

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements

- Upcoming holidays:



Announcements



- Upcoming holidays:
 - ▶ Monday, 12 February: No classes (Lincoln's birthday observed)
 - ▶ Monday, 19 February: No classes (Presidents' Day)
 - ▶ CUNY follows Monday class schedule on Tuesday, 20 February.
 - ▶ Lecture resumes in two weeks.
- Starting this week, we will end each lecture with a survey of computing research and the tech industry in NYC.

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?

No worries. If you miss a lecture slip, your final exam grade replaces that score.

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?
These are also on the class website.

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?
These are also on the class website.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?
These are also on the class website.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
Some of the grading scripts are really finicky about spacing and new lines. Let us know— some we can fix, some have to match exactly.

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?
These are also on the class website.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
Some of the grading scripts are really finicky about spacing and new lines. Let us know— some we can fix, some have to match exactly.
- I'm confused by the reading— what should I be reading?

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?
These are also on the class website.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
Some of the grading scripts are really finicky about spacing and new lines. Let us know— some we can fix, some have to match exactly.
- I'm confused by the reading— what should I be reading?
For this week: Introductory chapters (1-4) and §8.10 on images.

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?
These are also on the class website.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
Some of the grading scripts are really finicky about spacing and new lines. Let us know— some we can fix, some have to match exactly.
- I'm confused by the reading— what should I be reading?
For this week: Introductory chapters (1-4) and §8.10 on images.
- What's the best way to study for the final? What should I read?

Frequently Asked Questions

From lecture slips & recitation sections.

- I didn't turn in a lecture slip last week. Did I miss it?
No worries. If you miss a lecture slip, your final exam grade replaces that score.
- Can I get a copy of the lecture slides?
Yes, the slides are posted on the class website.
- Where do I find the programs from lecture?
These are also on the class website.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
Some of the grading scripts are really finicky about spacing and new lines. Let us know— some we can fix, some have to match exactly.
- I'm confused by the reading— what should I be reading?
For this week: Introductory chapters (1-4) and §8.10 on images.
- What's the best way to study for the final? What should I read?
Do all the programming assignments & quizzes. Practice final exams will be available mid-March.

Today's Topics



- Indexing and Slicing Lists
- Colors
- Hexadecimal Notation
- 2D Arrays & Image Files

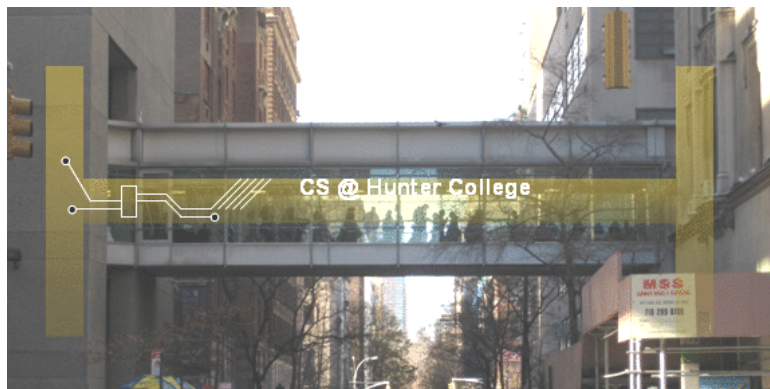
Today's Topics



- Indexing and Slicing Lists
- Colors
- Hexadecimal Notation
- 2D Arrays & Image Files

Due to special event, we're going to start with our CS Survey guest...

CS Surveys



- Survey of research at Hunter & tech industry in NYC...



Language is Hard for Computers

Learning Language is Easy for my 3-year-old twins

CSCI 12700 Guest Bullet Talk

William Gregory Sakas



*M.A./Ph.D. Program in Linguistics
@ The City University of New York*

1





Language is Hard

- *Buffalo buffalo, Buffalo buffalo buffalo, buffalo, Buffalo buffalo*
- *Someone shot the servant of the actress who was on the balcony. Who was on the balcony?*
- *Who do you think Mary kissed?*
- *Who do you think that Mary kissed?*
- *Who do you think bought a radio?*
- ** Who do you think that bought a radio?*





So how to explain language?

Treat Language as a **scientific field** - like **Physics**.

Example: A scientific principle about sentences:

Given $\langle p \rangle = [\alpha [H \ \beta]]$,
where $\alpha = \text{edge}(\text{Spec}'s)$ β then:
the head H of $\langle p \rangle$ is inert after the phase is
completed, triggering no further grammatical
operations.

Language is complex!!!
Understanding how language works is hard!!!

Unless you're 3.



CS Survey: Prof. Sakas, Computational Linguistics



Linguistic experts!

4



Lecture Slip



Linguistic experts!



Design a program that counts the number of plural nouns in a list of nouns. Think about:

- what the input is,
- what the output is, and
- how you can determine if a noun is plural.

Note: To simplify the problem, assume all plural nouns end in “s”.

Last Time: User Input

Covered in detail in Lab 2:

```
→ 1 mess = input('Please enter a message: ')\n   2 print("You entered", mess)
```

(Demo with pythonTutor)

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.

Side Note: '+' for numbers and strings



- $x = 3 + 5$ stores the number 8 in memory location x .
- $x = x + 1$ increases x by 1.

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.
- `x = x + 1` increases `x` by 1.
- `s = "hi" + "Mom"` stores "hiMom" in memory locations `s`.

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.
- `x = x + 1` increases `x` by 1.
- `s = "hi" + "Mom"` stores "hiMom" in memory locations `s`.
- `s = s + "A"` adds the letter "A" to the end of the strings `s`.

In Pairs or Triples...

Let's start (mostly) with review review:

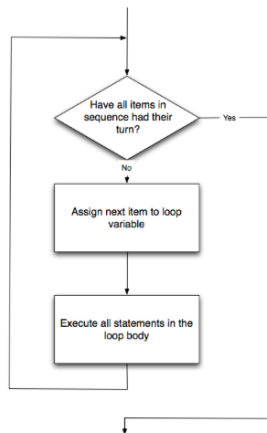
```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Python Tutor

```
1 for d in range(10, 0, -1):
2     print(d)
3     print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

(Demo with pythonTutor)

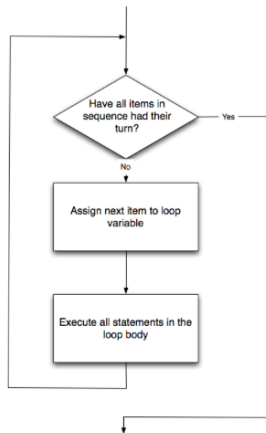
Review: for-loop



```
for i in list:  
    statement1  
    statement2  
    statement3
```

How to Think Like CS, §4.5

Review: for-loop



How to Think Like CS, §4.5

```
for i in list:  
    statement1  
    statement2  
    statement3
```

where `list` is a list of items:

- stated explicitly (e.g. `[1,2,3]`) or
- generated by a function, e.g. `range()`.

range()

What if you wanted to count by twos, or some other number:

```
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)
```

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`

```
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)
```

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
[start, start+step, start+2*step..., last]
(where last is the largest start+k*step less than stop)

```
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)
```

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
[start, start+step, start+2*step..., last]
(where last is the largest start+k*step less than stop)
- For example, if you want the the list [5,10,...,50] you would write:

```
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)
```

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
[start, start+step, start+2*step..., last]
(where last is the largest start+k*step less than stop)
- For example, if you want the the list [5,10,...,50] you would write:

`range(5,51,5)`

```
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)
```

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8last off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[start:stop]`

gives `[start, start+1, start+2..., stop-1]`.

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8Last off!")
4
5 for num in range(5, 8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```


Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[start:stop]`

gives `[start, start+1, start+2..., stop-1]`.

- Also works for lists:

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8Last off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[start:stop]`

gives `[start, start+1, start+2..., stop-1]`.

- Also works for lists:

`names[1:3]`

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8Last off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[start:stop]`

gives `[start, start+1, start+2..., stop-1]`.

- Also works for lists:

`names[1:3]`

gives `["Anna", "Alice"]`

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8last off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[start:stop]`

gives `[start, start+1, start+2..., stop-1]`.

- Also works for lists:






`names[1:3]`

gives `["Anna", "Alice"]`

- Python also lets you “count backwards”: last element has index: `-1`.






```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8Last off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Colors

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






- Can specify by name.

Colors

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






- Can specify by name.
- Can specify by numbers:

Colors

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

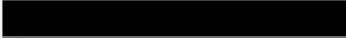




- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).

Colors

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






- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).
 - ▶ Adding light, not paint:

Colors

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






- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).
 - ▶ Adding light, not paint:
 - ★ Black: 0% red, 0% green, 0% blue

Colors

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






- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).
 - ▶ Adding light, not paint:
 - ★ Black: 0% red, 0% green, 0% blue
 - ★ White: 100% red, 100% green, 100% blue

Colors

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






- Can specify by numbers (RGB):

Colors

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






- Can specify by numbers (RGB):
 - ▶ Fractions of each:

Colors

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






- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.

Colors

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






- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:

Colors

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

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.

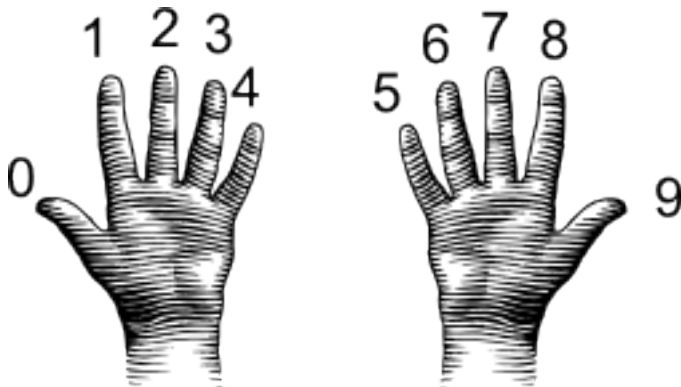
Colors

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

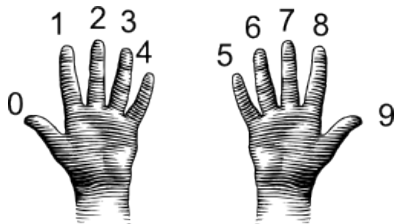
- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.
 - ▶ Hexcodes (base-16 numbers)...

Decimal & Hexadecimal Numbers

Counting with 10 digits:

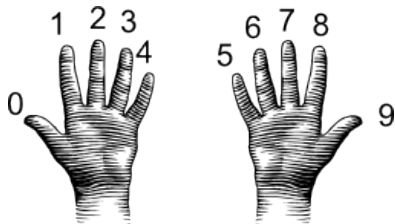


Decimal



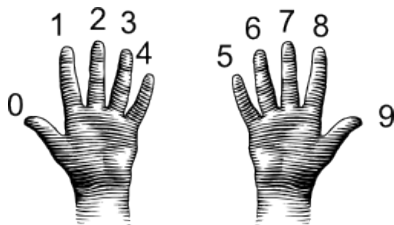
Decimal

00 01 02 03 04 05 06 07 08 09



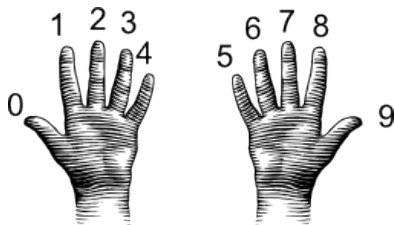
Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19

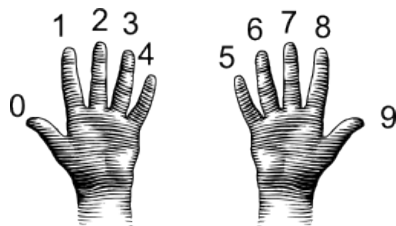


Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29

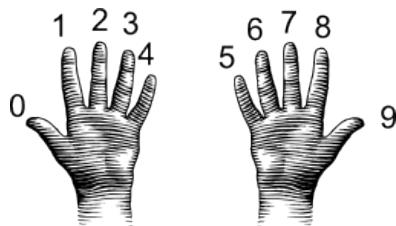


Decimal



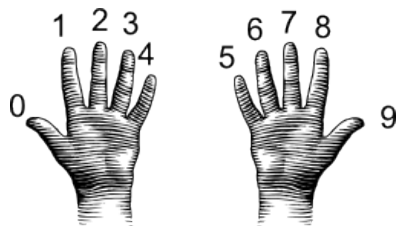
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

Decimal



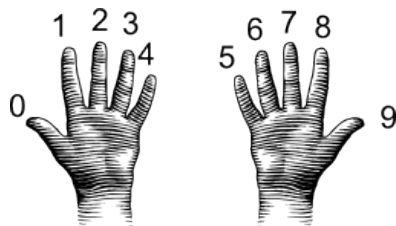
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49

Decimal



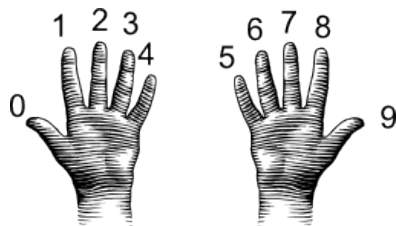
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

Decimal



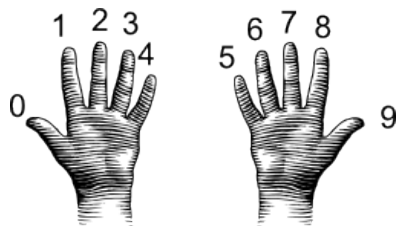
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69

Decimal



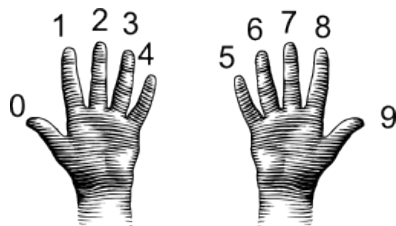
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79

Decimal



00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89

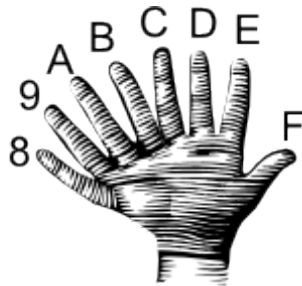
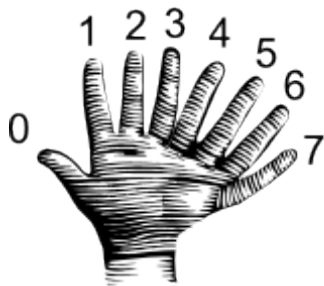
Decimal



00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

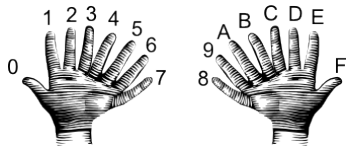
Decimal & Hexadecimal Numbers

Counting with 16 digits:



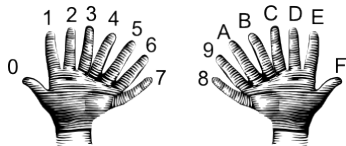
Hexadecimal

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F



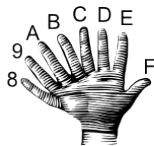
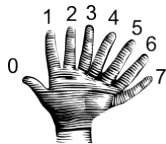
Hexadecimal

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F



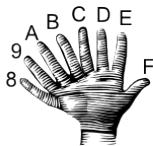
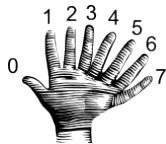
Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F

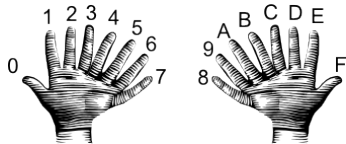


Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F

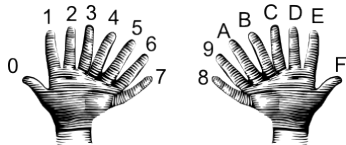


Hexadecimal



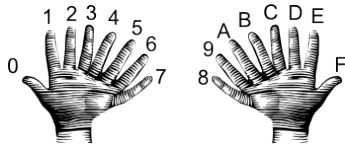
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Hexadecimal



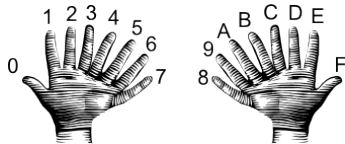
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

Hexadecimal



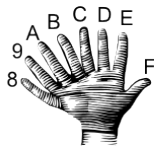
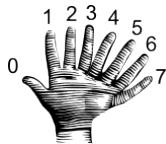
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F

Hexadecimal



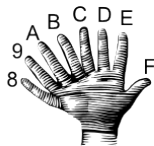
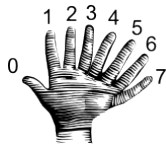
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F

Hexadecimal



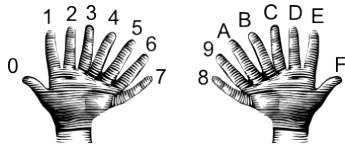
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

Hexadecimal



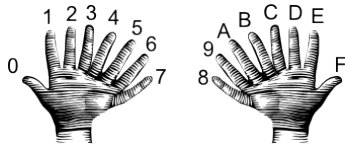
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F

Hexadecimal



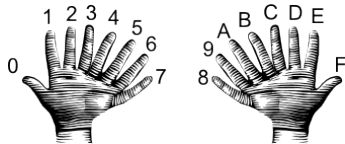
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF

Hexadecimal



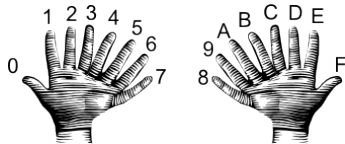
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF

Hexadecimal



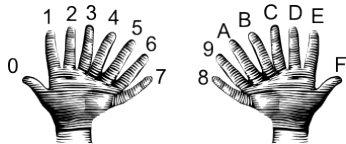
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Hexadecimal



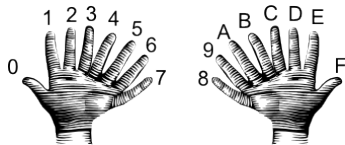
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

Hexadecimal



00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF

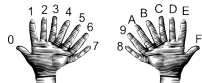
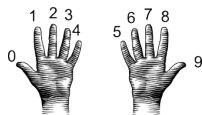
Hexadecimal



00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

Side Note: Listing the numbers






Used Python:



```
base = 10
digits = "0123456789ABCDEF"
```






```
for i in digits[:base]:
    for j in digits[:base]:
        x = str(i) + str(j)
        print(x, end=" ")
    print()
```

Colors

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

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.
 - ▶ Hexcodes (base-16 numbers):

Colors

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

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.
 - ▶ Hexcodes (base-16 numbers):
e.g. #0000FF is no red, no green, and 100% blue.

In Pairs or Triples...

Some review and some novel challenges:

```
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)
```

Trinkets

```
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)
```

(Demo with trinkets)

Images



Images



- We will use the standard portable network graphics (PNG) file format.

Images



- We will use the standard portable network graphics (PNG) file format.
- Saves every picture element (or 'pixel')—

Images



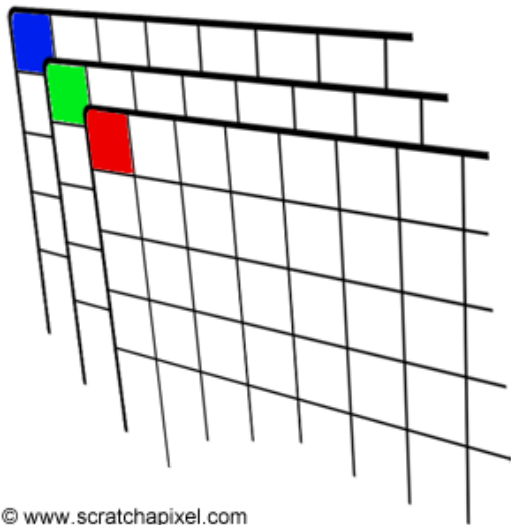
- We will use the standard portable network graphics (PNG) file format.
- Saves every picture element (or 'pixel')– often called a lossless format.

Images



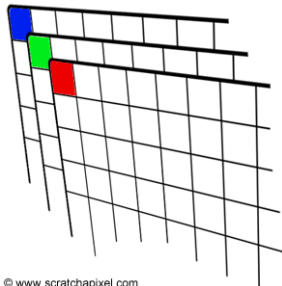
- We will use the standard portable network graphics (PNG) file format.
- Saves every picture element (or 'pixel')– often called a lossless format.
- Keeps track of the amount of red, blue, and green of each pixel.

Images



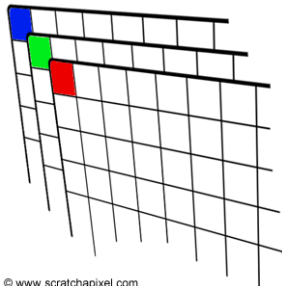
© www.scratchapixel.com

Images

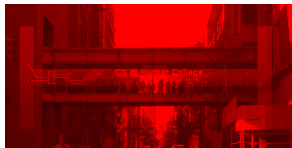


© www.scratchapixel.com

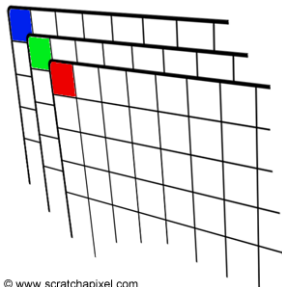
Images



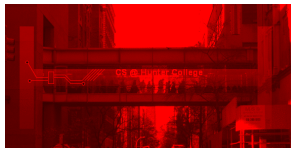
© www.scratchapixel.com



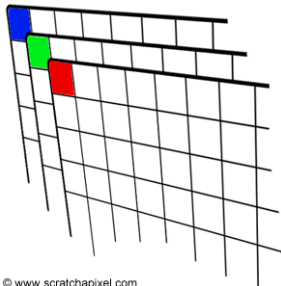
Images



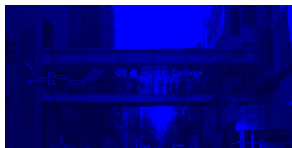
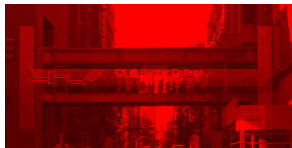
© www.scratchapixel.com



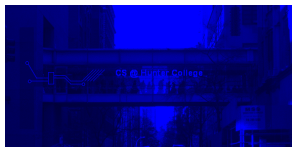
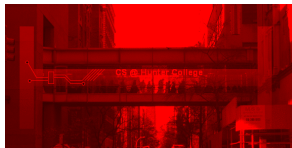
Images



© www.scratchapixel.com

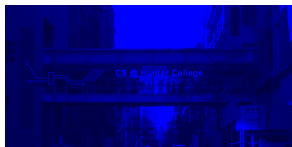
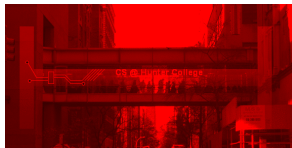


Useful Packages



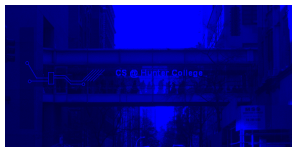
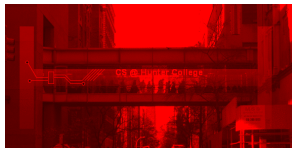
- We will use 2 useful packages for images:

Useful Packages



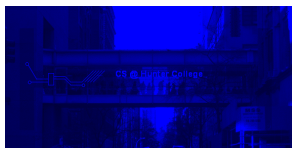
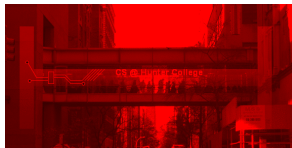
- We will use 2 useful packages for images:
 - ▶ `numpy`: numerical analysis package

Useful Packages



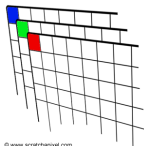
- We will use 2 useful packages for images:
 - ▶ `numpy`: numerical analysis package
 - ▶ `pyplot`: part of `matplotlib` for making graphs and plots

Useful Packages



- We will use 2 useful packages for images:
 - ▶ `numpy`: numerical analysis package
 - ▶ `pyplot`: part of `matplotlib` for making graphs and plots
- See lab notes for installing on your home machine.

Images with pyplot and numpy



#Import the packages for images and arrays:

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
img = plt.imread('csBridge.png') #Read in image from csBridge.png  
plt.imshow(img)                  #Load image into pyplot  
plt.show()                       #Show the image (waits until closed)
```

```
img2 = img.copy()                #make a copy of our image  
img2[:, :, 1] = 0                #Set the green channel to 0  
img2[:, :, 2] = 0                #Set the blue channel to 0
```

```
plt.imshow(img2)                 #Load our new image into pyplot  
plt.show()                       #Show the image (waits until closed to continue)
```

```
plt.imsave('reds.png', img2)    #Save the image we created to the file: reds.png
```

More on numpy arrays

```
>>> 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

numpy tutorial

In Pairs or Triples...

*Some review and
some novel challenges:*

1. Fill in the values in the array:

```
import numpy as np
```

```
A = np.zeros( (4,5) )
```

```
A[0,0] = 1.0
```

```
A[:,1] = 0.75
```

```
A[3,:] = 0.5
```


(If a cell has value 0, you can leave it blank.)

2. Write code that will generate the array with the following values:

Your code here:

1.0				1.0
1.0				1.0
1.0	1.0	1.0	1.0	1.0
1.0				1.0
1.0				1.0

Python Tutor

1. Fill in the values in the array:

```
import numpy as np
```

```
A = np.zeros( (4,5) )
```

```
A[0,0] = 1.0
```

```
A[:,1] = 0.75
```

```
A[3,:] = 0.5
```


(If a cell has value 0, you can leave it blank.)

(Demo with idle3)

2. Write code that will generate the array with the following values:

Your code here:

1.0				1.0
1.0				1.0
1.0	1.0	1.0	1.0	1.0
1.0				1.0
1.0				1.0

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).
- In Python, we introduced:

Recap



- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ Indexing and Slicing Lists

Recap



- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ Indexing and Slicing Lists
 - ▶ Colors

Recap



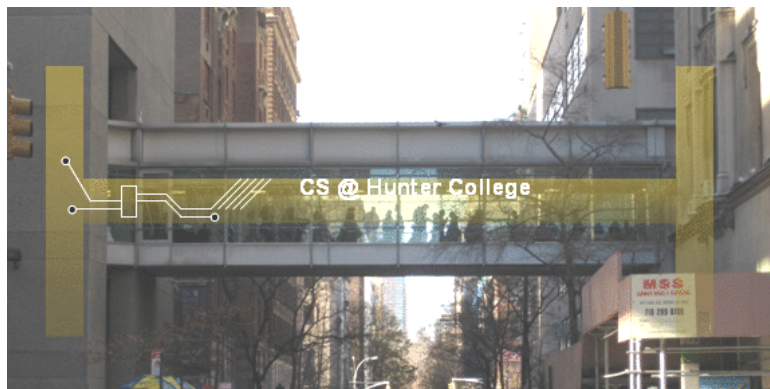
- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ Indexing and Slicing Lists
 - ▶ Colors
 - ▶ Hexadecimal Notation

Recap



- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ Indexing and Slicing Lists
 - ▶ Colors
 - ▶ Hexadecimal Notation
 - ▶ 2D Arrays & Image Files

Lecture Slips & Writing Boards



- Turn in lecture slips & writing boards as you leave...