

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Lecture Slip: tinyurl.com/yb8lcvl7

Exit Slip for Lecture 11

1

Of the programs 16 through 30, which did you enjoy the most?

- 16. Days Until Final
- 17. 5 Number Loop
- 18. Turtle String
- 19. Even/Odd Turtle
- 20. Flood Map
- 21. California Snow Pack
- 22. MetroCard Prices
- 23. Majority Circuit
- 24. NAND Circuit
- 25. Two Digit Numbers
- 26. Cropping Images
- 27. Population Fractions
- 28. Population Stats
- 29. Directory Shell Script
- 30. Always True Circuit
- Other: _____

Why?

Your answer

BACK

NEXT

Page 4 of 5

Never submit passwords through Google Forms.

Exit Slip for Lecture 11

2

Of the programs 16 through 30, on which did you spend the most time?

- 16. Days Until Final
- 17. 5 Number Loop
- 18. Turtle String
- 19. Even/Odd Turtle
- 20. Flood Map
- 21. California Snow Pack
- 22. MetroCard Prices
- 23. Majority Circuit
- 24. NAND Circuit
- 25. Two Digit Numbers
- 26. Cropping Images
- 27. Population Fractions
- 28. Population Stats
- 29. Directory Shell Script
- 30. Always True Circuit
- Other: _____

Why?

Your answer

BACK

SUBMIT

Page 5 of 5

Never submit passwords through Google Forms.

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

*It was difficult to lecture with folks walking in front of screen, and
it was difficult for tutoring during quiz & lecture.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and it was difficult for tutoring during quiz & lecture.

- Why do I need to sign in when dropping in?

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and it was difficult for tutoring during quiz & lecture.

- Why do I need to sign in when dropping in?

Our current space is temporary.

We're collecting data to bolster our case for more space (& more tutors)!

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and it was difficult for tutoring during quiz & lecture.

- Why do I need to sign in when dropping in?

Our current space is temporary.

We're collecting data to bolster our case for more space (& more tutors)!

- Why can't in-class quizzes taken at midnight count towards my grade?

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and it was difficult for tutoring during quiz & lecture.

- Why do I need to sign in when dropping in?

Our current space is temporary.

We're collecting data to bolster our case for more space (& more tutors)!

- Why can't in-class quizzes taken at midnight count towards my grade?

The in-class quizzes are in place of midterm exams.

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and it was difficult for tutoring during quiz & lecture.

- Why do I need to sign in when dropping in?

Our current space is temporary.

We're collecting data to bolster our case for more space (& more tutors)!

- Why can't in-class quizzes taken at midnight count towards my grade?

The in-class quizzes are in place of midterm exams.

We want to know that you (and not friend/family) took the in-class quiz.

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and it was difficult for tutoring during quiz & lecture.

- Why do I need to sign in when dropping in?

Our current space is temporary.

We're collecting data to bolster our case for more space (& more tutors)!

- Why can't in-class quizzes taken at midnight count towards my grade?

The in-class quizzes are in place of midterm exams.

We want to know that you (and not friend/family) took the in-class quiz.

- What's my grade in the class?

Frequently Asked Questions

From lecture slips & recitation sections.

- Why did you reorganize the lab, 1001E?

Lab serves dual purposes: recitation and drop-in tutoring.

Having the teachers' desk in front of the door was bad for both:

It was difficult to lecture with folks walking in front of screen, and it was difficult for tutoring during quiz & lecture.

- Why do I need to sign in when dropping in?

Our current space is temporary.

We're collecting data to bolster our case for more space (& more tutors)!

- Why can't in-class quizzes taken at midnight count towards my grade?

The in-class quizzes are in place of midterm exams.

We want to know that you (and not friend/family) took the in-class quiz.

- What's my grade in the class?

*Easy estimate: 30% Programs (drop 5 lowest), 30% Quizzes (drop 2 lowest)**

Of first 40: missed 8 or less, have > 90%

For in-class quizzes: drop 2 (or replace 5 with final grade).

Today's Topics



- Recap of Python & Circuits
- High vs. Low-Level Programming
- A Simplified Machine Language
- Final Exam Overview

Python & Circuits Review: 10 Weeks in 10 Minutes



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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

- As well as definite loops & the turtle package:

The screenshot shows a Python code editor interface. The left pane displays the code file `main.py` with the following content:

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

The right pane has two tabs: `Result` and `Instructions`. The `Result` tab shows the output of the program, which is a regular hexagon drawn in purple. Each vertex of the hexagon has a small purple star-like stamp.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

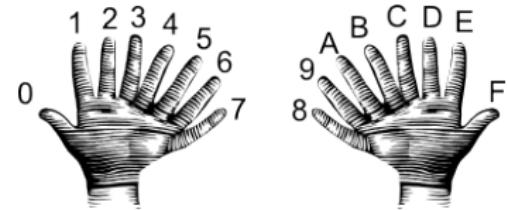
- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items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

- More on loops & ranges:

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

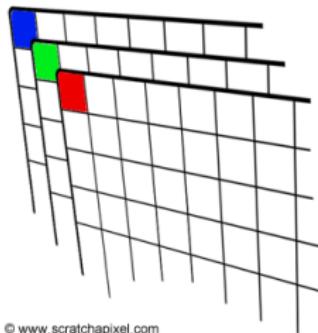
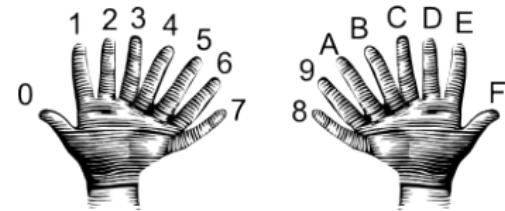
Week 3: colors, hex, slices, numpy & images

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



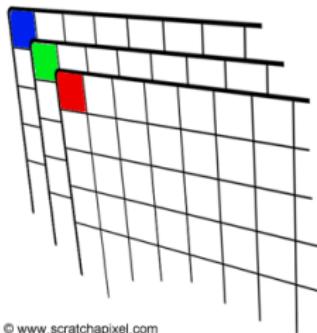
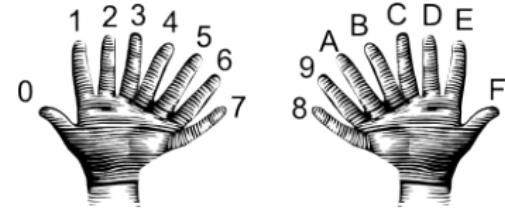
Week 3: colors, hex, slices, numpy & images

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



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

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



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

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

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

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

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

Week 4: design problem (cropping images) & decisions



Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions

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

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

Week 5: logical operators, truth tables & logical circuits

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

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

Week 5: logical operators, truth tables & logical circuits

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

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

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



Week 6: structured data, pandas, & more design

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

.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1690,1,037,203,727,784,2,037
1771,21843,36231,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,67541,5544,6241,1755,4543,75934
1820,123704,11487,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,18013,14081,5346,10965,391114
1850,35541,21801,18891,5346,10965,391115
1860,813469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33021,1911801
1890,1367711,70001,6544,58160,33021,1911814
1900,185093,116582,152999,200567,67921,2437202
1910,223342,1634351,28491,430980,8569,4766803
1920,2241103,2018354,44601,44601,73201,11651,50048
1930,1867128,1797128,1797128,1797128,1797128,4930446
1940,1889924,2498285,1297634,1394711,374441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1690311,2319319,1899319,1481277,191555,7981984
1970,1539231,2465701,1471701,135443,7981984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,379977,7322564
2000,1537195,2485326,2229379,1332650,419782,8080879
2010,1583873,2504705,2216722,1385108,4175133,8175133
2015,1444018,2646733,2339150,1459446,474558,8059405

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
All population figures are consistent with present-day boundaries.....
Five census after the consolidation of the five boroughs.....
.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1890,4937,2017,...,727,7881
1871,21843,36231,...,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,63541,5544,6240,1755,4543,75934
1820,123704,11187,8246,2792,4135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,18013,14049,5346,10965,391114
1850,35549,12891,12891,12891,12891,12891
1860,813469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33029,1911801
1890,1367111,66582,6354,58160,31861,21861
1900,185093,116582,152999,200567,67621,2437202
1910,2233142,1634351,2841,430980,8569,4766803
1920,22161103,2018354,4460,4460,73201,11651,55048
1930,1867111,66582,152999,200567,67621,2437202
1940,1889924,2498285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1690101,2319319,1809049,1451277,191555,7981984
1970,1539231,2460711,1471701,1471701,135443,7981984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,379877,7322564
2000,1537195,2485326,2229379,1332450,419782,8080879
2010,1583873,2340705,2216722,1385108,419782,8175133
2015,1444018,2046733,2339150,1459446,474558,8059405

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2037,727,788,102  
1771,21843,3623,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67530,6240,6840,2000,4543,85934  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3032,7082,242278  
1840,31150,10,113,1403,5346,10965,391114  
1850,35549,12850,1885,1085,5346,10965,411115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911801  
1890,1380000,710000,65000,58000,33029,2347134  
1900,185093,116582,152999,200567,67621,2437202  
1910,223342,1634351,2841,430980,8569,476683  
1920,22461103,2018354,446070,720201,116582,591048  
1930,26671103,2079128,446070,720201,116582,6930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7891957  
1960,1690101,2738175,1550949,1451277,191555,7891984  
1970,1639231,2465705,1471705,1271705,135443,7891984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,379877,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2271722,1385108,474558,8155405  
2015,1444018,2646733,2339150,1459446,474558,8056405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Population  
1690,203,2037,...,727,7181  
1771,21843,26232,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,72000,62000,68000,18000,49734  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31510,19013,14000,5348,10965,391114  
1850,35549,21890,18890,5348,10965,391115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1384093,72000,68000,51980,33091,1911804  
1900,1850993,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22461103,2018354,44607,72021,116582,593048  
1930,26671128,2079128,17254,593048,593048,4930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690211,2738175,1550949,1451277,191555,7991957  
1970,1639231,2465701,1472701,1235443,7981984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2227722,1385108,474558,8155405  
2015,1444518,2646733,2339150,1459446,474558,8155405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

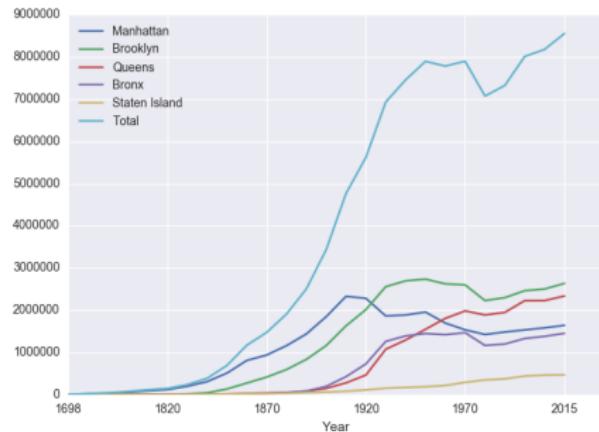
```
import matplotlib.pyplot as plt  
import pandas as pd
```

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Borough,Population  
1698,Manhattan,223,703  
1711,Brooklyn,238,432  
1731,Bronx,284,728  
1742,Queens,1,037,727  
1751,Bronx,1,181,718  
1760,Manhattan,3,131,454  
1770,Brooklyn,3,281,615  
1780,Bronx,3,459,615  
1790,Bronx,3,615,178  
1800,Bronx,3,827,494  
1800,Manhattan,6,015,574  
1800,Brooklyn,6,642,175  
1810,Bronx,4,543,795  
1810,Brooklyn,7,500,134  
1820,Bronx,5,975,34  
1820,Brooklyn,12,370,612  
1820,Manhattan,12,370,612  
1830,Bronx,12,370,612  
1830,Brooklyn,12,370,612  
1840,Bronx,12,370,612  
1840,Brooklyn,12,370,612  
1850,Bronx,12,370,612  
1850,Brooklyn,12,370,612  
1860,Bronx,12,370,612  
1860,Brooklyn,12,370,612  
1870,Bronx,12,370,612  
1870,Brooklyn,12,370,612  
1880,Bronx,12,370,612  
1880,Brooklyn,12,370,612  
1890,Bronx,12,370,612  
1890,Brooklyn,12,370,612  
1900,Bronx,12,370,612  
1900,Brooklyn,12,370,612  
1910,Bronx,12,370,612  
1910,Brooklyn,12,370,612  
1920,Bronx,12,370,612  
1920,Brooklyn,12,370,612  
1930,Bronx,12,370,612  
1930,Brooklyn,12,370,612  
1940,Bronx,12,370,612  
1940,Brooklyn,12,370,612  
1950,Bronx,12,370,612  
1950,Brooklyn,12,370,612  
1960,Bronx,12,370,612  
1960,Brooklyn,12,370,612  
1970,Bronx,12,370,612  
1970,Brooklyn,12,370,612  
1980,Bronx,12,370,612  
1980,Brooklyn,12,370,612  
1990,Bronx,12,370,612  
1990,Brooklyn,12,370,612  
2000,Bronx,12,370,612  
2000,Brooklyn,12,370,612  
2010,Bronx,12,370,612  
2010,Brooklyn,12,370,612  
2015,Bronx,12,370,612  
2015,Brooklyn,12,370,612
```

nycHistPop.csv

In Lab 6



Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 8: function parameters, github

- Functions can have **input parameters**.

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

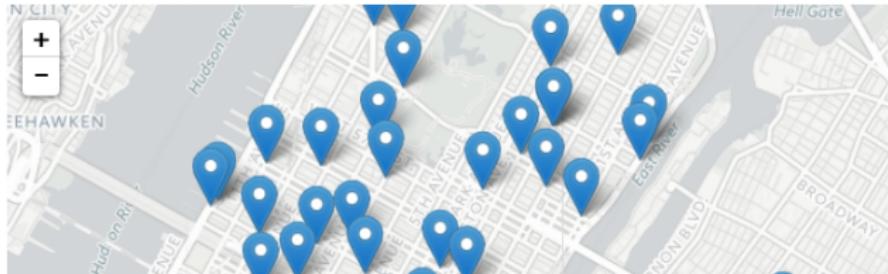
```
def totalWithTax(food, tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

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

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

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

Week 9: top-down design, folium



```
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: indefinite loops, searching data, 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: indefinite loops, searching data, 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: indefinite loops, searching data, random()

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

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

```
import turtle
import random

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

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

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

Week 10: indefinite loops, searching data, 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.`

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`

Python & Circuits Review: 10 Weeks in 10 Minutes



A whirlwind tour with
10 (or so) challenges...

In Pairs or Triples: Week 1

Predict what the code will do:

```
1 #Predict what will be printed:  
2  
3 for i in range(4):  
4     print('The world turned upside down')  
5  
6 for j in [0,1,2,3,4,5]:  
7     print(j)  
8  
9 for count in range(6):  
10    print(count)  
11  
12 for color in ['red', 'green', 'blue']:  
13    print(color)  
14  
15 print()  
16 print()  
17  
18 for i in range(2):  
19     for j in range(2):  
20         print('Look around,')  
21     print('How lucky we are to be alive!')
```

In Pairs or Triples: Week 2

Predict what the code will do:

```
1 #Predict what will be printed:  
2  
3 for c in range(65,90):  
4     print(chr(c))  
5  
6 message = "I love Python"  
7 newMessage = ""  
8 for c in message:  
9     print(ord(c))    #Print the Unicode of each number  
10    print(chr(ord(c)+1))    #Print the next character  
11    newMessage = newMessage + chr(ord(c)+1) #add to the new message  
12 print("The coded message is", newMessage)  
13  
14 word = "zebra"  
15 codedWord = ""  
16 for ch in word:  
17     offset = ord(ch) - ord('a') + 1 #how many letters past 'a'  
18     wrap = offset % 26    #if larger than 26, wrap back to 0  
19     newChar = chr(ord('a') + wrap)    #compute the new letter  
20     print(wrap, chr(ord('a') + wrap))    #print the wrap & new letter  
21     codedWord = codedWord + newChar #add the newChar to the coded word  
22  
23 print("The coded word (with wrap) is", codedWord)
```

Decimal	Hex	Char	Decimal	Hex	Char
64	40	@	96	60	`
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
82	52	R	114	72	r
83	53	S	115	73	s
84	54	T	116	74	t
85	55	U	117	75	u
86	56	V	118	76	v
87	57	W	119	77	w
88	58	X	120	78	x
89	59	Y	121	79	y
90	5A	Z	122	7A	z
91	5B	\	123	7B	{
92	5C	/	124	7C	
93	5D	I	125	7D	}
94	5E	^	126	7E	~
95	5F	DEL	127	7F	/DEL/

In Pairs or Triples: Week 3

Predict what the code will do:

```
1 import turtle
2 teddy = turtle.Turtle()
3
4 names = ["violet", "purple", "indigo", "lavender"]
5 for c in names:
6     teddy.color(c)
7     teddy.left(60)
8     teddy.forward(40)
9     teddy.dot(10)
10
11 teddy.penup()
12 teddy.forward(100)
13 teddy.pendown()
14
15 hexNames = ["#FF00FF", "#990099", "#550055", "#111111"]
16 for c in hexNames:
17     teddy.color(c)
18     teddy.left(60)
19     teddy.forward(40)
20     teddy.dot(10)
```

In Pairs or Triples: Week 4

Extend this program to also allow drawing in purple & stamping:

```
import turtle

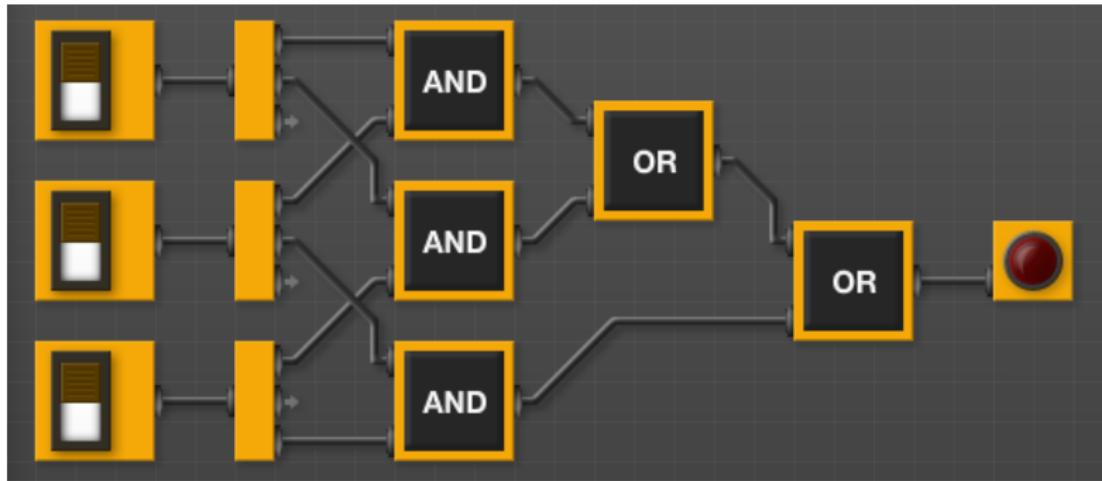
tess = turtle.Turtle()
myWin = turtle.Screen()      #The graphics window
commands = input("Please enter a command string: ")

for ch in commands:
    #perform action indicated by the character
    if ch == 'F':          #move forward
        tess.forward(50)
    elif ch == 'L':         #turn left
        tess.left(90)
    elif ch == 'R':         #turn right
        tess.right(90)
    elif ch == '^':         #lift pen
        tess.penup()
    elif ch == 'v':          #lower pen
        tess.pendown()
    elif ch == 'B':          #go backwards
        tess.backward(50)
    elif ch == 'r':          #turn red
        tess.color("red")
    elif ch == 'g':          #turn green
        tess.color("green")
    elif ch == 'b':          #turn blue
        tess.color("blue")
    else:                  #for any other character
        print("Error: do not know the command:", c)
```

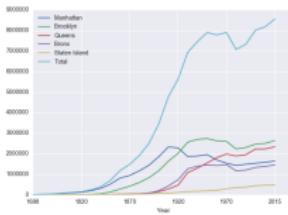
In Pairs or Triples: Week 5

When does this circuit yield true?

That is, what values for the inputs give an output value of true?



In Pairs or Triples: Week 6



Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`
- `print("S I:", pop["Staten Island"].mean())`
- `print("S I:", pop["Staten Island"].std())`
- `pop.plot.bar(x="Year")`
- `pop.plot.scatter(x="Brooklyn", y= "Total")`
- `pop["Fraction"] = pop["Bronx"]/pop["Total"]`

In Pairs or Triples: Week 7

Fill in the function body:

```
def monthString(monthNum):
    """
        Takes as input a number, monthNum, and
        returns the corresponding month name as a string.
        Example: monthString(1) returns "January".
        Assumes that input is an integer ranging from 1 to 12
    """

    monthString = ""

    #####
    ### FILL IN YOUR CODE HERE      ###
    ### Other than your name above, ###
    ### this is the only section   ###
    ### you change in this program. ###
    #####

    return(monthString)

def main():
    n = int(input('Enter the number of the month: '))
    mString = monthString(n)
    print('The month is', mString)
```

In Pairs or Triples: Week 8

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

- What are the formal parameters for the functions?
- What is the output of:

```
r = foo([1,2,3,4])
print("Return: ", r)
```

- What is the output of:

```
r = foo([1024,512,256,128])
print("Return: ", r)
```

In Pairs or Triples: Week 9

What does this code do?

```
import folium
import pandas as pd

cuny = pd.read_csv('cunyLocations.csv')
mapCUNY = folium.Map(location=[40.75, -74.125])

for index, row in cuny.iterrows():
    lat = row["Latitude"]
    lon = row["Longitude"]
    name = row["Campus"]
    if row["College or Institution Type"] == "Senior Colleges":
        collegeIcon = folium.Icon(color="purple")
    else:
        collegeIcon = folium.Icon(color="blue")
    newMarker = folium.Marker([lat, lon], popup=name, icon=collegeIcon)
    newMarker.add_to(mapCUNY)

mapCUNY.save(outfile='cunyLocationsSenior.html')
```

In Pairs or Triples: Week 10

- Predict what the code will do:

```
nums = [1, 4, 10, 6, 5, 42, 9, 8, 12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

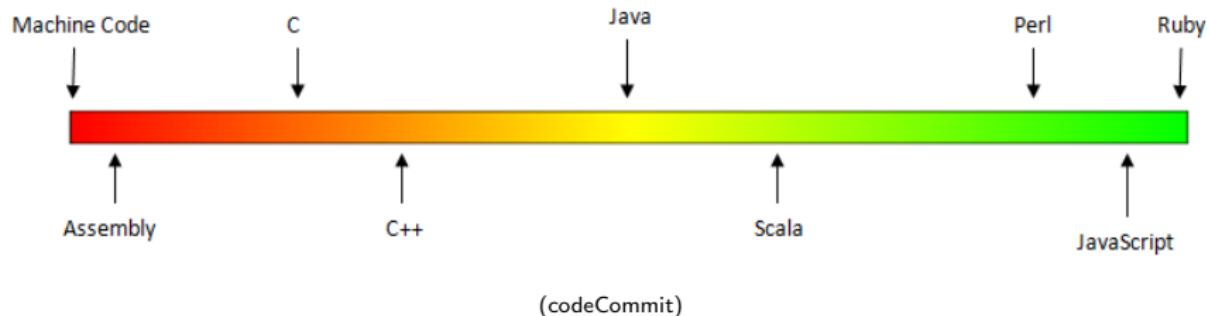
- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

Python & Circuits Review: 10 Weeks in 10 Minutes



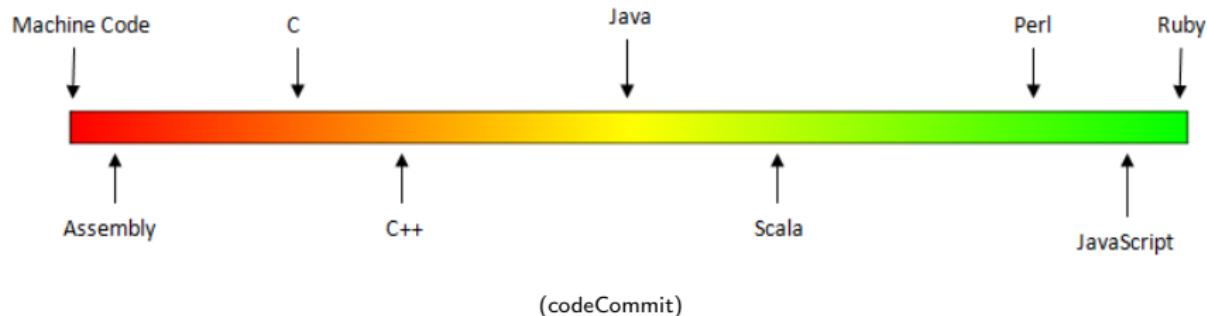
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
 - ▶ Primitive: `int`, `float`, `bool`, `string`;
 - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`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

High-Level vs. Low-Level Languages



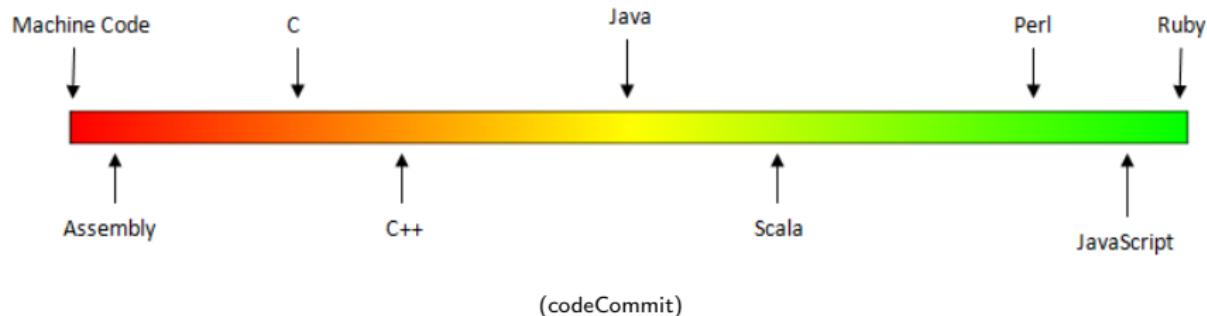
- Can view programming languages on a continuum.

High-Level vs. Low-Level Languages



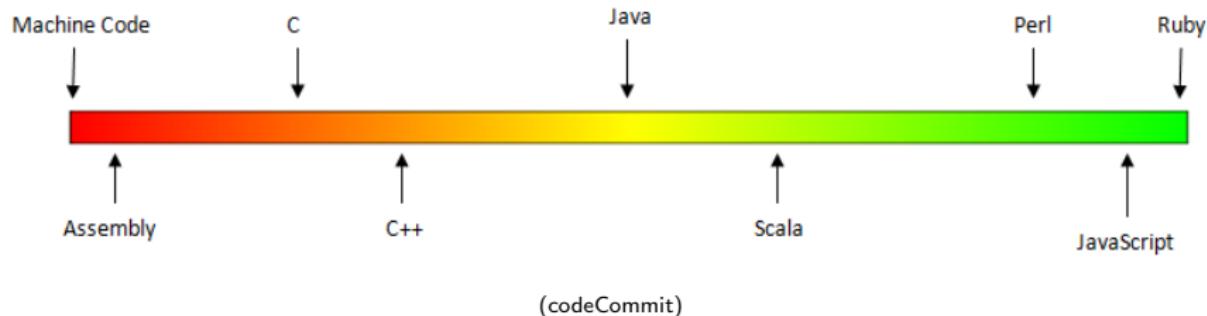
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

High-Level vs. Low-Level Languages



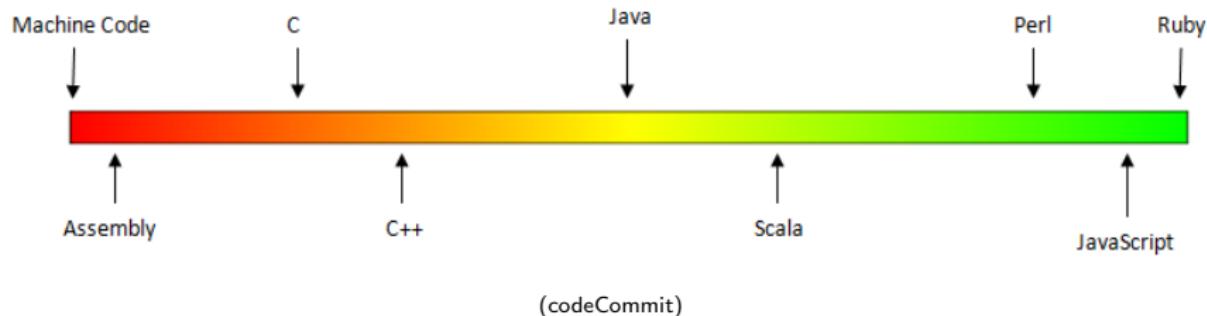
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

High-Level vs. Low-Level Languages



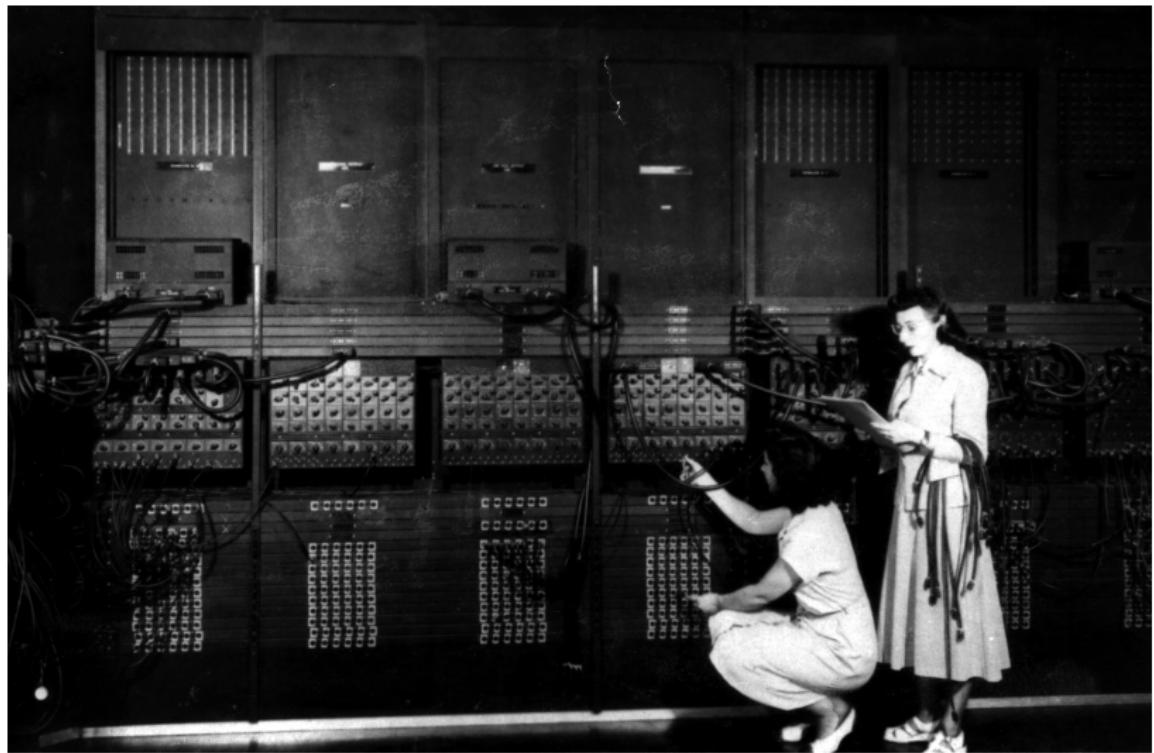
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine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

High-Level vs. Low-Level Languages



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

Machine Language



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

Machine Language

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

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

BREAK

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

(wiki)

Machine Language

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

```

A 002000 C2 3B REP #3B
A 002002 00 CLC
A 002004 F3 SED
A 002006 00 STP
A 002008 34 JZ LD #1224
A 00200E 60 21 45 ADC #1231
A 00200F 00 01 7F LD #17783
A 002010 D6 CLD
A 002011 E2 3B SEP #3B
A 002011 90 BYK
A 2012

```

(wiki)

Machine Language

(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.
 - It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.

Machine Language

```

A 002000 C2 3B REP #3B
A 002002 7F CLC
A 002003 FB SED
A 002004 34 32 #1234
A 002007 69 21 43 ADC #43
A 002008 8F B3 7F B1 STA #877F83
A 00200E D9 CLD
A 00200F E2 3B SEP #3B
A 002011 90 BHK
A 002012

```

F

```

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

BREAK

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

#10000000 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

(wiki)

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

Machine Language



The screenshot shows a computer interface for a MIPS-like processor. At the top, there's a status bar with memory addresses (0x00000000, 0x00000000, 0x00000000, 0x00000000) and other information. Below the status bar is a text area containing assembly code:

```
    C2 30    REP $#30
    00000002 1B    CLD
    00000003 FB    SEI
    00000004 09 34 12    LDA #1234
    00000007 69 21 43    LDH #4321
    00000008 8F 03 7F 01    STW #017F03
    00000009 E9 30    CLD
    0000000A 00 00    SEI #30
    00000011 00 00    SWI
    00000012 00 00    BREAK

PC PC      MU#00000000 A X Y SP DP R0
: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
& 20000000

BREAK

PC PC      MU#00000000 A X Y SP DP R0
: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
& 20000000
```

Below the assembly code, there are two rows of registers labeled 'PC' and 'R0'.

(wiki)

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

"Hello World!" in Simplified Machine Language

Line: 3 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

Step	Run	<input checked="" type="checkbox"/> Enable auto switching			
S	T	A	V	Stack	Log
s0:				10	
s1:				9	
s2:				9	
s3:				22	
s4:				696	
s5:				976	
s6:				927	
s7:				418	

(WeMIPS)



WeMIPS

The screenshot shows the WeMIPS IDE interface. At the top, there are tabs for 'Run', 'Step', and 'Break'. Below the tabs, there are several buttons: 'Addition Doubler', 'Stax', 'Looper', 'Stack Test', 'Hello World', 'Code Gen Save String', 'Interactive', 'Binary2 Decimal', 'Decimal2 Binary', and 'Debug'. The main area displays assembly code for printing 'Hello world!' to the screen. To the right of the assembly code is a debugger window with columns for Step, T, A, V, Stack, and Log. The Stack column shows memory addresses with their corresponding values.

```
# Store 'Hello world!' at the top of the stack
1    ADDI $t0, $zero, 72 # N
2    ADDI $t1, $zero, 101 # e
3    ADDI $t2, $zero, 101 # n
4    ADDI $t3, $zero, 101 # t
5    ADDI $t4, $zero, 101 # o
6    ADDI $t5, $zero, 101 # l
7    ADDI $t6, $zero, 101 # w
8    ADDI $t7, $zero, 101 # r
9    ADDI $t8, $zero, 101 # d
10   ADDI $t9, $zero, 101 # i
11   ADDI $t10, $zero, 101 # f
12   ADDI $t11, $zero, 101 # p
13   ADDI $t12, $zero, 101 # a
14   ADDI $t13, $zero, 101 # s
15   ADDI $t14, $zero, 101 # z
16   ADDI $t15, $zero, 101 # o
17   ADDI $t16, $zero, 101 # n
18   ADDI $t17, $zero, 101 # t
19   ADDI $t18, $zero, 101 # e
20   ADDI $t19, $zero, 101 # l
21   ADDI $t20, $zero, 101 # w
22   ADDI $t21, $zero, 101 # r
23   ADDI $t22, $zero, 101 # d
24   ADDI $t23, $zero, 101 # i
25   ADDI $t24, $zero, 101 # f
26   ADDI $t25, $zero, 101 # p
27   ADDI $t26, $zero, 101 # a
28   ADDI $t27, $zero, 101 # s
29   ADDI $t28, $zero, 101 # z
30   ADDI $t29, $zero, 101 # o
31   ADDI $t30, $zero, 101 # n
32   ADDI $t31, $zero, 101 # t
33   ADDI $t32, $zero, 101 # e
34   ADDI $t33, $zero, 101 # l
35   ADDI $t34, $zero, 101 # w
36   ADDI $t35, $zero, 101 # r
37   ADDI $t36, $zero, 101 # d
38   ADDI $t37, $zero, 101 # i
39   ADDI $t38, $zero, 101 # f
40   ADDI $t39, $zero, 101 # p
41   ADDI $t40, $zero, 101 # a
42   ADDI $t41, $zero, 101 # s
43   ADDI $t42, $zero, 101 # z
44   ADDI $t43, $zero, 101 # o
45   ADDI $t44, $zero, 101 # n
46   ADDI $t45, $zero, 101 # t
47   ADDI $t46, $zero, 101 # e
48   ADDI $t47, $zero, 101 # l
49   ADDI $t48, $zero, 101 # w
50   ADDI $t49, $zero, 101 # r
51   ADDI $t50, $zero, 101 # d
52   ADDI $t51, $zero, 101 # i
53   ADDI $t52, $zero, 101 # f
54   ADDI $t53, $zero, 101 # p
55   ADDI $t54, $zero, 101 # a
56   ADDI $t55, $zero, 101 # s
57   ADDI $t56, $zero, 101 # z
58   ADDI $t57, $zero, 101 # o
59   ADDI $t58, $zero, 101 # n
60   ADDI $t59, $zero, 101 # t
61   ADDI $t60, $zero, 101 # e
62   ADDI $t61, $zero, 101 # l
63   ADDI $t62, $zero, 101 # w
64   ADDI $t63, $zero, 101 # r
65   ADDI $t64, $zero, 101 # d
66   ADDI $t65, $zero, 101 # i
67   ADDI $t66, $zero, 101 # f
68   ADDI $t67, $zero, 101 # p
69   ADDI $t68, $zero, 101 # a
70   ADDI $t69, $zero, 101 # s
71   ADDI $t70, $zero, 101 # z
72   ADDI $t71, $zero, 101 # o
73   ADDI $t72, $zero, 101 # n
74   ADDI $t73, $zero, 101 # t
75   ADDI $t74, $zero, 101 # e
76   ADDI $t75, $zero, 101 # l
77   ADDI $t76, $zero, 101 # w
78   ADDI $t77, $zero, 101 # r
79   ADDI $t78, $zero, 101 # d
80   ADDI $t79, $zero, 101 # i
81   ADDI $t80, $zero, 101 # f
82   ADDI $t81, $zero, 101 # p
83   ADDI $t82, $zero, 101 # a
84   ADDI $t83, $zero, 101 # s
85   ADDI $t84, $zero, 101 # z
86   ADDI $t85, $zero, 101 # o
87   ADDI $t86, $zero, 101 # n
88   ADDI $t87, $zero, 101 # t
89   ADDI $t88, $zero, 101 # e
90   ADDI $t89, $zero, 101 # l
91   ADDI $t90, $zero, 101 # w
92   ADDI $t91, $zero, 101 # r
93   ADDI $t92, $zero, 101 # d
94   ADDI $t93, $zero, 101 # i
95   ADDI $t94, $zero, 101 # f
96   ADDI $t95, $zero, 101 # p
97   ADDI $t96, $zero, 101 # a
98   ADDI $t97, $zero, 101 # s
99   ADDI $t98, $zero, 101 # z
100  ADDI $t99, $zero, 101 # o
101  ADDI $t100, $zero, 101 # n
102  ADDI $t101, $zero, 101 # t
103  ADDI $t102, $zero, 101 # e
104  ADDI $t103, $zero, 101 # l
105  ADDI $t104, $zero, 101 # w
106  ADDI $t105, $zero, 101 # r
107  ADDI $t106, $zero, 101 # d
108  ADDI $t107, $zero, 101 # i
109  ADDI $t108, $zero, 101 # f
110  ADDI $t109, $zero, 101 # p
111  ADDI $t110, $zero, 101 # a
112  ADDI $t111, $zero, 101 # s
113  ADDI $t112, $zero, 101 # z
114  ADDI $t113, $zero, 101 # o
115  ADDI $t114, $zero, 101 # n
116  ADDI $t115, $zero, 101 # t
117  ADDI $t116, $zero, 101 # e
118  ADDI $t117, $zero, 101 # l
119  ADDI $t118, $zero, 101 # w
120  ADDI $t119, $zero, 101 # r
121  ADDI $t120, $zero, 101 # d
122  ADDI $t121, $zero, 101 # i
123  ADDI $t122, $zero, 101 # f
124  ADDI $t123, $zero, 101 # p
125  ADDI $t124, $zero, 101 # a
126  ADDI $t125, $zero, 101 # s
127  ADDI $t126, $zero, 101 # z
128  ADDI $t127, $zero, 101 # o
129  ADDI $t128, $zero, 101 # n
130  ADDI $t129, $zero, 101 # t
131  ADDI $t130, $zero, 101 # e
132  ADDI $t131, $zero, 101 # l
133  ADDI $t132, $zero, 101 # w
134  ADDI $t133, $zero, 101 # r
135  ADDI $t134, $zero, 101 # d
136  ADDI $t135, $zero, 101 # i
137  ADDI $t136, $zero, 101 # f
138  ADDI $t137, $zero, 101 # p
139  ADDI $t138, $zero, 101 # a
140  ADDI $t139, $zero, 101 # s
141  ADDI $t140, $zero, 101 # z
142  ADDI $t141, $zero, 101 # o
143  ADDI $t142, $zero, 101 # n
144  ADDI $t143, $zero, 101 # t
145  ADDI $t144, $zero, 101 # e
146  ADDI $t145, $zero, 101 # l
147  ADDI $t146, $zero, 101 # w
148  ADDI $t147, $zero, 101 # r
149  ADDI $t148, $zero, 101 # d
150  ADDI $t149, $zero, 101 # i
151  ADDI $t150, $zero, 101 # f
152  ADDI $t151, $zero, 101 # p
153  ADDI $t152, $zero, 101 # a
154  ADDI $t153, $zero, 101 # s
155  ADDI $t154, $zero, 101 # z
156  ADDI $t155, $zero, 101 # o
157  ADDI $t156, $zero, 101 # n
158  ADDI $t157, $zero, 101 # t
159  ADDI $t158, $zero, 101 # e
160  ADDI $t159, $zero, 101 # l
161  ADDI $t160, $zero, 101 # w
162  ADDI $t161, $zero, 101 # r
163  ADDI $t162, $zero, 101 # d
164  ADDI $t163, $zero, 101 # i
165  ADDI $t164, $zero, 101 # f
166  ADDI $t165, $zero, 101 # p
167  ADDI $t166, $zero, 101 # a
168  ADDI $t167, $zero, 101 # s
169  ADDI $t168, $zero, 101 # z
170  ADDI $t169, $zero, 101 # o
171  ADDI $t170, $zero, 101 # n
172  ADDI $t171, $zero, 101 # t
173  ADDI $t172, $zero, 101 # e
174  ADDI $t173, $zero, 101 # l
175  ADDI $t174, $zero, 101 # w
176  ADDI $t175, $zero, 101 # r
177  ADDI $t176, $zero, 101 # d
178  ADDI $t177, $zero, 101 # i
179  ADDI $t178, $zero, 101 # f
180  ADDI $t179, $zero, 101 # p
181  ADDI $t180, $zero, 101 # a
182  ADDI $t181, $zero, 101 # s
183  ADDI $t182, $zero, 101 # z
184  ADDI $t183, $zero, 101 # o
185  ADDI $t184, $zero, 101 # n
186  ADDI $t185, $zero, 101 # t
187  ADDI $t186, $zero, 101 # e
188  ADDI $t187, $zero, 101 # l
189  ADDI $t188, $zero, 101 # w
190  ADDI $t189, $zero, 101 # r
191  ADDI $t190, $zero, 101 # d
192  ADDI $t191, $zero, 101 # i
193  ADDI $t192, $zero, 101 # f
194  ADDI $t193, $zero, 101 # p
195  ADDI $t194, $zero, 101 # a
196  ADDI $t195, $zero, 101 # s
197  ADDI $t196, $zero, 101 # z
198  ADDI $t197, $zero, 101 # o
199  ADDI $t198, $zero, 101 # n
200  ADDI $t199, $zero, 101 # t
201  ADDI $t200, $zero, 101 # e
202  ADDI $t201, $zero, 101 # l
203  ADDI $t202, $zero, 101 # w
204  ADDI $t203, $zero, 101 # r
205  ADDI $t204, $zero, 101 # d
206  ADDI $t205, $zero, 101 # i
207  ADDI $t206, $zero, 101 # f
208  ADDI $t207, $zero, 101 # p
209  ADDI $t208, $zero, 101 # a
210  ADDI $t209, $zero, 101 # s
211  ADDI $t210, $zero, 101 # z
212  ADDI $t211, $zero, 101 # o
213  ADDI $t212, $zero, 101 # n
214  ADDI $t213, $zero, 101 # t
215  ADDI $t214, $zero, 101 # e
216  ADDI $t215, $zero, 101 # l
217  ADDI $t216, $zero, 101 # w
218  ADDI $t217, $zero, 101 # r
219  ADDI $t218, $zero, 101 # d
220  ADDI $t219, $zero, 101 # i
221  ADDI $t220, $zero, 101 # f
222  ADDI $t221, $zero, 101 # p
223  ADDI $t222, $zero, 101 # a
224  ADDI $t223, $zero, 101 # s
225  ADDI $t224, $zero, 101 # z
226  ADDI $t225, $zero, 101 # o
227  ADDI $t226, $zero, 101 # n
228  ADDI $t227, $zero, 101 # t
229  ADDI $t228, $zero, 101 # e
230  ADDI $t229, $zero, 101 # l
231  ADDI $t230, $zero, 101 # w
232  ADDI $t231, $zero, 101 # r
233  ADDI $t232, $zero, 101 # d
234  ADDI $t233, $zero, 101 # i
235  ADDI $t234, $zero, 101 # f
236  ADDI $t235, $zero, 101 # p
237  ADDI $t236, $zero, 101 # a
238  ADDI $t237, $zero, 101 # s
239  ADDI $t238, $zero, 101 # z
240  ADDI $t239, $zero, 101 # o
241  ADDI $t240, $zero, 101 # n
242  ADDI $t241, $zero, 101 # t
243  ADDI $t242, $zero, 101 # e
244  ADDI $t243, $zero, 101 # l
245  ADDI $t244, $zero, 101 # w
246  ADDI $t245, $zero, 101 # r
247  ADDI $t246, $zero, 101 # d
248  ADDI $t247, $zero, 101 # i
249  ADDI $t248, $zero, 101 # f
250  ADDI $t249, $zero, 101 # p
251  ADDI $t250, $zero, 101 # a
252  ADDI $t251, $zero, 101 # s
253  ADDI $t252, $zero, 101 # z
254  ADDI $t253, $zero, 101 # o
255  ADDI $t254, $zero, 101 # n
256  ADDI $t255, $zero, 101 # t
257  ADDI $t256, $zero, 101 # e
258  ADDI $t257, $zero, 101 # l
259  ADDI $t258, $zero, 101 # w
260  ADDI $t259, $zero, 101 # r
261  ADDI $t260, $zero, 101 # d
262  ADDI $t261, $zero, 101 # i
263  ADDI $t262, $zero, 101 # f
264  ADDI $t263, $zero, 101 # p
265  ADDI $t264, $zero, 101 # a
266  ADDI $t265, $zero, 101 # s
267  ADDI $t266, $zero, 101 # z
268  ADDI $t267, $zero, 101 # o
269  ADDI $t268, $zero, 101 # n
270  ADDI $t269, $zero, 101 # t
271  ADDI $t270, $zero, 101 # e
272  ADDI $t271, $zero, 101 # l
273  ADDI $t272, $zero, 101 # w
274  ADDI $t273, $zero, 101 # r
275  ADDI $t274, $zero, 101 # d
276  ADDI $t275, $zero, 101 # i
277  ADDI $t276, $zero, 101 # f
278  ADDI $t277, $zero, 101 # p
279  ADDI $t278, $zero, 101 # a
280  ADDI $t279, $zero, 101 # s
281  ADDI $t280, $zero, 101 # z
282  ADDI $t281, $zero, 101 # o
283  ADDI $t282, $zero, 101 # n
284  ADDI $t283, $zero, 101 # t
285  ADDI $t284, $zero, 101 # e
286  ADDI $t285, $zero, 101 # l
287  ADDI $t286, $zero, 101 # w
288  ADDI $t287, $zero, 101 # r
289  ADDI $t288, $zero, 101 # d
290  ADDI $t289, $zero, 101 # i
291  ADDI $t290, $zero, 101 # f
292  ADDI $t291, $zero, 101 # p
293  ADDI $t292, $zero, 101 # a
294  ADDI $t293, $zero, 101 # s
295  ADDI $t294, $zero, 101 # z
296  ADDI $t295, $zero, 101 # o
297  ADDI $t296, $zero, 101 # n
298  ADDI $t297, $zero, 101 # t
299  ADDI $t298, $zero, 101 # e
300  ADDI $t299, $zero, 101 # l
301  ADDI $t300, $zero, 101 # w
302  ADDI $t301, $zero, 101 # r
303  ADDI $t302, $zero, 101 # d
304  ADDI $t303, $zero, 101 # i
305  ADDI $t304, $zero, 101 # f
306  ADDI $t305, $zero, 101 # p
307  ADDI $t306, $zero, 101 # a
308  ADDI $t307, $zero, 101 # s
309  ADDI $t308, $zero, 101 # z
310  ADDI $t309, $zero, 101 # o
311  ADDI $t310, $zero, 101 # n
312  ADDI $t311, $zero, 101 # t
313  ADDI $t312, $zero, 101 # e
314  ADDI $t313, $zero, 101 # l
315  ADDI $t314, $zero, 101 # w
316  ADDI $t315, $zero, 101 # r
317  ADDI $t316, $zero, 101 # d
318  ADDI $t317, $zero, 101 # i
319  ADDI $t318, $zero, 101 # f
320  ADDI $t319, $zero, 101 # p
321  ADDI $t320, $zero, 101 # a
322  ADDI $t321, $zero, 101 # s
323  ADDI $t322, $zero, 101 # z
324  ADDI $t323, $zero, 101 # o
325  ADDI $t324, $zero, 101 # n
326  ADDI $t325, $zero, 101 # t
327  ADDI $t326, $zero, 101 # e
328  ADDI $t327, $zero, 101 # l
329  ADDI $t328, $zero, 101 # w
330  ADDI $t329, $zero, 101 # r
331  ADDI $t330, $zero, 101 # d
332  ADDI $t331, $zero, 101 # i
333  ADDI $t332, $zero, 101 # f
334  ADDI $t333, $zero, 101 # p
335  ADDI $t334, $zero, 101 # a
336  ADDI $t335, $zero, 101 # s
337  ADDI $t336, $zero, 101 # z
338  ADDI $t337, $zero, 101 # o
339  ADDI $t338, $zero, 101 # n
340  ADDI $t339, $zero, 101 # t
341  ADDI $t340, $zero, 101 # e
342  ADDI $t341, $zero, 101 # l
343  ADDI $t342, $zero, 101 # w
344  ADDI $t343, $zero, 101 # r
345  ADDI $t344, $zero, 101 # d
346  ADDI $t345, $zero, 101 # i
347  ADDI $t346, $zero, 101 # f
348  ADDI $t347, $zero, 101 # p
349  ADDI $t348, $zero, 101 # a
350  ADDI $t349, $zero, 101 # s
351  ADDI $t350, $zero, 101 # z
352  ADDI $t351, $zero, 101 # o
353  ADDI $t352, $zero, 101 # n
354  ADDI $t353, $zero, 101 # t
355  ADDI $t354, $zero, 101 # e
356  ADDI $t355, $zero, 101 # l
357  ADDI $t356, $zero, 101 # w
358  ADDI $t357, $zero, 101 # r
359  ADDI $t358, $zero, 101 # d
360  ADDI $t359, $zero, 101 # i
361  ADDI $t360, $zero, 101 # f
362  ADDI $t363, $zero, 101 # p
363  ADDI $t364, $zero, 101 # a
364  ADDI $t365, $zero, 101 # s
365  ADDI $t366, $zero, 101 # z
366  ADDI $t367, $zero, 101 # o
367  ADDI $t368, $zero, 101 # n
368  ADDI $t369, $zero, 101 # t
369  ADDI $t370, $zero, 101 # e
370  ADDI $t371, $zero, 101 # l
371  ADDI $t372, $zero, 101 # w
372  ADDI $t373, $zero, 101 # r
373  ADDI $t374, $zero, 101 # d
374  ADDI $t375, $zero, 101 # i
375  ADDI $t376, $zero, 101 # f
376  ADDI $t377, $zero, 101 # p
377  ADDI $t378, $zero, 101 # a
378  ADDI $t379, $zero, 101 # s
379  ADDI $t380, $zero, 101 # z
380  ADDI $t381, $zero, 101 # o
381  ADDI $t382, $zero, 101 # n
382  ADDI $t383, $zero, 101 # t
383  ADDI $t384, $zero, 101 # e
384  ADDI $t385, $zero, 101 # l
385  ADDI $t386, $zero, 101 # w
386  ADDI $t387, $zero, 101 # r
387  ADDI $t388, $zero, 101 # d
388  ADDI $t389, $zero, 101 # i
389  ADDI $t390, $zero, 101 # f
390  ADDI $t391, $zero, 101 # p
391  ADDI $t392, $zero, 101 # a
392  ADDI $t393, $zero, 101 # s
393  ADDI $t394, $zero, 101 # z
394  ADDI $t395, $zero, 101 # o
395  ADDI $t396, $zero, 101 # n
396  ADDI $t397, $zero, 101 # t
397  ADDI $t398, $zero, 101 # e
398  ADDI $t399, $zero, 101 # l
399  ADDI $t400, $zero, 101 # w
400  ADDI $t401, $zero, 101 # r
401  ADDI $t402, $zero, 101 # d
402  ADDI $t403, $zero, 101 # i
403  ADDI $t404, $zero, 101 # f
404  ADDI $t405, $zero, 101 # p
405  ADDI $t406, $zero, 101 # a
406  ADDI $t407, $zero, 101 # s
407  ADDI $t408, $zero, 101 # z
408  ADDI $t409, $zero, 101 # o
409  ADDI $t410, $zero, 101 # n
410  ADDI $t411, $zero, 101 # t
411  ADDI $t412, $zero, 101 # e
412  ADDI $t413, $zero, 101 # l
413  ADDI $t414, $zero, 101 # w
414  ADDI $t415, $zero, 101 # r
415  ADDI $t416, $zero, 101 # d
416  ADDI $t417, $zero, 101 # i
417  ADDI $t418, $zero, 101 # f
418  ADDI $t419, $zero, 101 # p
419  ADDI $t420, $zero, 101 # a
420  ADDI $t421, $zero, 101 # s
421  ADDI $t422, $zero, 101 # z
422  ADDI $t423, $zero, 101 # o
423  ADDI $t424, $zero, 101 # n
424  ADDI $t425, $zero, 101 # t
425  ADDI $t426, $zero, 101 # e
426  ADDI $t427, $zero, 101 # l
427  ADDI $t428, $zero, 101 # w
428  ADDI $t429, $zero, 101 # r
429  ADDI $t430, $zero, 101 # d
430  ADDI $t431, $zero, 101 # i
431  ADDI $t432, $zero, 101 # f
432  ADDI $t433, $zero, 101 # p
433  ADDI $t434, $zero, 101 # a
434  ADDI $t435, $zero, 101 # s
435  ADDI $t436, $zero, 101 # z
436  ADDI $t437, $zero, 101 # o
437  ADDI $t438, $zero, 101 # n
438  ADDI $t439, $zero, 101 # t
439  ADDI $t440, $zero, 101 # e
440  ADDI $t441, $zero, 101 # l
441  ADDI $t442, $zero, 101 # w
442  ADDI $t443, $zero, 101 # r
443  ADDI $t444, $zero, 101 # d
444  ADDI $t445, $zero, 101 # i
445  ADDI $t446, $zero, 101 # f
446  ADDI $t447, $zero, 101 # p
447  ADDI $t448, $zero, 101 # a
448  ADDI $t449, $zero, 101 # s
449  ADDI $t450, $zero, 101 # z
450  ADDI $t451, $zero, 101 # o
451  ADDI $t452, $zero, 101 # n
452  ADDI $t453, $zero, 101 # t
453  ADDI $t454, $zero, 101 # e
454  ADDI $t455, $zero, 101 # l
455  ADDI $t456, $zero, 101 # w
456  ADDI $t457, $zero, 101 # r
457  ADDI $t458, $zero, 101 # d
458  ADDI $t459, $zero, 101 # i
459  ADDI $t460, $zero, 101 # f
460  ADDI $t461, $zero, 101 # p
461  ADDI $t462, $zero, 101 # a
462  ADDI $t463, $zero, 101 # s
463  ADDI $t464, $zero, 101 # z
464  ADDI $t465, $zero, 101 # o
465  ADDI $t466, $zero, 101 # n
466  ADDI $t467, $zero, 101 # t
467  ADDI $t468, $zero, 101 # e
468  ADDI $t469, $zero, 101 # l
469  ADDI $t470, $zero, 101 # w
470  ADDI $t471, $zero, 101 # r
471  ADDI $t472, $zero, 101 # d
472  ADDI $t473, $zero, 101 # i
473  ADDI $t474, $zero, 101 # f
474  ADDI $t475, $zero, 101 # p
475  ADDI $t476, $zero, 101 # a
476  ADDI $t477, $zero, 101 # s
477  ADDI $t478, $zero, 101 # z
478  ADDI $t479, $zero, 101 # o
479  ADDI $t480, $zero, 101 # n
480  ADDI $t481, $zero, 101 # t
481  ADDI $t482, $zero, 101 # e
482  ADDI $t483, $zero, 101 # l
483  ADDI $t484, $zero, 101 # w
484  ADDI $t485, $zero, 101 # r
485  ADDI $t486, $zero, 101 # d
486  ADDI $t487, $zero, 101 # i
487  ADDI $t488, $zero, 101 # f
488  ADDI $t489, $zero, 101 # p
489  ADDI $t490, $zero, 101 # a
490  ADDI $t491, $zero, 101 # s
491  ADDI $t492, $zero, 101 # z
492  ADDI $t493, $zero, 101 # o
493  ADDI $t494, $zero, 101 # n
494  ADDI $t495, $zero, 101 # t
495  ADDI $t496, $zero, 101 # e
496  ADDI $t497, $zero, 101 # l
497  ADDI $t498, $zero, 101 # w
498  ADDI $t499, $zero, 101 # r
499  ADDI $t500, $zero, 101 # d
500  ADDI $t501, $zero, 101 # i
501  ADDI $t502, $zero, 101 # f
502  ADDI $t503, $zero, 101 # p
503  ADDI $t504, $zero, 101 # a
504  ADDI $t505, $zero, 101 # s
505  ADDI $t506, $zero, 101 # z
506  ADDI $t507, $zero, 101 # o
507  ADDI $t508, $zero, 101 # n
508  ADDI $t509, $zero, 101 # t
509  ADDI $t510, $zero, 101 # e
510  ADDI $t511, $zero, 101 # l
511  ADDI $t512, $zero, 101 # w
512  ADDI $t513, $zero, 101 # r
513  ADDI $t514, $zero, 101 # d
514  ADDI $t515, $zero, 101 # i
515  ADDI $t516, $zero, 101 # f
516  ADDI $t517, $zero, 101 # p
517  ADDI $t518, $zero, 101 # a
518  ADDI $t519, $zero, 101 # s
519  ADDI $t520, $zero, 101 # z
520  ADDI $t521, $zero, 101 # o
521  ADDI $t522, $zero, 101 # n
522  ADDI $t523, $zero, 101 # t
523  ADDI $t524, $zero, 101 # e
524  ADDI $t525, $zero, 101 # l
525  ADDI $t526, $zero, 101 # w
526  ADDI $t527, $zero, 101 # r
527  ADDI $t528, $zero, 101 # d
528  ADDI $t529, $zero, 101 # i
529  ADDI $t530, $zero, 101 # f
530  ADDI $t531, $zero, 101 # p
531  ADDI $t532, $zero, 101 # a
532  ADDI $t533, $zero, 101 # s
533  ADDI $t534, $zero, 101 # z
534  ADDI $t535, $zero, 101 # o
535  ADDI $t536, $zero, 101 # n
536  ADDI $t537, $zero, 101 # t
537  ADDI $t538, $zero, 101 # e
538  ADDI $t539, $zero, 101 # l
539  ADDI $t540, $zero, 101 # w
540  ADDI $t541, $zero, 101 # r
541  ADDI $t542, $zero, 101 # d
542  ADDI $t543, $zero, 101 # i
543  ADDI $t544, $zero, 101 # f
544  ADDI $t545, $zero, 101 # p
545  ADDI $t546, $zero, 101 # a
546  ADDI $t547, $zero, 101 # s
547  ADDI $t548, $zero, 101 # z
548  ADDI $t549, $zero, 101 # o
549  ADDI $t550, $zero, 101 # n
550  ADDI $t551, $zero, 101 # t
551  ADDI $t552, $zero, 101 # e
552  ADDI $t553, $zero, 101 # l
553  ADDI $t554, $zero, 101 # w
554  ADDI $t555, $zero, 101 # r
555  ADDI $t556, $zero, 101 # d
556  ADDI $t557, $zero, 101 # i
557  ADDI $t558, $zero, 101 # f
558  ADDI $t559, $zero, 101 # p
559  ADDI $t560, $zero, 101 # a
560  ADDI $t561, $zero, 101 # s
561  ADDI $t562, $zero, 101 # z
562  ADDI $t563, $zero, 101 # o

```

In Pairs or Triples:

Line: 3 Go! Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

Step Run Enable auto switching

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

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

WeMIPS

```

Line 3 Out ShowWhile Demo User Guide | Unit Tests | Docs

Addition Doubler Stmt Looper Stack Test Hello World
Code Gen Save String Interactive Binary0 Decimal Decimal2 Binary

Debug Step Run | # Enable auto switching
S T A V Stack Log
#d 10
#e 8
#h 9
#b 22
#d 890
#b 976
#d 977
#f 419

# Prints "Hello world!" at the top of the stack
#EQU1 std::cout, 72 #'H
#EQU1 std::endl, 10 #'n
#EQU1 std::wcout, 100 #'\u000d
#EQU1 std::wendl, 101 #'\u000a
#EQU1 std::hex, 198 #'1
#EQU1 std::dec, 199 #'0
#EQU1 std::oct, 197 #'2
#EQU1 std::wcout, 111 #'o
#EQU1 std::wendl, 112 #'\u000a
#EQU1 std::wcout, 41 #'open parenthesis
#D 94s, 5(#expr)
#EQU1 std::wcout, 113 #'w
#EQU1 std::wcout, 114 #'w
#EQU1 std::wcout, 115 #'n
#EQU1 std::wcout, 71(#expr)
#EQU1 std::wcout, 116 #'c
#D 95s, 6(#expr)
#EQU1 std::wcout, 118 #'1
#EQU1 std::wcout, 119 #'0
#EQU1 std::wcout, 120 #'d
#EQU1 std::wcout, 101 #'1
#EQU1 std::wcout, 33 #'#
#EQU1 std::wcout, 34 #'-
#EQU1 std::wcout, 0 #'(null)
#D 96s, 121(#expr)

# Prints 100, hex 20, & 4 is for print string
#EQU1 std::wcout, 100, 4 # prints to the log
#EQU1 std::wendl

```

(Demo with WeMIPS)

Lecture Slip: tinyurl.com/yb8lcvl7

Exit Slip for Lecture 11

1

Of the programs 16 through 30, which did you enjoy the most?

- 16. Days Until Final
- 17. 5 Number Loop
- 18. Turtle String
- 19. Even/Odd Turtle
- 20. Flood Map
- 21. California Snow Pack
- 22. MetroCard Prices
- 23. Majority Circuit
- 24. NAND Circuit
- 25. Two Digit Numbers
- 26. Cropping Images
- 27. Population Fractions
- 28. Population Stats
- 29. Directory Shell Script
- 30. Always True Circuit
- Other: _____

Why?

Your answer

BACK

NEXT

Page 4 of 5

Never submit passwords through Google Forms.

Exit Slip for Lecture 11

2

Of the programs 16 through 30, on which did you spend the most time?

- 16. Days Until Final
- 17. 5 Number Loop
- 18. Turtle String
- 19. Even/Odd Turtle
- 20. Flood Map
- 21. California Snow Pack
- 22. MetroCard Prices
- 23. Majority Circuit
- 24. NAND Circuit
- 25. Two Digit Numbers
- 26. Cropping Images
- 27. Population Fractions
- 28. Population Stats
- 29. Directory Shell Script
- 30. Always True Circuit
- Other: _____

Why?

Your answer

BACK

SUBMIT

Page 5 of 5

Never submit passwords through Google Forms.

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.
 - ▶ 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.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
- Sample exam available on-line after Thanksgiving Break.

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.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
- Sample exam available on-line after Thanksgiving Break.
- Mock exam: last day of lecture and discussed in recitation.

Final Overview

For each question below, write **only the function header (name & inputs) and return values** (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.
- Write a function that takes a weight in kilograms and returns the weight in pounds.
- Write a function that, given a DataFrame, returns the minimal value in the first column.
- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.
- Write a function that takes a string and returns its length.

(Hint: highlight key words, make list of inputs, list of outputs, then put together.)

Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the first column.

Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

Recap: Python, Languages, & Design

```
#Name: your name here  
#Date: October 2017  
#This program, uses functions,  
#     says hello to the world!  
  
def main():  
    print("Hello, World!")  
  
if __name__ == "__main__":  
    main()
```

- Python language

Recap: Python, Languages, & Design

```
#Name: your name here  
#Date: October 2017  
#This program, uses functions,  
#     says hello to the world!  
  
def main():  
    print("Hello, World!")  
  
if __name__ == "__main__":  
    main()
```

- Python language
- Logical Circuits

Recap: Python, Languages, & Design

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

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

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

- Python language
- Logical Circuits
- Simplified Machine Language

Recap: Python, Languages, & Design

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

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

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

- ➊ Python language
- ➋ Logical Circuits
- ➌ Simplified Machine Language
- ➍ Design: from written description ('specs') to function inputs & outputs ('APIs')

Lecture Slip: tinyurl.com/yb8lcvl7

Exit Slip for Lecture 11

1

Of the programs 16 through 30, which did you enjoy the most?

- 16. Days Until Final
- 17. 5 Number Loop
- 18. Turtle String
- 19. Even/Odd Turtle
- 20. Flood Map
- 21. California Snow Pack
- 22. MetroCard Prices
- 23. Majority Circuit
- 24. NAND Circuit
- 25. Two Digit Numbers
- 26. Cropping Images
- 27. Population Fractions
- 28. Population Stats
- 29. Directory Shell Script
- 30. Always True Circuit
- Other: _____

Why?

Your answer

BACK

NEXT

Page 4 of 5

Never submit passwords through Google Forms.

Exit Slip for Lecture 11

2

Of the programs 16 through 30, on which did you spend the most time?

- 16. Days Until Final
- 17. 5 Number Loop
- 18. Turtle String
- 19. Even/Odd Turtle
- 20. Flood Map
- 21. California Snow Pack
- 22. MetroCard Prices
- 23. Majority Circuit
- 24. NAND Circuit
- 25. Two Digit Numbers
- 26. Cropping Images
- 27. Population Fractions
- 28. Population Stats
- 29. Directory Shell Script
- 30. Always True Circuit
- Other: _____

Why?

Your answer

BACK

SUBMIT

Page 5 of 5

Never submit passwords through Google Forms.