

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & email

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must:*

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN);*

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN);*

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!);*

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

- I missed class. Do you need documentation?

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

- I missed class. Do you need documentation?

No, but If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

- I missed class. Do you need documentation?

No, but If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- I have not received any emails from this course.

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

- I missed class. Do you need documentation?

No, but If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- I have not received any emails from this course.

That is a big problem! *We send tons of important information through email. Please update your email on Blackboard to one you check regularly.*

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

- I missed class. Do you need documentation?

No, but If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- I have not received any emails from this course.

That is a big problem! *We send tons of important information through email. Please update your email on Blackboard to one you check regularly.*

- Can I work ahead?

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

- I missed class. Do you need documentation?

No, but If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- I have not received any emails from this course.

That is a big problem! *We send tons of important information through email. Please update your email on Blackboard to one you check regularly.*

- Can I work ahead?

Absolutely! Submission is open on Gradescope, 3 weeks before the deadline. Start right away (after Lab1 you can submit the first 5 problems)

Frequently Asked Questions

From lecture slips & email

- Am I only responsible for reading the weekly Lab?

No. *Each week you must: Take a lecture preview (available Mondays); Come to lecture (Tuesday 9:45-11 118 HN); Take a Quiz and Code Review (self-scheduled, 1001E HN); Read the weekly lab (online, see Course Outline on course website, find it on Blackboard!!!); Submit programming assignments to Gradescope (approx. 5 per lab)*

- When is the midterm?

There is no midterm. Instead there's required weekly quizzes and daily programming assignments.

- I missed class. Do you need documentation?

No, but If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- I have not received any emails from this course.

That is a big problem! *We send tons of important information through email. Please update your email on Blackboard to one you check regularly.*

- Can I work ahead?

Absolutely! Submission is open on Gradescope, 3 weeks before the deadline. Start right away (after Lab1 you can submit the first 5 problems)

Today's Topics



- **For-loops**
- `range()`
- Variables
- Characters
- Strings
- CS Survey (Dr. Sakas, Computational Linguistics)

In Pairs or Triples...

Some review and some novel challenges:

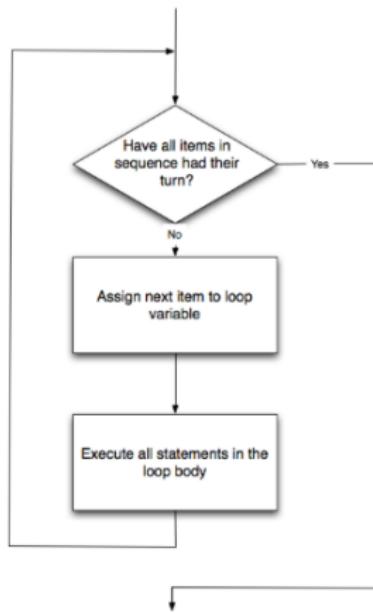
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

Python Tutor

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

(Demo with pythonTutor)

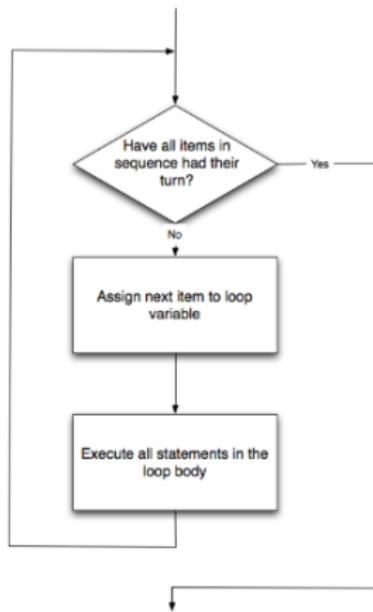
for-loop



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

How to Think Like CS, §4.5

for-loop



```
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().

How to Think Like CS, §4.5

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- Strings
- CS Survey (Dr. Sakas, Computational Linguistics)

More on range():

```
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)
```

Python Tutor

```
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)
```

(Demo with pythonTutor)

range()

Simplest version:

- `range(stop)`



range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`

range()



Simplest version:

- `range(stop)`
- Produces a list: $[0,1,2,3,\dots,stop-1]$
- For example, if you want the list $[0,1,2,3,\dots,100]$, you would write:

range()



Simplest version:

- `range(stop)`
- Produces a list: $[0,1,2,3,\dots,stop-1]$
- For example, if you want the list $[0,1,2,3,\dots,100]$, you would write:

```
range(101)
```

`range()`

What if you wanted to start somewhere else:



range()

What if you wanted to start somewhere else:

- `range(start, stop)`



range()

What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start,start+1,...,stop-1]`



range()

What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start,start+1,...,stop-1]`
- For example, if you want the list
`[10,11,...,20]`
you would write:



range()

What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start,start+1,...,stop-1]`
- For example, if you want the list
`[10,11,...,20]`
you would write:



```
range(10,21)
```

`range()`

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



range()

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

- `range(start, stop, step)`



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 $\text{start}+k*\text{step}$ less than stop)



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 list
`[5,10,...,50]`
you would write:

range()

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

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

```
range(5, 51, 5)
```



In summary: range()



The three versions:

In summary: range()



The three versions:

- `range(stop)`

In summary: range()



The three versions:

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

In summary: range()



The three versions:

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

Today's Topics



- For-loops
- `range()`
- **Variables**
- Characters
- Strings
- CS Survey (Dr. Sakas, Computational Linguistics)

Variables

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



Variables

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



Variables

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



Variables



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

Variables



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

Variables



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

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
 - e.g. [3, 1, 4, 5, 9] or
 - ['violet', 'purple', 'indigo']
 - ▶ **class variables**: for complex objects, like turtles.
- In Python (unlike other languages) you don't need to specify the type; it is deduced by its value.

Variable Names

- There's some rules about valid names for variables.



Variable Names

- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.



Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.
- Can't use some words that Python has reserved for itself (e.g. `for`).
(List of reserved words in *Think CS*, §2.5.)

Today's Topics



- For-loops
- `range()`
- Variables
- **Characters**
- Strings
- CS Survey (Dr. Sakas, Computational Linguistics)

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.
(New version called: Unicode).

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.
(New version called: Unicode).

ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---------|-----|------------------------|---------|-----|---------|---------|-----|------|---------|-----|-------|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | \$ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | , | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | (| 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 |) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [END OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | \ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | | 124 | 7C | |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D |] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | - |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

(wiki)

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

ASCII TABLE

| Binary Hex Char | Binary Hex Char | Decimal Hex Char | Decimal Non Char |
|-----------------|-----------------|------------------|------------------|
| 00000000 | 00000000 | 00 | 0 |
| 00000001 | 00000001 | 01 | 1 |
| 00000002 | 00000002 | 02 | 2 |
| 00000003 | 00000003 | 03 | 3 |
| 00000004 | 00000004 | 04 | 4 |
| 00000005 | 00000005 | 05 | 5 |
| 00000006 | 00000006 | 06 | 6 |
| 00000007 | 00000007 | 07 | 7 |
| 00000008 | 00000008 | 08 | 8 |
| 00000009 | 00000009 | 09 | 9 |
| 0000000A | 0000000A | 0A | A |
| 0000000B | 0000000B | 0B | B |
| 0000000C | 0000000C | 0C | C |
| 0000000D | 0000000D | 0D | D |
| 0000000E | 0000000E | 0E | E |
| 0000000F | 0000000F | 0F | F |
| 00000010 | 00000010 | 10 | 10 |
| 00000011 | 00000011 | 11 | 11 |
| 00000012 | 00000012 | 12 | 12 |
| 00000013 | 00000013 | 13 | 13 |
| 00000014 | 00000014 | 14 | 14 |
| 00000015 | 00000015 | 15 | 15 |
| 00000016 | 00000016 | 16 | 16 |
| 00000017 | 00000017 | 17 | 17 |
| 00000018 | 00000018 | 18 | 18 |
| 00000019 | 00000019 | 19 | 19 |
| 0000001A | 0000001A | 1A | 1A |
| 0000001B | 0000001B | 1B | 1B |
| 0000001C | 0000001C | 1C | 1C |
| 0000001D | 0000001D | 1D | 1D |
| 0000001E | 0000001E | 1E | 1E |
| 0000001F | 0000001F | 1F | 1F |
| 00000020 | 00000020 | 20 | 20 |
| 00000021 | 00000021 | 21 | 21 |
| 00000022 | 00000022 | 22 | 22 |
| 00000023 | 00000023 | 23 | 23 |
| 00000024 | 00000024 | 24 | 24 |
| 00000025 | 00000025 | 25 | 25 |
| 00000026 | 00000026 | 26 | 26 |
| 00000027 | 00000027 | 27 | 27 |
| 00000028 | 00000028 | 28 | 28 |
| 00000029 | 00000029 | 29 | 29 |
| 0000002A | 0000002A | 2A | 2A |
| 0000002B | 0000002B | 2B | 2B |
| 0000002C | 0000002C | 2C | 2C |
| 0000002D | 0000002D | 2D | 2D |
| 0000002E | 0000002E | 2E | 2E |
| 0000002F | 0000002F | 2F | 2F |
| 00000030 | 00000030 | 30 | 30 |
| 00000031 | 00000031 | 31 | 31 |
| 00000032 | 00000032 | 32 | 32 |
| 00000033 | 00000033 | 33 | 33 |
| 00000034 | 00000034 | 34 | 34 |
| 00000035 | 00000035 | 35 | 35 |
| 00000036 | 00000036 | 36 | 36 |
| 00000037 | 00000037 | 37 | 37 |
| 00000038 | 00000038 | 38 | 38 |
| 00000039 | 00000039 | 39 | 39 |
| 0000003A | 0000003A | 3A | 3A |
| 0000003B | 0000003B | 3B | 3B |
| 0000003C | 0000003C | 3C | 3C |
| 0000003D | 0000003D | 3D | 3D |
| 0000003E | 0000003E | 3E | 3E |
| 0000003F | 0000003F | 3F | 3F |
| 00000040 | 00000040 | 40 | 40 |
| 00000041 | 00000041 | 41 | 41 |
| 00000042 | 00000042 | 42 | 42 |
| 00000043 | 00000043 | 43 | 43 |
| 00000044 | 00000044 | 44 | 44 |
| 00000045 | 00000045 | 45 | 45 |
| 00000046 | 00000046 | 46 | 46 |
| 00000047 | 00000047 | 47 | 47 |
| 00000048 | 00000048 | 48 | 48 |
| 00000049 | 00000049 | 49 | 49 |
| 0000004A | 0000004A | 4A | 4A |
| 0000004B | 0000004B | 4B | 4B |
| 0000004C | 0000004C | 4C | 4C |
| 0000004D | 0000004D | 4D | 4D |
| 0000004E | 0000004E | 4E | 4E |
| 0000004F | 0000004F | 4F | 4F |
| 00000050 | 00000050 | 50 | 50 |
| 00000051 | 00000051 | 51 | 51 |
| 00000052 | 00000052 | 52 | 52 |
| 00000053 | 00000053 | 53 | 53 |
| 00000054 | 00000054 | 54 | 54 |
| 00000055 | 00000055 | 55 | 55 |
| 00000056 | 00000056 | 56 | 56 |
| 00000057 | 00000057 | 57 | 57 |
| 00000058 | 00000058 | 58 | 58 |
| 00000059 | 00000059 | 59 | 59 |
| 0000005A | 0000005A | 5A | 5A |
| 0000005B | 0000005B | 5B | 5B |
| 0000005C | 0000005C | 5C | 5C |
| 0000005D | 0000005D | 5D | 5D |
| 0000005E | 0000005E | 5E | 5E |
| 0000005F | 0000005F | 5F | 5F |
| 00000060 | 00000060 | 60 | 60 |
| 00000061 | 00000061 | 61 | 61 |
| 00000062 | 00000062 | 62 | 62 |
| 00000063 | 00000063 | 63 | 63 |
| 00000064 | 00000064 | 64 | 64 |
| 00000065 | 00000065 | 65 | 65 |
| 00000066 | 00000066 | 66 | 66 |
| 00000067 | 00000067 | 67 | 67 |
| 00000068 | 00000068 | 68 | 68 |
| 00000069 | 00000069 | 69 | 69 |
| 0000006A | 0000006A | 6A | 6A |
| 0000006B | 0000006B | 6B | 6B |
| 0000006C | 0000006C | 6C | 6C |
| 0000006D | 0000006D | 6D | 6D |
| 0000006E | 0000006E | 6E | 6E |
| 0000006F | 0000006F | 6F | 6F |
| 00000070 | 00000070 | 70 | 70 |
| 00000071 | 00000071 | 71 | 71 |
| 00000072 | 00000072 | 72 | 72 |
| 00000073 | 00000073 | 73 | 73 |
| 00000074 | 00000074 | 74 | 74 |
| 00000075 | 00000075 | 75 | 75 |
| 00000076 | 00000076 | 76 | 76 |
| 00000077 | 00000077 | 77 | 77 |
| 00000078 | 00000078 | 78 | 78 |
| 00000079 | 00000079 | 79 | 79 |
| 0000007A | 0000007A | 7A | 7A |
| 0000007B | 0000007B | 7B | 7B |
| 0000007C | 0000007C | 7C | 7C |
| 0000007D | 0000007D | 7D | 7D |
| 0000007E | 0000007E | 7E | 7E |
| 0000007F | 0000007F | 7F | 7F |
| 00000080 | 00000080 | 80 | 80 |
| 00000081 | 00000081 | 81 | 81 |
| 00000082 | 00000082 | 82 | 82 |
| 00000083 | 00000083 | 83 | 83 |
| 00000084 | 00000084 | 84 | 84 |
| 00000085 | 00000085 | 85 | 85 |
| 00000086 | 00000086 | 86 | 86 |
| 00000087 | 00000087 | 87 | 87 |
| 00000088 | 00000088 | 88 | 88 |
| 00000089 | 00000089 | 89 | 89 |
| 0000008A | 0000008A | 8A | 8A |
| 0000008B | 0000008B | 8B | 8B |
| 0000008C | 0000008C | 8C | 8C |
| 0000008D | 0000008D | 8D | 8D |
| 0000008E | 0000008E | 8E | 8E |
| 0000008F | 0000008F | 8F | 8F |
| 00000090 | 00000090 | 90 | 90 |
| 00000091 | 00000091 | 91 | 91 |
| 00000092 | 00000092 | 92 | 92 |
| 00000093 | 00000093 | 93 | 93 |
| 00000094 | 00000094 | 94 | 94 |
| 00000095 | 00000095 | 95 | 95 |
| 00000096 | 00000096 | 96 | 96 |
| 00000097 | 00000097 | 97 | 97 |
| 00000098 | 00000098 | 98 | 98 |
| 00000099 | 00000099 | 99 | 99 |
| 0000009A | 0000009A | 9A | 9A |
| 0000009B | 0000009B | 9B | 9B |
| 0000009C | 0000009C | 9C | 9C |
| 0000009D | 0000009D | 9D | 9D |
| 0000009E | 0000009E | 9E | 9E |
| 0000009F | 0000009F | 9F | 9F |
| 000000A0 | 000000A0 | A0 | 100 |

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

| Decimal Num Char | Octal Num Char | Hex Num Char | Decimal Num Char | Octal Num Char | Hex Num Char |
|------------------|----------------|--------------|------------------|----------------|--------------|
| ' ' | '0' | '0000' | '0' | '0000' | '0000' |
| '0' | '10' | '0030' | '160' | '400' | 'A0 |
| '1' | '11' | '0031' | '161' | '401' | 'A1' |
| '2' | '12' | '0032' | '162' | '402' | 'A2' |
| '3' | '13' | '0033' | '163' | '403' | 'A3' |
| '4' | '14' | '0034' | '164' | '404' | 'A4' |
| '5' | '15' | '0035' | '165' | '405' | 'A5' |
| '6' | '16' | '0036' | '166' | '406' | 'A6' |
| '7' | '17' | '0037' | '167' | '407' | 'A7' |
| '8' | '20' | '0038' | '170' | '410' | 'A8' |
| '9' | '21' | '0039' | '171' | '411' | 'A9' |
| '.' | '22' | '003A' | '172' | '412' | 'AA' |
| '.' | '23' | '003B' | '173' | '413' | 'AB' |
| '.' | '24' | '003C' | '174' | '414' | 'AC' |
| '.' | '25' | '003D' | '175' | '415' | 'AD' |
| '.' | '26' | '003E' | '176' | '416' | 'AE' |
| '.' | '27' | '003F' | '177' | '417' | 'AF' |
| '.' | '30' | '0040' | '180' | '420' | 'B0' |
| '.' | '31' | '0041' | '181' | '421' | 'B1' |
| '.' | '32' | '0042' | '182' | '422' | 'B2' |
| '.' | '33' | '0043' | '183' | '423' | 'B3' |
| '.' | '34' | '0044' | '184' | '424' | 'B4' |
| '.' | '35' | '0045' | '185' | '425' | 'B5' |
| '.' | '36' | '0046' | '186' | '426' | 'B6' |
| '.' | '37' | '0047' | '187' | '427' | 'B7' |
| '.' | '40' | '0048' | '188' | '430' | 'B8' |
| '.' | '41' | '0049' | '189' | '431' | 'B9' |
| '.' | '42' | '004A' | '190' | '432' | 'BA' |
| '.' | '43' | '004B' | '191' | '433' | 'BB' |
| '.' | '44' | '004C' | '192' | '434' | 'BC' |
| '.' | '45' | '004D' | '193' | '435' | 'BD' |
| '.' | '46' | '004E' | '194' | '436' | 'BE' |
| '.' | '47' | '004F' | '195' | '437' | 'BF' |
| '.' | '50' | '0050' | '196' | '440' | 'C0' |
| '.' | '51' | '0051' | '197' | '441' | 'C1' |
| '.' | '52' | '0052' | '198' | '442' | 'C2' |
| '.' | '53' | '0053' | '199' | '443' | 'C3' |
| '.' | '54' | '0054' | '200' | '444' | 'C4' |
| '.' | '55' | '0055' | '201' | '445' | 'C5' |
| '.' | '56' | '0056' | '202' | '446' | 'C6' |
| '.' | '57' | '0057' | '203' | '447' | 'C7' |
| '.' | '60' | '0058' | '204' | '450' | 'C8' |
| '.' | '61' | '0059' | '205' | '451' | 'C9' |
| '.' | '62' | '005A' | '206' | '452' | 'CA' |
| '.' | '63' | '005B' | '207' | '453' | 'CB' |
| '.' | '64' | '005C' | '208' | '454' | 'CC' |
| '.' | '65' | '005D' | '209' | '455' | 'CD' |
| '.' | '66' | '005E' | '210' | '456' | 'CE' |
| '.' | '67' | '005F' | '211' | '457' | 'CF' |
| '.' | '70' | '0060' | '212' | '460' | 'D0' |
| '.' | '71' | '0061' | '213' | '461' | 'D1' |
| '.' | '72' | '0062' | '214' | '462' | 'D2' |
| '.' | '73' | '0063' | '215' | '463' | 'D3' |
| '.' | '74' | '0064' | '216' | '464' | 'D4' |
| '.' | '75' | '0065' | '217' | '465' | 'D5' |
| '.' | '76' | '0066' | '218' | '466' | 'D6' |
| '.' | '77' | '0067' | '219' | '467' | 'D7' |
| '.' | '80' | '0068' | '220' | '470' | 'E0' |
| '.' | '81' | '0069' | '221' | '471' | 'E1' |
| '.' | '82' | '006A' | '222' | '472' | 'E2' |
| '.' | '83' | '006B' | '223' | '473' | 'E3' |
| '.' | '84' | '006C' | '224' | '474' | 'E4' |
| '.' | '85' | '006D' | '225' | '475' | 'E5' |
| '.' | '86' | '006E' | '226' | '476' | 'E6' |
| '.' | '87' | '006F' | '227' | '477' | 'E7' |
| '.' | '90' | '0070' | '228' | '480' | 'F0' |
| '.' | '91' | '0071' | '229' | '481' | 'F1' |
| '.' | '92' | '0072' | '230' | '482' | 'F2' |
| '.' | '93' | '0073' | '231' | '483' | 'F3' |
| '.' | '94' | '0074' | '232' | '484' | 'F4' |
| '.' | '95' | '0075' | '233' | '485' | 'F5' |
| '.' | '96' | '0076' | '234' | '486' | 'F6' |
| '.' | '97' | '0077' | '235' | '487' | 'F7' |
| '.' | '98' | '0078' | '236' | '488' | 'F8' |
| '.' | '99' | '0079' | '237' | '489' | 'F9' |
| '.' | '9A' | '007A' | '238' | '490' | 'FA' |
| '.' | '9B' | '007B' | '239' | '491' | 'FB' |
| '.' | '9C' | '007C' | '240' | '492' | 'FC' |
| '.' | '9D' | '007D' | '241' | '493' | 'FD' |
| '.' | '9E' | '007E' | '242' | '494' | 'FE' |
| '.' | '9F' | '007F' | '243' | '495' | 'FF' |
| '.' | 'A0' | '0080' | '244' | '496' | 'FF' |
| '.' | 'A1' | '0081' | '245' | '497' | 'FF' |
| '.' | 'A2' | '0082' | '246' | '498' | 'FF' |
| '.' | 'A3' | '0083' | '247' | '499' | 'FF' |
| '.' | 'A4' | '0084' | '248' | '4A0' | 'FF' |
| '.' | 'A5' | '0085' | '249' | '4A1' | 'FF' |
| '.' | 'A6' | '0086' | '250' | '4A2' | 'FF' |
| '.' | 'A7' | '0087' | '251' | '4A3' | 'FF' |
| '.' | 'A8' | '0088' | '252' | '4A4' | 'FF' |
| '.' | 'A9' | '0089' | '253' | '4A5' | 'FF' |
| '.' | 'AA' | '008A' | '254' | '4A6' | 'FF' |
| '.' | 'AB' | '008B' | '255' | '4A7' | 'FF' |
| '.' | 'AC' | '008C' | '256' | '4A8' | 'FF' |
| '.' | 'AD' | '008D' | '257' | '4A9' | 'FF' |
| '.' | 'AE' | '008E' | '258' | '4AA' | 'FF' |
| '.' | 'AF' | '008F' | '259' | '4AB' | 'FF' |
| '.' | 'B0' | '0090' | '260' | '4AC' | 'FF' |
| '.' | 'B1' | '0091' | '261' | '4AD' | 'FF' |
| '.' | 'B2' | '0092' | '262' | '4AE' | 'FF' |
| '.' | 'B3' | '0093' | '263' | '4AF' | 'FF' |
| '.' | 'B4' | '0094' | '264' | '4B0' | 'FF' |
| '.' | 'B5' | '0095' | '265' | '4B1' | 'FF' |
| '.' | 'B6' | '0096' | '266' | '4B2' | 'FF' |
| '.' | 'B7' | '0097' | '267' | '4B3' | 'FF' |
| '.' | 'C0' | '0098' | '268' | '4B4' | 'FF' |
| '.' | 'C1' | '0099' | '269' | '4B5' | 'FF' |
| '.' | 'C2' | '009A' | '270' | '4B6' | 'FF' |
| '.' | 'C3' | '009B' | '271' | '4B7' | 'FF' |
| '.' | 'C4' | '009C' | '272' | '4B8' | 'FF' |
| '.' | 'C5' | '009D' | '273' | '4B9' | 'FF' |
| '.' | 'C6' | '009E' | '274' | '4BA' | 'FF' |
| '.' | 'C7' | '009F' | '275' | '4BAA' | 'FF' |
| '.' | 'D0' | '00A0' | '276' | '4BAA' | 'FF' |
| '.' | 'D1' | '00A1' | '277' | '4BAA' | 'FF' |
| '.' | 'D2' | '00A2' | '278' | '4BAA' | 'FF' |
| '.' | 'D3' | '00A3' | '279' | '4BAA' | 'FF' |
| '.' | 'D4' | '00A4' | '27A' | '4BAA' | 'FF' |
| '.' | 'D5' | '00A5' | '27B' | '4BAA' | 'FF' |
| '.' | 'D6' | '00A6' | '27C' | '4BAA' | 'FF' |
| '.' | 'D7' | '00A7' | '27D' | '4BAA' | 'FF' |
| '.' | 'E0' | '00A8' | '27E' | '4BAA' | 'FF' |
| '.' | 'E1' | '00A9' | '27F' | '4BAA' | 'FF' |
| '.' | 'E2' | '00AA' | '280' | '4BAA' | 'FF' |
| '.' | 'E3' | '00AB' | '281' | '4BAA' | 'FF' |
| '.' | 'E4' | '00AC' | '282' | '4BAA' | 'FF' |
| '.' | 'E5' | '00AD' | '283' | '4BAA' | 'FF' |
| '.' | 'E6' | '00AE' | '284' | '4BAA' | 'FF' |
| '.' | 'E7' | '00AF' | '285' | '4BAA' | 'FF' |
| '.' | 'F0' | '00A0' | '286' | '4BAA' | 'FF' |
| '.' | 'F1' | '00A1' | '287' | '4BAA' | 'FF' |
| '.' | 'F2' | '00A2' | '288' | '4BAA' | 'FF' |
| '.' | 'F3' | '00A3' | '289' | '4BAA' | 'FF' |
| '.' | 'F4' | '00A4' | '28A' | '4BAA' | 'FF' |
| '.' | 'F5' | '00A5' | '28B' | '4BAA' | 'FF' |
| '.' | 'F6' | '00A6' | '28C' | '4BAA' | 'FF' |
| '.' | 'F7' | '00A7' | '28D' | '4BAA' | 'FF' |
| '.' | 'FA' | '00A8' | '28E' | '4BAA' | 'FF' |
| '.' | 'FB' | '00A9' | '28F' | '4BAA' | 'FF' |
| '.' | 'FC' | '00AA' | '290' | '4BAA' | 'FF' |
| '.' | 'FD' | '00AB' | '291' | '4BAA' | 'FF' |
| '.' | 'FE' | '00AC' | '292' | '4BAA' | 'FF' |
| '.' | 'FF' | '00AD' | '293' | '4BAA' | 'FF' |
| '.' | 'FF' | '00AE' | '294' | '4BAA' | 'FF' |
| '.' | 'FF' | '00AF' | '295' | '4BAA' | 'FF' |

- `ord(c)`: returns Unicode (ASCII) of the character.

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

| Octal | Hex | Char | Octal | Hex | Char | Octal | Hex | Char |
|-------|-----|------|-------|-----|------|-------|-----|------|
| 000 | 00 | | 030 | 22 |   | 060 | 38 |   |
| 001 | 01 |   | 031 | 23 |   | 061 | 39 |   |
| 002 | 02 |   | 032 | 24 |   | 062 | 3A |   |
| 003 | 03 |   | 033 | 25 |   | 063 | 3B |   |
| 004 | 04 |   | 034 | 26 |   | 064 | 3C |   |
| 005 | 05 |   | 035 | 27 |   | 065 | 3D |   |
| 006 | 06 |   | 036 | 28 |   | 066 | 3E |   |
| 007 | 07 |   | 037 | 29 |   | 067 | 3F |   |
| 010 | 08 |   | 040 | 30 |   | 070 | 48 |   |
| 011 | 09 |   | 041 | 31 |   | 071 | 49 |   |
| 012 | 0A |   | 042 | 32 |   | 072 | 4A |   |
| 013 | 0B |   | 043 | 33 |   | 073 | 4B |   |
| 014 | 0C |   | 044 | 34 |   | 074 | 4C |   |
| 015 | 0D |   | 045 | 35 |   | 075 | 4D |   |
| 016 | 0E |   | 046 | 36 |   | 076 | 4E |   |
| 017 | 0F |   | 047 | 37 |   | 077 | 4F |   |
| 020 | 10 |   | 050 | 50 |   | 080 | 58 |   |
| 021 | 11 |   | 051 | 51 |   | 081 | 59 |   |
| 022 | 12 |   | 052 | 52 |   | 082 | 5A |   |
| 023 | 13 |   | 053 | 53 |   | 083 | 5B |   |
| 024 | 14 |   | 054 | 54 |   | 084 | 5C |   |
| 025 | 15 |   | 055 | 55 |   | 085 | 5D |   |
| 026 | 16 |   | 056 | 56 |   | 086 | 5E |   |
| 027 | 17 |   | 057 | 57 |   | 087 | 5F |   |
| 030 | 20 |   | 060 | 60 |   | 090 | 68 |   |
| 031 | 21 |   | 061 | 61 |   | 091 | 69 |   |
| 032 | 22 |   | 062 | 62 |   | 092 | 6A |   |
| 033 | 23 |   | 063 | 63 |   | 093 | 6B |   |
| 034 | 24 |   | 064 | 64 |   | 094 | 6C |   |
| 035 | 25 |   | 065 | 65 |   | 095 | 6D |   |
| 036 | 26 |   | 066 | 66 |   | 096 | 6E |   |
| 037 | 27 |   | 067 | 67 |   | 097 | 6F |   |
| 040 | 30 |   | 070 | 70 |   | 098 | 78 |   |
| 041 | 31 |   | 071 | 71 |   | 099 | 79 |   |
| 042 | 32 |   | 072 | 72 |   | 100 | 7A |   |
| 043 | 33 |   | 073 | 73 |   | | | |
| 044 | 34 |   | 074 | 74 |   | | | |
| 045 | 35 |   | 075 | 75 |   | | | |
| 046 | 36 |   | 076 | 76 |   | | | |
| 047 | 37 |   | 077 | 77 |   | | | |
| 050 | 50 |   | 080 | 58 |   | | | |
| 051 | 51 |   | 081 | 59 |   | | | |
| 052 | 52 |   | 082 | 5A |   | | | |
| 053 | 53 |   | 083 | 5B |   | | | |
| 054 | 54 |   | 084 | 5C |   | | | |
| 055 | 55 |   | 085 | 5D |   | | | |
| 056 | 56 |   | 086 | 5E |   | | | |
| 057 | 57 |   | 087 | 5F |   | | | |
| 060 | 60 |   | 090 | 68 |   | | | |
| 061 | 61 |   | 091 | 69 |   | | | |
| 062 | 62 |   | 092 | 6A |   | | | |
| 063 | 63 |   | 093 | 6B |   | | | |
| 064 | 64 |   | 094 | 6C |   | | | |
| 065 | 65 |   | 095 | 6D |   | | | |
| 066 | 66 |   | 096 | 6E |   | | | |
| 067 | 67 |   | 097 | 6F |   | | | |
| 070 | 70 |   | 098 | 78 |   | | | |
| 071 | 71 |   | 099 | 79 |   | | | |
| 072 | 72 |   | 100 | 7A |   | | | |

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

| Decimal Num Char | Octal Num Char | Hex Num Char | Decimal Num Char | Octal Num Char | Hex Num Char |
|------------------|----------------|--------------|------------------|----------------|--------------|
| '\000' | '000' | '000' | '\001' | '001' | '001' |
| '\002' | '002' | '002' | '\003' | '003' | '003' |
| '\004' | '004' | '004' | '\005' | '005' | '005' |
| '\006' | '006' | '006' | '\007' | '007' | '007' |
| '\010' | '010' | '00A' | '\011' | '011' | '00B' |
| '\012' | '012' | '00C' | '\013' | '013' | '00D' |
| '\014' | '014' | '00E' | '\015' | '015' | '00F' |
| '\016' | '016' | '010' | '\017' | '017' | '011' |
| '\020' | '020' | '012' | '\021' | '021' | '013' |
| '\022' | '022' | '014' | '\023' | '023' | '015' |
| '\024' | '024' | '016' | '\025' | '025' | '017' |
| '\026' | '026' | '018' | '\027' | '027' | '019' |
| '\030' | '030' | '01A' | '\031' | '031' | '01B' |
| '\032' | '032' | '01C' | '\033' | '033' | '01D' |
| '\034' | '034' | '01E' | '\035' | '035' | '01F' |
| '\036' | '036' | '020' | '\037' | '037' | '021' |
| '\040' | '040' | '022' | '\041' | '041' | '023' |
| '\042' | '042' | '024' | '\043' | '043' | '025' |
| '\044' | '044' | '026' | '\045' | '045' | '027' |
| '\046' | '046' | '028' | '\047' | '047' | '029' |
| '\048' | '048' | '02A' | '\049' | '049' | '02B' |
| '\050' | '050' | '02C' | '\051' | '051' | '02D' |
| '\052' | '052' | '02E' | '\053' | '053' | '02F' |
| '\054' | '054' | '030' | '\055' | '055' | '031' |
| '\056' | '056' | '032' | '\057' | '057' | '033' |
| '\060' | '060' | '034' | '\061' | '061' | '035' |
| '\062' | '062' | '036' | '\063' | '063' | '037' |
| '\064' | '064' | '038' | '\065' | '065' | '039' |
| '\066' | '066' | '03A' | '\067' | '067' | '03B' |
| '\068' | '068' | '03C' | '\069' | '069' | '03D' |
| '\070' | '070' | '03E' | '\071' | '071' | '03F' |
| '\072' | '072' | '040' | '\073' | '073' | '041' |
| '\074' | '074' | '042' | '\075' | '075' | '043' |
| '\076' | '076' | '044' | '\077' | '077' | '045' |
| '\080' | '080' | '048' | '\081' | '081' | '049' |
| '\082' | '082' | '04A' | '\083' | '083' | '04B' |
| '\084' | '084' | '04C' | '\085' | '085' | '04D' |
| '\086' | '086' | '04E' | '\087' | '087' | '04F' |
| '\088' | '088' | '050' | '\089' | '089' | '051' |
| '\090' | '090' | '052' | '\091' | '091' | '053' |
| '\092' | '092' | '054' | '\093' | '093' | '055' |
| '\094' | '094' | '056' | '\095' | '095' | '057' |
| '\096' | '096' | '058' | '\097' | '097' | '059' |
| '\098' | '098' | '05A' | '\099' | '099' | '05B' |
| '\09A' | '09A' | '05C' | '\09B' | '09B' | '05D' |
| '\09C' | '09C' | '05E' | '\09D' | '09D' | '05F' |
| '\09E' | '09E' | '060' | '\09F' | '09F' | '061' |
| '\0A0' | '0A0' | '062' | '\0A1' | '0A1' | '063' |
| '\0A2' | '0A2' | '064' | '\0A3' | '0A3' | '065' |
| '\0A4' | '0A4' | '066' | '\0A5' | '0A5' | '067' |
| '\0A6' | '0A6' | '068' | '\0A7' | '0A7' | '069' |
| '\0A8' | '0A8' | '06A' | '\0A9' | '0A9' | '06B' |
| '\0AA' | '0AA' | '06C' | '\0AB' | '0AB' | '06D' |
| '\0AC' | '0AC' | '06E' | '\0AD' | '0AD' | '06F' |
| '\0AE' | '0AE' | '070' | '\0AF' | '0AF' | '071' |
| '\0B0' | '0B0' | '072' | '\0B1' | '0B1' | '073' |
| '\0B2' | '0B2' | '074' | '\0B3' | '0B3' | '075' |
| '\0B4' | '0B4' | '076' | '\0B5' | '0B5' | '077' |
| '\0B6' | '0B6' | '078' | '\0B7' | '0B7' | '079' |
| '\0B8' | '0B8' | '07A' | '\0B9' | '0B9' | '07B' |
| '\0BA' | '0BA' | '07C' | '\0BB' | '0BB' | '07D' |
| '\0BC' | '0BC' | '07E' | '\0BD' | '0BD' | '07F' |
| '\0BE' | '0BE' | '080' | '\0BF' | '0BF' | '081' |
| '\0C0' | '0C0' | '082' | '\0C1' | '0C1' | '083' |
| '\0C2' | '0C2' | '084' | '\0C3' | '0C3' | '085' |
| '\0C4' | '0C4' | '086' | '\0C5' | '0C5' | '087' |
| '\0C6' | '0C6' | '088' | '\0C7' | '0C7' | '089' |
| '\0C8' | '0C8' | '08A' | '\0C9' | '0C9' | '08B' |
| '\0CA' | '0CA' | '08C' | '\0CB' | '0CB' | '08D' |
| '\0CC' | '0CC' | '08E' | '\0CD' | '0CD' | '08F' |
| '\0CE' | '0CE' | '090' | '\0CF' | '0CF' | '091' |
| '\0D0' | '0D0' | '092' | '\0D1' | '0D1' | '093' |
| '\0D2' | '0D2' | '094' | '\0D3' | '0D3' | '095' |
| '\0D4' | '0D4' | '096' | '\0D5' | '0D5' | '097' |
| '\0D6' | '0D6' | '098' | '\0D7' | '0D7' | '099' |
| '\0D8' | '0D8' | '09A' | '\0D9' | '0D9' | '09B' |
| '\0DA' | '0DA' | '09C' | '\0DB' | '0DB' | '09D' |
| '\0DC' | '0DC' | '09E' | '\0DD' | '0DD' | '09F' |
| '\0DE' | '0DE' | '0A0' | '\0EF' | '0EF' | '0A1' |
| '\0F0' | '0F0' | '0A2' | '\0F1' | '0F1' | '0A3' |
| '\0F2' | '0F2' | '0A4' | '\0F3' | '0F3' | '0A5' |
| '\0F4' | '0F4' | '0A6' | '\0F5' | '0F5' | '0A7' |
| '\0F6' | '0F6' | '0A8' | '\0F7' | '0F7' | '0A9' |
| '\0F8' | '0F8' | '0AA' | '\0F9' | '0F9' | '0AB' |
| '\0FA' | '0FA' | '0AC' | '\0FB' | '0FB' | '0AD' |
| '\0FC' | '0FC' | '0AE' | '\0FD' | '0FD' | '0BD' |
| '\0FE' | '0FE' | '0C0' | '\0FF' | '0FF' | '0C1' |

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is `x`.

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

| Decimal Num Char | Octal Num Char | Hex Num Char | Decimal Num Char | Octal Num Char | Hex Num Char |
|------------------|----------------|--------------|------------------|----------------|--------------|
| '\000' | '000' | '000' | '\001' | '001' | '001' |
| '\002' | '002' | '002' | '\003' | '003' | '003' |
| '\004' | '004' | '004' | '\005' | '005' | '005' |
| '\006' | '006' | '006' | '\007' | '007' | '007' |
| '\010' | '010' | '00A' | '\011' | '011' | '00B' |
| '\012' | '012' | '00C' | '\013' | '013' | '00D' |
| '\014' | '014' | '00E' | '\015' | '015' | '00F' |
| '\016' | '016' | '010' | '\017' | '017' | '011' |
| '\020' | '020' | '012' | '\021' | '021' | '013' |
| '\022' | '022' | '014' | '\023' | '023' | '015' |
| '\024' | '024' | '016' | '\025' | '025' | '017' |
| '\026' | '026' | '018' | '\027' | '027' | '019' |
| '\030' | '030' | '01A' | '\031' | '031' | '01B' |
| '\032' | '032' | '01C' | '\033' | '033' | '01D' |
| '\034' | '034' | '01E' | '\035' | '035' | '01F' |
| '\036' | '036' | '020' | '\037' | '037' | '021' |
| '\040' | '040' | '022' | '\041' | '041' | '023' |
| '\042' | '042' | '024' | '\043' | '043' | '025' |
| '\044' | '044' | '026' | '\045' | '045' | '027' |
| '\046' | '046' | '028' | '\047' | '047' | '029' |
| '\048' | '048' | '02A' | '\049' | '049' | '02B' |
| '\050' | '050' | '02C' | '\051' | '051' | '02D' |
| '\052' | '052' | '02E' | '\053' | '053' | '02F' |
| '\054' | '054' | '030' | '\055' | '055' | '031' |
| '\056' | '056' | '032' | '\057' | '057' | '033' |
| '\060' | '060' | '034' | '\061' | '061' | '035' |
| '\062' | '062' | '036' | '\063' | '063' | '037' |
| '\064' | '064' | '038' | '\065' | '065' | '039' |
| '\066' | '066' | '03A' | '\067' | '067' | '03B' |
| '\068' | '068' | '03C' | '\069' | '069' | '03D' |
| '\070' | '070' | '03E' | '\071' | '071' | '03F' |
| '\072' | '072' | '040' | '\073' | '073' | '041' |
| '\074' | '074' | '042' | '\075' | '075' | '043' |
| '\076' | '076' | '044' | '\077' | '077' | '045' |
| '\080' | '080' | '048' | '\081' | '081' | '049' |
| '\082' | '082' | '04A' | '\083' | '083' | '04B' |
| '\084' | '084' | '04C' | '\085' | '085' | '04D' |
| '\086' | '086' | '04E' | '\087' | '087' | '04F' |
| '\088' | '088' | '050' | '\089' | '089' | '051' |
| '\090' | '090' | '052' | '\091' | '091' | '053' |
| '\092' | '092' | '054' | '\093' | '093' | '055' |
| '\094' | '094' | '056' | '\095' | '095' | '057' |
| '\096' | '096' | '058' | '\097' | '097' | '059' |
| '\098' | '098' | '05A' | '\099' | '099' | '05B' |
| '\09A' | '09A' | '05C' | '\09B' | '09B' | '05D' |
| '\09C' | '09C' | '05E' | '\09D' | '09D' | '05F' |
| '\09E' | '09E' | '060' | '\09F' | '09F' | '061' |
| '\0A0' | '0A0' | '062' | '\0A1' | '0A1' | '063' |
| '\0A2' | '0A2' | '064' | '\0A3' | '0A3' | '065' |
| '\0A4' | '0A4' | '066' | '\0A5' | '0A5' | '067' |
| '\0A6' | '0A6' | '068' | '\0A7' | '0A7' | '069' |
| '\0A8' | '0A8' | '06A' | '\0A9' | '0A9' | '06B' |
| '\0AA' | '0AA' | '06C' | '\0AB' | '0AB' | '06D' |
| '\0AC' | '0AC' | '06E' | '\0AD' | '0AD' | '06F' |
| '\0AE' | '0AE' | '070' | '\0AF' | '0AF' | '071' |
| '\0B0' | '0B0' | '072' | '\0B1' | '0B1' | '073' |
| '\0B2' | '0B2' | '074' | '\0B3' | '0B3' | '075' |
| '\0B4' | '0B4' | '076' | '\0B5' | '0B5' | '077' |
| '\0B6' | '0B6' | '078' | '\0B7' | '0B7' | '079' |
| '\0B8' | '0B8' | '07A' | '\0B9' | '0B9' | '07B' |
| '\0BA' | '0BA' | '07C' | '\0BB' | '0BB' | '07D' |
| '\0BC' | '0BC' | '07E' | '\0BD' | '0BD' | '07F' |
| '\0BE' | '0BE' | '080' | '\0BF' | '0BF' | '081' |
| '\0C0' | '0C0' | '082' | '\0C1' | '0C1' | '083' |
| '\0C2' | '0C2' | '084' | '\0C3' | '0C3' | '085' |
| '\0C4' | '0C4' | '086' | '\0C5' | '0C5' | '087' |
| '\0C6' | '0C6' | '088' | '\0C7' | '0C7' | '089' |
| '\0C8' | '0C8' | '08A' | '\0C9' | '0C9' | '08B' |
| '\0CA' | '0CA' | '08C' | '\0CB' | '0CB' | '08D' |
| '\0CC' | '0CC' | '08E' | '\0CD' | '0CD' | '08F' |
| '\0CE' | '0CE' | '090' | '\0CF' | '0CF' | '091' |
| '\0D0' | '0D0' | '092' | '\0D1' | '0D1' | '093' |
| '\0D2' | '0D2' | '094' | '\0D3' | '0D3' | '095' |
| '\0D4' | '0D4' | '096' | '\0D5' | '0D5' | '097' |
| '\0D6' | '0D6' | '098' | '\0D7' | '0D7' | '099' |
| '\0D8' | '0D8' | '09A' | '\0D9' | '0D9' | '09B' |
| '\0DA' | '0DA' | '09C' | '\0DB' | '0DB' | '09D' |
| '\0DC' | '0DC' | '09E' | '\0DD' | '0DD' | '09F' |
| '\0DE' | '0DE' | '0A0' | '\0EF' | '0EF' | '0A1' |
| '\0F0' | '0F0' | '0A2' | '\0F1' | '0F1' | '0A3' |
| '\0F2' | '0F2' | '0A4' | '\0F3' | '0F3' | '0A5' |
| '\0F4' | '0F4' | '0A6' | '\0F5' | '0F5' | '0A7' |
| '\0F6' | '0F6' | '0A8' | '\0F7' | '0F7' | '0A9' |
| '\0F8' | '0F8' | '0AA' | '\0F9' | '0F9' | '0AB' |
| '\0FA' | '0FA' | '0AC' | '\0FB' | '0FB' | '0AD' |
| '\0FC' | '0FC' | '0AE' | '\0FD' | '0FD' | '0BD' |
| '\0FE' | '0FE' | '0C0' | '\0FF' | '0FF' | '0C1' |

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.
- Example: `chr(97)` returns 'a'.

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

| Decimal Num Char | Octal Num Char | Hex Num Char | Decimal Num Char | Octal Num Char | Hex Num Char |
|------------------|----------------|--------------|------------------|----------------|--------------|
| ' ' | '0' | '0000' | '0' | '0000' | '0000' |
| '0' | '10' | '0030' | '1' | '1010' | '0031' |
| '1' | '20' | '0031' | '2' | '2020' | '0032' |
| '2' | '30' | '0032' | '3' | '3030' | '0033' |
| '3' | '40' | '0033' | '4' | '4040' | '0034' |
| '4' | '50' | '0034' | '5' | '5050' | '0035' |
| '5' | '60' | '0035' | '6' | '6060' | '0036' |
| '6' | '70' | '0036' | '7' | '7070' | '0037' |
| '8' | '100' | '0038' | '9' | '10100' | '0039' |
| 'A' | '120' | '0041' | 'B' | '12100' | '0042' |
| 'C' | '140' | '0043' | 'D' | '14100' | '0044' |
| 'E' | '160' | '0045' | 'F' | '16100' | '0046' |
| 'G' | '170' | '0047' | 'H' | '17100' | '0048' |
| 'I' | '17200' | '0049' | 'J' | '172100' | '004A' |
| 'K' | '172200' | '004B' | 'L' | '172300' | '004C' |
| 'M' | '172400' | '004D' | 'N' | '172500' | '004E' |
| 'P' | '172700' | '004F' | 'Q' | '172800' | '0050' |
| 'R' | '172900' | '0051' | 'S' | '172A00' | '0052' |
| 'T' | '172B00' | '0053' | 'U' | '172C00' | '0054' |
| 'V' | '172D00' | '0055' | 'W' | '172E00' | '0056' |
| 'X' | '172F00' | '0057' | 'Y' | '173000' | '0058' |
| 'Z' | '173100' | '0059' | '_' | '173200' | '005A' |
| '`' | '173300' | '005B' | '@' | '173400' | '005C' |
| '`' | '173500' | '005D' | '`' | '173600' | '005E' |
| '`' | '173700' | '005F' | '`' | '173800' | '005F' |
| '`' | '173900' | '005F' | '`' | '173A00' | '005F' |
| '`' | '173B00' | '005F' | '`' | '173C00' | '005F' |
| '`' | '173D00' | '005F' | '`' | '173E00' | '005F' |
| '`' | '173F00' | '005F' | '`' | '174000' | '005F' |
| '`' | '174100' | '005F' | '`' | '174200' | '005F' |
| '`' | '174300' | '005F' | '`' | '174400' | '005F' |
| '`' | '174500' | '005F' | '`' | '174600' | '005F' |
| '`' | '174700' | '005F' | '`' | '174800' | '005F' |
| '`' | '174900' | '005F' | '`' | '174A00' | '005F' |
| '`' | '174B00' | '005F' | '`' | '174C00' | '005F' |
| '`' | '174D00' | '005F' | '`' | '174E00' | '005F' |
| '`' | '174F00' | '005F' | '`' | '175000' | '005F' |
| '`' | '175100' | '005F' | '`' | '175200' | '005F' |
| '`' | '175300' | '005F' | '`' | '175400' | '005F' |
| '`' | '175500' | '005F' | '`' | '175600' | '005F' |
| '`' | '175700' | '005F' | '`' | '175800' | '005F' |
| '`' | '175900' | '005F' | '`' | '175A00' | '005F' |
| '`' | '175B00' | '005F' | '`' | '175C00' | '005F' |
| '`' | '175D00' | '005F' | '`' | '175E00' | '005F' |
| '`' | '175F00' | '005F' | '`' | '176000' | '005F' |
| '`' | '176100' | '005F' | '`' | '176200' | '005F' |
| '`' | '176300' | '005F' | '`' | '176400' | '005F' |
| '`' | '176500' | '005F' | '`' | '176600' | '005F' |
| '`' | '176700' | '005F' | '`' | '176800' | '005F' |
| '`' | '176900' | '005F' | '`' | '176A00' | '005F' |
| '`' | '176B00' | '005F' | '`' | '176C00' | '005F' |
| '`' | '176D00' | '005F' | '`' | '176E00' | '005F' |
| '`' | '176F00' | '005F' | '`' | '177000' | '005F' |
| '`' | '177100' | '005F' | '`' | '177200' | '005F' |
| '`' | '177300' | '005F' | '`' | '177400' | '005F' |
| '`' | '177500' | '005F' | '`' | '177600' | '005F' |
| '`' | '177700' | '005F' | '`' | '177800' | '005F' |
| '`' | '177900' | '005F' | '`' | '177A00' | '005F' |
| '`' | '177B00' | '005F' | '`' | '177C00' | '005F' |
| '`' | '177D00' | '005F' | '`' | '177E00' | '005F' |
| '`' | '177F00' | '005F' | '`' | '178000' | '005F' |
| '`' | '178100' | '005F' | '`' | '178200' | '005F' |
| '`' | '178300' | '005F' | '`' | '178400' | '005F' |
| '`' | '178500' | '005F' | '`' | '178600' | '005F' |
| '`' | '178700' | '005F' | '`' | '178800' | '005F' |
| '`' | '178900' | '005F' | '`' | '178A00' | '005F' |
| '`' | '178B00' | '005F' | '`' | '178C00' | '005F' |
| '`' | '178D00' | '005F' | '`' | '178E00' | '005F' |
| '`' | '178F00' | '005F' | '`' | '179000' | '005F' |
| '`' | '179100' | '005F' | '`' | '179200' | '005F' |
| '`' | '179300' | '005F' | '`' | '179400' | '005F' |
| '`' | '179500' | '005F' | '`' | '179600' | '005F' |
| '`' | '179700' | '005F' | '`' | '179800' | '005F' |
| '`' | '179900' | '005F' | '`' | '179A00' | '005F' |
| '`' | '179B00' | '005F' | '`' | '179C00' | '005F' |
| '`' | '179D00' | '005F' | '`' | '179E00' | '005F' |
| '`' | '179F00' | '005F' | '`' | '17A000' | '005F' |
| '`' | '17A100' | '005F' | '`' | '17A200' | '005F' |
| '`' | '17A300' | '005F' | '`' | '17A400' | '005F' |
| '`' | '17A500' | '005F' | '`' | '17A600' | '005F' |
| '`' | '17A700' | '005F' | '`' | '17A800' | '005F' |
| '`' | '17A900' | '005F' | '`' | '17AA00' | '005F' |
| '`' | '17AB00' | '005F' | '`' | '17AC00' | '005F' |
| '`' | '17AD00' | '005F' | '`' | '17AE00' | '005F' |
| '`' | '17AF00' | '005F' | '`' | '17B000' | '005F' |
| '`' | '17B100' | '005F' | '`' | '17B200' | '005F' |
| '`' | '17B300' | '005F' | '`' | '17B400' | '005F' |
| '`' | '17B500' | '005F' | '`' | '17B600' | '005F' |
| '`' | '17B700' | '005F' | '`' | '17B800' | '005F' |
| '`' | '17B900' | '005F' | '`' | '17BA00' | '005F' |
| '`' | '17BB00' | '005F' | '`' | '17BC00' | '005F' |
| '`' | '17BD00' | '005F' | '`' | '17BE00' | '005F' |
| '`' | '17BF00' | '005F' | '`' | '17C000' | '005F' |
| '`' | '17C100' | '005F' | '`' | '17C200' | '005F' |
| '`' | '17C300' | '005F' | '`' | '17C400' | '005F' |
| '`' | '17C500' | '005F' | '`' | '17C600' | '005F' |
| '`' | '17C700' | '005F' | '`' | '17C800' | '005F' |
| '`' | '17C900' | '005F' | '`' | '17CA00' | '005F' |
| '`' | '17CB00' | '005F' | '`' | '17CC00' | '005F' |
| '`' | '17CD00' | '005F' | '`' | '17CE00' | '005F' |
| '`' | '17CF00' | '005F' | '`' | '17D000' | '005F' |
| '`' | '17D100' | '005F' | '`' | '17D200' | '005F' |
| '`' | '17D300' | '005F' | '`' | '17D400' | '005F' |
| '`' | '17D500' | '005F' | '`' | '17D600' | '005F' |
| '`' | '17D700' | '005F' | '`' | '17D800' | '005F' |
| '`' | '17D900' | '005F' | '`' | '17DA00' | '005F' |
| '`' | '17DB00' | '005F' | '`' | '17DC00' | '005F' |
| '`' | '17DD00' | '005F' | '`' | '17DE00' | '005F' |
| '`' | '17DF00' | '005F' | '`' | '17E000' | '005F' |
| '`' | '17E100' | '005F' | '`' | '17E200' | '005F' |
| '`' | '17E300' | '005F' | '`' | '17E400' | '005F' |
| '`' | '17E500' | '005F' | '`' | '17E600' | '005F' |
| '`' | '17E700' | '005F' | '`' | '17E800' | '005F' |
| '`' | '17E900' | '005F' | '`' | '17EA00' | '005F' |
| '`' | '17EB00' | '005F' | '`' | '17EC00' | '005F' |
| '`' | '17ED00' | '005F' | '`' | '17EE00' | '005F' |
| '`' | '17EF00' | '005F' | '`' | '17F000' | '005F' |
| '`' | '17F100' | '005F' | '`' | '17F200' | '005F' |
| '`' | '17F300' | '005F' | '`' | '17F400' | '005F' |
| '`' | '17F500' | '005F' | '`' | '17F600' | '005F' |
| '`' | '17F700' | '005F' | '`' | '17F800' | '005F' |
| '`' | '17F900' | '005F' | '`' | '17FA00' | '005F' |
| '`' | '17FB00' | '005F' | '`' | '17FC00' | '005F' |
| '`' | '17FD00' | '005F' | '`' | '17FE00' | '005F' |
| '`' | '17FF00' | '005F' | '`' | '17F000' | '005F' |

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.
- Example: `chr(97)` returns 'a'.
- What is `chr(33)`?

In Pairs or Triples...

Some review and some novel challenges:

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



Python Tutor

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

(Demo with pythonTutor)

User Input

Covered in detail in Lab 2:

```
→ 1 mess = input('Please enter a message: ')
  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`.

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- **Strings**
- CS Survey (Dr. Sakas, Computational Linguistics)

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: 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~~X~~Saturday~~X~~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.

```
"FridayXSaturdayXSunday"  
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.

```
"FridayXSaturdayXSunday"  
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.

```
"FridayXSaturdayXSunday"  
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")
```

```
"FriXXXsSaturdayXXXsSunday"
```

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")
```

```
"FriXXXySaturXXXySunXXX"  
days = ['Fri', 'sSatur', 'sSun']
```

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- Strings
- **CS Survey (Dr. Sakas, Computational Linguistics)**

CS Survey: Prof. Sakas, Computational Computational Linguistics



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

CS Survey: Prof. Sakas, Computational Linguistics



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?*

CS Survey: Prof. Sakas, Computational Linguistics



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!

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”.

Recap

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

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

Recap

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

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

Recap

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

► For-loops

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

Recap

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

- ▶ For-loops
- ▶ range()

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

Recap

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

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

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings

Recap

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

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

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic

Recap

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

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

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation

Recap

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

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

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation
- ▶ Functions: `ord()` and `char()`

Recap

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

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

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation
- ▶ Functions: `ord()` and `char()`
- ▶ String Manipulation

Recap

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

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

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation
- ▶ Functions: `ord()` and `char()`
- ▶ String Manipulation

- 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 Spring 2018, Mock Exam.

Writing Boards



- Return writing boards as you leave...