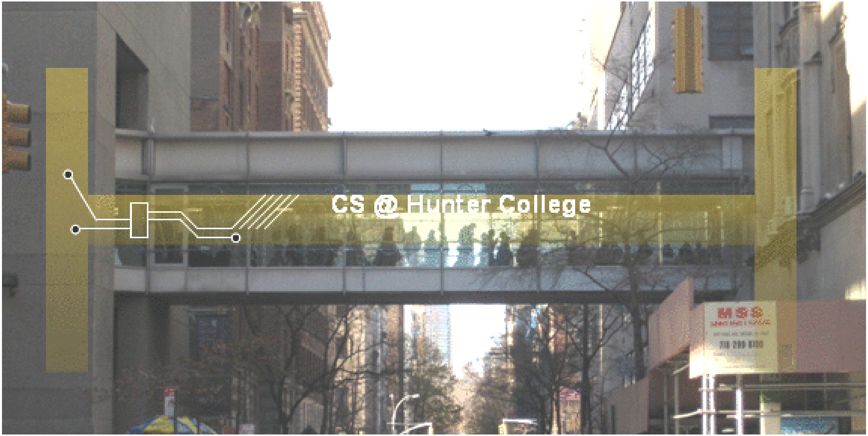
CSci 127: Introduction to Computer Science



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

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

Today's Topics

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

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
#Date:  September 1, 2017
#This program prints:  Hello, World!

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

← *These lines are comments*
← *(for us, not computer to read)*
← *(this one also)*
← *Prints the string "Hello, World!" to the screen*

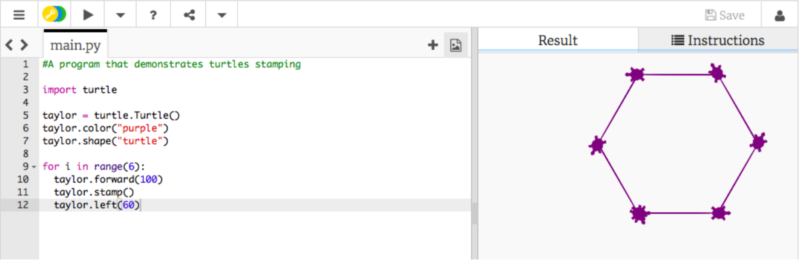
Week 1: print(), loops, comments, & turtles

- Introduced comments & print():

```
#Name:  Thomas Hunter
#Date:  September 1, 2017
#This program prints:  Hello, World!

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

← *These lines are comments*
← *(for us, not computer to read)*
← *(this one also)*
← *Prints the string "Hello, World!" to the screen*

- As well as definite loops & the turtle package:



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

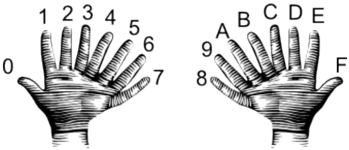
Week 2: variables, data types, more on loops & range()

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

```python
#Predict what will be printed:

for num in [2,4,6,8,10]:
    print(num)

sum = 0
for x in range(0,12,2):
    print(x)
    sum = sum + x

print(x)

for c in "ABCD":
    print(c)
```

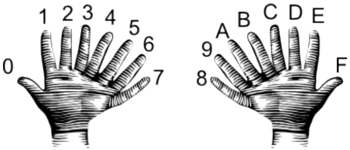
Week 3: colors, hex, slices, numpy & images

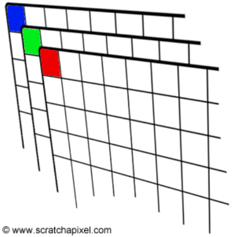
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



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

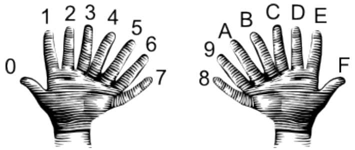
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

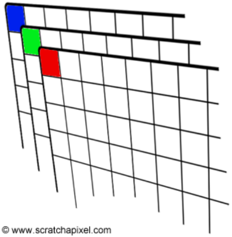




Week 3: colors, hex, slices, numpy & images

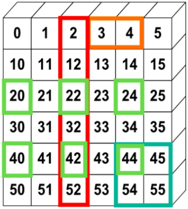
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	





© www.scratchapixel.com

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

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

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

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



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')
```

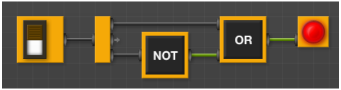
Week 5: logical operators, truth tables & logical circuits

```python
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

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

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

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



Week 6: structured data, pandas, & more design

```
Source:  https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,,
All population figures are consistent with present-day boundaries.,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,,
,,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

`nycHistPop.csv`

In Lab 6

Week 6: structured data, pandas, & more design

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

```
Source:  https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,,
All population figures are consistent with present-day boundaries.,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,,
,,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

`nycHistPop.csv`

In Lab 6

Week 6: structured data, pandas, & more design

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

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

```
Source:  https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,,
All population figures are consistent with present-day boundaries.,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,,
,,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

```
Source:  https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,,
All population figures are consistent with present-day boundaries.,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,,
,,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

`nycHistPop.csv`

In Lab 6

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

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

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

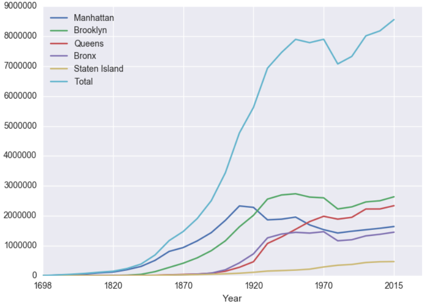
Week 6: structured data, pandas, & more design

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

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

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

```
Source:  https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,,
All population figures are consistent with present-day boundaries.,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,,
,,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6642,1755,4563,79215
1810,96373,8303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5346,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813669,279122,32903,23593,25492,1174779
1870,942292,419921,45468,37393,33029,1478103
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51693,2507414
1900,1850093,1166582,152999,200507,67021,3437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018356,469042,732016,116531,5620048
1930,1867312,2560401,1079129,1265258,158346,6930446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550849,1451277,191555,7891957
1960,1698281,2627319,1809578,1424815,221991,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1585873,2504700,2230722,1385108,468730,8175133
2015,1644518,2636735,2339150,1455444,474558,8550405
```

`nycHistPop.csv`

In Lab 6



Week 7: functions

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

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

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

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

Week 7: functions

```python
#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

```python
#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

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

Formal Parameters

Actual Parameters

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

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


```python
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()

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

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

```python
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()

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

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

```python
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.

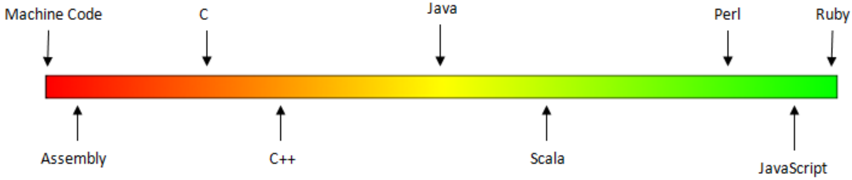
Python & Circuits Review: 10 Weeks in 10 Minutes

- 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

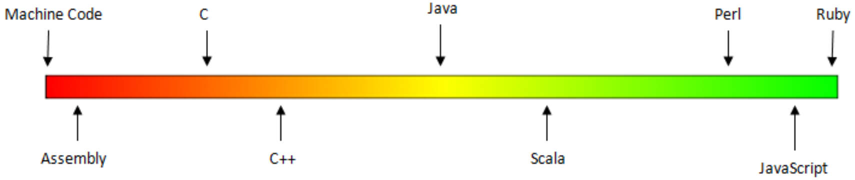
Low-Level vs. High-Level Languages



(codeCommit)

- Can view programming languages on a continuum.

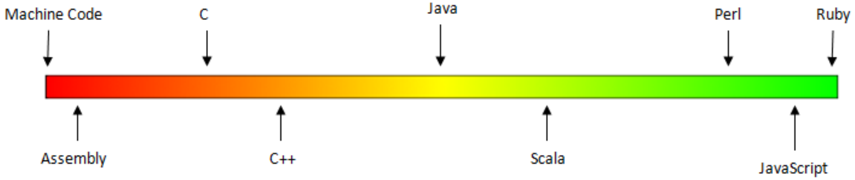
Low-Level vs. High-Level Languages



(codeCommit)

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

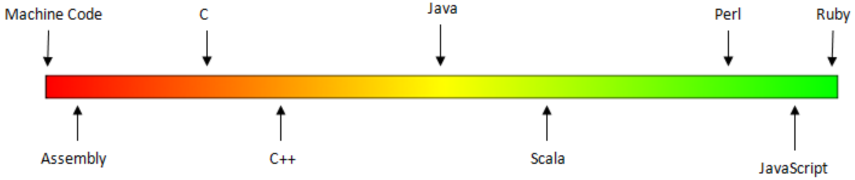
Low-Level vs. High-Level Languages



(codeCommit)

- 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).

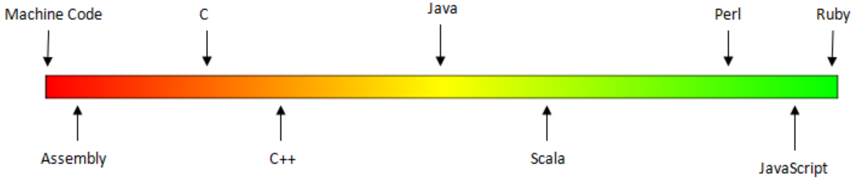
Low-Level vs. High-Level Languages



(codeCommit)

- 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**.

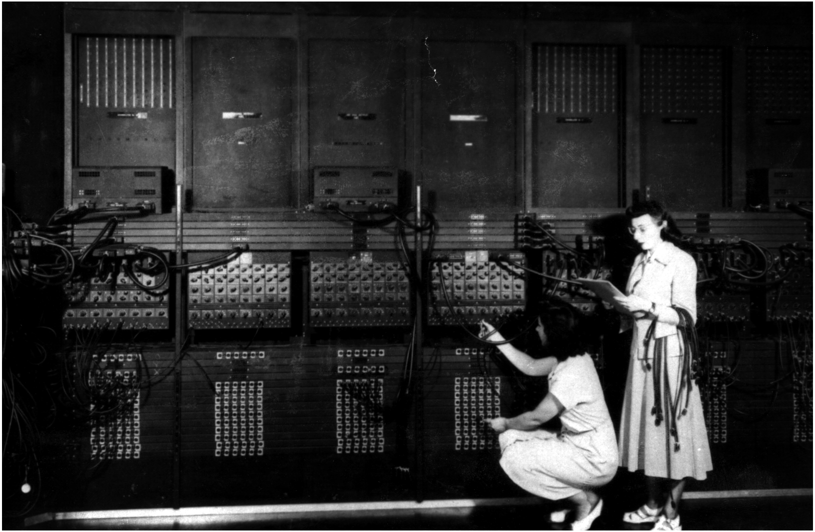
Low-Level vs. High-Level Languages



(codeCommit)

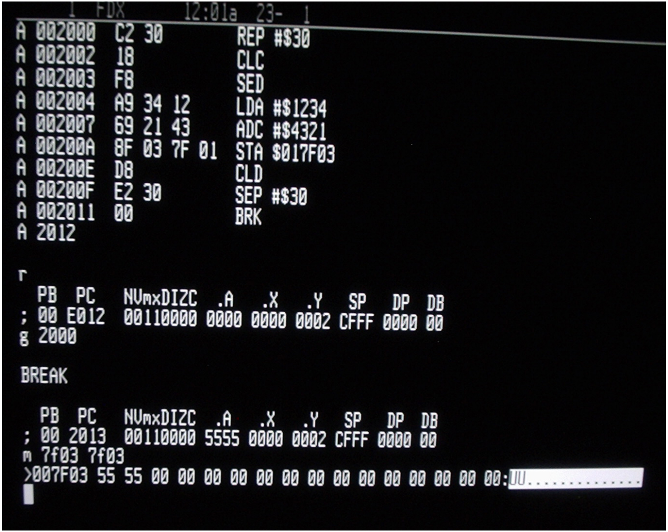
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine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



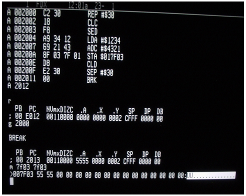
(Ruth Gordon & Ester Gerston programming the ENIAC, UPenn)

Machine Language



(wiki)

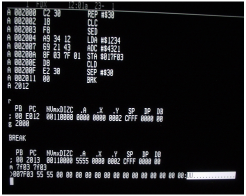
Machine Language



(wiki)

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

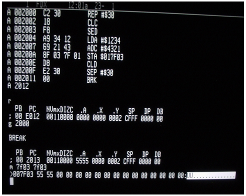
Machine Language



(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.

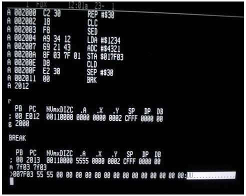
Machine Language



(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.

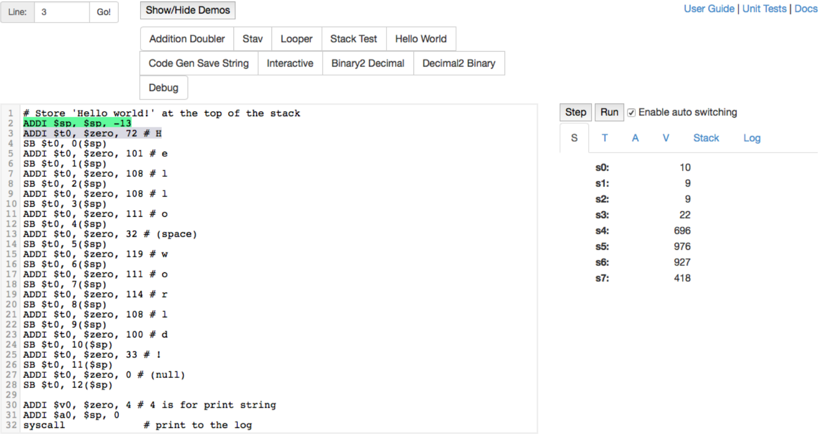
Machine Language



(wiki)

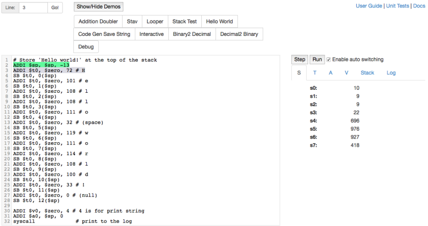
- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.
- Due to its small set of commands, processors can be designed to run those commands very efficiently.
- More in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

“Hello World!” in Simplified Machine Language



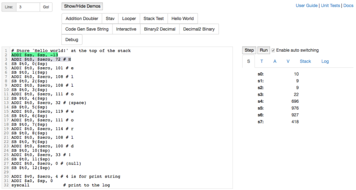
(WeMIPS)

WeMIPS



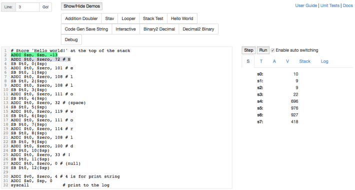
(Demo with `WeMIPS`)

MIPS Commands



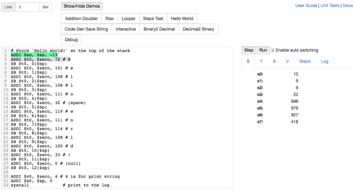
- **Registers:** locations for storing information that can be quickly accessed.

MIPS Commands



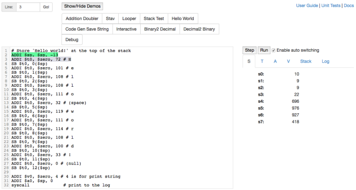
- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': `$s0`, `$s1`, `$t0`, `$t1`,...

MIPS Commands



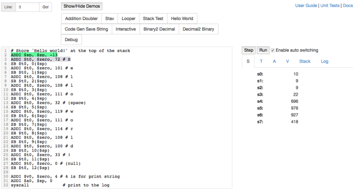
- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:

MIPS Commands



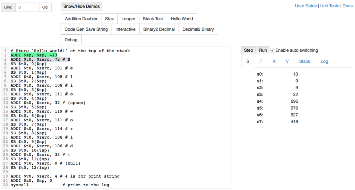
- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:
  `add $s1, $s2, $s3`

MIPS Commands



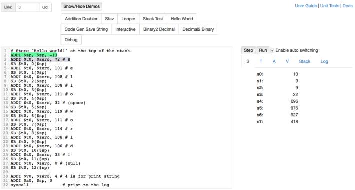
- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers: `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

MIPS Commands



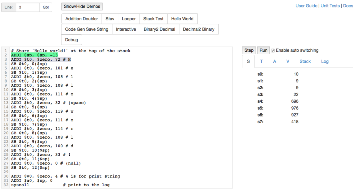
- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:
  `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

MIPS Commands



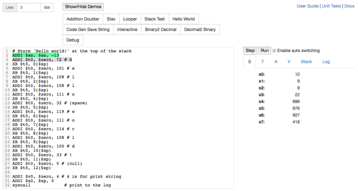
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:
  `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

MIPS Commands



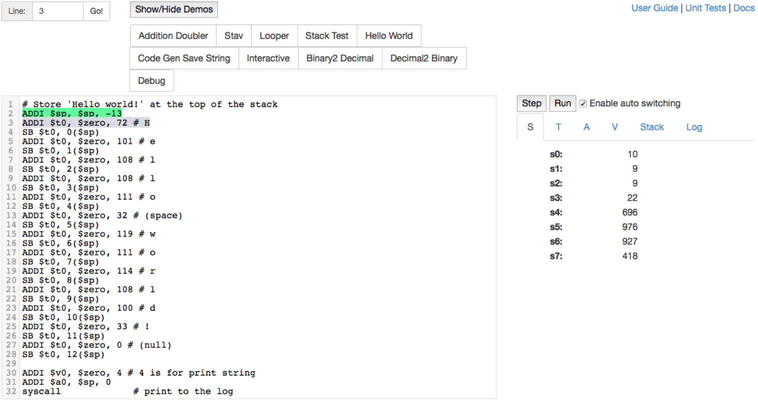
- **Registers:** locations for storing information that can be quickly accessed. Names start with '$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:
  `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

MIPS Commands



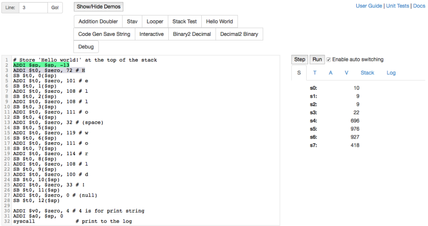
- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': `$s0`, `$s1`, `$t0`, `$t1`,...
- **R Instructions:** Commands that use data in the registers:
  `add $s1, $s2, $s3` (Basic form: `OP rd, rs, rt`)
- **I Instructions:** instructions that also use intermediate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

In Pairs or Triples:



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

WeMIPS

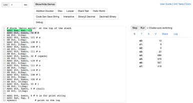


(Demo with `WeMIPS`)

Today's Topics

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

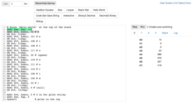
Loops & Jumps in Machine Language



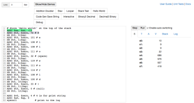
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.

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.

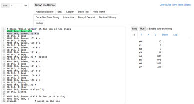


Loops & Jumps in Machine Language



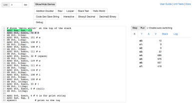
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.

Loops & Jumps in Machine Language



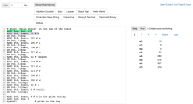
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:

Loops & Jumps in Machine Language



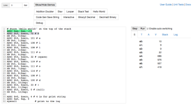
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
  - **Unconditional:**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

Loops & Jumps in Machine Language



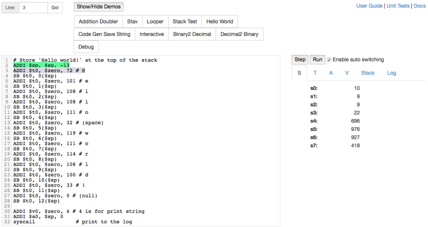
- Instead of built-in looping structures like `for` and `while`, you create your own loops by "jumping" to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
  - **Unconditional:**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

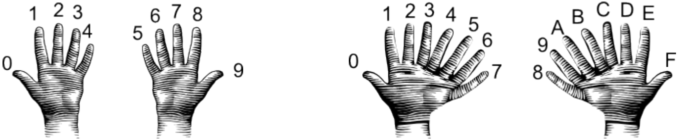


(Demo with `WeMIPS`)

Today's Topics

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

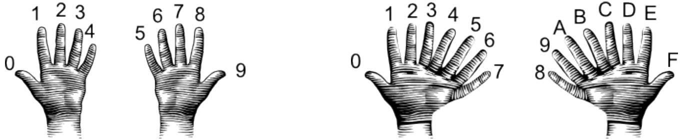
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

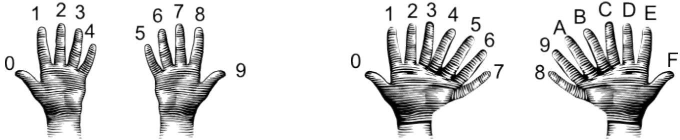
Hexadecimal to Decimal: Converting Between Bases



(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.

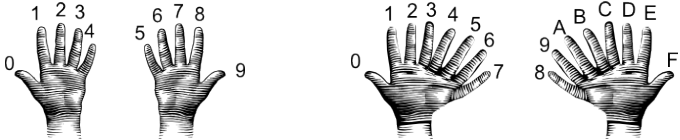
Hexadecimal to Decimal: Converting Between Bases



(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

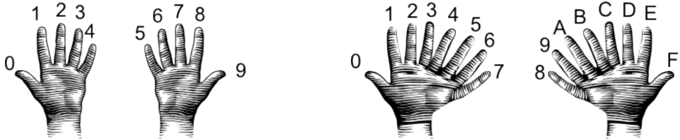


(from i-programmer.info)

- 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

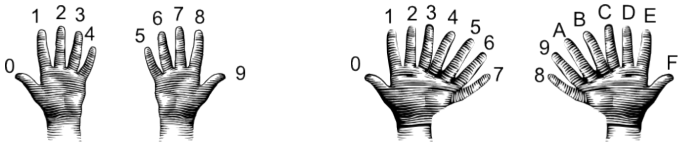


(from i-programmer.info)

- 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



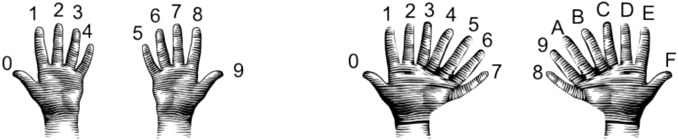
(from i-programmer.info)

- 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

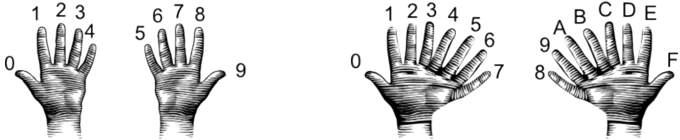


(from i-programmer.info)

- 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?

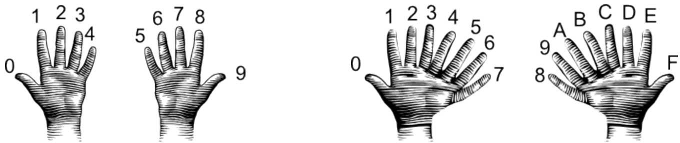
    `2 in decimal is 2. 2*16 is 32.`

    `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.`

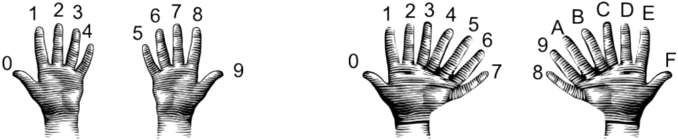
    `A in decimal digits is 10.`

    `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.`

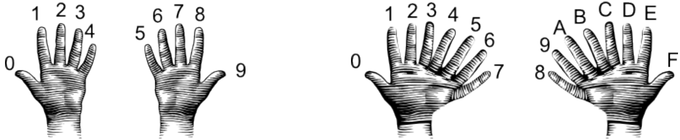
    `A in decimal digits is 10.`

    `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.`

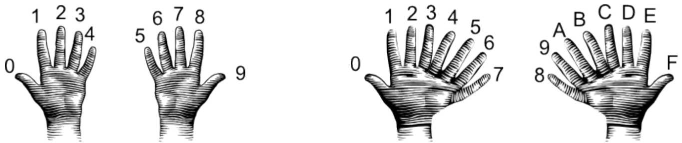
    `32 + 10 is 42.`

    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.`

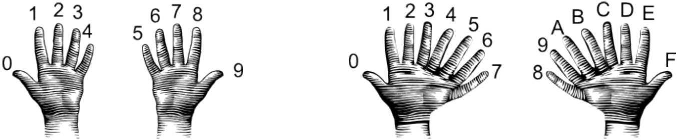
    Answer is 42.
  - Example: what is 99 as a decimal number?

    `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?

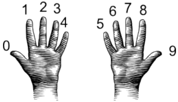
    `9 in decimal is 9. 9*16 is 144.`

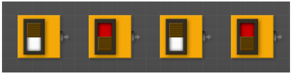
    `9 in decimal digits is 9`

    `144 + 9 is 153.`

    Answer is 153.

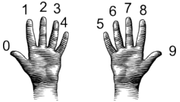
Decimal to Binary: Converting Between Bases

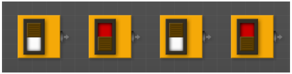




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.

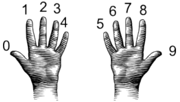
Decimal to Binary: Converting Between Bases

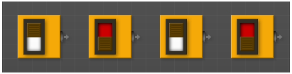




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.

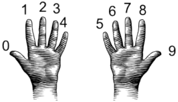
Decimal to Binary: Converting Between Bases

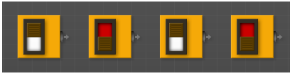




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.

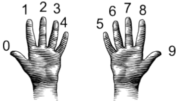
Decimal to Binary: Converting Between Bases

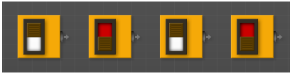




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.

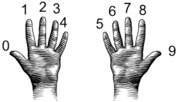
Decimal to Binary: Converting Between Bases

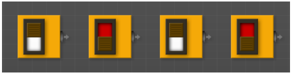




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

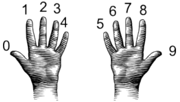
Decimal to Binary: Converting Between Bases

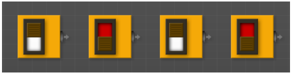




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

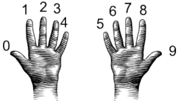
Decimal to Binary: Converting Between Bases

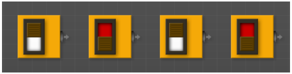




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

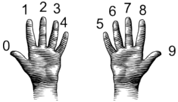
Decimal to Binary: Converting Between Bases

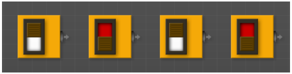




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

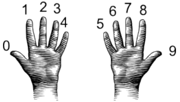
Decimal to Binary: Converting Between Bases

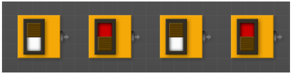




- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

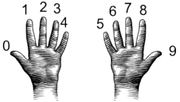


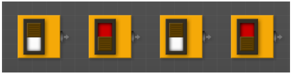


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

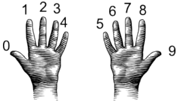


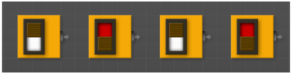


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

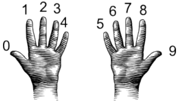


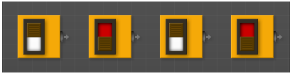


- From decimal to binary:
  - Divide by 128 $(= 2^7)$. Quotient is the first digit.
  - Divide remainder by 64 $(= 2^6)$. Quotient is the next digit.
  - Divide remainder by 32 $(= 2^5)$. Quotient is the next digit.
  - Divide remainder by 16 $(= 2^4)$. Quotient is the next digit.
  - Divide remainder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

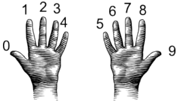


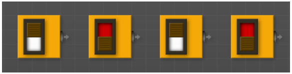


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

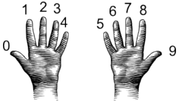


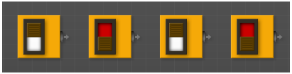


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

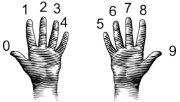


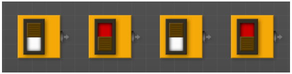


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

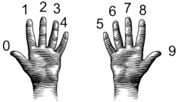


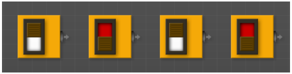


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

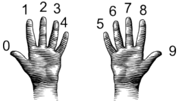


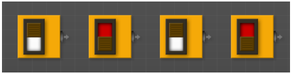


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

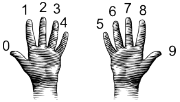


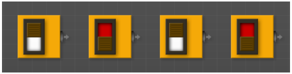


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

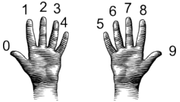


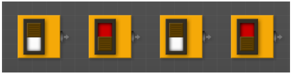


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

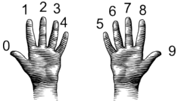


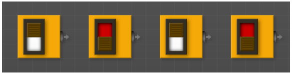


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

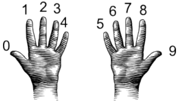


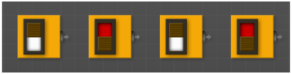


- From decimal to binary:
  - Divide by 128 $(= 2^7)$. Quotient is the first digit.
  - Divide remainder by 64 $(= 2^6)$. Quotient is the next digit.
  - Divide remainder by 32 $(= 2^5)$. Quotient is the next digit.
  - Divide remainder by 16 $(= 2^4)$. Quotient is the next digit.
  - Divide remainder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

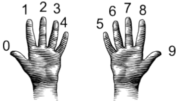


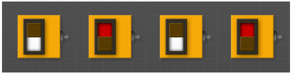


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

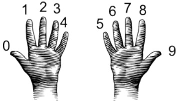


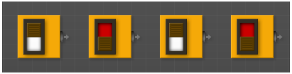


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

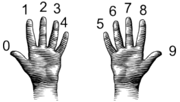


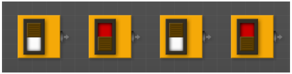


- From decimal to binary:
  - Divide by 128 $(= 2^7)$. Quotient is the first digit.
  - Divide remainder by 64 $(= 2^6)$. Quotient is the next digit.
  - Divide remainder by 32 $(= 2^5)$. Quotient is the next digit.
  - Divide remainder by 16 $(= 2^4)$. Quotient is the next digit.
  - Divide remainder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

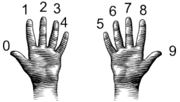


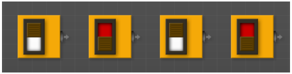


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

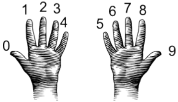


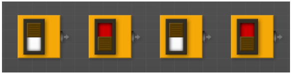


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

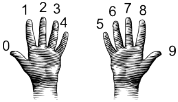


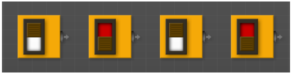


- From decimal to binary:
  - Divide by 128 $(= 2^7)$. Quotient is the first digit.
  - Divide remainder by 64 $(= 2^6)$. Quotient is the next digit.
  - Divide remainder by 32 $(= 2^5)$. Quotient is the next digit.
  - Divide remainder by 16 $(= 2^4)$. Quotient is the next digit.
  - Divide remainder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

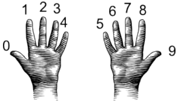


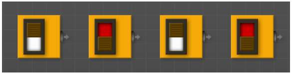


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

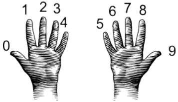


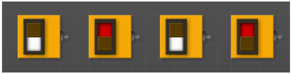


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

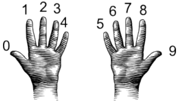


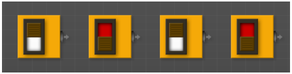


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

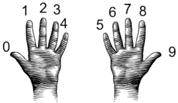


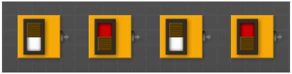


- From decimal to binary:
  - Divide by 128 ($= 2^7$). Quotient is the first digit.
  - Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
  - Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
  - Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
  - Divide remainder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

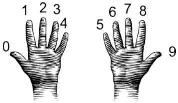
Decimal to Binary: Converting Between Bases

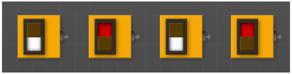




- Example: what is 99 in binary notation?

Decimal to Binary: Converting Between Bases

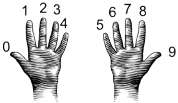


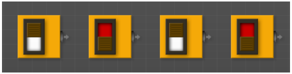


- Example: what is 99 in binary notation?

  `99/128 is 0 rem 99.`

Decimal to Binary: Converting Between Bases

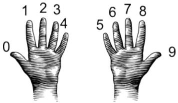


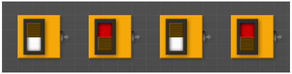


- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases

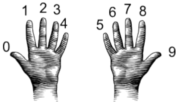


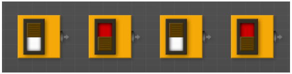


- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases

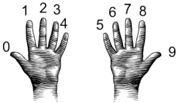


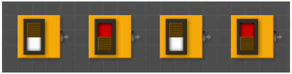


- Example: what is 99 in binary notation?

```
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

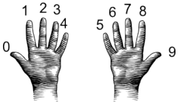


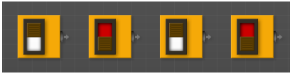


- Example: what is 99 in binary notation?

```
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

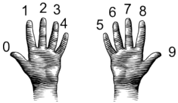


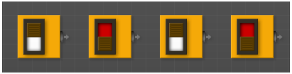


- Example: what is 99 in binary notation?

```
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

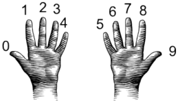


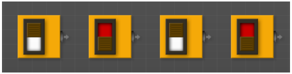


- Example: what is 99 in binary notation?

```
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

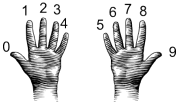


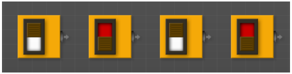


- Example: what is 99 in binary notation?

```
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

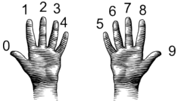


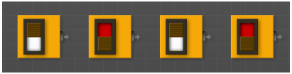


- Example: what is 99 in binary notation?

```
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

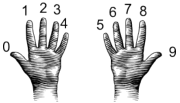


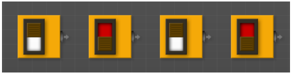


- Example: what is 99 in binary notation?

```
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

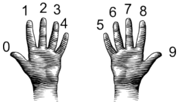


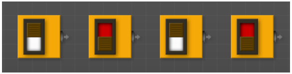


- Example: what is 99 in binary notation?

```
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

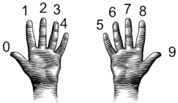


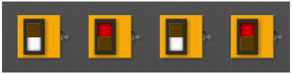


- Example: what is 99 in binary notation?

`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

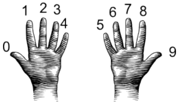


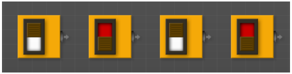


- Example: what is 99 in binary notation?

```
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

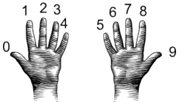


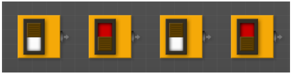


- Example: what is 99 in binary notation?

`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

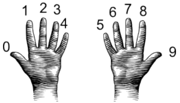


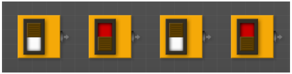


- Example: what is 99 in binary notation?

```
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

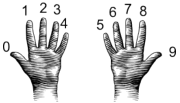


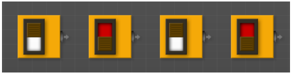


- Example: what is 99 in binary notation?

```
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

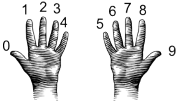


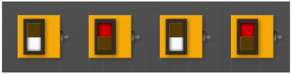


- Example: what is 99 in binary notation?

  `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

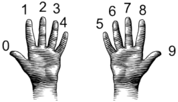


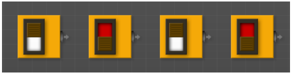


- Example: what is 99 in binary notation?

`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

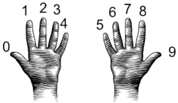


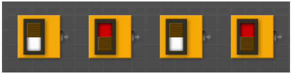


- Example: what is 99 in binary notation?

`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

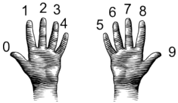


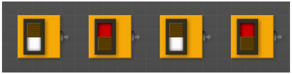


- Example: what is 99 in binary notation?

```
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

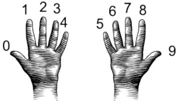


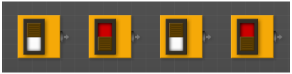


- Example: what is 99 in binary notation?

`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



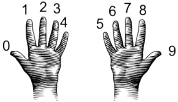


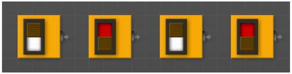
- Example: what is 99 in binary notation?

`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.

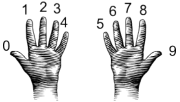
Binary to Decimal: Converting Between Bases

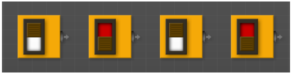




- From binary to decimal:
  - Set sum = last digit.

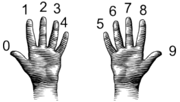
Binary to Decimal: Converting Between Bases

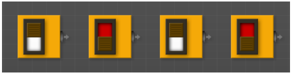




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.

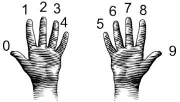
Binary to Decimal: Converting Between Bases

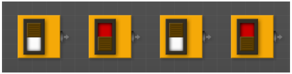




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.

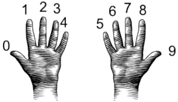
Binary to Decimal: Converting Between Bases

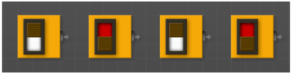




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.

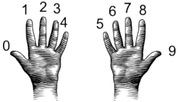
Binary to Decimal: Converting Between Bases

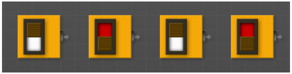




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.

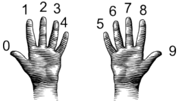
Binary to Decimal: Converting Between Bases

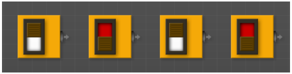




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.

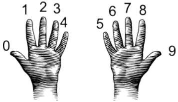
Binary to Decimal: Converting Between Bases

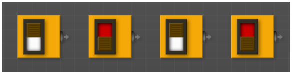




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

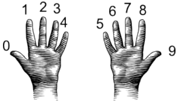
Binary to Decimal: Converting Between Bases

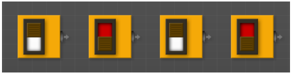




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

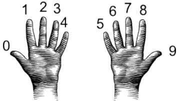
Binary to Decimal: Converting Between Bases

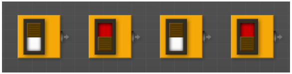




- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

Binary to Decimal: Converting Between Bases

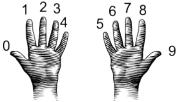


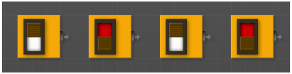


- From binary to decimal:
  - Set sum $=$ last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to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

    `Sum starts with:`

Binary to Decimal: Converting Between Bases

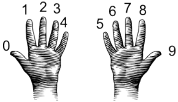


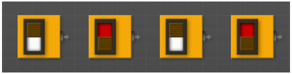


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

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

Binary to Decimal: Converting Between Bases

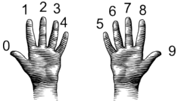


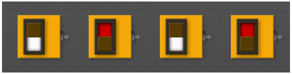


- From binary to decimal:
  - Set sum $=$ last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to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

Binary to Decimal: Converting Between Bases

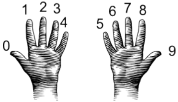


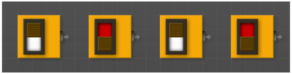


- From binary to decimal:
  - Set sum $=$ last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to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

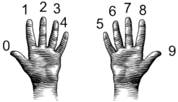


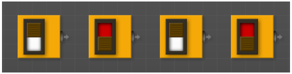


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

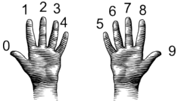


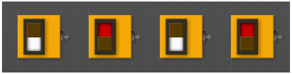


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

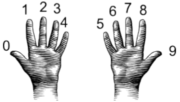


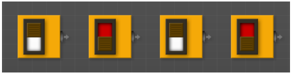


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

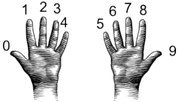


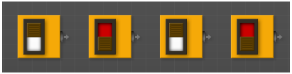


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

```
Sum starts with:              1
0*2 = 0.   Add 0 to sum:      1
1*4 = 4.   Add 4 to sum:      5
1*8 = 8.   Add 8 to sum:      13
1*16 = 16.  Add 16 to sum:
```

Binary to Decimal: Converting Between Bases

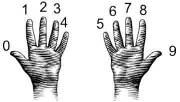


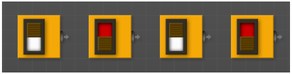


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

```
Sum starts with:           1
0*2 = 0.  Add 0 to sum:    1
1*4 = 4.  Add 4 to sum:    5
1*8 = 8.  Add 8 to sum:    13
1*16 = 16.  Add 16 to sum: 29
```

Binary to Decimal: Converting Between Bases

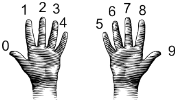


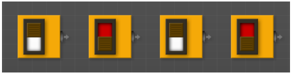


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

```
Sum starts with:            1
0*2 = 0.  Add 0 to sum:     1
1*4 = 4.  Add 4 to sum:     5
1*8 = 8.  Add 8 to sum:     13
1*16 = 16.  Add 16 to sum:  29
1*32 = 32.  Add 32 to sum:
```

Binary to Decimal: Converting Between Bases

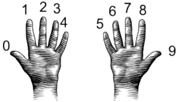


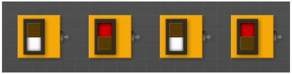


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

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

Binary to Decimal: Converting Between Bases

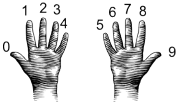


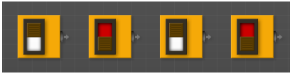


- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by $2 = 2^1$. Add to sum.
  - Multiply next digit by $4 = 2^2$. Add to sum.
  - Multiply next digit by $8 = 2^3$. Add to sum.
  - Multiply next digit by $16 = 2^4$. Add to sum.
  - Multiply next digit by $32 = 2^5$. Add to sum.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

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

Binary to Decimal: Converting Between Bases

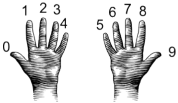


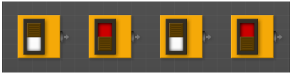


- Example: What is 10100100 in decimal?

  `Sum starts with:`

Binary to Decimal: Converting Between Bases

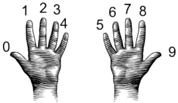


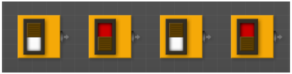


- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases

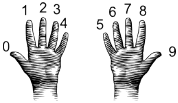


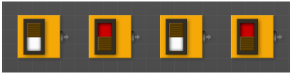


- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases

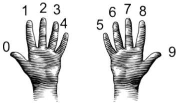


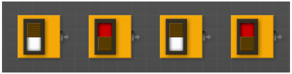


- Example: What is 10100100 in decimal?

```
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

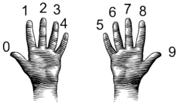


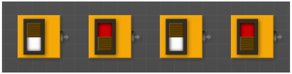


- Example: What is 10100100 in decimal?

```
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

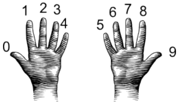


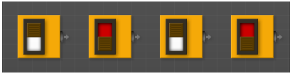


- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases

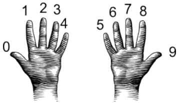


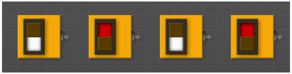


- Example: What is 10100100 in decimal?

```
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

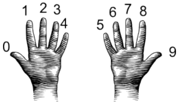


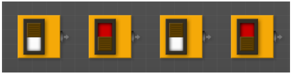


- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases

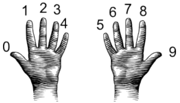


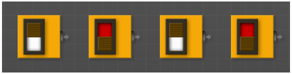


- Example: What is 10100100 in decimal?

```
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
0*16 = 0.   Add 0 to sum:    4
```

Binary to Decimal: Converting Between Bases

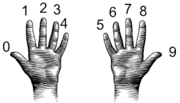


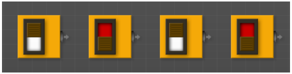


- Example: What is 10100100 in decimal?

```
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
0*16 = 0.   Add 0 to sum:     4
1*32 = 32.   Add 32 to sum:
```

Binary to Decimal: Converting Between Bases

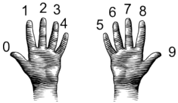


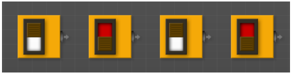


- Example: What is 10100100 in decimal?

```
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
0*16 = 0.   Add 0 to sum:  4
1*32 = 32.  Add 32 to sum: 36
```

Binary to Decimal: Converting Between Bases

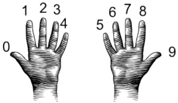


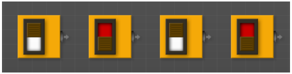


- Example: What is 10100100 in decimal?

```
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
0*16 = 0.  Add 0 to sum:   4
1*32 = 32.  Add 32 to sum: 36
0*64 = 0.  Add 0 to sum:
```

Binary to Decimal: Converting Between Bases

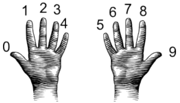


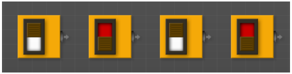


- Example: What is 10100100 in decimal?

```
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
0*16 = 0.   Add 0 to sum:     4
1*32 = 32.   Add 32 to sum:   36
0*64 = 0.   Add 0 to sum:     36
```

Binary to Decimal: Converting Between Bases

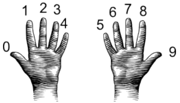


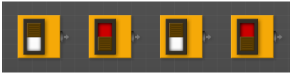


- Example: What is 10100100 in decimal?

```
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
0*16 = 0.   Add 0 to sum:     4
1*32 = 32.   Add 32 to sum:   36
0*64 = 0.   Add 0 to sum:     36
1*128 = 0.   Add 128 to sum:
```

Binary to Decimal: Converting Between Bases

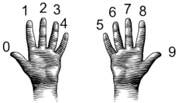


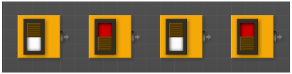


- Example: What is 10100100 in decimal?

```
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
0*16 = 0.  Add 0 to sum:    4
1*32 = 32.  Add 32 to sum:  36
0*64 = 0.  Add 0 to sum:    36
1*128 = 0.  Add 128 to sum: 164
```

Binary to Decimal: Converting Between Bases



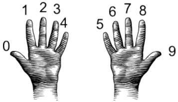


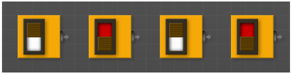
- Example: What is 10100100 in decimal?

```
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
0*16 = 0.   Add 0 to sum:     4
1*32 = 32.   Add 32 to sum:   36
0*64 = 0.   Add 0 to sum:     36
1*128 = 0.   Add 128 to sum:  164
```

The answer is 164.

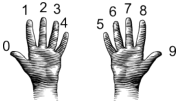
Design Challenge: Incrementers

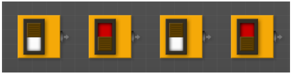




- Simplest arithmetic: add one ("increment") a variable.

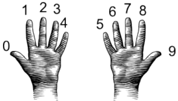
Design Challenge: Incrementers

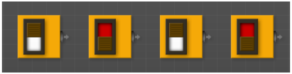




- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

Design Challenge: Incrementers

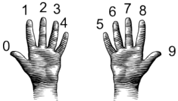


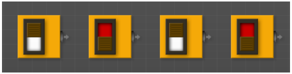


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

Design Challenge: Incrementers



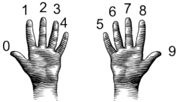


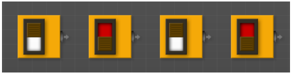
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

Design Challenge: Incrementers



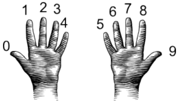


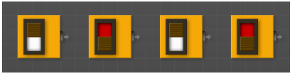
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



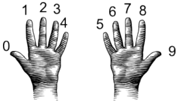


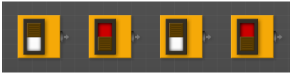
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



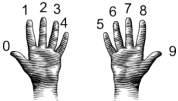


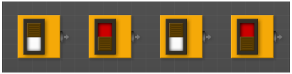
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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





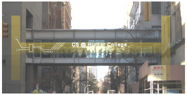
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

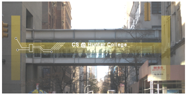
- Challenge: Write an algorithm for incrementing numbers expressed as words.
  Example: `"forty one"` → `"forty two"`
  Hint: Convert to numbers, increment, and convert back to strings.
- Challenge: Write an algorithm for incrementing binary numbers.
  Example: `"1001"` → `"1010"`

Recap

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

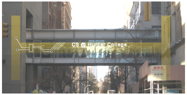


Recap



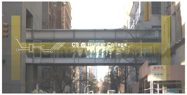
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Searching through data is a common task– built-in functions and standard design patterns for this.

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

19 December 2018

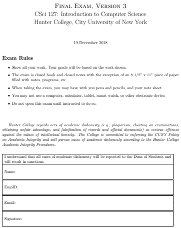
Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmpID:
Email:
Signature:

Exam Options



Final Exam, Version 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

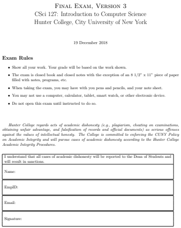
Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmpID:
Email:
Signature:

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 Options


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

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

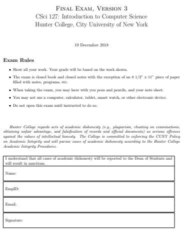
I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmpID:
Email:
Signature:

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:

Exam Options


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

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmplID:
Email:
Signature:

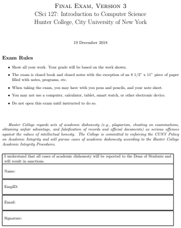
Exam Times:

- 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:

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

Exam Options


Final Exam, Version 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmplID:
Email:
Signature:

Exam Times:

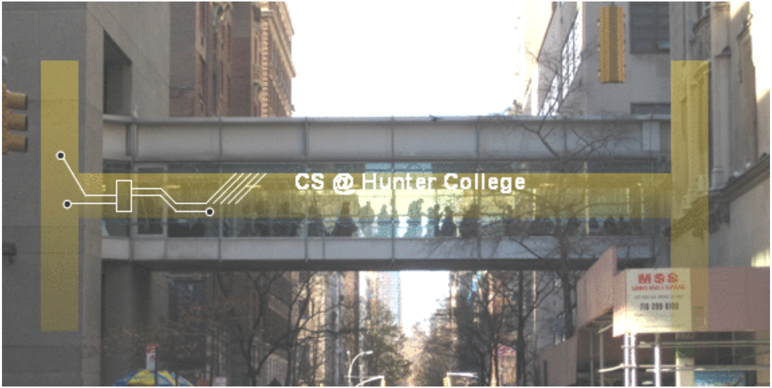
- Default: Regular Time: Monday, 16 December, 9-11am.
- Alternate Time: Reading Day, Friday, 13 December, 8:30am-10:30am.
- Accessibility Testing Center: Paperwork required. Must be completed on 13 December.

Grading Options:

- 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...