 151
56   addi   $s51,$zero,152 # $s51 = 152
57   addi   $s52,$zero,153 # $s52 = 153
58   addi   $s53,$zero,154 # $s53 = 154
59   addi   $s54,$zero,155 # $s54 = 155
60   addi   $s55,$zero,156 # $s55 = 156
61   addi   $s56,$zero,157 # $s56 = 157
62   addi   $s57,$zero,158 # $s57 = 158
63   addi   $s58,$zero,159 # $s58 = 159
64   addi   $s59,$zero,160 # $s59 = 160
65   addi   $s60,$zero,161 # $s60 = 161
66   addi   $s61,$zero,162 # $s61 = 162
67   addi   $s62,$zero,163 # $s62 = 163
68   addi   $s63,$zero,164 # $s63 = 164
69   addi   $s64,$zero,165 # $s64 = 165
70   addi   $s65,$zero,166 # $s65 = 166
71   addi   $s66,$zero,167 # $s66 = 167
72   addi   $s67,$zero,168 # $s67 = 168
73   addi   $s68,$zero,169 # $s68 = 169
74   addi   $s69,$zero,170 # $s69 = 170
75   addi   $s70,$zero,171 # $s70 = 171
76   addi   $s71,$zero,172 # $s71 = 172
77   addi   $s72,$zero,173 # $s72 = 173
78   addi   $s73,$zero,174 # $s73 = 174
79   addi   $s74,$zero,175 # $s74 = 175
80   addi   $s75,$zero,176 # $s75 = 176
81   addi   $s76,$zero,177 # $s76 = 177
82   addi   $s77,$zero,178 # $s77 = 178
83   addi   $s78,$zero,179 # $s78 = 179
84   addi   $s79,$zero,180 # $s79 = 180
85   addi   $s80,$zero,181 # $s80 = 181
86   addi   $s81,$zero,182 # $s81 = 182
87   addi   $s82,$zero,183 # $s82 = 183
88   addi   $s83,$zero,184 # $s83 = 184
89   addi   $s84,$zero,185 # $s84 = 185
90   addi   $s85,$zero,186 # $s85 = 186
91   addi   $s86,$zero,187 # $s86 = 187
92   addi   $s87,$zero,188 # $s87 = 188
93   addi   $s88,$zero,189 # $s88 = 189
94   addi   $s89,$zero,190 # $s89 = 190
95   addi   $s90,$zero,191 # $s90 = 191
96   addi   $s91,$zero,192 # $s91 = 192
97   addi   $s92,$zero,193 # $s92 = 193
98   addi   $s93,$zero,194 # $s93 = 194
99   addi   $s94,$zero,195 # $s94 = 195
100  addi   $s95,$zero,196 # $s95 = 196
101  addi   $s96,$zero,197 # $s96 = 197
102  addi   $s97,$zero,198 # $s97 = 198
103  addi   $s98,$zero,199 # $s98 = 199
104  addi   $s99,$zero,200 # $s99 = 200
105  addi   $s100,$zero,201 # $s100 = 201
106  addi   $s101,$zero,202 # $s101 = 202
107  addi   $s102,$zero,203 # $s102 = 203
108  addi   $s103,$zero,204 # $s103 = 204
109  addi   $s104,$zero,205 # $s104 = 205
110  addi   $s105,$zero,206 # $s105 = 206
111  addi   $s106,$zero,207 # $s106 = 207
112  addi   $s107,$zero,208 # $s107 = 208
113  addi   $s108,$zero,209 # $s108 = 209
114  addi   $s109,$zero,210 # $s109 = 210
115  addi   $s110,$zero,211 # $s110 = 211
116  addi   $s111,$zero,212 # $s111 = 212
117  addi   $s112,$zero,213 # $s112 = 213
118  addi   $s113,$zero,214 # $s113 = 214
119  addi   $s114,$zero,215 # $s114 = 215
120  addi   $s115,$zero,216 # $s115 = 216
121  addi   $s116,$zero,217 # $s116 = 217
122  addi   $s117,$zero,218 # $s117 = 218
123  addi   $s118,$zero,219 # $s118 = 219
124  addi   $s119,$zero,220 # $s119 = 220
125  addi   $s120,$zero,221 # $s120 = 221
126  addi   $s121,$zero,222 # $s121 = 222
127  addi   $s122,$zero,223 # $s122 = 223
128  addi   $s123,$zero,224 # $s123 = 224
129  addi   $s124,$zero,225 # $s124 = 225
130  addi   $s125,$zero,226 # $s125 = 226
131  addi   $s126,$zero,227 # $s126 = 227
132  addi   $s127,$zero,228 # $s127 = 228
133  addi   $s128,$zero,229 # $s128 = 229
134  addi   $s129,$zero,230 # $s129 = 230
135  addi   $s130,$zero,231 # $s130 = 231
136  addi   $s131,$zero,232 # $s131 = 232
137  addi   $s132,$zero,233 # $s132 = 233
138  addi   $s133,$zero,234 # $s133 = 234
139  addi   $s134,$zero,235 # $s134 = 235
140  addi   $s135,$zero,236 # $s135 = 236
141  addi   $s136,$zero,237 # $s136 = 237
142  addi   $s137,$zero,238 # $s137 = 238
143  addi   $s138,$zero,239 # $s138 = 239
144  addi   $s139,$zero,240 # $s139 = 240
145  addi   $s140,$zero,241 # $s140 = 241
146  addi   $s141,$zero,242 # $s141 = 242
147  addi   $s142,$zero,243 # $s142 = 243
148  addi   $s143,$zero,244 # $s143 = 244
149  addi   $s144,$zero,245 # $s144 = 245
150  addi   $s145,$zero,246 # $s145 = 246
151  addi   $s146,$zero,247 # $s146 = 247
152  addi   $s147,$zero,248 # $s147 = 248
153  addi   $s148,$zero,249 # $s148 = 249
154  addi   $s149,$zero,250 # $s149 = 250
155  addi   $s150,$zero,251 # $s150 = 251
156  addi   $s151,$zero,252 # $s151 = 252
157  addi   $s152,$zero,253 # $s152 = 253
158  addi   $s153,$zero,254 # $s153 = 254
159  addi   $s154,$zero,255 # $s154 = 255
160  addi   $s155,$zero,256 # $s155 = 256
161  addi   $s156,$zero,257 # $s156 = 257
162  addi   $s157,$zero,258 # $s157 = 258
163  addi   $s158,$zero,259 # $s158 = 259
164  addi   $s159,$zero,260 # $s159 = 260
165  addi   $s160,$zero,261 # $s160 = 261
166  addi   $s161,$zero,262 # $s161 = 262
167  addi   $s162,$zero,263 # $s162 = 263
168  addi   $s163,$zero,264 # $s163 = 264
169  addi   $s164,$zero,265 # $s164 = 265
170  addi   $s165,$zero,266 # $s165 = 266
171  addi   $s166,$zero,267 # $s166 = 267
172  addi   $s167,$zero,268 # $s167 = 268
173  addi   $s168,$zero,269 # $s168 = 269
174  addi   $s169,$zero,270 # $s169 = 270
175  addi   $s170,$zero,271 # $s170 = 271
176  addi   $s171,$zero,272 # $s171 = 272
177  addi   $s172,$zero,273 # $s172 = 273
178  addi   $s173,$zero,274 # $s173 = 274
179  addi   $s174,$zero,275 # $s174 = 275
180  addi   $s175,$zero,276 # $s175 = 276
181  addi   $s176,$zero,277 # $s176 = 277
182  addi   $s177,$zero,278 # $s177 = 278
183  addi   $s178,$zero,279 # $s178 = 279
184  addi   $s179,$zero,280 # $s179 = 280
185  addi   $s180,$zero,281 # $s180 = 281
186  addi   $s181,$zero,282 # $s181 = 282
187  addi   $s182,$zero,283 # $s182 = 283
188  addi   $s183,$zero,284 # $s183 = 284
189  addi   $s184,$zero,285 # $s184 = 285
190  addi   $s185,$zero,286 # $s185 = 286
191  addi   $s186,$zero,287 # $s186 = 287
192  addi   $s187,$zero,288 # $s187 = 288
193  addi   $s188,$zero,289 # $s188 = 289
194  addi   $s189,$zero,290 # $s189 = 290
195  addi   $s190,$zero,291 # $s190 = 291
196  addi   $s191,$zero,292 # $s191 = 292
197  addi   $s192,$zero,293 # $s192 = 293
198  addi   $s193,$zero,294 # $s193 = 294
199  addi   $s194,$zero,295 # $s194 = 295
200  addi   $s195,$zero,296 # $s195 = 296
201  addi   $s196,$zero,297 # $s196 = 297
202  addi   $s197,$zero,298 # $s197 = 298
203  addi   $s198,$zero,299 # $s198 = 299
204  addi   $s199,$zero,300 # $s199 = 300
205  addi   $s200,$zero,301 # $s200 = 301
206  addi   $s201,$zero,302 # $s201 = 302
207  addi   $s202,$zero,303 # $s202 = 303
208  addi   $s203,$zero,304 # $s203 = 304
209  addi   $s204,$zero,305 # $s204 = 305
210  addi   $s205,$zero,306 # $s205 = 306
211  addi   $s206,$zero,307 # $s206 = 307
212  addi   $s207,$zero,308 # $s207 = 308
213  addi   $s208,$zero,309 # $s208 = 309
214  addi   $s209,$zero,310 # $s209 = 310
215  addi   $s210,$zero,311 # $s210 = 311
216  addi   $s211,$zero,312 # $s211 = 312
217  addi   $s212,$zero,313 # $s212 = 313
218  addi   $s213,$zero,314 # $s213 = 314
219  addi   $s214,$zero,315 # $s214 = 315
220  addi   $s215,$zero,316 # $s215 = 316
221  addi   $s216,$zero,317 # $s216 = 317
222  addi   $s217,$zero,318 # $s217 = 318
223  addi   $s218,$zero,319 # $s218 = 319
224  addi   $s219,$zero,320 # $s219 = 320
225  addi   $s220,$zero,321 # $s220 = 321
226  addi   $s221,$zero,322 # $s221 = 322
227  addi   $s222,$zero,323 # $s222 = 323
228  addi   $s223,$zero,324 # $s223 = 324
229  addi   $s224,$zero,325 # $s224 = 325
230  addi   $s225,$zero,326 # $s225 = 326
231  addi   $s226,$zero,327 # $s226 = 327
232  addi   $s227,$zero,328 # $s227 = 328
233  addi   $s228,$zero,329 # $s228 = 329
234  addi   $s229,$zero,330 # $s229 = 330
235  addi   $s230,$zero,331 # $s230 = 331
236  addi   $s231,$zero,332 # $s231 = 332
237  addi   $s232,$zero,333 # $s232 = 333
238  addi   $s233,$zero,334 # $s233 = 334
239  addi   $s234,$zero,335 # $s234 = 335
240  addi   $s235,$zero,336 # $s235 = 336
241  addi   $s236,$zero,337 # $s236 = 337
242  addi   $s237,$zero,338 # $s237 = 338
243  addi   $s238,$zero,339 # $s238 = 339
244  addi   $s239,$zero,340 # $s239 = 340
245  addi   $s240,$zero,341 # $s240 = 341
246  addi   $s241,$zero,342 # $s241 = 342
247  addi   $s242,$zero,343 # $s242 = 343
248  addi   $s243,$zero,344 # $s243 = 344
249  addi   $s244,$zero,345 # $s244 = 345
250  addi   $s245,$zero,346 # $s245 = 346
251  addi   $s246,$zero,347 # $s246 = 347
252  addi   $s247,$zero,348 # $s247 = 348
253  addi   $s248,$zero,349 # $s248 = 349
254  addi   $s249,$zero,350 # $s249 = 350
255  addi   $s250,$zero,351 # $s250 = 351
256  addi   $s251,$zero,352 # $s251 = 352
257  addi   $s252,$zero,353 # $s252 = 353
258  addi   $s253,$zero,354 # $s253 = 354
259  addi   $s254,$zero,355 # $s254 = 355
260  addi   $s255,$zero,356 # $s255 = 356
261  addi   $s256,$zero,357 # $s256 = 357
262  addi   $s257,$zero,358 # $s257 = 358
263  addi   $s258,$zero,359 # $s258 = 359
264  addi   $s259,$zero,360 # $s259 = 360
265  addi   $s260,$zero,361 # $s260 = 361
266  addi   $s261,$zero,362 # $s261 = 362
267  addi   $s262,$zero,363 # $s262 = 363
268  addi   $s263,$zero,364 # $s263 = 364
269  addi   $s264,$zero,365 # $s264 = 365
270  addi   $s265,$zero,366 # $s265 = 366
271  addi   $s266,$zero,367 # $s266 = 367
272  addi   $s267,$zero,368 # $s267 = 368
273  addi   $s268,$zero,369 # $s268 = 369
274  addi   $s269,$zero,370 # $s269 = 370
275  addi   $s270,$zero,371 # $s270 = 371
276  addi   $s271,$zero,372 # $s271 = 372
277  addi   $s272,$zero,373 # $s272 = 373
278  addi   $s273,$zero,374 # $s273 = 374
279  addi   $s274,$zero,375 # $s274 = 375
280  addi   $s275,$zero,376 # $s275 = 376
281  addi   $s276,$zero,377 # $s276 = 377
282  addi   $s277,$zero,378 # $s277 = 378
283  addi   $s278,$zero,379 # $s278 = 379
284  addi   $s279,$zero,380 # $s279 = 380
285  addi   $s280,$zero,381 # $s280 = 381
286  addi   $s281,$zero,382 # $s281 = 382
287  addi   $s282,$zero,383 # $s282 = 383
288  addi   $s283,$zero,384 # $s283 = 384
289  addi   $s284,$zero,385 # $s284 = 385
290  addi   $s285,$zero,386 # $s285 = 386
291  addi   $s286,$zero,387 # $s286 = 387
292  addi   $s287,$zero,388 # $s287 = 388
293  addi   $s288,$zero,389 # $s288 = 389
294  addi   $s289,$zero,390 # $s289 = 390
295  addi   $s290,$zero,391 # $s290 = 391
296  addi   $s291,$zero,392 # $s291 = 392
297  addi   $s292,$zero,393 # $s292 = 393
298  addi   $s293,$zero,394 # $s293 = 394
299  addi   $s294,$zero,395 # $s294 = 395
200  addi   $s295,$zero,396 # $s295 = 396
201  addi   $s296,$zero,397 # $s296 = 397
202  addi   $s297,$zero,398 # $s297 = 398
203  addi   $s298,$zero,399 # $s298 = 399
204  addi   $s299,$zero,400 # $s299 = 400
205  addi   $s295,$zero,401 # $s295 = 401
206  addi   $s296,$zero,402 # $s296 = 402
207  addi   $s297,$zero,403 # $s297 = 403
208  addi   $s298,$zero,404 # $s298 = 404
209  addi   $s299,$zero,405 # $s299 = 405
210  addi   $s295,$zero,406 # $s295 = 406
211  addi   $s296,$zero,407 # $s296 = 407
212  addi   $s297,$zero,408 # $s297 = 408
213  addi   $s298,$zero,409 # $s298 = 409
214  addi   $s299,$zero,410 # $s299 = 410
215  addi   $s295,$zero,411 # $s295 = 411
216  addi   $s296,$zero,412 # $s296 = 412
217  addi   $s297,$zero,413 # $s297 = 413
218  addi   $s298,$zero,414 # $s298 = 414
219  addi   $s299,$zero,415 # $s299 = 415
220  addi   $s295,$zero,416 # $s295 = 416
221  addi   $s296,$zero,417 # $s296 = 417
222  addi   $s297,$zero,418 # $s297 = 418
223  addi   $s298,$zero,419 # $s298 = 419
224  addi   $s299,$zero,420 # $s299 = 420
225  addi   $s295,$zero,421 # $s295 = 421
226  addi   $s296,$zero,422 # $s296 = 422
227  addi   $s297,$zero,423 # $s297 = 423
228  addi   $s298,$zero,424 # $s298 = 424
229  addi   $s299,$zero,425 # $s299 = 425
230  addi   $s295,$zero,426 # $s295 = 426
231  addi   $s296,$zero,427 # $s296 = 427
232  addi   $s297,$zero,428 # $s297 = 428
233  addi   $s298,$zero,429 # $s298 = 429
234  addi   $s299,$zero,430 # $s299 = 430
235  addi   $s295,$zero,431 # $s295 = 431
236  addi   $s296,$zero,432 # $s296 = 432
237  addi   $s297,$zero,433 # $s297 = 433
238  addi   $s298,$zero,434 # $s298 = 434
239  addi   $s299,$zero,435 # $s299 = 435
240  addi   $s295,$zero,436 # $s295 = 436
241  addi   $s296,$zero,437 # $s296 = 437
242  addi   $s297,$zero,438 # $s297 = 438
243  addi   $s298,$zero,439 # $s298 = 439
244  addi   $s299,$zero,440 # $s299 = 440
245  addi   $s295,$zero,441 # $s295 = 441
246  addi   $s296,$zero,442 # $s296 = 442
247  addi   $s297,$zero,443 # $s297 = 443
248  addi   $s298,$zero,444 # $s298 = 444
249  addi   $s299,$zero,445 # $s299 = 445
250  addi   $s295,$zero,446 # $s295 = 446
251  addi   $s296,$zero,447 # $s296 = 447
252  addi   $s297,$zero,448 # $s297 = 448
253  addi   $s298,$zero,449 # $s298 = 449
254  addi   $s299,$zero,450 # $s299 = 450
255  addi   $s295,$zero,451 # $s295 = 451
256  addi   $s296,$zero,452 # $s296 = 452
257  addi   $s297,$zero,453 # $s297 = 453
258  addi   $s298,$zero,454 # $s298 = 454
259  addi   $s299,$zero,455 # $s299 = 455
260  addi   $s295,$zero,456 # $s295 = 456
261  addi   $s296,$zero,457 # $s296 = 457
262  addi   $s297,$zero,458 # $s297 = 458
263  addi   $s298,$zero,459 # $s298 = 459
264  addi   $s299,$zero,460 # $s299 = 460
265  addi   $s295,$zero,461 # $s295 = 461
266  addi   $s296,$zero,462 # $s296 = 462
267  addi   $s297,$zero,463 # $s297 = 463
268  addi   $s298,$zero,464 # $s298 = 464
269  addi   $s299,$zero,465 # $s299 = 465
270  addi   $s295,$zero,466 # $s295 = 466
271  addi   $s296,$zero,467 # $s296 = 467
272  addi   $s297,$zero,468 # $s297 = 468
273  addi   $s298,$zero,469 # $s298 = 469
274  addi   $s299,$zero,470 # $s299 = 470
275  addi   $s295,$zero,471 # $s295 = 471
276  addi   $s296,$zero,472 # $s296 = 472
277  addi   $s297,$zero,473 # $s297 = 473
278  addi   $s298,$zero,474 # $s298 = 474
279  addi   $s299,$zero,475 # $s299 = 475
280  addi   $s295,$zero,476 # $s295 = 476
281  addi   $s296,$zero,477 # $s296 = 477
282  addi   $s297,$zero,478 # $s297 = 478
283  addi   $s298,$zero,479 # $s298 = 479
284  addi   $s299,$zero,480 # $s299 = 480
285  addi   $s295,$zero,481 # $s295 = 481
286  addi   $s296,$zero,482 # $s296 = 482
287  addi   $s297,$zero,483 # $s297 = 483
288  addi   $s298,$zero,484 # $s298 = 484
289  addi   $s299,$zero,485 # $s299 = 485
290  addi   $s295,$zero,486 # $s295 = 486
291  addi   $s296,$zero,487 # $s296 = 487
292  addi   $s297,$zero,488 # $s297 = 488
293  addi   $s298,$zero,489 # $s298 = 489
294  addi   $s299,$zero,490 # $s299 = 490
295  addi   $s295,$zero,491 # $s295 = 491
296  addi   $s296,$zero,492 # $s296 = 492
297  addi   $s297,$zero,493 # $s297 = 493
298  addi   $s298,$zero,494 # $s298 = 494
299  addi   $s299,$zero,495 # $s299 = 495
200  addi   $s295,$zero,496 # $s295 = 496
201  addi   $s296,$zero,497 # $s296 = 497
202  addi   $s297,$zero,498 # $s297 = 498
203  addi   $s298,$zero,499 # $s298 = 499
204  addi   $s299,$zero,500 # $s299 = 500
205  addi   $s295,$zero,501 # $s295 = 501
206  addi   $s296,$zero,502 # $s296 = 502
207  addi   $s297,$zero,503 # $s297 = 503
208  addi   $s298,$zero,504 # $s298 = 504
209  addi   $s299,$zero,505 # $s299 = 505
210  addi   $s295,$zero,506 # $s295 = 506
211  addi   $s296,$zero,507 # $s296 = 507
212  addi   $s297,$zero,508 # $s297 = 508
213  addi   $s298,$zero,509 # $s298 = 509
214  addi   $s299,$zero,510 # $s299 = 510
215  addi   $s295,$zero,511 # $s295 = 511
216  addi   $s296,$zero,512 # $s296 = 512
217  addi   $s297,$zero,513 # $s297 = 513
218  addi   $s298,$zero,514 # $s298 = 514
219  addi   $s299,$zero,515 # $s299 = 515
220  addi   $
```

MIPS Commands

The screenshot shows the ShowMIPS Demo application window. At the top, there are tabs for User, ShowMIPS Demo, and Run. Below the tabs are buttons for Addition, Counter, IfElse, Looper, StackTest, and Hello World. Underneath those are buttons for CodeGen, SaveString, Interactive, DirectDecimal, DecimalBinary, and Debug. The main area contains assembly code and a register dump.

```
# Shows "Hello world" at the top of the stack
1    .text
2    .globl _start
3    .type _start, @function
4    _start:
5        addi $t0, $zero, 101 # $t0 = 101
6        addi $t1, $zero, 100 # $t1 = 100
7        addi $t2, $zero, 100 # $t2 = 100
8        addi $t3, $zero, 100 # $t3 = 100
9        addi $t4, $zero, 100 # $t4 = 100
10       addi $t5, $zero, 100 # $t5 = 100
11       addi $t6, $zero, 100 # $t6 = 100
12       addi $t7, $zero, 100 # $t7 = 100
13       addi $t8, $zero, 100 # $t8 = 100
14       addi $t9, $zero, 100 # $t9 = 100
15       addi $t10, $zero, 100 # $t10 = 100
16       addi $t11, $zero, 100 # $t11 = 100
17       addi $t12, $zero, 100 # $t12 = 100
18       addi $t13, $zero, 100 # $t13 = 100
19       addi $t14, $zero, 100 # $t14 = 100
20       addi $t15, $zero, 100 # $t15 = 100
21       addi $t16, $zero, 100 # $t16 = 100
22       addi $t17, $zero, 100 # $t17 = 100
23       addi $t18, $zero, 100 # $t18 = 100
24       addi $t19, $zero, 100 # $t19 = 100
25       addi $t20, $zero, 100 # $t20 = 100
26       addi $t21, $zero, 100 # $t21 = 100
27       addi $t22, $zero, 0 # (call)
28       addi $t23, $zero, 0 # (return)
29       addi $t24, $zero, 4 # $t24 = 4 for print string
30       addi $t25, $zero, 5 # print to the log
31       syscall
```

S	T	A	V	Stack	Log
\$0				10	
\$1				9	
\$2				8	
\$3				7	
\$4				6	
\$5				5	
\$6				4	
\$7				3	
\$8				2	
\$9				1	
\$10				0	
\$11				418	
\$12					
\$13					
\$14					
\$15					
\$16					
\$17					
\$18					
\$19					
\$20					
\$21					
\$22					
\$23					
\$24					
\$25					

- **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)

In Pairs or Triples:

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 # $t0 = 72
LD $t0, 0($sp) # Load 72 into $t0
SW $t0, 16($sp) # Store $t0 at address 16($sp)
ADDI $t0, $zero, 101 # $t0 = 101
LD $t0, 32($sp) # Load 101 into $t0
SW $t0, 48($sp) # Store $t0 at address 48($sp)
ADDI $t0, $zero, 108 # $t0 = 108
LD $t0, 48($sp) # Load 108 into $t0
SW $t0, 64($sp) # Store $t0 at address 64($sp)
ADDI $t0, $zero, 115 # $t0 = 115
LD $t0, 56($sp) # Load 115 into $t0
SW $t0, 80($sp) # Store $t0 at address 80($sp)
ADDI $t0, $zero, 111 # $t0 = 111
LD $t0, 64($sp) # Load 111 into $t0
SW $t0, 88($sp) # Store $t0 at address 88($sp)
ADDI $t0, $zero, 114 # $t0 = 114
LD $t0, 72($sp) # Load 114 into $t0
SW $t0, 96($sp) # Store $t0 at address 96($sp)
ADDI $t0, $zero, 109 # $t0 = 109
LD $t0, 80($sp) # Load 109 into $t0
SW $t0, 104($sp) # Store $t0 at address 104($sp)
ADDI $t0, $zero, 103 # $t0 = 103
LD $t0, 88($sp) # Load 103 into $t0
SW $t0, 112($sp) # Store $t0 at address 112($sp)
ADDI $t0, $zero, 99 # $t0 = 99
LD $t0, 96($sp) # Load 99 into $t0
SW $t0, 116($sp) # Store $t0 at address 116($sp)
ADDI $t0, $zero, 107 # $t0 = 107
LD $t0, 104($sp) # Load 107 into $t0
SW $t0, 120($sp) # Store $t0 at address 120($sp)
ADDI $t0, $zero, 102 # $t0 = 102
LD $t0, 112($sp) # Load 102 into $t0
SW $t0, 128($sp) # Store $t0 at address 128($sp)
ADDI $t0, $zero, 98 # $t0 = 98
LD $t0, 120($sp) # Load 98 into $t0
SW $t0, 136($sp) # Store $t0 at address 136($sp)
ADDI $t0, $zero, 94 # $t0 = 94
LD $t0, 128($sp) # Load 94 into $t0
SW $t0, 144($sp) # Store $t0 at address 144($sp)
ADDI $t0, $zero, 90 # $t0 = 90
LD $t0, 136($sp) # Load 90 into $t0
SW $t0, 152($sp) # Store $t0 at address 152($sp)
ADDI $t0, $zero, 86 # $t0 = 86
LD $t0, 144($sp) # Load 86 into $t0
SW $t0, 160($sp) # Store $t0 at address 160($sp)
ADDI $t0, $zero, 82 # $t0 = 82
LD $t0, 152($sp) # Load 82 into $t0
SW $t0, 168($sp) # Store $t0 at address 168($sp)
ADDI $t0, $zero, 78 # $t0 = 78
LD $t0, 160($sp) # Load 78 into $t0
SW $t0, 176($sp) # Store $t0 at address 176($sp)
ADDI $t0, $zero, 74 # $t0 = 74
LD $t0, 168($sp) # Load 74 into $t0
SW $t0, 184($sp) # Store $t0 at address 184($sp)
ADDI $t0, $zero, 70 # $t0 = 70
LD $t0, 176($sp) # Load 70 into $t0
SW $t0, 192($sp) # Store $t0 at address 192($sp)
ADDI $t0, $zero, 66 # $t0 = 66
LD $t0, 184($sp) # Load 66 into $t0
SW $t0, 200($sp) # Store $t0 at address 200($sp)
ADDI $t0, $zero, 62 # $t0 = 62
LD $t0, 192($sp) # Load 62 into $t0
SW $t0, 208($sp) # Store $t0 at address 208($sp)
ADDI $t0, $zero, 58 # $t0 = 58
LD $t0, 200($sp) # Load 58 into $t0
SW $t0, 216($sp) # Store $t0 at address 216($sp)
ADDI $t0, $zero, 54 # $t0 = 54
LD $t0, 208($sp) # Load 54 into $t0
SW $t0, 224($sp) # Store $t0 at address 224($sp)
ADDI $t0, $zero, 50 # $t0 = 50
LD $t0, 216($sp) # Load 50 into $t0
SW $t0, 232($sp) # Store $t0 at address 232($sp)
ADDI $t0, $zero, 46 # $t0 = 46
LD $t0, 224($sp) # Load 46 into $t0
SW $t0, 240($sp) # Store $t0 at address 240($sp)
ADDI $t0, $zero, 42 # $t0 = 42
LD $t0, 232($sp) # Load 42 into $t0
SW $t0, 248($sp) # Store $t0 at address 248($sp)
ADDI $t0, $zero, 38 # $t0 = 38
LD $t0, 240($sp) # Load 38 into $t0
SW $t0, 256($sp) # Store $t0 at address 256($sp)
ADDI $t0, $zero, 34 # $t0 = 34
LD $t0, 248($sp) # Load 34 into $t0
SW $t0, 264($sp) # Store $t0 at address 264($sp)
ADDI $t0, $zero, 30 # $t0 = 30
LD $t0, 256($sp) # Load 30 into $t0
SW $t0, 272($sp) # Store $t0 at address 272($sp)
ADDI $t0, $zero, 26 # $t0 = 26
LD $t0, 264($sp) # Load 26 into $t0
SW $t0, 280($sp) # Store $t0 at address 280($sp)
ADDI $t0, $zero, 22 # $t0 = 22
LD $t0, 272($sp) # Load 22 into $t0
SW $t0, 288($sp) # Store $t0 at address 288($sp)
ADDI $t0, $zero, 18 # $t0 = 18
LD $t0, 280($sp) # Load 18 into $t0
SW $t0, 296($sp) # Store $t0 at address 296($sp)
ADDI $t0, $zero, 14 # $t0 = 14
LD $t0, 288($sp) # Load 14 into $t0
SW $t0, 304($sp) # Store $t0 at address 304($sp)
ADDI $t0, $zero, 10 # $t0 = 10
LD $t0, 296($sp) # Load 10 into $t0
SW $t0, 312($sp) # Store $t0 at address 312($sp)
ADDI $t0, $zero, 6 # $t0 = 6
LD $t0, 304($sp) # Load 6 into $t0
SW $t0, 320($sp) # Store $t0 at address 320($sp)
ADDI $t0, $zero, 2 # $t0 = 2
LD $t0, 312($sp) # Load 2 into $t0
SW $t0, 328($sp) # Store $t0 at address 328($sp)
ADDI $t0, $zero, 0 # $t0 = 0
LD $t0, 320($sp) # Load 0 into $t0
SW $t0, 336($sp) # Store $t0 at address 336($sp)

# Print string
# $t0 = 4 is for print string
# $t1 = 144 is for print the log
# $t2 = 0 is for print the log
```

(Demo with WeMIPS)

Today's Topics



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

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 0x00000000, there is a label 'Label' followed by a colon and the assembly instruction 'JMP Label'. The bottom of the window has standard file menu options like File, Edit, View, Insert, Options, Window, and Help.

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.



Jump Demo

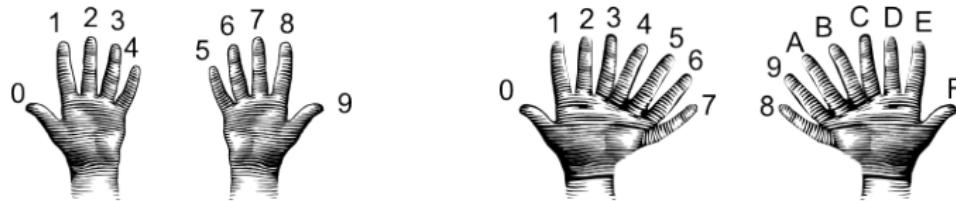
(Demo with WeMIPS)

Today's Topics



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

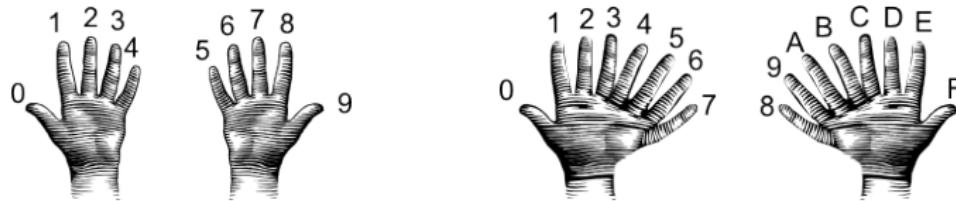
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

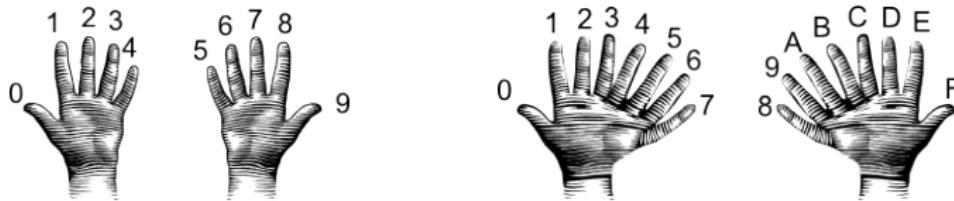
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

Hexadecimal to Decimal: Converting Between Bases

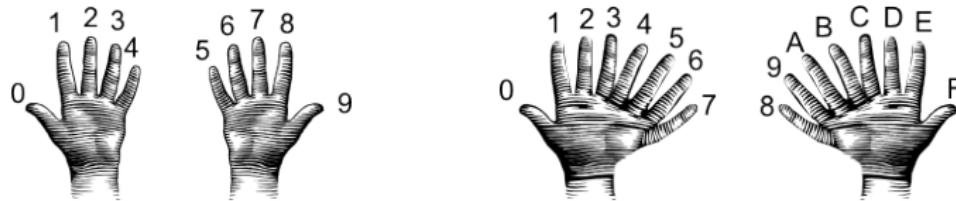


(from i-programmer.info)

- From hexadecimal to decimal:

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

Hexadecimal to Decimal: Converting Between Bases

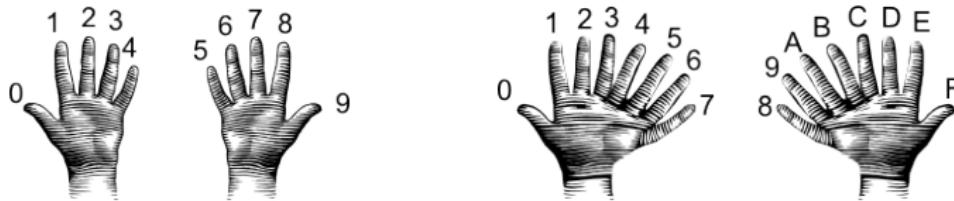


(from i-programmer.info)

- From hexadecimal to decimal:

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

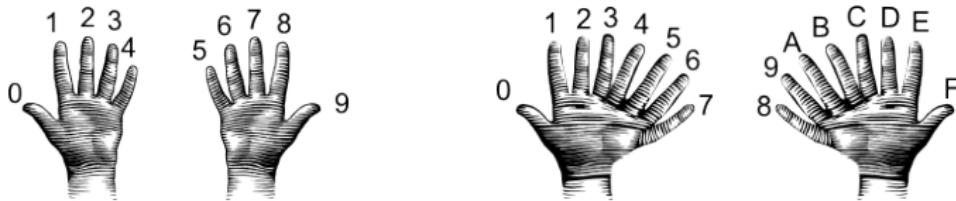
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:
 - Convert first digit to decimal and multiple by 16.
 - Convert second digit to decimal and add to total.
 - Example: what is 2A as a decimal number?
2 in decimal is 2. 2×16 is 32.

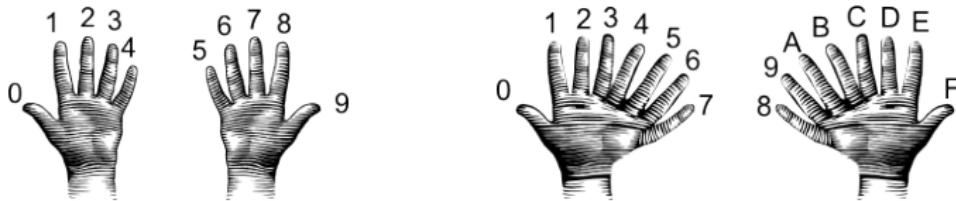
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

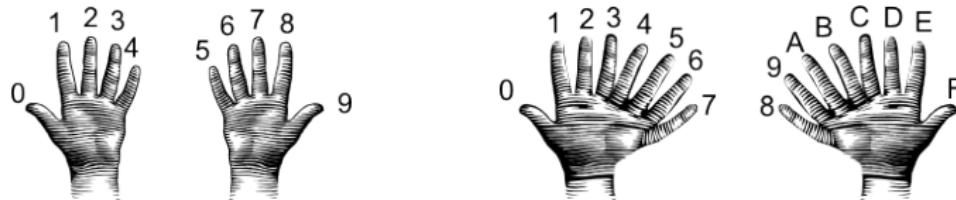
- Convert first digit to decimal and multiple by 16.
- Convert second digit to decimal and add to total.
- Example: what is 2A as a decimal number?

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

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

2 in decimal is 2. 2×16 is 32.

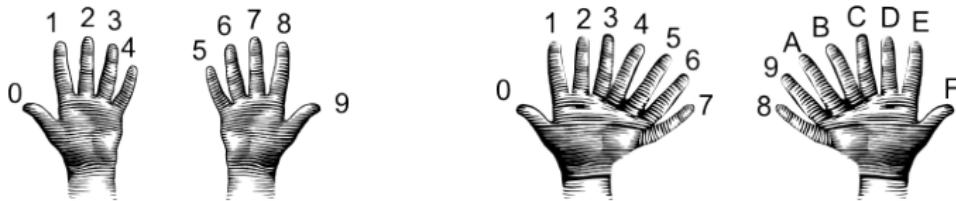
A in decimal digits is 10.

$32 + 10$ is 42.

Answer is 42.

- Example: what is 99 as a decimal number?

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

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

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

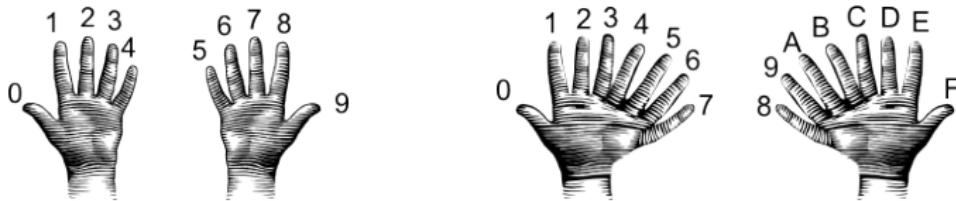
$32 + 10$ is 42.

Answer is 42.

- Example: what is 99 as a decimal number?

9 in decimal is 9.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

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

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

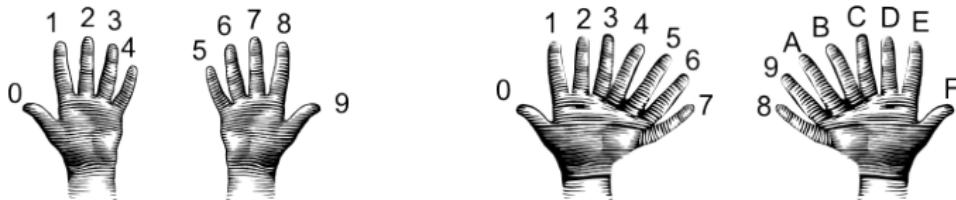
$32 + 10$ is 42.

Answer is 42.

- Example: what is 99 as a decimal number?

9 in decimal is 9. 9×16 is 144.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

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

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

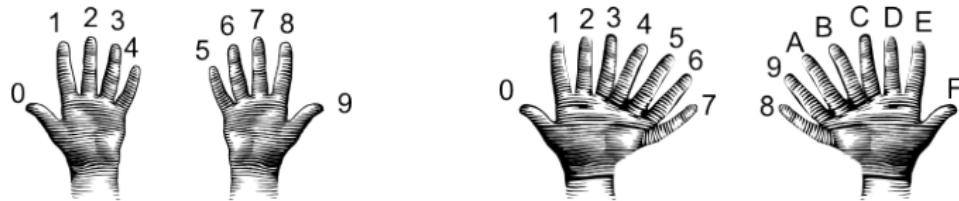
Answer is 42.

- Example: what is 99 as a decimal number?

9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

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

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

Answer is 42.

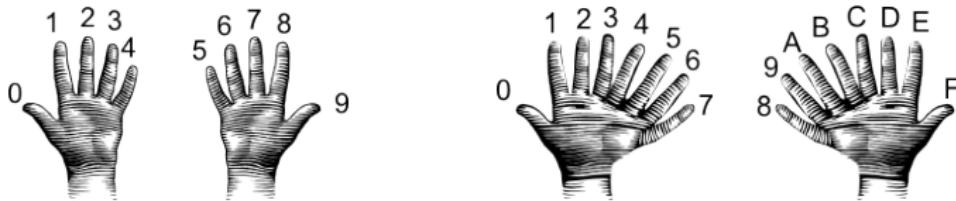
- Example: what is 99 as a decimal number?

9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

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

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

Answer is 42.

- Example: what is 99 as a decimal number?

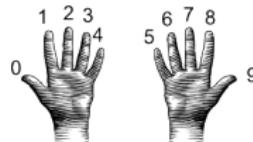
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

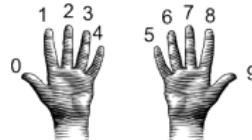
Answer is 153.

Decimal to Binary: Converting Between Bases



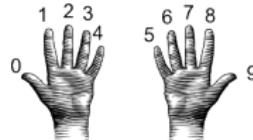
- From decimal to binary:
 - Divide by 128 ($= 2^7$). Quotient is the first digit.

Decimal to Binary: Converting Between Bases



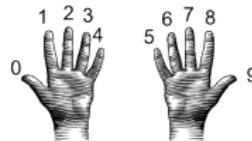
- From decimal to binary:
 - ▶ Divide by $128 (= 2^7)$. Quotient is the first digit.
 - ▶ Divide remainder by $64 (= 2^6)$. Quotient is the next digit.

Decimal to Binary: Converting Between Bases



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

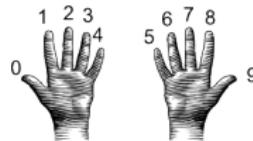
Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

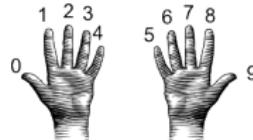
Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

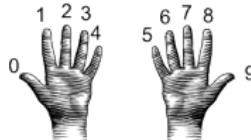
Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

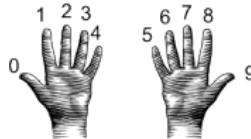
Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

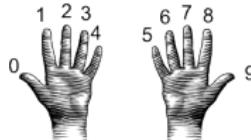
Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

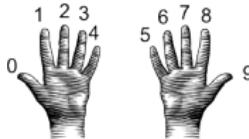
Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

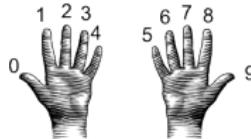


- From decimal to binary:

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

130/128 is 1 rem 2.

Decimal to Binary: Converting Between Bases

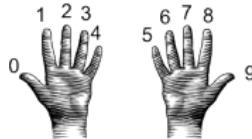


- From decimal to binary:

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

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

Decimal to Binary: Converting Between Bases



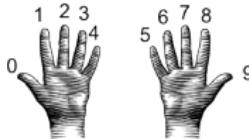
- From decimal to binary:

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

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

2/64 is 0 rem 2.

Decimal to Binary: Converting Between Bases



- From decimal to binary:

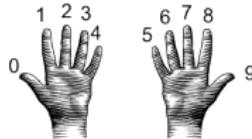
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

- Example: what is 130 in binary notation?

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

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

Decimal to Binary: Converting Between Bases



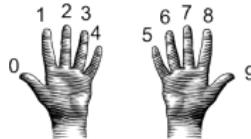
- From decimal to binary:

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

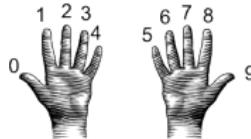
- ▶ Divide by $128 (= 2^7)$. Quotient is the first digit.
- ▶ Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- ▶ Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- ▶ Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- ▶ Divide remainder by $8 (= 2^3)$. Quotient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

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

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

2/32 is 0 rem 2.

Decimal to Binary: Converting Between Bases



- From decimal to binary:

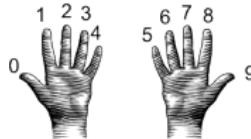
- ▶ Divide by $128 (= 2^7)$. Quotient is the first digit.
- ▶ Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- ▶ Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- ▶ Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- ▶ Divide remainder by $8 (= 2^3)$. Quotient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

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

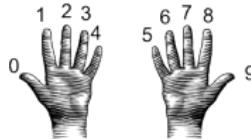
- ▶ Divide by $128 (= 2^7)$. Quotient is the first digit.
- ▶ Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- ▶ Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- ▶ Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- ▶ Divide remainder by $8 (= 2^3)$. Quotient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

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

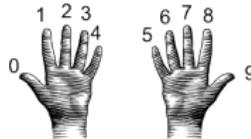
130/128 is 1 rem 2. First digit is 1: 1...

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

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

2/16 is 0 rem 2.

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

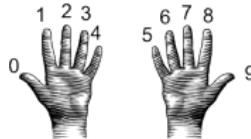
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

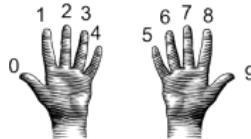
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

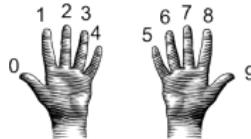
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

2/8 is 0 rem 2.

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

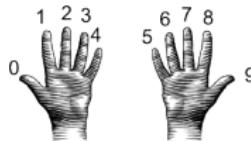
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

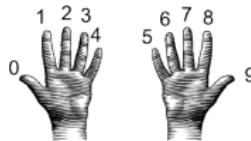
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

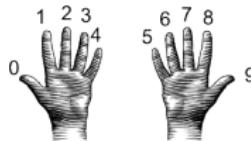
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

2/4 is 0 remainder 2.

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

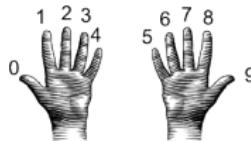
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

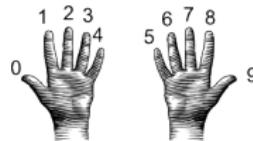
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

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

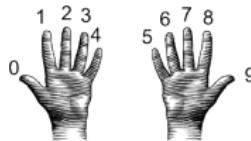
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

2/2 is 1 rem 0.

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

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

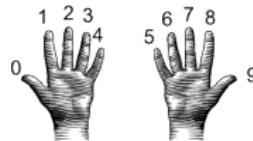
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

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

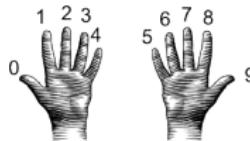
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

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

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

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

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

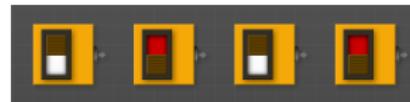
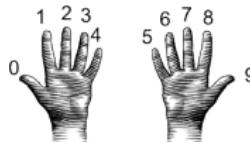
2/8 is 0 rem 2. Next digit is 0: 10000...

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

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

Adding the last remainder: 10000010

Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

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

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

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

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

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

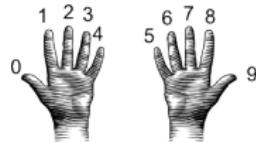
2/4 is 0 remainder 2. Next digit is 0: 100000...

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

Adding the last remainder:

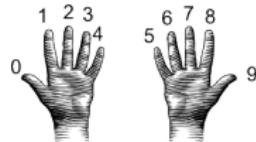
10000010

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

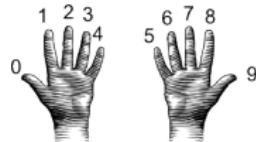
Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

$99/128$ is 0 rem 99.

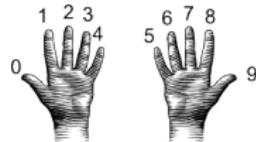
Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases

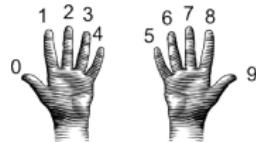


- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

Decimal to Binary: Converting Between Bases

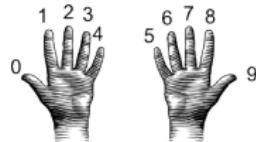


- Example: what is 99 in binary notation?

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

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

Decimal to Binary: Converting Between Bases

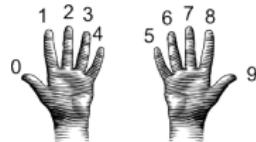


- Example: what is 99 in binary notation?

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

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

Decimal to Binary: Converting Between Bases



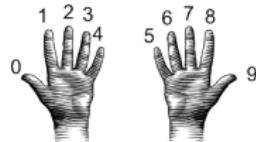
- Example: what is 99 in binary notation?

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

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

35/32 is 1 rem 3.

Decimal to Binary: Converting Between Bases



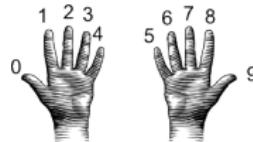
- Example: what is 99 in binary notation?

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

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

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

Decimal to Binary: Converting Between Bases



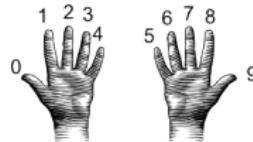
- Example: what is 99 in binary notation?

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

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

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

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

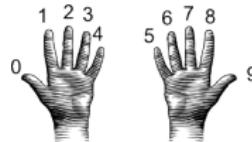
99/128 is 0 rem 99. First digit is 0: 0...

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

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

3/16 is 0 rem 3.

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

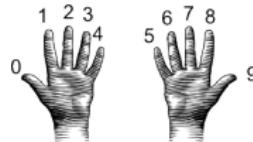
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

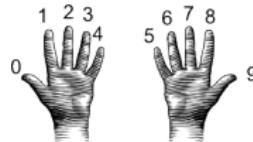
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

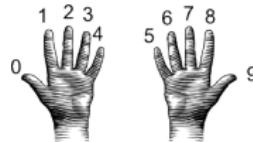
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

3/8 is 0 rem 3.

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

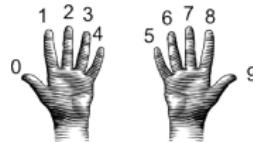
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

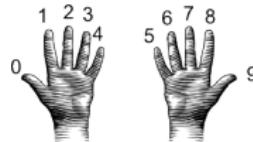
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

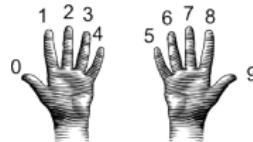
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

3/4 is 0 remainder 3.

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

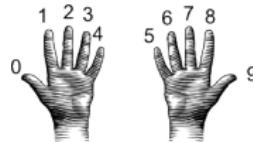
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

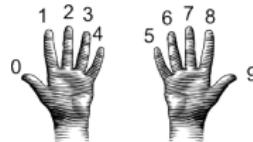
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

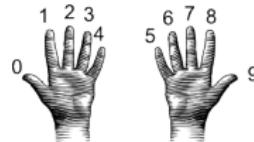
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1.

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

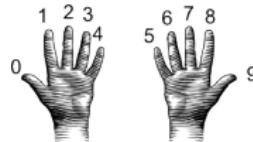
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1. Next digit is 1:

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

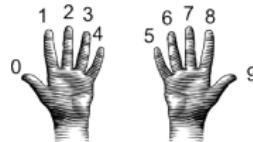
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1. Next digit is 1: 0110001...

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

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

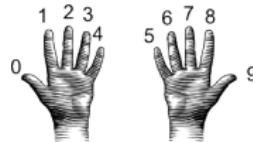
3/8 is 0 rem 3. Next digit is 0: 01100...

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

3/2 is 1 rem 1. Next digit is 1: 0110001...

Adding the last remainder: 01100011

Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

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

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

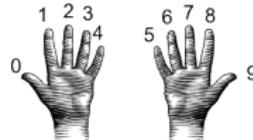
3/4 is 0 remainder 3. Next digit is 0: 011000...

3/2 is 1 rem 1. Next digit is 1: 0110001...

Adding the last remainder: 01100011

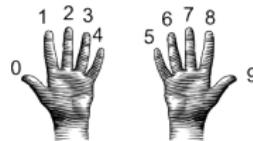
Answer is 1100011.

Binary to Decimal: Converting Between Bases



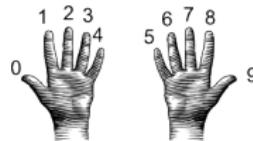
- From binary to decimal:
 - Set sum = last digit.

Binary to Decimal: Converting Between Bases



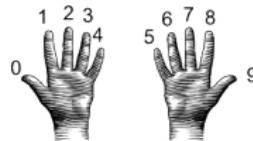
- From binary to decimal:
 - Set sum = last digit.
 - Multiply next digit by 2^1 . Add to sum.

Binary to Decimal: Converting Between Bases



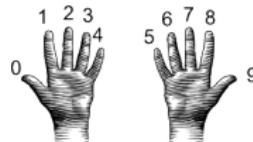
- From binary to decimal:
 - Set sum = last digit.
 - Multiply next digit by $2 = 2^1$. Add to sum.
 - Multiply next digit by $4 = 2^2$. Add to sum.

Binary to Decimal: Converting Between Bases



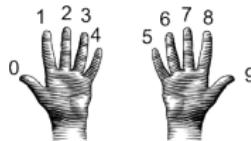
- From binary to decimal:
 - ▶ Set sum = last digit.
 - ▶ Multiply next digit by 2^1 . Add to sum.
 - ▶ Multiply next digit by 2^2 . Add to sum.
 - ▶ Multiply next digit by 2^3 . Add to sum.

Binary to Decimal: Converting Between Bases



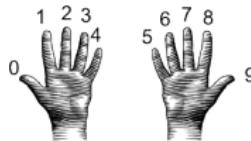
- From binary to decimal:
 - Set sum = last digit.
 - Multiply next digit by 2^1 . Add to sum.
 - Multiply next digit by $4 = 2^2$. Add to sum.
 - Multiply next digit by $8 = 2^3$. Add to sum.
 - Multiply next digit by $16 = 2^4$. Add to sum.

Binary to Decimal: Converting Between Bases



- 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.

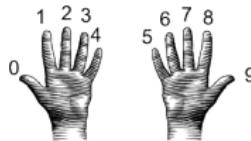
Binary to Decimal: Converting Between Bases



- 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.

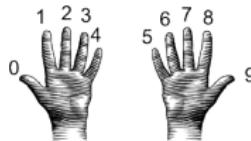
Binary to Decimal: Converting Between Bases



- 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.

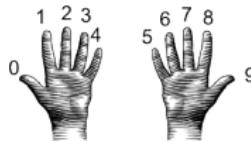
Binary to Decimal: Converting Between Bases



- 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.

Binary to Decimal: Converting Between Bases

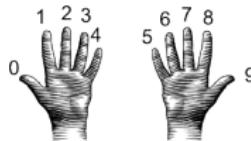


- From binary to decimal:

- 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:

Binary to Decimal: Converting Between Bases

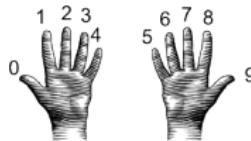


- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases

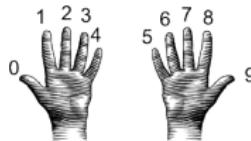


- From binary to decimal:

- 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



- From binary to decimal:

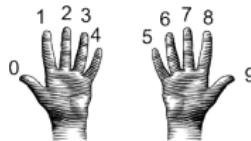
- Set sum = last digit.
- Multiply next digit by 2^1 . Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$. Add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

Sum starts with: 1

$0 * 2 = 0$. Add 0 to sum: 1

$1 * 4 = 4$. Add 4 to sum:

Binary to Decimal: Converting Between Bases

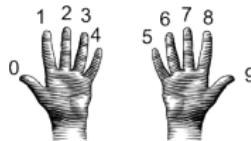


- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



- From binary to decimal:

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

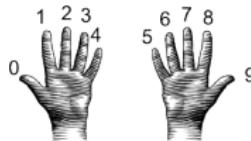
Sum starts with: 1

$0 * 2 = 0$. Add 0 to sum: 1

$1 * 4 = 4$. Add 4 to sum: 5

$1 * 8 = 8$. Add 8 to sum:

Binary to Decimal: Converting Between Bases



- From binary to decimal:

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

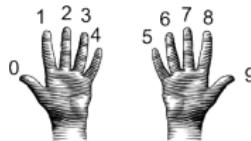
Sum starts with: 1

$0 * 2 = 0$. Add 0 to sum: 1

$1 * 4 = 4$. Add 4 to sum: 5

$1 * 8 = 8$. Add 8 to sum: 13

Binary to Decimal: Converting Between Bases

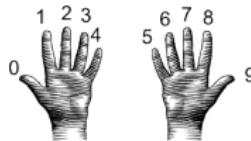


- From binary to decimal:

- 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
 $0 \cdot 2 = 0$. Add 0 to sum: 1
 $1 \cdot 4 = 4$. Add 4 to sum: 5
 $1 \cdot 8 = 8$. Add 8 to sum: 13
 $1 \cdot 16 = 16$. Add 16 to sum:

Binary to Decimal: Converting Between Bases

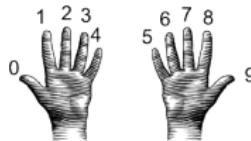


- From binary to decimal:

- 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
 $0 \cdot 2 = 0$. Add 0 to sum: 1
 $1 \cdot 4 = 4$. Add 4 to sum: 5
 $1 \cdot 8 = 8$. Add 8 to sum: 13
 $1 \cdot 16 = 16$. Add 16 to sum: 29

Binary to Decimal: Converting Between Bases

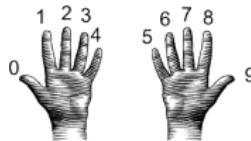


- From binary to decimal:

- 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
$0 \cdot 2 = 0$. Add 0 to sum:	1
$1 \cdot 4 = 4$. Add 4 to sum:	5
$1 \cdot 8 = 8$. Add 8 to sum:	13
$1 \cdot 16 = 16$. Add 16 to sum:	29
$1 \cdot 32 = 32$. Add 32 to sum:	

Binary to Decimal: Converting Between Bases

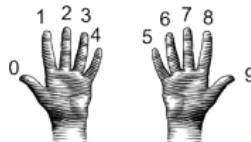


- From binary to decimal:

- 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
$0 \cdot 2 = 0$. Add 0 to sum:	1
$1 \cdot 4 = 4$. Add 4 to sum:	5
$1 \cdot 8 = 8$. Add 8 to sum:	13
$1 \cdot 16 = 16$. Add 16 to sum:	29
$1 \cdot 32 = 32$. Add 32 to sum:	61

Binary to Decimal: Converting Between Bases

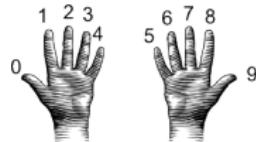


- From binary to decimal:

- 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
$0 \cdot 2 = 0$. Add 0 to sum:	1
$1 \cdot 4 = 4$. Add 4 to sum:	5
$1 \cdot 8 = 8$. Add 8 to sum:	13
$1 \cdot 16 = 16$. Add 16 to sum:	29
$1 \cdot 32 = 32$. Add 32 to sum:	61

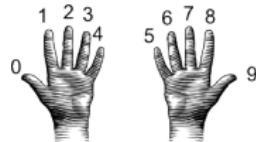
Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:

Binary to Decimal: Converting Between Bases

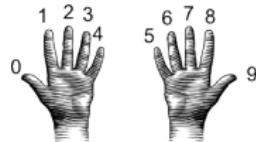


- Example: What is 10100100 in decimal?

Sum starts with: 0

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

Binary to Decimal: Converting Between Bases

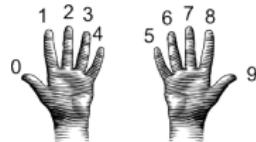


- Example: What is 10100100 in decimal?

Sum starts with: 0

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

Binary to Decimal: Converting Between Bases



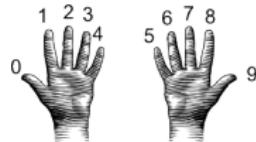
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

$1 * 4 = 4$. Add 4 to sum:

Binary to Decimal: Converting Between Bases



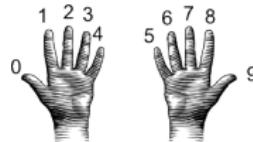
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

$1 * 4 = 4$. Add 4 to sum: 4

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

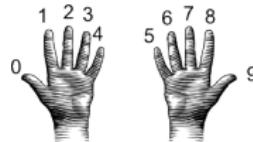
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

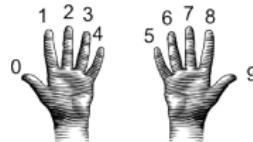
Sum starts with: 0

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

$1 * 4 = 4$. Add 4 to sum: 4

$0 * 8 = 0$. Add 0 to sum: 4

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

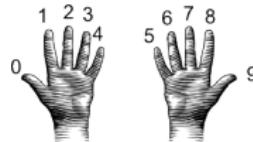
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

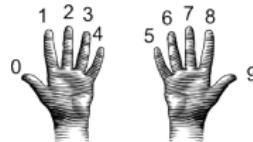
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

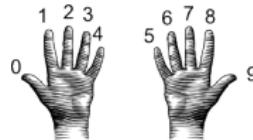
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

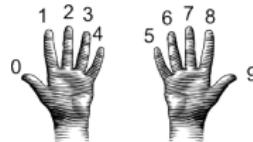
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

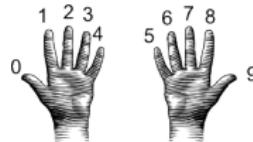
$0 \times 8 = 0$. Add 0 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

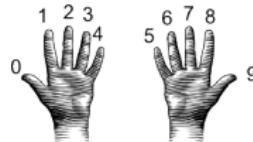
$0 \times 8 = 0$. Add 0 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

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

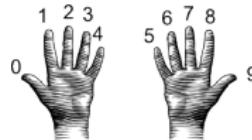
$0 \times 16 = 0$. Add 0 to sum: 4

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

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

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

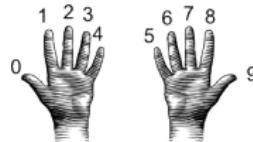
Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases

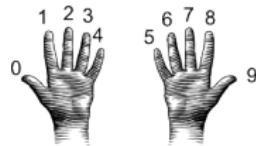


- Example: What is 10100100 in decimal?

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

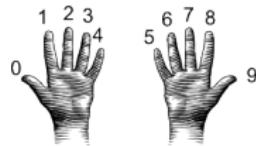
The answer is 164.

Design Challenge: Incrementers



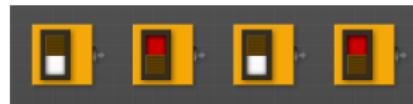
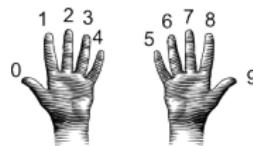
- Simplest arithmetic: add one ("increment") a variable.

Design Challenge: Incrementers



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

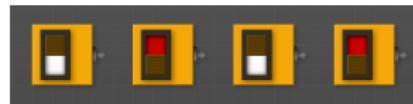
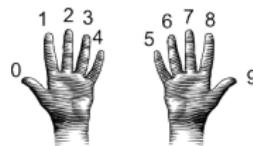
Design Challenge: Incrementers



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

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

Design Challenge: Incrementers

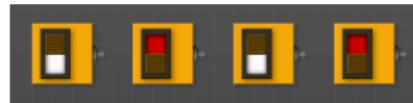
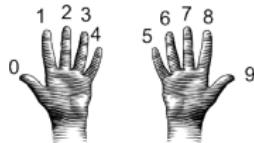


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

Design Challenge: Incrementers

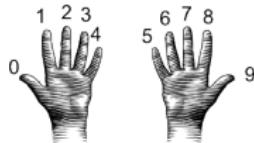


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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

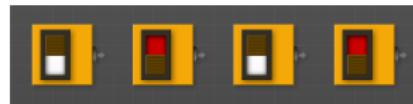
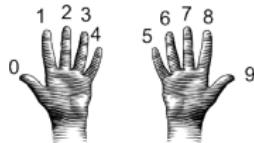


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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

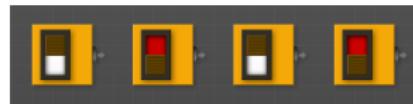
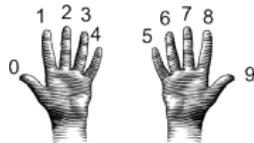


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



- 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

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



Recap



- On lecture slip, write down a topic you wish we had spent more time (and why).
- Searching through data is a common task— built-in functions and standard design patterns for this.

Recap



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



- On lecture slip, write down a topic you wish we had spent more time (and why).
- 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.
- Pass your lecture slips to the aisles for the UTAs to collect.

Today's Topics



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

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.
 - ▶ 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

18 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- You may bring one closed book and three notes with the exception of an 8.5" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have white pens and pencils, and your note sheet.
- You may not use calculators, electronic calculators, mobile phones, or other electronic devices.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, offering unfair advantages, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The Hunter College Code of Academic Integrity (Section 127.20) defines academic dishonesty and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedure.

<small>I understand that all cases of academic dishonesty will be reported to the Dean of Students and Student Conduct.</small>
Name _____
Signature _____

Exam Options

Exam Times:

- Default: Regular Time: Monday, 16 December, 9-11am.
- Alternate Time: Reading Day, Friday, 13 December, 8:30am-10:30am.
- Accessibility Testing Center: Paperwork required. Must be completed on 13 December.

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

18 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- You may bring one closed book and that book with the exception of an 8.5" x 11" piece of paper that with notes, programs, etc.
- When taking the exam, you may have whatever pens and pencils, and your note sheet.
- You may not use calculators, electronic calculators, online search, or other electronic devices.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, offering unfair advantages, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook contains the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Office of Student and Conduct Accountability.	
Name:	
Sig#:	
Date:	
Signature:	

Exam Options

Exam Times:

- Default: Regular Time: Monday, 16 December, 9-11am.
- Alternate Time: Reading Day, Friday, 13 December, 8:30am-10:30am.
- Accessibility Testing Center: Paperwork required. Must be completed on 13 December.

Grading Options:

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

13 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- You may bring one closed book and that book with the exception of an 8.5" x 11" piece of paper that with notes, programs, etc.
- When taking the exam, you may have whatever pens and pencils, and your note sheet.
- You may not use calculators, electronic calculators, online search, or other electronic devices.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, offering unfair advantages, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook contains the policy on dealing with acts of academic dishonesty and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedure.

I understand that all cases of academic dishonesty will be reported to the Office of Student and Academic Conduct.	
Name:	
Signature:	
Date:	
Signature:	

Exam Options

Exam Times:

- Default: Regular Time: Monday, 16 December, 9-11am.
- Alternate Time: Reading Day, Friday, 13 December, 8:30am-10:30am.
- Accessibility Testing Center: Paperwork required. Must be completed on 13 December.

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- You may bring one closed book and that book with the exception of an 8.5" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have whatever pens and pencils, and your note sheet.
- You may not use calculators, electronic calculators, online search, or other electronic devices.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, offering unfair advantages, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook contains the Hunter College Academic Integrity Procedures.

Students must affix their name to the line of handwriting and print below.
Name:
Signature:
Handwriting:
Date:
Signature:

Grading Options:

- Default: Letter Grade.
- Credit/NoCredit grade— availability depends on major and academic standing.

Exam Options

Exam Times:

- Default: Regular Time: Monday, 16 December, 9-11am.
- Alternate Time: Reading Day, Friday, 13 December, 8:30am-10:30am.
- Accessibility Testing Center: Paperwork required. Must be completed on 13 December.

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

13 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- You may bring one book and that book with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have whatever pens and pencils, and your note sheet.
- You may not use calculators, electronic calculators, online search, or other electronic devices.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, offering unfair advantages, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook contains the policy on Academic Honesty and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedure.

I understand that all cases of academic dishonesty will be reported to the Office of Student and Conduct Accountability.
Name _____
Signature _____

Grading Options:

- Default: Letter Grade.
- Credit/NoCredit grade— availability depends on major and academic standing.

Forms for your choices (“pink slips”) available next lecture.

Writing Boards



- Return writing boards as you leave...