

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)*

- 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)*

- 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)*

- 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)

- 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)*

- 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)

- 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)*

- 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)

- 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)*

- 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)

- 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)*

- 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)

- 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)*

- 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)

- 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)*

- 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)

- 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.*

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:

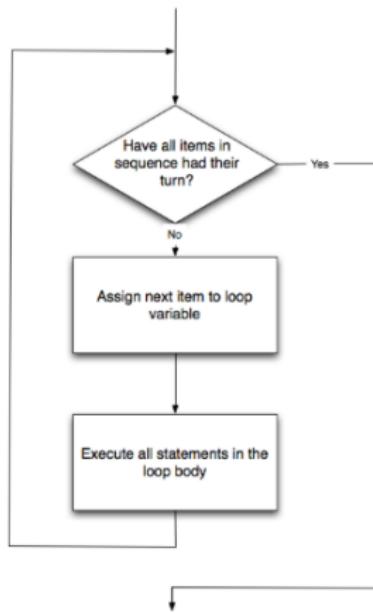
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11        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)

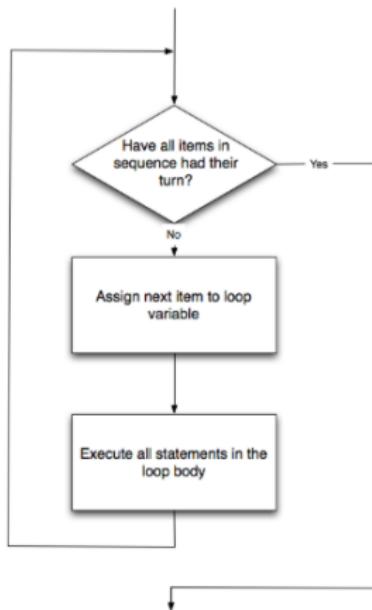
for-loop



```
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(sum)  
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(sum)  
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.)

ASCII TABLE														
Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char
0	00	0	NULL	\0	32	20	40	SIGKILL	\000	64	40	100	SIGPOLL	\001
1	01	1	SOH	\001	33	21	41	SIGALRM	\002	65	41	101	SIGSTOP	\003
2	02	2	STX	\002	34	22	42	SIGPOLL	\004	66	42	102	SIGCONT	\005
3	03	3	ETX	\003	35	23	43	SIGPOLL	\006	67	43	103	SIGKILL	\007
4	04	4	ENQ	\004	36	24	44	SIGPOLL	\008	68	44	104	SIGPOLL	\009
5	05	5	KSYN	\005	37	25	45	SIGPOLL	\010	69	45	105	SIGPOLL	\011
6	06	6	ACK	\006	38	26	46	SIGPOLL	\012	70	46	106	SIGPOLL	\013
7	07	7	NAK	\007	39	27	47	SIGPOLL	\014	71	47	107	SIGPOLL	\015
8	08	10	SYN	\008	40	28	48	SIGPOLL	\016	72	48	108	SIGPOLL	\017
9	09	11	BT	\009	41	29	49	SIGPOLL	\018	73	49	109	SIGPOLL	\019
10	0A	12	HT	\00A	42	2A	4A	SIGPOLL	\01A	74	4A	110	SIGPOLL	\01B
11	0B	13	LF	\00B	43	2B	4B	SIGPOLL	\01B	75	4B	111	SIGPOLL	\01C
12	0C	14	VT	\00C	44	2C	4C	SIGPOLL	\01C	76	4C	112	SIGPOLL	\01D
13	0D	15	FF	\00D	45	2D	4D	SIGPOLL	\01D	77	4D	113	SIGPOLL	\01E
14	0E	16	CR	\00E	46	2E	4E	SIGPOLL	\01E	78	4E	114	SIGPOLL	\01F
15	0F	17	SO	\00F	47	2F	4F	SIGPOLL	\01F	79	4F	115	SIGPOLL	\01F
16	10	20	SI	\010	48	30	50	SIGPOLL	\01F	80	50	116	SIGPOLL	\01F
17	11	21	DC1	\011	49	31	51	SIGPOLL	\01F	81	51	117	SIGPOLL	\01F
18	12	22	DC2	\012	50	32	52	SIGPOLL	\01F	82	52	118	SIGPOLL	\01F
19	13	23	DC3	\013	51	33	53	SIGPOLL	\01F	83	53	119	SIGPOLL	\01F
20	14	24	DC4	\014	52	34	54	SIGPOLL	\01F	84	54	120	SIGPOLL	\01F
21	15	25	NAK	\015	53	35	55	SIGPOLL	\01F	85	55	121	SIGPOLL	\01F
22	16	26	SYN	\016	54	36	56	SIGPOLL	\01F	86	56	122	SIGPOLL	\01F
23	17	27	ETX	\017	55	37	57	SIGPOLL	\01F	87	57	123	SIGPOLL	\01F
24	18	28	ENQ	\018	56	38	58	SIGPOLL	\01F	88	58	124	SIGPOLL	\01F
25	19	29	ACK	\019	57	39	59	SIGPOLL	\01F	89	59	125	SIGPOLL	\01F
26	1A	2A	BT	\01A	58	3A	5A	SIGPOLL	\01F	90	5A	126	SIGPOLL	\01F
27	1B	2B	HT	\01B	59	3B	5B	SIGPOLL	\01F	91	5B	127	SIGPOLL	\01F
28	1C	2C	LF	\01C	60	3C	5C	SIGPOLL	\01F	92	5C	128	SIGPOLL	\01F
29	1D	2D	VT	\01D	61	3D	5D	SIGPOLL	\01F	93	5D	129	SIGPOLL	\01F
30	1E	2E	FF	\01E	62	3E	5E	SIGPOLL	\01F	94	5E	130	SIGPOLL	\01F
31	1F	2F	CR	\01F	63	3F	5F	SIGPOLL	\01F	95	5F	131	SIGPOLL	\01F
32	20	30	SO	\020	64	40	60	SIGPOLL	\01F	96	60	132	SIGPOLL	\01F
33	21	31	SI	\021	65	41	61	SIGPOLL	\01F	97	61	133	SIGPOLL	\01F
34	22	32	DC1	\022	66	42	62	SIGPOLL	\01F	98	62	134	SIGPOLL	\01F
35	23	33	DC2	\023	67	43	63	SIGPOLL	\01F	99	63	135	SIGPOLL	\01F
36	24	34	DC3	\024	68	44	64	SIGPOLL	\01F	100	64	136	SIGPOLL	\01F
37	25	35	DC4	\025	69	45	65	SIGPOLL	\01F	101	65	137	SIGPOLL	\01F
38	26	36	NAK	\026	70	46	66	SIGPOLL	\01F	102	66	138	SIGPOLL	\01F
39	27	37	SYN	\027	71	47	67	SIGPOLL	\01F	103	67	139	SIGPOLL	\01F
40	28	38	ETX	\028	72	48	68	SIGPOLL	\01F	104	68	140	SIGPOLL	\01F
41	29	39	ENQ	\029	73	49	69	SIGPOLL	\01F	105	69	141	SIGPOLL	\01F
42	2A	3A	ACK	\02A	74	4A	6A	SIGPOLL	\01F	106	6A	142	SIGPOLL	\01F
43	2B	3B	BT	\02B	75	4B	6B	SIGPOLL	\01F	107	6B	143	SIGPOLL	\01F
44	2C	3C	HT	\02C	76	4C	6C	SIGPOLL	\01F	108	6C	144	SIGPOLL	\01F
45	2D	3D	LF	\02D	77	4D	6D	SIGPOLL	\01F	109	6D	145	SIGPOLL	\01F
46	2E	3E	VT	\02E	78	4E	6E	SIGPOLL	\01F	110	6E	146	SIGPOLL	\01F
47	2F	3F	FF	\02F	79	4F	6F	SIGPOLL	\01F	111	6F	147	SIGPOLL	\01F
48	30	40	CR	\030	80	50	70	SIGPOLL	\01F	112	70	148	SIGPOLL	\01F
49	31	41	SO	\031	81	51	71	SIGPOLL	\01F	113	71	149	SIGPOLL	\01F
50	32	42	SI	\032	82	52	72	SIGPOLL	\01F	114	72	150	SIGPOLL	\01F
51	33	43	DC1	\033	83	53	73	SIGPOLL	\01F	115	73	151	SIGPOLL	\01F
52	34	44	DC2	\034	84	54	74	SIGPOLL	\01F	116	74	152	SIGPOLL	\01F
53	35	45	DC3	\035	85	55	75	SIGPOLL	\01F	117	75	153	SIGPOLL	\01F
54	36	46	DC4	\036	86	56	76	SIGPOLL	\01F	118	76	154	SIGPOLL	\01F
55	37	47	NAK	\037	87	57	77	SIGPOLL	\01F	119	77	155	SIGPOLL	\01F
56	38	48	SYN	\038	88	58	78	SIGPOLL	\01F	120	78	156	SIGPOLL	\01F
57	39	49	ETX	\039	89	59	79	SIGPOLL	\01F	121	79	157	SIGPOLL	\01F
58	3A	4A	ENQ	\03A	90	5A	7A	SIGPOLL	\01F	122	7A	158	SIGPOLL	\01F
59	3B	4B	ACK	\03B	91	5B	7B	SIGPOLL	\01F	123	7B	159	SIGPOLL	\01F
60	3C	4C	BT	\03C	92	5C	7C	SIGPOLL	\01F	124	7C	160	SIGPOLL	\01F
61	3D	4D	HT	\03D	93	5D	7D	SIGPOLL	\01F	125	7D	161	SIGPOLL	\01F
62	3E	4E	LF	\03E	94	5E	7E	SIGPOLL	\01F	126	7E	162	SIGPOLL	\01F
63	3F	4F	VT	\03F	95	5F	7F	SIGPOLL	\01F	127	7F	163	SIGPOLL	\01F
64	40	50	FF	\040	96	60	80	SIGPOLL	\01F	128	80	164	SIGPOLL	\01F
65	41	51	CR	\041	97	61	81	SIGPOLL	\01F	129	81	165	SIGPOLL	\01F
66	42	52	SO	\042	98	62	82	SIGPOLL	\01F	130	82	166	SIGPOLL	\01F
67	43	53	SI	\043	99	63	83	SIGPOLL	\01F	131	83	167	SIGPOLL	\01F
68	44	54	DC1	\044	100	64	84	SIGPOLL	\01F	132	84	168	SIGPOLL	\01F
69	45	55	DC2	\045	101	65	85	SIGPOLL	\01F	133	85	169	SIGPOLL	\01F
70	46	56	DC3	\046	102	66	86	SIGPOLL	\01F	134	86	170	SIGPOLL	\01F
71	47	57	DC4	\047	103	67	87	SIGPOLL	\01F	135	87	171	SIGPOLL	\01F
72	48	58	NAK	\048	104	68	88	SIGPOLL	\01F	136	88	172	SIGPOLL	\01F
73	49	59	SYN	\049	105	69	89	SIGPOLL	\01F	137	89	173	SIGPOLL	\01F
74	4A	5A	ETX	\04A	106	6A	8A	SIGPOLL	\01F	138	8A	174	SIGPOLL	\01F
75	4B	5B	ENQ	\04B	107	6B	8B	SIGPOLL	\01F	139	8B	175	SIGPOLL	\01F
76	4C	5C	ACK	\04C	108	6C	8C	SIGPOLL	\01F	140	8C	176	SIGPOLL	\01F
77	4D	5D	BT	\04D	109	6D	8D	SIGPOLL	\01F	141	8D	177	SIGPOLL	\01F
78	4E	5E	HT	\04E	110	6E	8E	SIGPOLL	\01F	142	8E	178	SIGPOLL	\01F
79	4F	5F	LF	\04F	111	6F	8F	SIGPOLL	\01F	143	8F	179	SIGPOLL	\01F
80	50	60	VT	\050	112	70	90	SIGPOLL	\01F	144	90	180	SIGPOLL	\01F
81	51	61	FF	\051	113	71	91	SIGPOLL	\01F	145	91	181	SIGPOLL	\01F
82	52	62	CR	\052	114	72	92	SIGPOLL	\01F	146	92	182	SIGPOLL	\01F
83	53	63	SO	\053	115	73	93	SIGPOLL	\01F	147	93	183	SIGPOLL	\01F
84	54	64	SI	\054	116	74	94	SIGPOLL	\01F	148	94	184	SIGPOLL	\01F
85	55	65	DC1	\055	117	75	95	SIGPOLL	\01F	149	95	185	SIGPOLL	\01F
86	56	66	DC2	\056	118	76	96	SIGPOLL	\01F	150	96	186	SIGPOLL	\01F
87	57	67	DC3	\057	119	77	97	SIGPOLL	\01F	151	97	187	SIGPOLL	\01F
88	58	68	DC4	\058	120	78	98	SIGPOLL	\01F	152	98	188	SIGPOLL	\01F
89	59	69	NAK	\059	121	79	99	SIGPOLL	\01F	153	99	189	SIGPOLL	\01F
90	5A	6A	SYN	\05A	122	7A	9A	SIGPOLL	\01F	154	9A	190	SIGPOLL	\01F
91	5B	6B	ETX	\05B	123	7B	9B	SIGPOLL	\01F	155	9B	191	SIGPOLL	\01F
92	5C	6C	ENQ	\05C	124	7C	9C	SIGPOLL	\01F	156	9C	192	SIGPOLL	\01F
93	5D	6D	ACK	\05D	125	7D	9D	SIGPOLL	\01F	157	9D	193	SIGPOLL	\01F
94	5E	6E	BT	\05E	126	7E	9E	SIGPOLL	\01F	158	9E	194	SIGPOLL	\01F
95	5F	6F	HT	\05F	127	7F	9F	SIGPOLL	\01F	159	9F	195	SIGPOLL	\01F
96	60	70	LF	\060	128	80	A0	SIGPOLL	\01F	160	A0	196	SIGPOLL	\01F
97	61	71	VT	\061	129	81	A1	SIGPOLL	\01F	161	A1	197	SIGPOLL	\01F
98	62	72	FF	\062	130	82	A2	SIGPOLL	\01F	162	A2	198	SIGPOLL	\01F
99	63	73	CR	\063	131	83	A3	SIGPOLL	\01F	163	A3	199	SIGPOLL	\01F
100	64	74	SO	\064	132	84	A4	SIGPOLL	\01F	164	A4	200	SIGPOLL	\01F
101	65	75	SI	\065	133	85	A5	SIGPOLL	\01F	165	A5	201	SIGPOLL	\01F
102	66	76	DC1	\066	134	86	A6	SIGPOLL	\01F	166	A6	202	SIGPOLL	\01F
103	67	77	DC2	\067	135	87	A7	SIGP						

- `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.)

Decimal Num Char	Octal Num Char	Hex Num Char	Decimal Num Char	Octal Num Char	Hex Num Char
'\0'	'000'	'000'	'\n'	'000'	'000'
'\t'	'009'	'009'	'\r'	'00D'	'00D'
'\v'	'012'	'012'	'\f'	'014'	'014'
'\b'	'010'	'010'	'\a'	'007'	'007'
'\x00'	'000'	'000'	'\x01'	'001'	'001'
'\x02'	'002'	'002'	'\x03'	'003'	'003'
'\x04'	'004'	'004'	'\x05'	'005'	'005'
'\x06'	'006'	'006'	'\x07'	'007'	'007'
'\x08'	'010'	'010'	'\x09'	'012'	'012'
'\x0A'	'014'	'014'	'\x0B'	'015'	'015'
'\x0C'	'016'	'016'	'\x0D'	'017'	'017'
'\x0E'	'020'	'020'	'\x0F'	'021'	'021'
'\x10'	'022'	'022'	'\x11'	'023'	'023'
'\x12'	'024'	'024'	'\x13'	'025'	'025'
'\x14'	'026'	'026'	'\x15'	'027'	'027'
'\x16'	'030'	'030'	'\x17'	'031'	'031'
'\x18'	'032'	'032'	'\x19'	'033'	'033'
'\x1A'	'034'	'034'	'\x1B'	'035'	'035'
'\x1C'	'036'	'036'	'\x1D'	'037'	'037'
'\x1E'	'040'	'040'	'\x1F'	'041'	'041'
'\x20'	'042'	'042'	'\x21'	'043'	'043'
'\x22'	'044'	'044'	'\x23'	'045'	'045'
'\x24'	'046'	'046'	'\x25'	'047'	'047'
'\x26'	'050'	'050'	'\x27'	'051'	'051'
'\x28'	'052'	'052'	'\x29'	'053'	'053'
'\x2A'	'054'	'054'	'\x2B'	'055'	'055'
'\x2C'	'056'	'056'	'\x2D'	'057'	'057'
'\x2E'	'060'	'060'	'\x2F'	'061'	'061'
'\x2E'	'060'	'060'	'\x30'	'062'	'062'
'\x31'	'063'	'063'	'\x32'	'064'	'064'
'\x33'	'065'	'065'	'\x34'	'066'	'066'
'\x35'	'067'	'067'	'\x36'	'070'	'070'
'\x37'	'071'	'071'	'\x38'	'072'	'072'
'\x39'	'073'	'073'	'\x3A'	'074'	'074'
'\x3B'	'075'	'075'	'\x3C'	'076'	'076'
'\x3D'	'077'	'077'	'\x3E'	'080'	'080'
'\x3F'	'081'	'081'	'\x3F'	'082'	'082'
'\x40'	'083'	'083'	'\x40'	'084'	'084'
'\x41'	'085'	'085'	'\x42'	'086'	'086'
'\x43'	'087'	'087'	'\x44'	'088'	'088'
'\x45'	'089'	'089'	'\x46'	'090'	'090'
'\x47'	'091'	'091'	'\x48'	'092'	'092'
'\x49'	'093'	'093'	'\x4A'	'094'	'094'
'\x4B'	'095'	'095'	'\x4C'	'096'	'096'
'\x4D'	'097'	'097'	'\x4E'	'098'	'098'
'\x4F'	'099'	'099'	'\x4F'	'099'	'099'
'\x50'	'100'	'0A0'	'\x51'	'101'	'0A1'
'\x52'	'102'	'0A2'	'\x53'	'103'	'0A3'
'\x54'	'104'	'0A4'	'\x55'	'105'	'0A5'
'\x56'	'106'	'0A6'	'\x57'	'107'	'0A7'
'\x58'	'108'	'0A8'	'\x59'	'109'	'0A9'
'\x5A'	'110'	'0AA'	'\x5B'	'111'	'0AB'
'\x5C'	'112'	'0AC'	'\x5D'	'113'	'0AD'
'\x5E'	'114'	'0AE'	'\x5F'	'115'	'0AF'
'\x60'	'116'	'0B0'	'\x61'	'117'	'0B1'
'\x62'	'118'	'0B2'	'\x63'	'119'	'0B3'
'\x64'	'120'	'0B4'	'\x65'	'121'	'0B5'
'\x66'	'122'	'0B6'	'\x67'	'123'	'0B7'
'\x68'	'124'	'0B8'	'\x69'	'125'	'0B9'
'\x6A'	'126'	'0BA'	'\x6B'	'127'	'0BB'
'\x6C'	'128'	'0BC'	'\x6D'	'129'	'0BD'
'\x6E'	'130'	'0BE'	'\x6F'	'131'	'0BF'
'\x70'	'132'	'0C0'	'\x71'	'133'	'0C1'
'\x72'	'134'	'0C2'	'\x73'	'135'	'0C3'
'\x74'	'136'	'0C4'	'\x75'	'137'	'0C5'
'\x76'	'138'	'0C6'	'\x77'	'139'	'0C7'
'\x78'	'140'	'0C8'	'\x79'	'141'	'0C9'
'\x7A'	'142'	'0CA'	'\x7B'	'143'	'0CB'
'\x7C'	'144'	'0CC'	'\x7D'	'145'	'0CD'
'\x7E'	'146'	'0CE'	'\x7F'	'147'	'0CF'
'\x80'	'148'	'0D0'	'\x81'	'149'	'0D1'
'\x82'	'150'	'0D2'	'\x83'	'151'	'0D3'
'\x84'	'152'	'0D4'	'\x85'	'153'	'0D5'
'\x86'	'154'	'0D6'	'\x87'	'155'	'0D7'
'\x88'	'156'	'0D8'	'\x89'	'157'	'0D9'
'\x8A'	'158'	'0DA'	'\x8B'	'159'	'0DB'
'\x8C'	'160'	'0DC'	'\x8D'	'161'	'0DD'
'\x8E'	'162'	'0DE'	'\x8F'	'163'	'0DF'
'\x90'	'164'	'0E0'	'\x91'	'165'	'0E1'
'\x92'	'166'	'0E2'	'\x93'	'167'	'0E3'
'\x94'	'168'	'0E4'	'\x95'	'169'	'0E5'
'\x96'	'170'	'0E6'	'\x97'	'171'	'0E7'
'\x98'	'172'	'0E8'	'\x99'	'173'	'0E9'
'\x9A'	'174'	'0EA'	'\x9B'	'175'	'0EB'
'\x9C'	'176'	'0EC'	'\x9D'	'177'	'0ED'
'\x9E'	'178'	'0EE'	'\x9F'	'179'	'0EF'
'\xA0'	'180'	'0F0'	'\xA1'	'181'	'0F1'
'\xA2'	'182'	'0F2'	'\xA3'	'183'	'0F3'
'\xA4'	'184'	'0F4'	'\xA5'	'185'	'0F5'
'\xA6'	'186'	'0F6'	'\xA7'	'187'	'0F7'
'\xA8'	'188'	'0F8'	'\xA9'	'189'	'0F9'
'\xA9'	'190'	'0FA'	'\xA9'	'191'	'0FB'
'\xA9'	'192'	'0FC'	'\xA9'	'193'	'0FD'
'\xA9'	'194'	'0FE'	'\xA9'	'195'	'0FF'

- `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.)

ASCII TABLE														
Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char
0	00	0	NULL	\0	32	20	40	SIGKILL	\000	64	40	100	SIGPOLL	\001
1	01	1	SOH	\001	33	21	41	SIGALRM	\002	65	41	101	SIGSTOP	\003
2	02	2	STX	\002	34	22	42	SIGPOLL	\004	66	42	102	SIGCONT	\005
3	03	3	ETX	\003	35	23	43	SIGPOLL	\006	67	43	103	SIGKILL	\007
4	04	4	EOT	\004	36	24	44	SIGPOLL	\008	68	44	104	SIGPOLL	\009
5	05	5	ENQ	\005	37	25	45	SIGPOLL	\010	69	45	105	SIGPOLL	\011
6	06	6	ACK	\006	38	26	46	SIGPOLL	\012	70	46	106	SIGPOLL	\013
7	07	7	NAK	\007	39	27	47	SIGPOLL	\014	71	47	107	SIGPOLL	\015
8	08	10	SYN	\008	40	28	48	SIGPOLL	\016	72	48	108	SIGPOLL	\017
9	09	11	BT	\009	41	29	49	SIGPOLL	\018	73	49	109	SIGPOLL	\019
10	0A	12	HT	\00A	42	2A	4A	SIGPOLL	\01A	74	4A	110	SIGPOLL	\01B
11	0B	13	LF	\00B	43	2B	4B	SIGPOLL	\01B	75	4B	111	SIGPOLL	\01C
12	0C	14	VT	\00C	44	2C	4C	SIGPOLL	\01C	76	4C	112	SIGPOLL	\01D
13	0D	15	FF	\00D	45	2D	4D	SIGPOLL	\01D	77	4D	113	SIGPOLL	\01E
14	0E	16	CR	\00E	46	2E	4E	SIGPOLL	\01E	78	4E	114	SIGPOLL	\01F
15	0F	17	SO	\00F	47	2F	4F	SIGPOLL	\01F	79	4F	115	SIGPOLL	\020
16	10	20	SI	\010	48	30	50	SIGPOLL	\01F	80	50	116	SIGPOLL	\021
17	11	21	DC1	\011	49	31	51	SIGPOLL	\01F	81	51	117	SIGPOLL	\022
18	12	22	DC2	\012	50	32	52	SIGPOLL	\01F	82	52	118	SIGPOLL	\023
19	13	23	DC3	\013	51	33	53	SIGPOLL	\01F	83	53	119	SIGPOLL	\024
20	14	24	DC4	\014	52	34	54	SIGPOLL	\01F	84	54	120	SIGPOLL	\025
21	15	25	NAK	\015	53	35	55	SIGPOLL	\01F	85	55	121	SIGPOLL	\026
22	16	26	SYN	\016	54	36	56	SIGPOLL	\01F	86	56	122	SIGPOLL	\027
23	17	27	ETX	\017	55	37	57	SIGPOLL	\01F	87	57	123	SIGPOLL	\028
24	18	28	ENQ	\018	56	38	58	SIGPOLL	\01F	88	58	124	SIGPOLL	\029
25	19	29	ACK	\019	57	39	59	SIGPOLL	\01F	89	59	125	SIGPOLL	\02A
26	1A	2A	BT	\01A	58	3A	5A	SIGPOLL	\01F	90	5A	126	SIGPOLL	\02B
27	1B	2B	HT	\01B	59	3B	5B	SIGPOLL	\01F	91	5B	127	SIGPOLL	\02C
28	1C	2C	LF	\01C	60	3C	5C	SIGPOLL	\01F	92	5C	128	SIGPOLL	\02D
29	1D	2D	VT	\01D	61	3D	5D	SIGPOLL	\01F	93	5D	129	SIGPOLL	\02E
30	1E	2E	FF	\01E	62	3E	5E	SIGPOLL	\01F	94	5E	130	SIGPOLL	\02F
31	1F	2F	CR	\01F	63	3F	5F	SIGPOLL	\01F	95	5F	131	SIGPOLL	\030
32	20	30	SO	\020	64	40	60	SIGPOLL	\01F	96	60	132	SIGPOLL	\031
33	21	31	SI	\021	65	41	61	SIGPOLL	\01F	97	61	133	SIGPOLL	\032
34	22	32	DC1	\022	66	42	62	SIGPOLL	\01F	98	62	134	SIGPOLL	\033
35	23	33	DC2	\023	67	43	63	SIGPOLL	\01F	99	63	135	SIGPOLL	\034
36	24	34	DC3	\024	68	44	64	SIGPOLL	\01F	100	64	136	SIGPOLL	\035
37	25	35	DC4	\025	69	45	65	SIGPOLL	\01F	101	65	137	SIGPOLL	\036
38	26	36	NAK	\026	70	46	66	SIGPOLL	\01F	102	66	138	SIGPOLL	\037
39	27	37	SYN	\027	71	47	67	SIGPOLL	\01F	103	67	139	SIGPOLL	\038
40	28	38	ETX	\028	72	48	68	SIGPOLL	\01F	104	68	140	SIGPOLL	\039
41	29	39	ENQ	\029	73	49	69	SIGPOLL	\01F	105	69	141	SIGPOLL	\03A
42	2A	3A	ACK	\02A	74	4A	6A	SIGPOLL	\01F	106	6A	142	SIGPOLL	\03B
43	2B	3B	BT	\02B	75	4B	6B	SIGPOLL	\01F	107	6B	143	SIGPOLL	\03C
44	2C	3C	HT	\02C	76	4C	6C	SIGPOLL	\01F	108	6C	144	SIGPOLL	\03D
45	2D	3D	LF	\02D	77	4D	6D	SIGPOLL	\01F	109	6D	145	SIGPOLL	\03E
46	2E	3E	VT	\02E	78	4E	6E	SIGPOLL	\01F	110	6E	146	SIGPOLL	\03F
47	2F	3F	FF	\02F	79	4F	6F	SIGPOLL	\01F	111	6F	147	SIGPOLL	\040
48	30	40	CR	\030	80	50	70	SIGPOLL	\01F	112	70	148	SIGPOLL	\041
49	31	41	SO	\031	81	51	71	SIGPOLL	\01F	113	71	149	SIGPOLL	\042
50	32	42	SI	\032	82	52	72	SIGPOLL	\01F	114	72	150	SIGPOLL	\043
51	33	43	DC1	\033	83	53	73	SIGPOLL	\01F	115	73	151	SIGPOLL	\044
52	34	44	DC2	\034	84	54	74	SIGPOLL	\01F	116	74	152	SIGPOLL	\045
53	35	45	DC3	\035	85	55	75	SIGPOLL	\01F	117	75	153	SIGPOLL	\046
54	36	46	DC4	\036	86	56	76	SIGPOLL	\01F	118	76	154	SIGPOLL	\047
55	37	47	NAK	\037	87	57	77	SIGPOLL	\01F	119	77	155	SIGPOLL	\048
56	38	48	SYN	\038	88	58	78	SIGPOLL	\01F	120	78	156	SIGPOLL	\049
57	39	49	ETX	\039	89	59	79	SIGPOLL	\01F	121	79	157	SIGPOLL	\04A
58	3A	4A	ENQ	\03A	90	5A	7A	SIGPOLL	\01F	122	7A	158	SIGPOLL	\04B
59	3B	4B	ACK	\03B	91	5B	7B	SIGPOLL	\01F	123	7B	159	SIGPOLL	\04C
60	3C	4C	BT	\03C	92	5C	7C	SIGPOLL	\01F	124	7C	160	SIGPOLL	\04D
61	3D	4D	HT	\03D	93	5D	7D	SIGPOLL	\01F	125	7D	161	SIGPOLL	\04E
62	3E	4E	LF	\03E	94	5E	7E	SIGPOLL	\01F	126	7E	162	SIGPOLL	\04F
63	3F	4F	VT	\03F	95	5F	7F	SIGPOLL	\01F	127	7F	163	SIGPOLL	\050
64	40	50	FF	\040	96	60	80	SIGPOLL	\01F	128	80	164	SIGPOLL	\051
65	41	51	CR	\041	97	61	81	SIGPOLL	\01F	129	81	165	SIGPOLL	\052
66	42	52	SO	\042	98	62	82	SIGPOLL	\01F	130	82	166	SIGPOLL	\053
67	43	53	SI	\043	99	63	83	SIGPOLL	\01F	131	83	167	SIGPOLL	\054
68	44	54	DC1	\044	100	64	84	SIGPOLL	\01F	132	84	168	SIGPOLL	\055
69	45	55	DC2	\045	101	65	85	SIGPOLL	\01F	133	85	169	SIGPOLL	\056
70	46	56	DC3	\046	102	66	86	SIGPOLL	\01F	134	86	170	SIGPOLL	\057
71	47	57	DC4	\047	103	67	87	SIGPOLL	\01F	135	87	171	SIGPOLL	\058
72	48	58	NAK	\048	104	68	88	SIGPOLL	\01F	136	88	172	SIGPOLL	\059
73	49	59	SYN	\049	105	69	89	SIGPOLL	\01F	137	89	173	SIGPOLL	\05A
74	4A	5A	ETX	\04A	106	6A	8A	SIGPOLL	\01F	138	8A	174	SIGPOLL	\05B
75	4B	5B	ENQ	\04B	107	6B	8B	SIGPOLL	\01F	139	8B	175	SIGPOLL	\05C
76	4C	5C	ACK	\04C	108	6C	8C	SIGPOLL	\01F	140	8C	176	SIGPOLL	\05D
77	4D	5D	BT	\04D	109	6D	8D	SIGPOLL	\01F	141	8D	177	SIGPOLL	\05E
78	4E	5E	HT	\04E	110	6E	8E	SIGPOLL	\01F	142	8E	178	SIGPOLL	\05F
79	4F	5F	LF	\04F	111	6F	8F	SIGPOLL	\01F	143	8F	179	SIGPOLL	\060
80	50	60	VT	\050	112	70	90	SIGPOLL	\01F	144	90	180	SIGPOLL	\061
81	51	61	FF	\051	113	71	91	SIGPOLL	\01F	145	91	181	SIGPOLL	\062
82	52	62	CR	\052	114	72	92	SIGPOLL	\01F	146	92	182	SIGPOLL	\063
83	53	63	SO	\053	115	73	93	SIGPOLL	\01F	147	93	183	SIGPOLL	\064
84	54	64	SI	\054	116	74	94	SIGPOLL	\01F	148	94	184	SIGPOLL	\065
85	55	65	DC1	\055	117	75	95	SIGPOLL	\01F	149	95	185	SIGPOLL	\066
86	56	66	DC2	\056	118	76	96	SIGPOLL	\01F	150	96	186	SIGPOLL	\067
87	57	67	DC3	\057	119	77	97	SIGPOLL	\01F	151	97	187	SIGPOLL	\068
88	58	68	DC4	\058	120	78	98	SIGPOLL	\01F	152	98	188	SIGPOLL	\069
89	59	69	NAK	\059	121	79	99	SIGPOLL	\01F	153	99	189	SIGPOLL	\06A
90	5A	6A	SYN	\05A	122	7A	9A	SIGPOLL	\01F	154	9A	190	SIGPOLL	\06B
91	5B	6B	ETX	\05B	123	7B	9B	SIGPOLL	\01F	155	9B	191	SIGPOLL	\06C
92	5C	6C	ENQ	\05C	124	7C	9C	SIGPOLL	\01F	156	9C	192	SIGPOLL	\06D
93	5D	6D	ACK	\05D	125	7D	9D	SIGPOLL	\01F	157	9D	193	SIGPOLL	\06E
94	5E	6E	BT	\05E	126	7E	9E	SIGPOLL	\01F	158	9E	194	SIGPOLL	\06F
95	5F	6F	HT	\05F	127	7F	9F	SIGPOLL	\01F	159	9F	195	SIGPOLL	\070
96	60	70	LF	\060	128	80	A0	SIGPOLL	\01F	160	A0	196	SIGPOLL	\071
97	61	71	VT	\061	129	81	A1	SIGPOLL	\01F	161	A1	197	SIGPOLL	\072
98	62	72	FF	\062	130	82	A2	SIGPOLL	\01F	162	A2	198	SIGPOLL	\073
99	63	73	CR	\063	131	83	A3	SIGPOLL	\01F	163	A3	199	SIGPOLL	\074
100	64	74	SO	\064	132	84	A4	SIGPOLL	\01F	164	A4	200	SIGPOLL	\075
101	65	75	SI	\065	133	85	A5	SIGPOLL	\01F	165	A5	201	SIGPOLL	\076
102	66	76	DC1	\066	134	86	A6	SIGPOLL	\01F	166	A6	202	SIGPOLL	\077
103	67	77	DC2	\067	135	87	A7	SIGPOLL						

- `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 New Char	Octal New Char	Hex New Char	Decimal New Char	Octal New Char	Hex New Char
0	000	000	1	001	001
2	002	002	3	003	003
4	004	004	5	005	005
6	006	006	7	007	007
8	010	008	9	011	009
10	012	00A	11	013	00B
12	014	00C	13	015	00D
14	016	00E	15	017	00F
16	020	010	17	021	011
18	022	012	19	023	013
20	024	014	21	025	015
22	026	016	23	027	017
24	030	018	25	031	019
26	032	01A	27	033	01B
28	034	01C	29	035	01D
30	036	01E	31	037	01F
32	040	020	33	041	021
34	042	022	35	043	023
36	044	024	37	045	025
38	046	026	39	047	027
40	050	028	41	051	029
42	052	02A	43	053	02B
44	054	02C	45	055	02D
46	056	02E	47	057	02F
48	060	030	49	061	031
50	062	032	51	063	033
52	064	034	53	065	035
54	066	036	55	067	037
56	070	038	57	071	039
58	072	03A	59	073	03B
60	074	03C	61	075	03D
62	076	03E	63	077	03F
64	080	040	65	081	041
66	082	042	67	083	043
68	084	044	69	085	045
70	086	046	71	087	047
72	090	048	73	091	049
74	092	04A	75	093	04B
76	094	04C	77	095	04D
78	096	04E	79	097	04F
80	100	050	81	101	051
82	102	052	83	103	053
84	104	054	85	105	055
86	106	056	87	107	057
88	110	05A	89	111	05B
90	112	05C	91	113	05D
92	114	05E	93	115	05F
94	116	060	95	117	061
96	120	064	97	121	065
98	122	066	99	123	067
100	124	068	101	125	069
102	126	06A	103	127	06B
104	130	06C	105	131	06D
106	132	06E	107	133	06F
108	134	070	109	135	071
110	136	072	111	137	073
112	138	074	113	139	075
114	140	076	115	141	077
116	144	07A	117	145	07B
118	146	07C	119	147	07D
120	148	07E	121	149	07F
122	150	080	123	151	081
124	152	082	125	153	083
126	154	084	127	155	085
128	156	086	129	157	087
130	160	090	131	161	091
132	162	092	133	163	093
134	164	094	135	165	095
136	166	096	137	167	097
138	170	0A0	139	171	0A1
140	172	0A2	141	173	0A3
142	174	0A4	143	175	0A5
144	176	0A6	145	177	0A7
146	180	0B0	147	181	0B1
148	182	0B2	149	183	0B3
150	184	0B4	151	185	0B5
152	186	0B6	153	187	0B7
154	190	0C0	155	191	0C1
156	192	0C2	157	193	0C3
158	194	0C4	159	195	0C5
160	196	0C6	161	197	0C7
162	200	0D0	163	201	0D1
164	202	0D2	165	203	0D3
166	204	0D4	167	205	0D5
168	206	0D6	169	207	0D7
170	210	0E0	171	211	0E1
172	212	0E2	173	213	0E3
174	214	0E4	175	215	0E5
176	216	0E6	177	217	0E7
178	220	0F0	179	221	0F1
180	222	0F2	181	223	0F3
182	224	0F4	183	225	0F5
184	226	0F6	185	227	0F7
186	230	100	187	231	101
188	232	102	189	233	103
190	234	104	191	235	105
192	236	106	193	237	107
194	240	108	195	241	109
196	242	10A	197	243	10B
198	244	10C	199	245	10D
200	246	10E	201	247	10F
202	250	110	203	251	111
204	252	112	205	253	113
206	254	114	207	255	115
208	256	116	209	257	117
210	260	118	211	261	119
212	262	11A	213	263	11B
214	264	11C	215	265	11D
216	266	11E	217	267	11F
218	270	120	219	271	121
220	272	122	221	273	123
222	274	124	223	275	125
224	276	126	225	277	127
226	280	130	227	281	131
228	282	132	229	283	133
230	284	134	231	285	135
232	286	136	233	287	137
234	290	138	235	291	139
236	292	13A	237	293	13B
238	294	13C	239	295	13D
240	296	13E	241	297	13F
242	300	140	243	301	141
244	302	142	245	303	143
246	304	144	247	305	145
248	306	146	249	307	147
250	310	148	251	311	149
252	312	14A	253	313	14B
254	314	14C	255	315	14D
256	316	14E	257	317	14F
258	320	150	259	321	151
260	322	152	261	323	153
262	324	154	263	325	155
264	326	156	265	327	157
266	330	158	267	331	159
268	332	15A	269	333	15B
270	334	15C	271	335	15D
272	336	15E	273	337	15F
274	340	160	275	341	161
276	342	162	277	343	163
278	344	164	279	345	165
280	346	166	281	347	167
282	350	168	283	351	169
284	352	16A	285	353	16B
286	354	16C	287	355	16D
288	356	16E	289	357	16F
290	360	170	291	361	171
292	362	172	293	363	173
294	364	174	295	365	175
296	366	176	297	367	177
298	370	178	299	371	179
300	372	17A	301	373	17B
302	374	17C	303	375	17D
304	376	17E	305	377	17F
306	380	180	307	381	181
308	382	182	309	383	183
310	384	184	311	385	185
312	386	186	313	387	187
314	390	188	315	391	189
316	392	18A	317	393	18B
318	394	18C	319	395	18D
320	396	18E	321	397	18F
322	400	190	323	401	191
324	402	192	325	403	193
326	404	194	327	405	195
328	406	196	329	407	197
330	410	198	331	411	199
332	412	19A	333	413	19B
334	414	19C	335	415	19D
336	416	19E	337	417	19F
338	420	200	339	421	201
340	422	202	341	423	203
342	424	204	343	425	205
344	426	206	345	427	207
346	430	208	347	431	209
348	432	20A	349	433	20B
350	434	20C	351	435	20D
352	436	20E	353	437	20F
354	440	210	355	441	211
356	442	212	357	443	213
358	444	214	359	445	215
360	446	216	361	447	217
362	450	218	363	451	219
364	452	21A	365	453	21B
366	454	21C	367	455	21D
368	456	21E	369	457	21F
370	460	220	371	461	221
372	462	222	373	463	223
374	464	224	375	465	225
376	466	226	377	467	227
378	470	228	379	471	229
380	472	22A	381	473	22B
382	474	22C	383	475	22D
384	476	22E	385	477	22F
386	480	230	387	481	231
388	482	232	389	483	233
390	484	234	391	485	235
392	486	236	393	487	237
394	490	238	395	491	239
396	492	23A	397	493	23B
398	494	23C	399	495	23D
400	496	23E	401	497	23F
402	500	240	403	501	241
404	502	242	405	503	243
406	504	244	407	505	245
408	506	246	409	507	247
410	510	248	411	511	249
412	512	24A	413	513	24B
414	514	24C	415	515	24D
416	516	24E	417	517	24F
418	520	250	419	521	251
420	522	252	421	523	253
422	524	254	423	525	255
424	526	256	425	527	257
426	530	258	427	531	259
428	532	25A	429	533	25B
430	534	25C	431	535	25D
432	536	25E	433	537	25F
434	540	260	435	541	261
436	542	262	437	543	263
438	544	264	439	545	265
440	546	266	441	547	267
442	550	268	443	551	269
444	552	26A	445	553	26B
446	554	26C	447	555	26D
448	556	26E	449	557	26F
450	560	270	451	561	271
452	562	272	453	563	273
454	564	274	455	565	275
456	566	276	457	567	277
458	570	278	459	571	279
460	572	27A	461	573	27B
462	574	27C	463	575	27D
464</					

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, it wraps back to 0  
19     newChar = chr(ord(ch) + wrap) #compute the new letter  
20     print(wrap, chr(ord(ch) + 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...