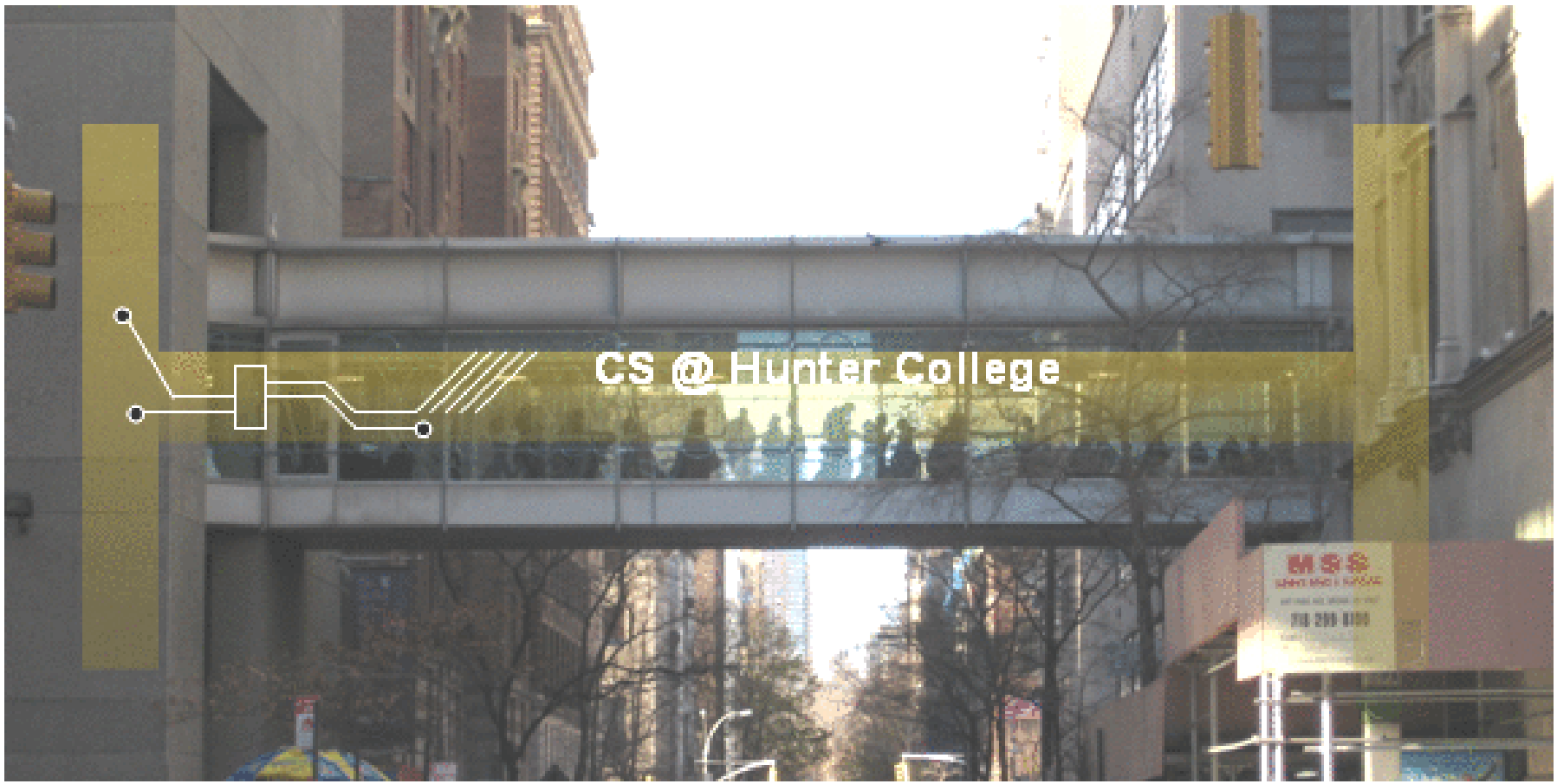
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & emails.

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!
- **What do you mean by Input and Output?**

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!
- **What do you mean by Input and Output?**
  Input is data provided to a program each time it runs (e.g. input() in Lab2); Output is data produced by a program each time it runs (e.g. display text or graphics on screen). Not all programs have Input or Output.

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!
- **What do you mean by Input and Output?**
  Input is data provided to a program each time it runs (e.g. input() in Lab2); Output is data produced by a program each time it runs (e.g. display text or graphics on screen). Not all programs have Input or Output.
- **I don't understand the ASCII Table?**

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!
- **What do you mean by Input and Output?**
  Input is data provided to a program each time it runs (e.g. input() in Lab2); Output is data produced by a program each time it runs (e.g. display text or graphics on screen). Not all programs have Input or Output.
- **I don't understand the ASCII Table?**
  Intro/Survey course: introduce high-level concepts before low-level. Cannot store characters on a computer chip, only numbers. ASCII is simply an agreement on how to map characters to numbers so they can be stored on computer chips.

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!
- **What do you mean by Input and Output?**
  Input is data provided to a program each time it runs (e.g. input() in Lab2); Output is data produced by a program each time it runs (e.g. display text or graphics on screen). Not all programs have Input or Output.
- **I don't understand the ASCII Table?**
  Intro/Survey course: introduce high-level concepts before low-level. Cannot store characters on a computer chip, only numbers. ASCII is simply an agreement on how to map characters to numbers so they can be stored on computer chips.
- **Why are we learning about the command line?**

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!
- **What do you mean by Input and Output?**
  Input is data provided to a program each time it runs (e.g. input() in Lab2); Output is data produced by a program each time it runs (e.g. display text or graphics on screen). Not all programs have Input or Output.
- **I don't understand the ASCII Table?**
  Intro/Survey course: introduce high-level concepts before low-level. Cannot store characters on a computer chip, only numbers. ASCII is simply an agreement on how to map characters to numbers so they can be stored on computer chips.
- **Why are we learning about the command line?**
  Starting with Lab2, bottom section will introduce shell commands. Command line is widely used among Computer Scientists and in Industry; very useful for automating tasks and working remotely. Do not overlook!!! Will be tested on both Quizzes and Final Exam.

Frequently Asked Questions

From lecture slips & emails.

- **What does it mean that this course is hybrid?**
  Instead of having a second weekly lecture, you work independently through the weekly **Online Lab***: reinforces material introduced in lecture, introduces new material, and provide knowledge necessary to work on programming assignments.*
- **Can I work ahead on programs?**
  Absolutely!!! You should be 5 or so programs ahead. If you are working on today's program you are already falling behind!!!
- **What do you mean by Input and Output?**
  Input is data provided to a program each time it runs (e.g. input() in Lab2); Output is data produced by a program each time it runs (e.g. display text or graphics on screen). Not all programs have Input or Output.
- **I don't understand the ASCII Table?**
  Intro/Survey course: introduce high-level concepts before low-level. Cannot store characters on a computer chip, only numbers. ASCII is simply an agreement on how to map characters to numbers so they can be stored on computer chips.
- **Why are we learning about the command line?**
  Starting with Lab2, bottom section will introduce shell commands. Command line is widely used among Computer Scientists and in Industry; very useful for automating tasks and working remotely. Do not overlook!!! Will be tested on both Quizzes and Final Exam.

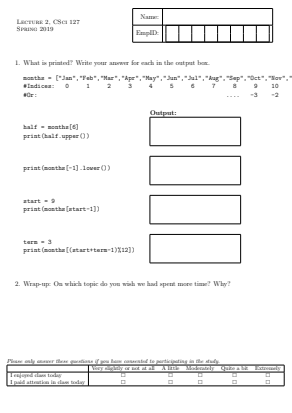
Today's Topics

- Research Survey
- More on Strings
- Arithmetic
- Indexing and Slicing Lists
- Colors & Hexadecimal Notation

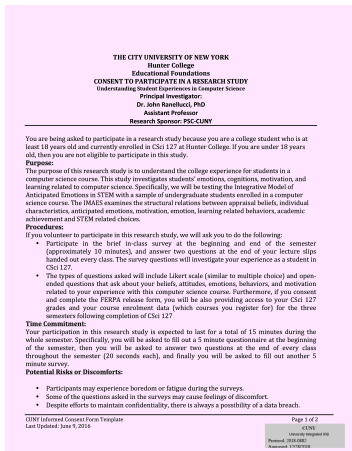
Today's Topics

- **Research Survey**
- More on Strings
- Arithmetic
- Indexing and Slicing Lists
- Colors & Hexadecimal Notation

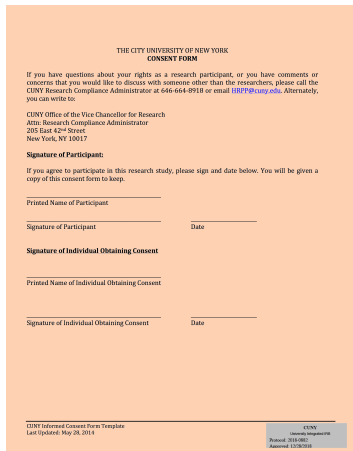
Why All the Handouts Today?



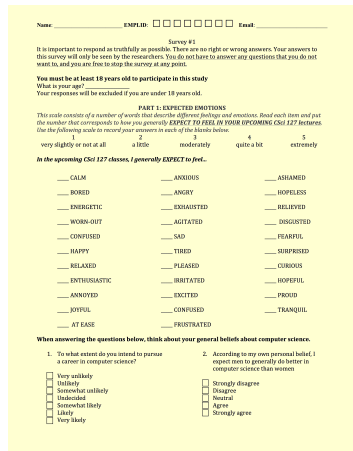
Lecture Slip



Overview



Consent Form



Survey

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.

Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.

Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer three extra questions at the end of your "lecture slips".

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.

Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer three extra questions at the end of your "lecture slips".

Part 3: Consists of six questions per week for 10 weeks (three before class and three after class) **via text message.**

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.

Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer three extra questions at the end of your "lecture slips".

Part 3: Consists of six questions per week for 10 weeks (three before class and three after class) **via text message.**
(Participants will be compensated with a $20 Amazon gift certificate for completing the text-message portion of the survey - $1 for 3-question sets)

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.

Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer three extra questions at the end of your "lecture slips".

Part 3: Consists of six questions per week for 10 weeks (three before class and three after class) **via text message.**
(Participants will be compensated with a $20 Amazon gift certificate for completing the text-message portion of the survey - $1 for 3-question sets)

This study is not part of the class, and no individual analyses will be shared with your instructor.

Today's Topics

- Research Survey
- **More on Strings**
- Arithmetic
- Indexing and Slicing Lists
- Colors & Hexadecimal Notation

More on Strings...

From Final Exam, Fall 2017, Version 1, #1:

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.
- First, go through and write down what we know:

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
  - There are 3 `print()`.

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
  - There are 3 `print()`.
  - Output will have at least:

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
  - There are 3 `print()`.
  - Output will have at least:
    `There are ???  fun days in a week`

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
  - There are 3 `print()`.
  - Output will have at least:

    `There are ???  fun days in a week`

    `Two of them are ???`

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
  - There are 3 `print()`.
  - Output will have at least:

```
There are ???  fun days in a week
Two of them are ???
My favorite ???  is Saturday.
```

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank– write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
  - There are 3 `print()`.
  - Output will have at least:

    ```
    There are ???  fun days in a week
    Two of them are ???
    My favorite ???  is Saturday.
    ```

- *Will get* 1/3 *to* 1/2 *points for writing down the basic structure.*

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in s.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in s.
  - `s.count("s")` counts the number of lower case `s` that occurs.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in s.
  - `s.count("s")` counts the number of lower case `s` that occurs.
  - `num = s.count("s")` stores the result in the variable `num`, for later.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in s.
  - `s.count("s")` counts the number of lower case `s` that occurs.
  - `num = s.count("s")` stores the result in the variable `num`, for later.
  - What would `print(s.count("sS"))` output?

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in s.
  - `s.count("s")` counts the number of lower case `s` that occurs.
  - `num = s.count("s")` stores the result in the variable `num`, for later.
  - What would `print(s.count("sS"))` output?
  - What about:

    ```
    mess = "10 20 21 9 101 35"
    mults = mess.count("0 ")
    print(mults)
    ```

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank– write what you know & puzzle out as much as possible:

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank– write what you know & puzzle out as much as possible:

```
There are 3 fun days in a week
Two of them are ???
My favorite ???  is Saturday.
```

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is `'F'`.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is `'r'`.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is `'s'`.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is `'day'`.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is `'Fri'`.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a "substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is 'FridaysSaturdaysSunday'.
  (no trailing 's' at the end)

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"
days = ['Friday', 'Saturday', 'Sunday']
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXSaturdayXSunday"
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")
"FridaysSaturdaysSunday"
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXSaturdayXSunday"
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")
"FriXXXsSaturXXXsSunXXX"
days = ['Fri', 'sSatur', 'sSun']
```

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank– write what you know & puzzle out as much as possible:

More on Strings...

Name: **EmpID:** CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank– write what you know & puzzle out as much as possible:

```
There are 3 fun days in a week
Two of them are Friday Sunday
My favorite ???  is Saturday.
```

Today's Topics

- Research Survey
- More on Strings
- **Arithmetic**
- Indexing and Slicing Lists
- Colors & Hexadecimal Notation

Arithmetic

Some arithmetic operators in Python:

- Addition:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`

15 // 7 = 2

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`

  15 // 7 = 2
- Remainder or Modulus:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7` *15 // 7 = 2*
- Remainder or Modulus:
  `days = totalDays % 7` *15 % 7 = 1*

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7` *15 // 7 = 2*
- Remainder or Modulus:
  `days = totalDays % 7` *15 % 7 = 1*
- Exponentiaion:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`
  15 // 7 = 2
- Remainder or Modulus:
  `days = totalDays % 7`
  15 % 7 = 1
- Exponentiaion:
  `pop = 2**time`

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.
- If the user enters, 8 and 20.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.
- If the user enters, 8 and 20.
- If the user enters, 11 and 1.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

```
Enter starting time: 9
Enter how long: 2
Your event starts at 9 o'clock.
Your event ends at 11 o'clock.
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 12 and 4.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 12 and 4.

```
Enter starting time: 12
Enter how long: 4
Your event starts at 12 o'clock.
Your event ends at 4 o'clock.
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 8 and 20.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 8 and 20.

```
Enter starting time: 8
Enter how long: 20
Your event starts at 8 o'clock.
Your event ends at 4 o'clock.
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 11 and 1.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 11 and 1.

```
Enter starting time: 11
Enter how long: 1
Your event starts at 11 o'clock.
Your event ends at 0 o'clock.
```

Today's Topics

- Research Survey
- More on Strings
- Arithmetic
- **Indexing and Slicing Lists**
- Colors & Hexadecimal Notation

In Pairs or Triples...

Mostly 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: range()

The three versions:

Review: `range()`

The three versions:

- `range(stop)`

Review: `range()`

The three versions:

- `range(stop)`
- `range(start, stop)`

Review: range()

The three versions:

- `range(stop)`
- `range(start, stop)`
- `range(start, stop, step)`

Slices

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

```
for d in range(10, 0, -1):
    print(d)
print("Blast off!")

for num in range(5,8):
    print(num, 2*num)

s = "City University of New York"
print(s[3], s[0:3], s[:3])
print(s[5:8], s[-1])

names = ["Eleanor", "Anna", "Alice", "Edith"]
for n in names:
    print(n)
```

Slices

```
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)
```

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

  `s[5:8]`

  gives: `"Uni"`

Slices

```
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)
```

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

  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

Slices

```
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)
```

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

  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

  `names[1:3]`

Slices

```
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)
```

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

  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

  `names[1:3]`

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

Slices

```
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)
```

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

  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

  `names[1:3]`

  gives: `["Anna", "Alice"]`
- Python also lets you "count backwards": last element has index: `-1`.

Today's Topics

- Arithmetic
- Indexing and Slicing Lists
- Design Challenge: Planes
- **Colors & Hexadecimal Notation**

Colors

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

- Can specify by name.

Colors

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

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

Colors

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

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

Colors

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

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

Colors

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

- 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
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):

Colors

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

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

Colors

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

- 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
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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.

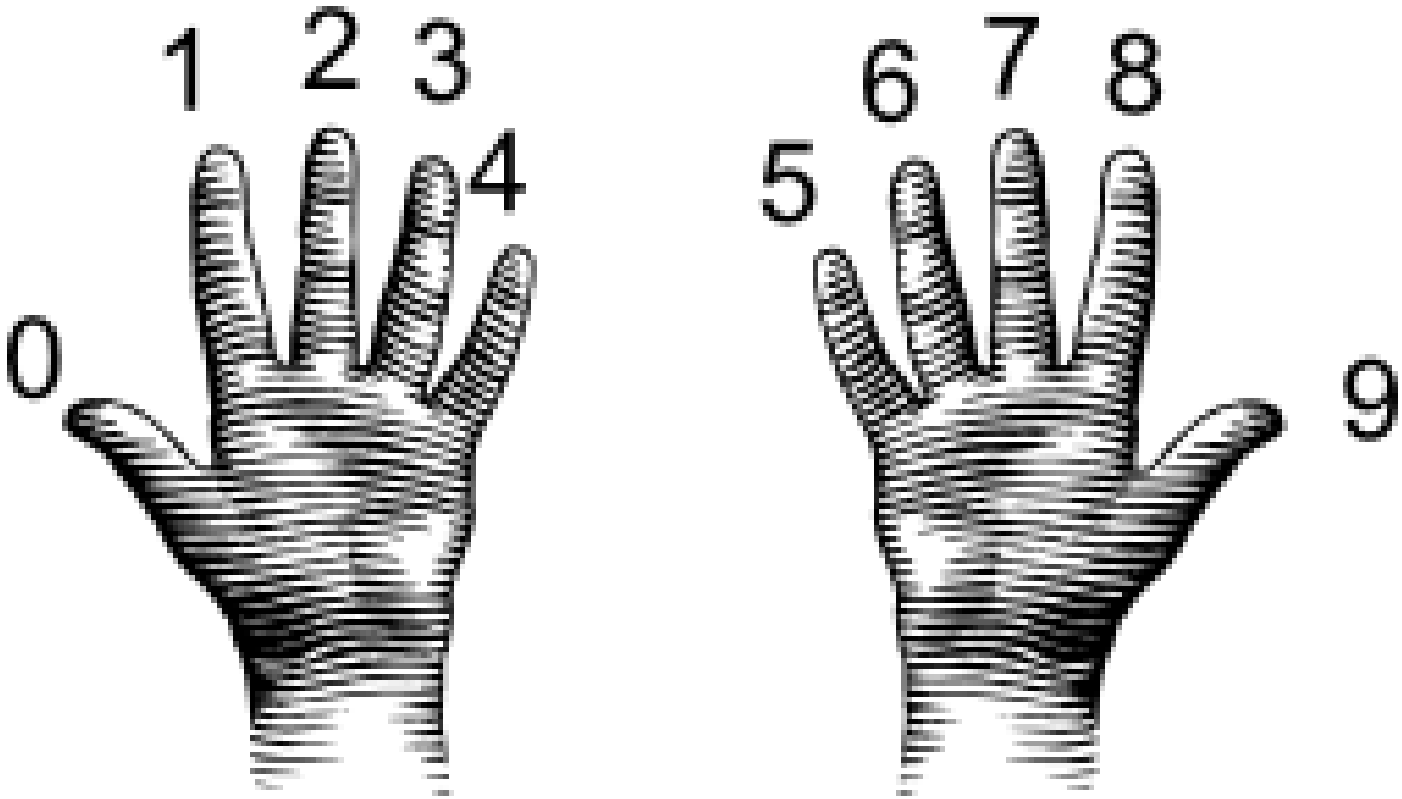
Colors

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

- 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)...

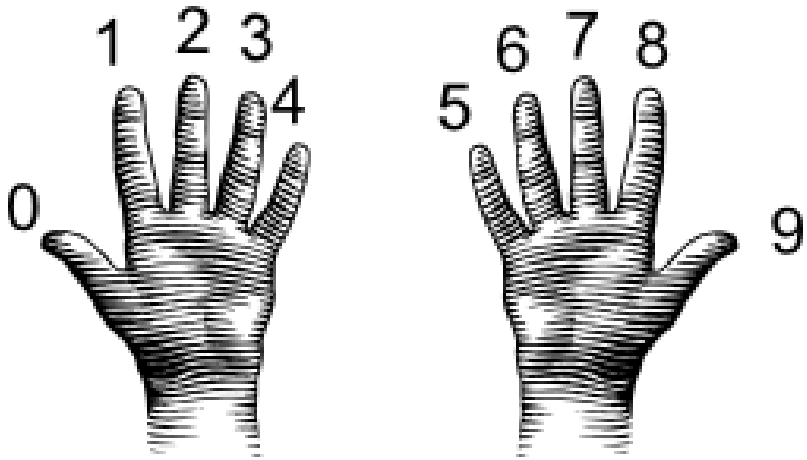
Decimal & Hexadecimal Numbers

Counting with 10 digits:



(from i-programmer.info)

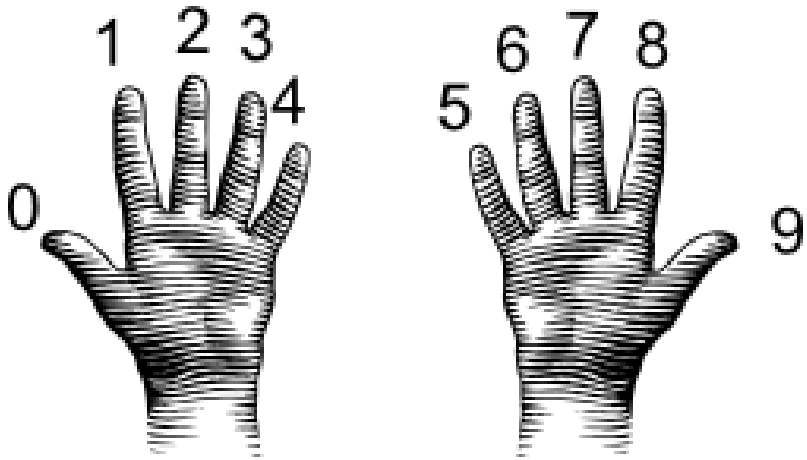
Decimal



(from i-programmer.info)

Decimal

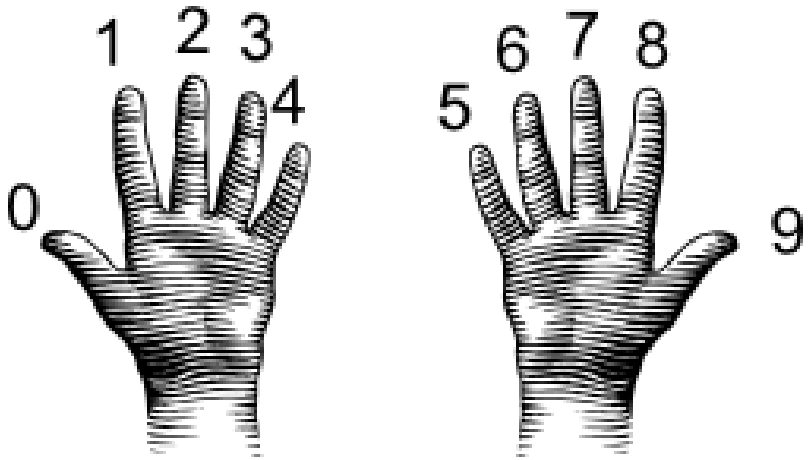
00 01 02 03 04 05 06 07 08 09



(from i-programmer.info)

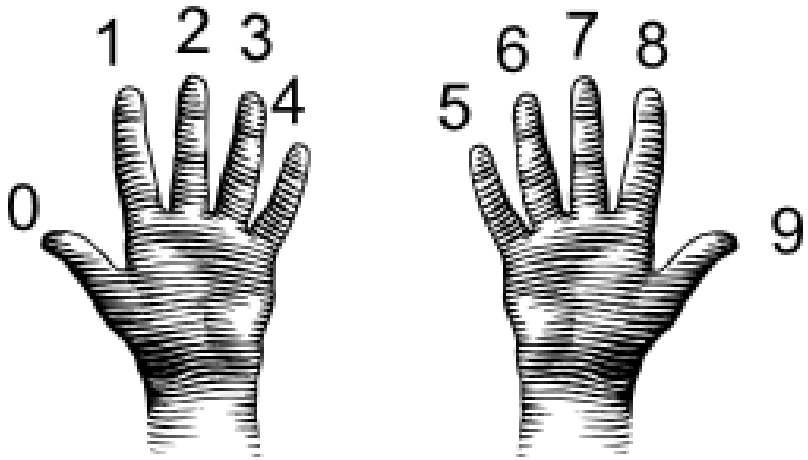
Decimal

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



(from i-programmer.info)

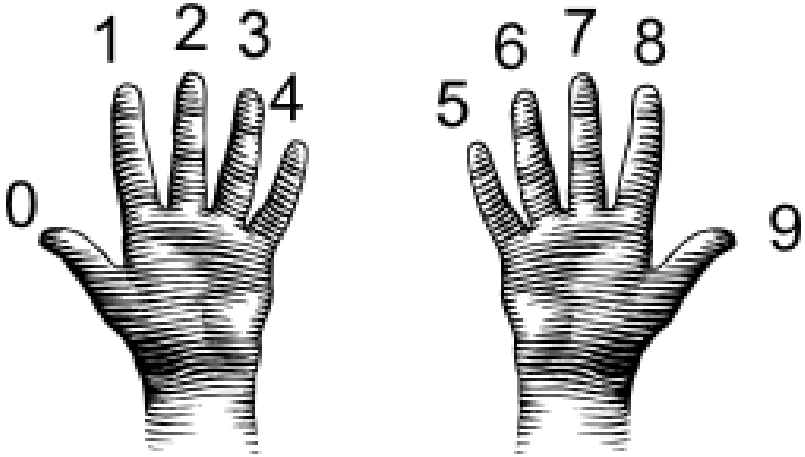
Decimal



(from i-programmer.info)

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

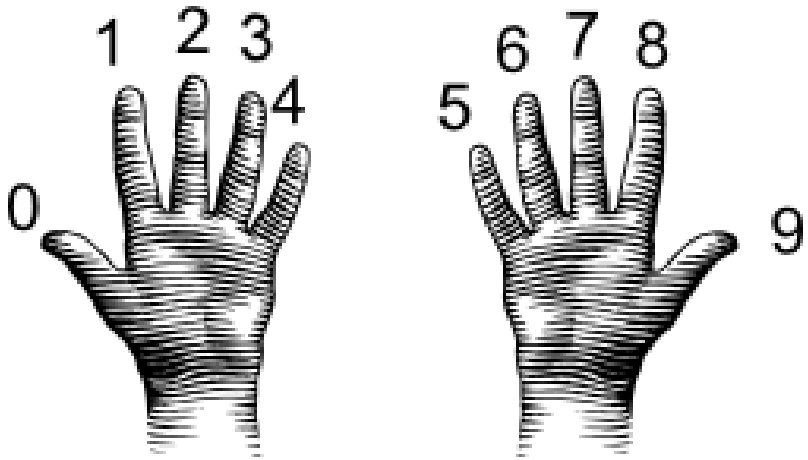
Decimal



(from i-programmer.info)

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

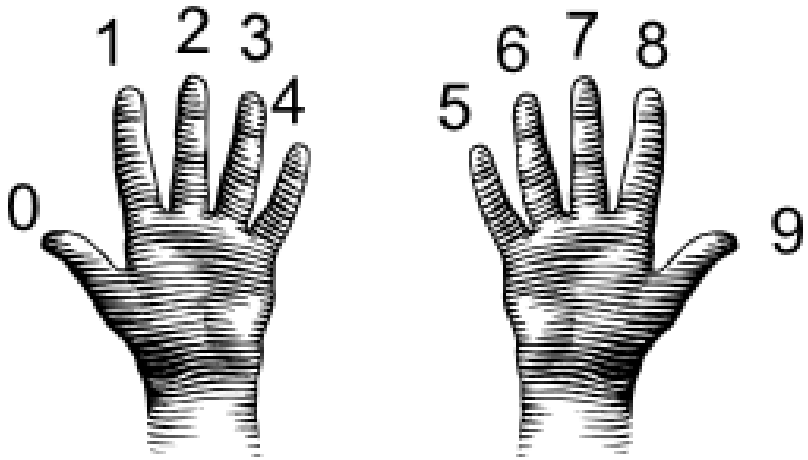
Decimal



(from i-programmer.info)

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

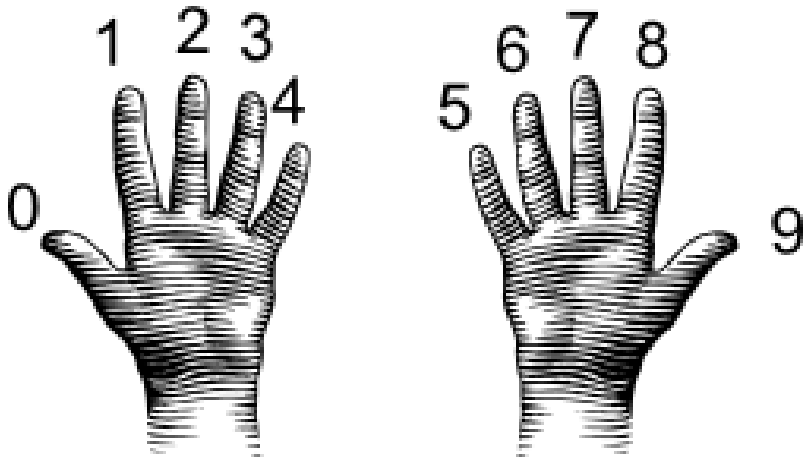
Decimal



(from i-programmer.info)

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

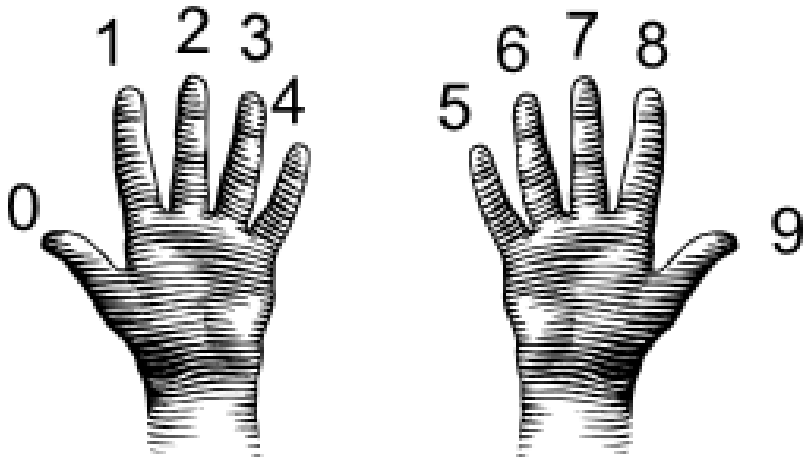
Decimal



(from i-programmer.info)

```
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
```

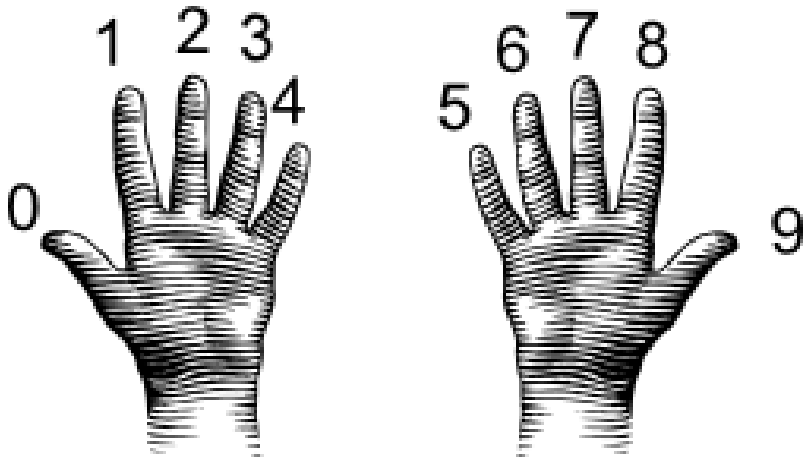
Decimal



(from i-programmer.info)

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

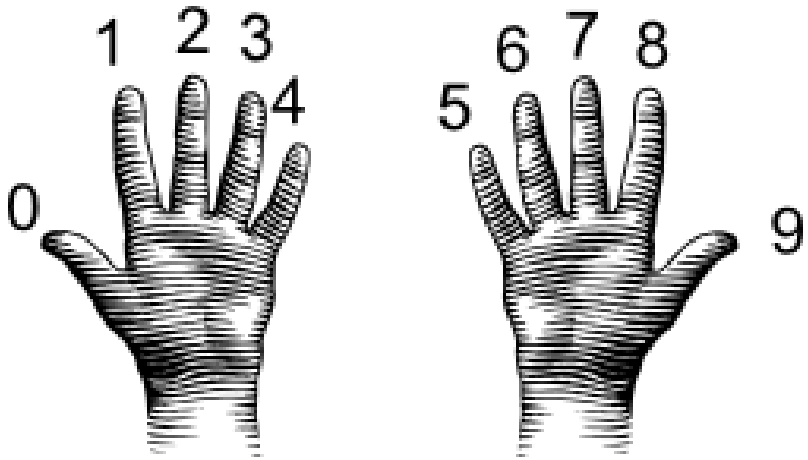
Decimal



(from i-programmer.info)

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



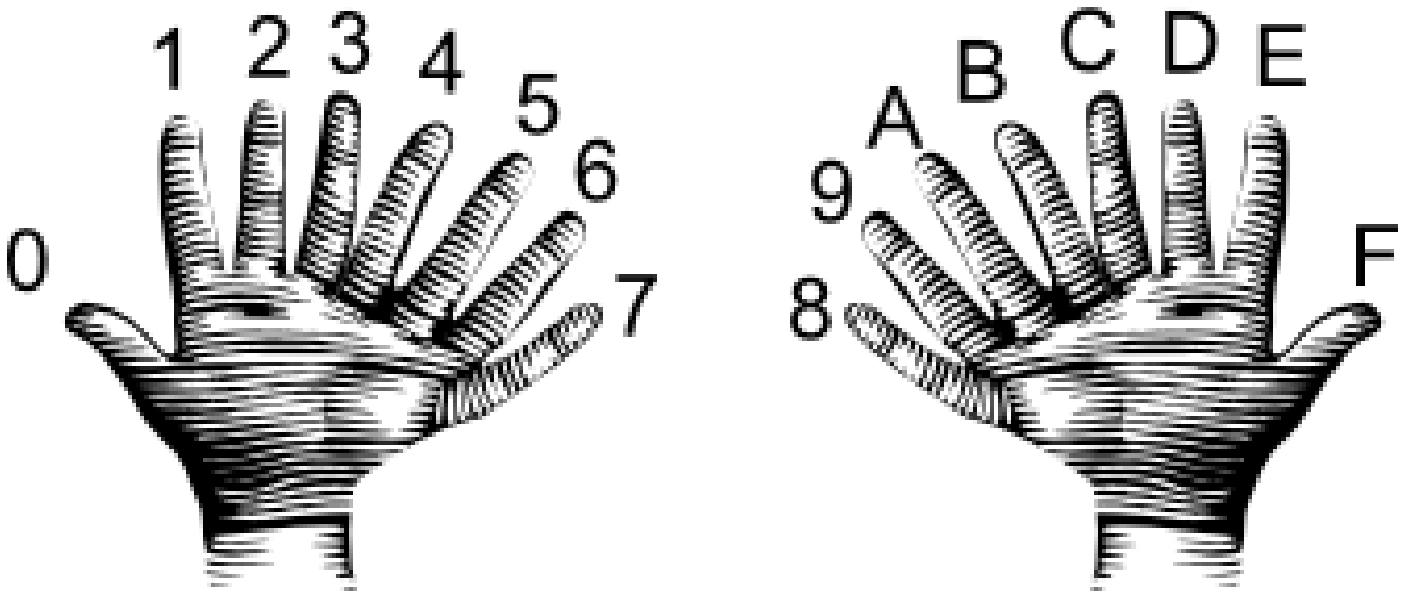
(from i-programmer.info)

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

Max Number = 99

Decimal & Hexadecimal Numbers

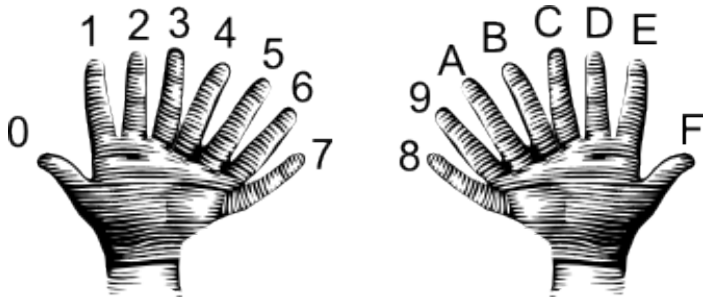
Counting with 16 digits:



(from i-programmer.info)

Hexadecimal

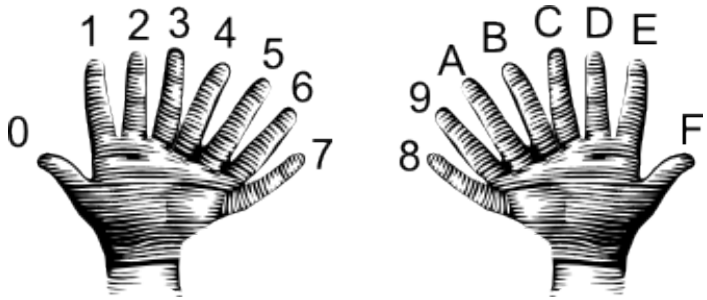
```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```



(from i-programmer.info)

Hexadecimal

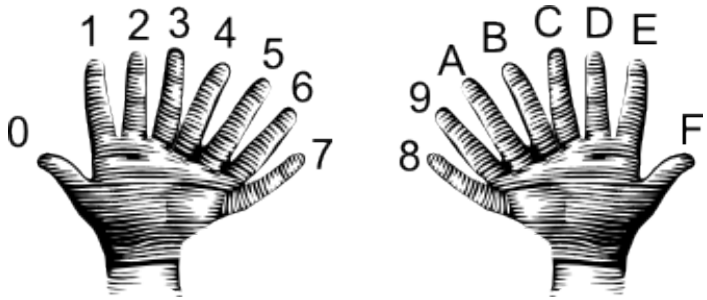
```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
```



(from i-programmer.info)

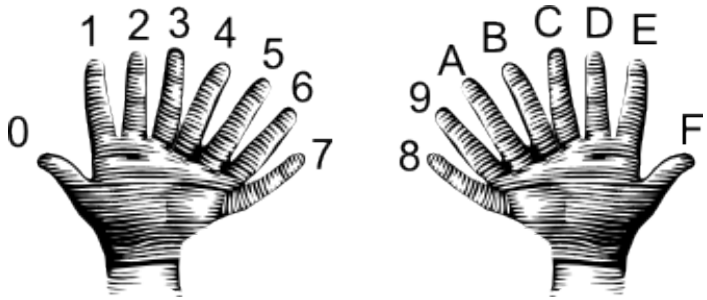
Hexadecimal

```
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
```



(from i-programmer.info)

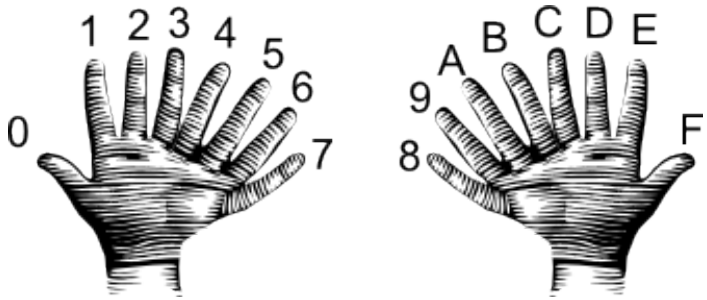
Hexadecimal



(from i-programmer.info)

```
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
```

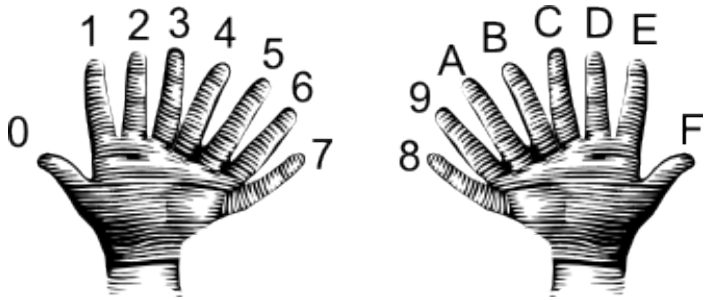
Hexadecimal



(from i-programmer.info)

```
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
```

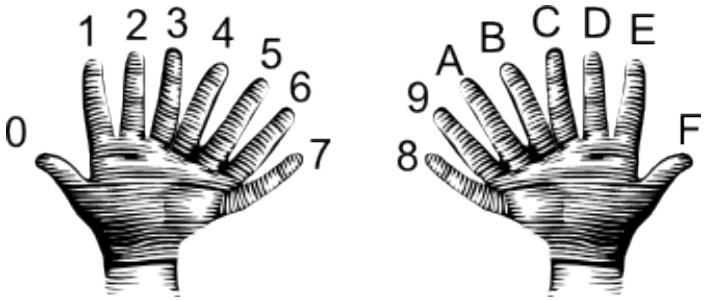
Hexadecimal



(from i-programmer.info)

```
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
```

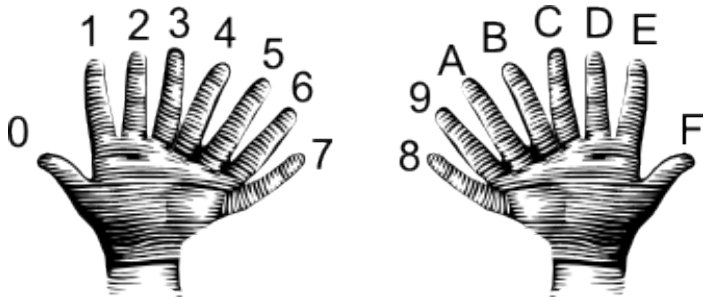
Hexadecimal



(from i-programmer.info)

```
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
```

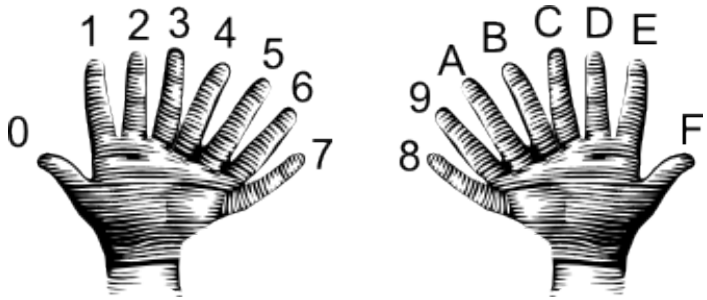
Hexadecimal



(from i-programmer.info)

```
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
```

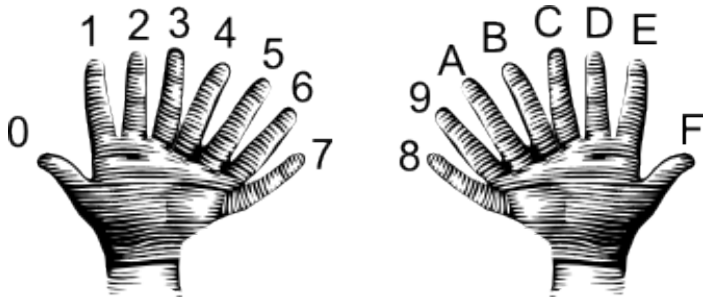
Hexadecimal



(from i-programmer.info)

```
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
```

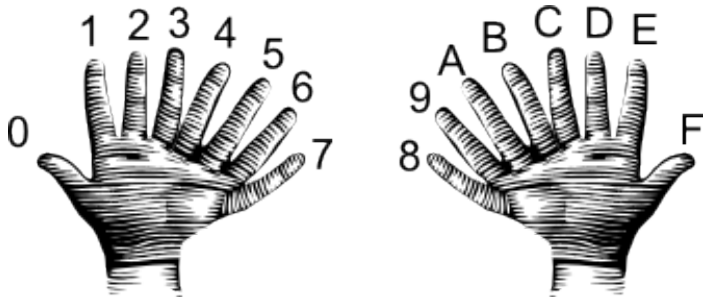
Hexadecimal



(from i-programmer.info)

```
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
```

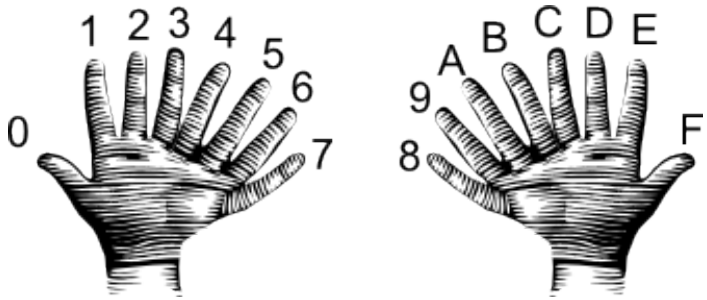
Hexadecimal



(from i-programmer.info)

```
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
```

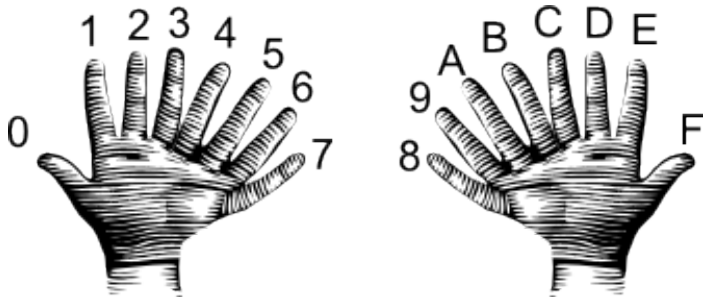
Hexadecimal



(from i-programmer.info)

```
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
```

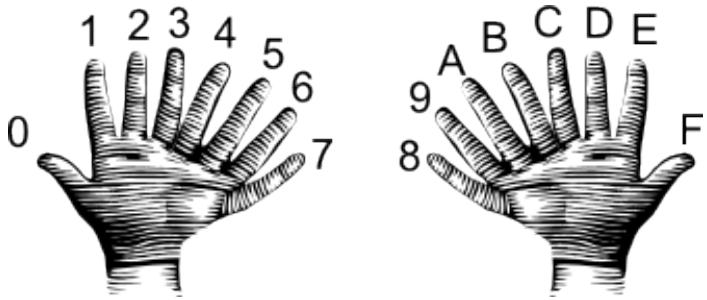
Hexadecimal



(from i-programmer.info)

```
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
```

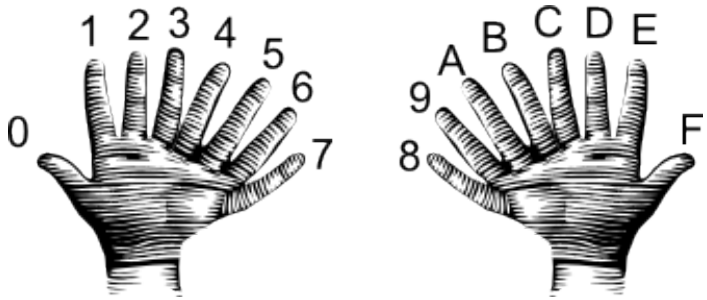
Hexadecimal



(from i-programmer.info)

```
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
```

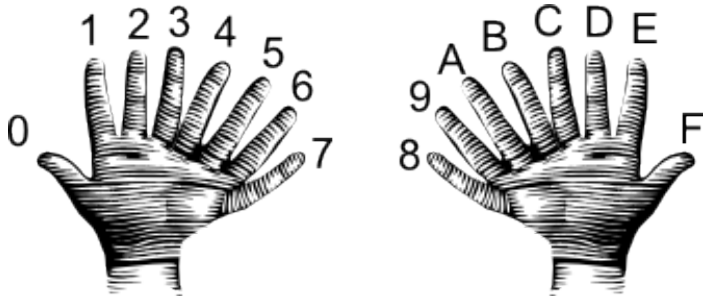
Hexadecimal



(from i-programmer.info)

```
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



(from i-programmer.info)

```
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
```

Max Number = 255

Colors

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

- 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
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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`)

Lecture Slip

LECTURE 3, CSCI 127
SPRING 2020

Name:	
EmpID:	

1. Write down the names of your team members:

2. What is printed? Write your answer for each in the output box.

```
months = ["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]
#Indices:   0     1     2     3     4     5     6     7     8     9    10    11
#Or:                                                      ....   -3    -2    -1
```

Output:

```
half = months[6]
print(half.upper())
```

```
print(months[-1].lower())
```

```
start = 9
print(months[start-1])
```

```
term = 3
print(months[(start+term-1)%12])
```

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
- Pass your lecture slips to the end of the rows for the UTA's to collect.

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.

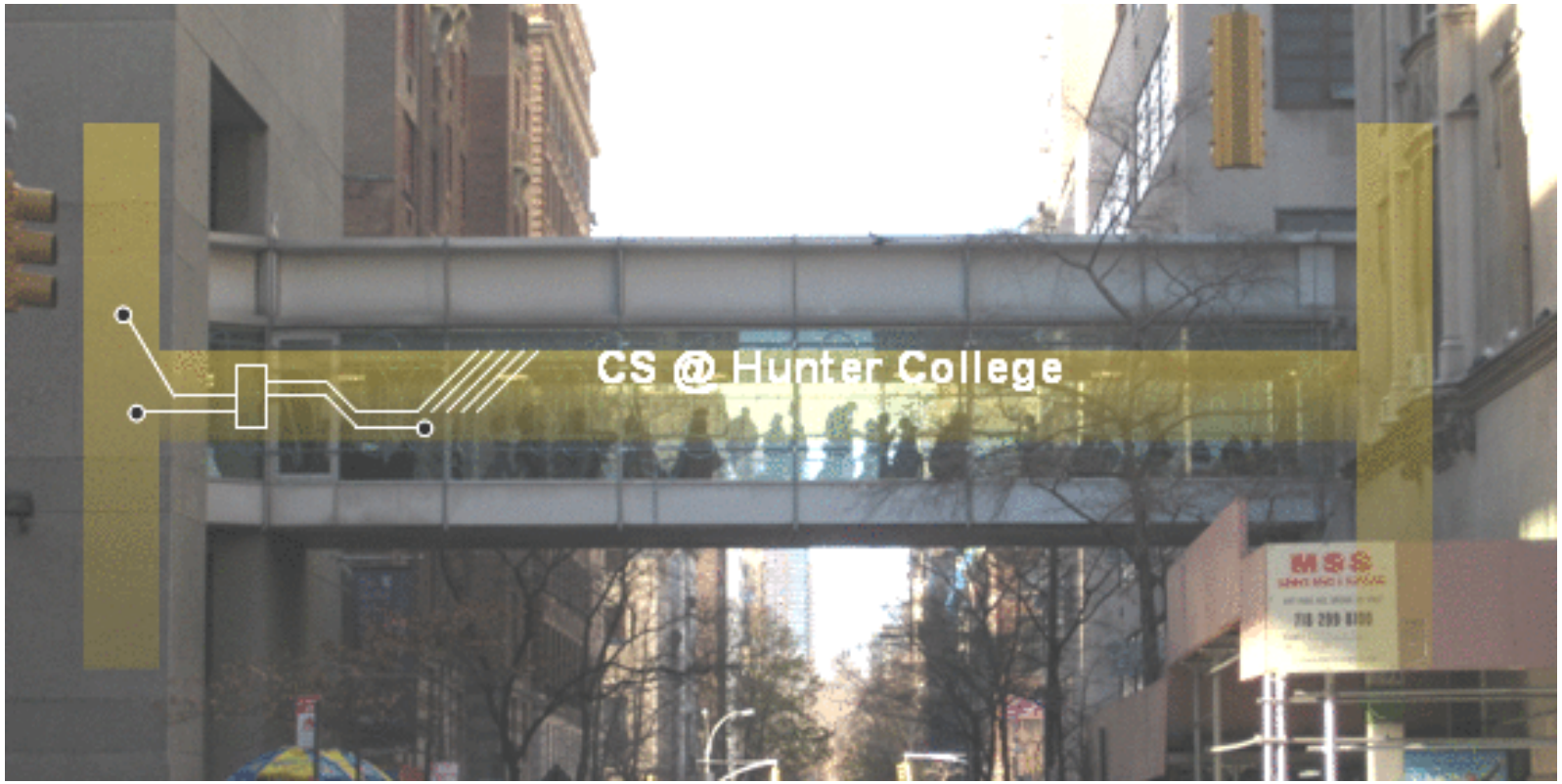
Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final Exam Information).

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final Exam Information).
- We're starting with Fall 2017, Version 2.

Writing Boards



- Return writing boards as you leave...