

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:

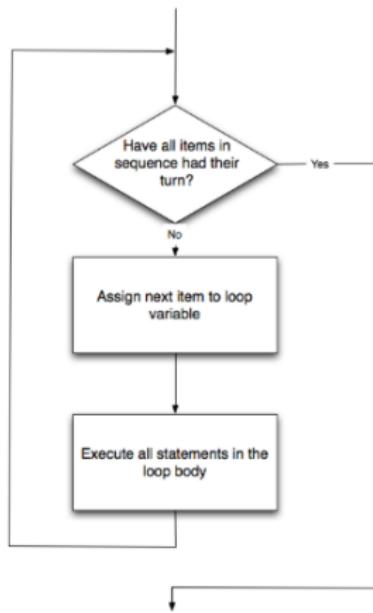
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

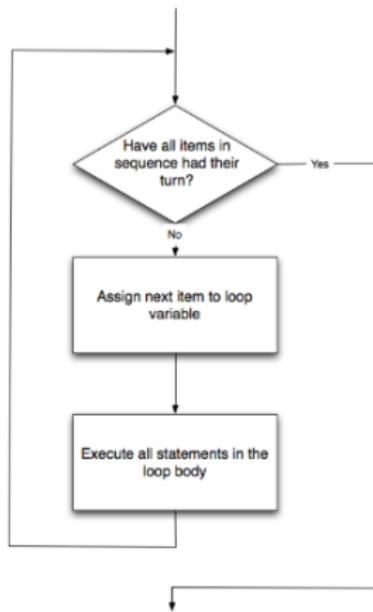
for-loop



```
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	16
00000011	00000011	11	17
00000012	00000012	12	18
00000013	00000013	13	19
00000014	00000014	14	20
00000015	00000015	15	21
00000016	00000016	16	22
00000017	00000017	17	23
00000018	00000018	18	24
00000019	00000019	19	25
0000001A	0000001A	1A	26
0000001B	0000001B	1B	27
0000001C	0000001C	1C	28
0000001D	0000001D	1D	29
0000001E	0000001E	1E	30
0000001F	0000001F	1F	31
00000020	00000020	20	32
00000021	00000021	21	33
00000022	00000022	22	34
00000023	00000023	23	35
00000024	00000024	24	36
00000025	00000025	25	37
00000026	00000026	26	38
00000027	00000027	27	39
00000028	00000028	28	40
00000029	00000029	29	41
0000002A	0000002A	2A	42
0000002B	0000002B	2B	43
0000002C	0000002C	2C	44
0000002D	0000002D	2D	45
0000002E	0000002E	2E	46
0000002F	0000002F	2F	47
00000030	00000030	30	48
00000031	00000031	31	49
00000032	00000032	32	50
00000033	00000033	33	51
00000034	00000034	34	52
00000035	00000035	35	53
00000036	00000036	36	54
00000037	00000037	37	55
00000038	00000038	38	56
00000039	00000039	39	57
0000003A	0000003A	3A	58
0000003B	0000003B	3B	59
0000003C	0000003C	3C	60
0000003D	0000003D	3D	61
0000003E	0000003E	3E	62
0000003F	0000003F	3F	63
00000040	00000040	40	64
00000041	00000041	41	65
00000042	00000042	42	66
00000043	00000043	43	67
00000044	00000044	44	68
00000045	00000045	45	69
00000046	00000046	46	70
00000047	00000047	47	71
00000048	00000048	48	72
00000049	00000049	49	73
0000004A	0000004A	4A	74
0000004B	0000004B	4B	75
0000004C	0000004C	4C	76
0000004D	0000004D	4D	77
0000004E	0000004E	4E	78
0000004F	0000004F	4F	79
00000050	00000050	50	80
00000051	00000051	51	81
00000052	00000052	52	82
00000053	00000053	53	83
00000054	00000054	54	84
00000055	00000055	55	85
00000056	00000056	56	86
00000057	00000057	57	87
00000058	00000058	58	88
00000059	00000059	59	89
0000005A	0000005A	5A	90
0000005B	0000005B	5B	91
0000005C	0000005C	5C	92
0000005D	0000005D	5D	93
0000005E	0000005E	5E	94
0000005F	0000005F	5F	95
00000060	00000060	60	96
00000061	00000061	61	97
00000062	00000062	62	98
00000063	00000063	63	99
00000064	00000064	64	100
00000065	00000065	65	101
00000066	00000066	66	102
00000067	00000067	67	103
00000068	00000068	68	104
00000069	00000069	69	105
0000006A	0000006A	6A	106
0000006B	0000006B	6B	107
0000006C	0000006C	6C	108
0000006D	0000006D	6D	109
0000006E	0000006E	6E	110
0000006F	0000006F	6F	111
00000070	00000070	70	112
00000071	00000071	71	113
00000072	00000072	72	114
00000073	00000073	73	115
00000074	00000074	74	116
00000075	00000075	75	117
00000076	00000076	76	118
00000077	00000077	77	119
00000078	00000078	78	120
00000079	00000079	79	121
0000007A	0000007A	7A	122
0000007B	0000007B	7B	123
0000007C	0000007C	7C	124
0000007D	0000007D	7D	125
0000007E	0000007E	7E	126
0000007F	0000007F	7F	127
00000080	00000080	80	128
00000081	00000081	81	129
00000082	00000082	82	130
00000083	00000083	83	131
00000084	00000084	84	132
00000085	00000085	85	133
00000086	00000086	86	134
00000087	00000087	87	135
00000088	00000088	88	136
00000089	00000089	89	137
0000008A	0000008A	8A	138
0000008B	0000008B	8B	139
0000008C	0000008C	8C	140
0000008D	0000008D	8D	141
0000008E	0000008E	8E	142
0000008F	0000008F	8F	143
00000090	00000090	90	144
00000091	00000091	91	145
00000092	00000092	92	146
00000093	00000093	93	147
00000094	00000094	94	148
00000095	00000095	95	149
00000096	00000096	96	150
00000097	00000097	97	151
00000098	00000098	98	152
00000099	00000099	99	153
0000009A	0000009A	9A	154
0000009B	0000009B	9B	155
0000009C	0000009C	9C	156
0000009D	0000009D	9D	157
0000009E	0000009E	9E	158
0000009F	0000009F	9F	159
000000A0	000000A0	A0	160
000000A1	000000A1	A1	161
000000A2	000000A2	A2	162
000000A3	000000A3	A3	163
000000A4	000000A4	A4	164
000000A5	000000A5	A5	165
000000A6	000000A6	A6	166
000000A7	000000A7	A7	167
000000A8	000000A8	A8	168
000000A9	000000A9	A9	169
000000AA	000000AA	A9	170
000000AB	000000AB	AB	171
000000AC	000000AC	AC	172
000000AD	000000AD	AD	173
000000AE	000000AE	AE	174
000000AF	000000AF	AF	175
000000B0	000000B0	B0	176
000000B1	000000B1	B1	177
000000B2	000000B2	B2	178
000000B3	000000B3	B3	179
000000B4	000000B4	B4	180
000000B5	000000B5	B5	181
000000B6	000000B6	B6	182
000000B7	000000B7	B7	183
000000B8	000000B8	B8	184
000000B9	000000B9	B9	185
000000BA	000000BA	BA	186
000000BB	000000BB	BB	187
000000BC	000000BC	BC	188
000000BD	000000BD	BD	189
000000BE	000000BE	BE	190
000000BF	000000BF	BF	191
000000C0	000000C0	C0	192
000000C1	000000C1	C1	193
000000C2	000000C2	C2	194
000000C3	000000C3	C3	195
000000C4	000000C4	C4	196
000000C5	000000C5	C5	197
000000C6	000000C6	C6	198
000000C7	000000C7	C7	199
000000C8	000000C8	C8	200
000000C9	000000C9	C9	201
000000CA	000000CA	CA	202
000000CB	000000CB	CB	203
000000CC	000000CC	CC	204
000000CD	000000CD	CD	205
000000CE	000000CE	CE	206
000000CF	000000CF	CF	207
000000D0	000000D0	D0	208
000000D1	000000D1	D1	209
000000D2	000000D2	D2	210
000000D3	000000D3	D3	211
000000D4	000000D4	D4	212
000000D5	000000D5	D5	213
000000D6	000000D6	D6	214
000000D7	000000D7	D7	215
000000D8	000000D8	D8	216
000000D9	000000D9	D9	217
000000DA	000000DA	DA	218
000000DB	000000DB	DB	219
000000DC	000000DC	DC	220
000000DD	000000DD	DD	221
000000DE	000000DE	DE	222
000000DF	000000DF	DF	223
000000E0	000000E0	E0	224
000000E1	000000E1	E1	225
000000E2	000000E2	E2	226
000000E3	000000E3	E3	227
000000E4	000000E4	E4	228
000000E5	000000E5	E5	229
000000E6	000000E6	E6	230
000000E7	000000E7	E7	231
000000E8	000000E8	E8	232
000000E9	000000E9	E9	233
000000EA	000000EA	EA	234
000000EB	000000EB	EB	235
000000EC	000000EC	EC	236
000000ED	000000ED	ED	237
000000EE	000000EE	EE	238
000000EF	000000EF	EF	239
000000F0	000000F0	F0	240
000000F1	000000F1	F1	241
000000F2	000000F2	F2	242
000000F3	000000F3	F3	243
000000F4	000000F4	F4	244
000000F5	000000F5	F5	245
000000F6	000000F6	F6	246
000000F7	000000F7	F7	247
000000F8	000000F8	F8	248
000000F9	000000F9	F9	249
000000FA	000000FA	FA	250
000000FB	000000FB	FB	251
000000FC	000000FC	FC	252
000000FD	000000FD	FD	253
000000FE	000000FE	FE	254
000000FF	000000FF	FF	255

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	DLE	\006	38	26	46	SIGPOLL	\012	70	46	106	SIGPOLL	\013
7	07	7	DC1	\007	39	27	47	SIGPOLL	\014	71	47	107	SIGPOLL	\015
8	08	10	DC2	\008	40	28	48	SIGPOLL	\016	72	48	108	SIGPOLL	\017
9	09	11	DC3	\009	41	29	49	SIGPOLL	\018	73	49	109	SIGPOLL	\019
10	0A	12	DC4	\00A	42	2A	4A	SIGPOLL	\01A	74	4A	110	SIGPOLL	\01B
11	0B	13	SUSP	\00B	43	2B	4B	SIGPOLL	\01C	75	4B	111	SIGPOLL	\01D
12	0C	14	DC5	\00C	44	2C	4C	SIGPOLL	\01E	76	4C	112	SIGPOLL	\01F
13	0D	15	DC6	\00D	45	2D	4D	SIGPOLL	\01G	77	4D	113	SIGPOLL	\01H
14	0E	16	NAK	\00E	46	2E	4E	SIGPOLL	\01I	78	4E	114	SIGPOLL	\01J
15	0F	17	SYN	\00F	47	2F	4F	SIGPOLL	\01K	79	4F	115	SIGPOLL	\01L
16	10	20	CAN	\010	48	30	50	SIGPOLL	\01M	80	50	116	SIGPOLL	\01N
17	11	21	EM	\011	49	31	51	SIGPOLL	\01O	81	51	117	SIGPOLL	\01P
18	12	22	FS	\012	50	32	52	SIGPOLL	\01R	82	52	118	SIGPOLL	\01S
19	13	23	GS	\013	51	33	53	SIGPOLL	\01T	83	53	119	SIGPOLL	\01U
20	14	24	RS	\014	52	34	54	SIGPOLL	\01V	84	54	120	SIGPOLL	\01W
21	15	25	US	\015	53	35	55	SIGPOLL	\01X	85	55	121	SIGPOLL	\01Y
22	16	26	SP	\016	54	36	56	SIGPOLL	\01Z	86	56	122	SIGPOLL	\01[
23	17	27	DEL	\017	55	37	57	SIGPOLL	\01\`	87	57	123	SIGPOLL	\01\`
24	18	28		\018	56	38	58	SIGPOLL	\01\`	88	58	124	SIGPOLL	\01\`
25	19	29		\019	57	39	59	SIGPOLL	\01\`	89	59	125	SIGPOLL	\01\`
26	1A	2A		\01A	58	3A	5A	SIGPOLL	\01\`	90	5A	126	SIGPOLL	\01\`
27	1B	2B		\01B	59	3B	5B	SIGPOLL	\01\`	91	5B	127	SIGPOLL	\01\`

- `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'	'\177'	'377'	'7F'
'\t'	'010'	'009'	'\128'	'320'	'80'
'\n'	'012'	'00A'	'\141'	'341'	'A1'
'\r'	'015'	'00D'	'\172'	'372'	'C2'
'\f'	'020'	'00C'	'\174'	'374'	'C4'
'\v'	'022'	'00B'	'\176'	'376'	'C6'
'\b'	'0100'	'0040'	'\170'	'370'	'80'
'\a'	'0007'	'0007'	'\177'	'377'	'7F'
'\x00'	'0000'	'0000'	'\177'	'377'	'7F'
'\x01'	'0001'	'0001'	'\177'	'377'	'7F'
'\x02'	'0002'	'0002'	'\177'	'377'	'7F'
'\x03'	'0003'	'0003'	'\177'	'377'	'7F'
'\x04'	'0004'	'0004'	'\177'	'377'	'7F'
'\x05'	'0005'	'0005'	'\177'	'377'	'7F'
'\x06'	'0006'	'0006'	'\177'	'377'	'7F'
'\x07'	'0007'	'0007'	'\177'	'377'	'7F'
'\x08'	'0010'	'0040'	'\170'	'370'	'80'
'\x09'	'0012'	'0050'	'\172'	'372'	'C2'
'\x0A'	'0015'	'0060'	'\174'	'374'	'C4'
'\x0B'	'0020'	'0070'	'\176'	'376'	'C6'
'\x0C'	'0040'	'0100'	'\170'	'370'	'80'
'\x0D'	'0007'	'0007'	'\177'	'377'	'7F'
'\x0E'	'0001'	'0001'	'\177'	'377'	'7F'
'\x0F'	'0002'	'0002'	'\177'	'377'	'7F'
'\x10'	'0003'	'0003'	'\177'	'377'	'7F'
'\x11'	'0004'	'0004'	'\177'	'377'	'7F'
'\x12'	'0005'	'0005'	'\177'	'377'	'7F'
'\x13'	'0006'	'0006'	'\177'	'377'	'7F'
'\x14'	'0007'	'0007'	'\177'	'377'	'7F'
'\x15'	'0010'	'0040'	'\170'	'370'	'80'
'\x16'	'0012'	'0050'	'\172'	'372'	'C2'
'\x17'	'0015'	'0060'	'\174'	'374'	'C4'
'\x18'	'0020'	'0070'	'\176'	'376'	'C6'
'\x19'	'0040'	'0100'	'\170'	'370'	'80'
'\x1A'	'0007'	'0007'	'\177'	'377'	'7F'
'\x1B'	'0001'	'0001'	'\177'	'377'	'7F'
'\x1C'	'0002'	'0002'	'\177'	'377'	'7F'
'\x1D'	'0003'	'0003'	'\177'	'377'	'7F'
'\x1E'	'0004'	'0004'	'\177'	'377'	'7F'
'\x1F'	'0005'	'0005'	'\177'	'377'	'7F'
'\x20'	'0006'	'0006'	'\177'	'377'	'7F'
'\x21'	'0007'	'0007'	'\177'	'377'	'7F'
'\x22'	'0010'	'0040'	'\170'	'370'	'80'
'\x23'	'0012'	'0050'	'\172'	'372'	'C2'
'\x24'	'0015'	'0060'	'\174'	'374'	'C4'
'\x25'	'0020'	'0070'	'\176'	'376'	'C6'
'\x26'	'0040'	'0100'	'\170'	'370'	'80'
'\x27'	'0007'	'0007'	'\177'	'377'	'7F'
'\x28'	'0001'	'0001'	'\177'	'377'	'7F'
'\x29'	'0002'	'0002'	'\177'	'377'	'7F'
'\x2A'	'0003'	'0003'	'\177'	'377'	'7F'
'\x2B'	'0004'	'0004'	'\177'	'377'	'7F'
'\x2C'	'0005'	'0005'	'\177'	'377'	'7F'
'\x2D'	'0006'	'0006'	'\177'	'377'	'7F'
'\x2E'	'0007'	'0007'	'\177'	'377'	'7F'
'\x2F'	'0010'	'0040'	'\170'	'370'	'80'
'\x30'	'0012'	'0050'	'\172'	'372'	'C2'
'\x31'	'0015'	'0060'	'\174'	'374'	'C4'
'\x32'	'0020'	'0070'	'\176'	'376'	'C6'
'\x33'	'0040'	'0100'	'\170'	'370'	'80'
'\x34'	'0007'	'0007'	'\177'	'377'	'7F'
'\x35'	'0001'	'0001'	'\177'	'377'	'7F'
'\x36'	'0002'	'0002'	'\177'	'377'	'7F'
'\x37'	'0003'	'0003'	'\177'	'377'	'7F'
'\x38'	'0004'	'0004'	'\177'	'377'	'7F'
'\x39'	'0005'	'0005'	'\177'	'377'	'7F'
'\x3A'	'0006'	'0006'	'\177'	'377'	'7F'
'\x3B'	'0007'	'0007'	'\177'	'377'	'7F'
'\x3C'	'0010'	'0040'	'\170'	'370'	'80'
'\x3D'	'0012'	'0050'	'\172'	'372'	'C2'
'\x3E'	'0015'	'0060'	'\174'	'374'	'C4'
'\x3F'	'0020'	'0070'	'\176'	'376'	'C6'
'\x40'	'0040'	'0100'	'\170'	'370'	'80'
'\x41'	'0007'	'0007'	'\177'	'377'	'7F'
'\x42'	'0001'	'0001'	'\177'	'377'	'7F'
'\x43'	'0002'	'0002'	'\177'	'377'	'7F'
'\x44'	'0003'	'0003'	'\177'	'377'	'7F'
'\x45'	'0004'	'0004'	'\177'	'377'	'7F'
'\x46'	'0005'	'0005'	'\177'	'377'	'7F'
'\x47'	'0006'	'0006'	'\177'	'377'	'7F'
'\x48'	'0007'	'0007'	'\177'	'377'	'7F'
'\x49'	'0010'	'0040'	'\170'	'370'	'80'
'\x4A'	'0012'	'0050'	'\172'	'372'	'C2'
'\x4B'	'0015'	'0060'	'\174'	'374'	'C4'
'\x4C'	'0020'	'0070'	'\176'	'376'	'C6'
'\x4D'	'0040'	'0100'	'\170'	'370'	'80'
'\x4E'	'0007'	'0007'	'\177'	'377'	'7F'
'\x4F'	'0001'	'0001'	'\177'	'377'	'7F'
'\x50'	'0002'	'0002'	'\177'	'377'	'7F'
'\x51'	'0003'	'0003'	'\177'	'377'	'7F'
'\x52'	'0004'	'0004'	'\177'	'377'	'7F'
'\x53'	'0005'	'0005'	'\177'	'377'	'7F'
'\x54'	'0006'	'0006'	'\177'	'377'	'7F'
'\x55'	'0007'	'0007'	'\177'	'377'	'7F'
'\x56'	'0010'	'0040'	'\170'	'370'	'80'
'\x57'	'0012'	'0050'	'\172'	'372'	'C2'
'\x58'	'0015'	'0060'	'\174'	'374'	'C4'
'\x59'	'0020'	'0070'	'\176'	'376'	'C6'
'\x5A'	'0040'	'0100'	'\170'	'370'	'80'
'\x5B'	'0007'	'0007'	'\177'	'377'	'7F'
'\x5C'	'0001'	'0001'	'\177'	'377'	'7F'
'\x5D'	'0002'	'0002'	'\177'	'377'	'7F'
'\x5E'	'0003'	'0003'	'\177'	'377'	'7F'
'\x5F'	'0004'	'0004'	'\177'	'377'	'7F'
'\x60'	'0005'	'0005'	'\177'	'377'	'7F'
'\x61'	'0006'	'0006'	'\177'	'377'	'7F'
'\x62'	'0007'	'0007'	'\177'	'377'	'7F'
'\x63'	'0010'	'0040'	'\170'	'370'	'80'
'\x64'	'0012'	'0050'	'\172'	'372'	'C2'
'\x65'	'0015'	'0060'	'\174'	'374'	'C4'
'\x66'	'0020'	'0070'	'\176'	'376'	'C6'
'\x67'	'0040'	'0100'	'\170'	'370'	'80'
'\x68'	'0007'	'0007'	'\177'	'377'	'7F'
'\x69'	'0001'	'0001'	'\177'	'377'	'7F'
'\x6A'	'0002'	'0002'	'\177'	'377'	'7F'
'\x6B'	'0003'	'0003'	'\177'	'377'	'7F'
'\x6C'	'0004'	'0004'	'\177'	'377'	'7F'
'\x6D'	'0005'	'0005'	'\177'	'377'	'7F'
'\x6E'	'0006'	'0006'	'\177'	'377'	'7F'
'\x6F'	'0007'	'0007'	'\177'	'377'	'7F'
'\x70'	'0010'	'0040'	'\170'	'370'	'80'
'\x71'	'0012'	'0050'	'\172'	'372'	'C2'
'\x72'	'0015'	'0060'	'\174'	'374'	'C4'
'\x73'	'0020'	'0070'	'\176'	'376'	'C6'
'\x74'	'0040'	'0100'	'\170'	'370'	'80'
'\x75'	'0007'	'0007'	'\177'	'377'	'7F'
'\x76'	'0001'	'0001'	'\177'	'377'	'7F'
'\x77'	'0002'	'0002'	'\177'	'377'	'7F'
'\x78'	'0003'	'0003'	'\177'	'377'	'7F'
'\x79'	'0004'	'0004'	'\177'	'377'	'7F'
'\x7A'	'0005'	'0005'	'\177'	'377'	'7F'
'\x7B'	'0006'	'0006'	'\177'	'377'	'7F'
'\x7C'	'0007'	'0007'	'\177'	'377'	'7F'
'\x7D'	'0010'	'0040'	'\170'	'370'	'80'
'\x7E'	'0012'	'0050'	'\172'	'372'	'C2'
'\x7F'	'0015'	'0060'	'\174'	'374'	'C4'
'\x80'	'0020'	'0070'	'\176'	'376'	'C6'
'\x81'	'0040'	'0100'	'\170'	'370'	'80'
'\x82'	'0007'	'0007'	'\177'	'377'	'7F'
'\x83'	'0001'	'0001'	'\177'	'377'	'7F'
'\x84'	'0002'	'0002'	'\177'	'377'	'7F'
'\x85'	'0003'	'0003'	'\177'	'377'	'7F'
'\x86'	'0004'	'0004'	'\177'	'377'	'7F'
'\x87'	'0005'	'0005'	'\177'	'377'	'7F'
'\x88'	'0006'	'0006'	'\177'	'377'	'7F'
'\x89'	'0007'	'0007'	'\177'	'377'	'7F'
'\x8A'	'0010'	'0040'	'\170'	'370'	'80'
'\x8B'	'0012'	'0050'	'\172'	'372'	'C2'
'\x8C'	'0015'	'0060'	'\174'	'374'	'C4'
'\x8D'	'0020'	'0070'	'\176'	'376'	'C6'
'\x8E'	'0040'	'0100'	'\170'	'370'	'80'
'\x8F'	'0007'	'0007'	'\177'	'377'	'7F'
'\x90'	'0001'	'0001'	'\177'	'377'	'7F'
'\x91'	'0002'	'0002'	'\177'	'377'	'7F'
'\x92'	'0003'	'0003'	'\177'	'377'	'7F'
'\x93'	'0004'	'0004'	'\177'	'377'	'7F'
'\x94'	'0005'	'0005'	'\177'	'377'	'7F'
'\x95'	'0006'	'0006'	'\177'	'377'	'7F'
'\x96'	'0007'	'0007'	'\177'	'377'	'7F'
'\x97'	'0010'	'0040'	'\170'	'370'	'80'
'\x98'	'0012'	'0050'	'\172'	'372'	'C2'
'\x99'	'0015'	'0060'	'\174'	'374'	'C4'
'\x9A'	'0020'	'0070'	'\176'	'376'	'C6'
'\x9B'	'0040'	'0100'	'\170'	'370'	'80'
'\x9C'	'0007'	'0007'	'\177'	'377'	'7F'
'\x9D'	'0001'	'0001'	'\177'	'377'	'7F'
'\x9E'	'0002'	'0002'	'\177'	'377'	'7F'
'\x9F'	'0003'	'0003'	'\177'	'377'	'7F'
'\xA0'	'0004'	'0004'	'\177'	'377'	'7F'
'\xA1'	'0005'	'0005'	'\177'	'377'	'7F'
'\xA2'	'0006'	'0006'	'\177'	'377'	'7F'
'\xA3'	'0007'	'0007'	'\177'	'377'	'7F'
'\xA4'	'0010'	'0040'	'\170'	'370'	'80'
'\xA5'	'0012'	'0050'	'\172'	'372'	'C2'
'\xA6'	'0015'	'0060'	'\174'	'374'	'C4'
'\xA7'	'0020'	'0070'	'\176'	'376'	'C6'
'\xA8'	'0040'	'0100'	'\170'	'370'	'80'
'\xA9'	'0007'	'0007'	'\177'	'377'	'7F'
'\xAA'	'0001'	'0001'	'\177'	'377'	'7F'
'\xAB'	'0002'	'0002'	'\177'	'377'	'7F'
'\xAC'	'0003'	'0003'	'\177'	'377'	'7F'
'\xAD'	'0004'	'0004'	'\177'	'377'	'7F'
'\xAE'	'0005'	'0005'	'\177'	'377'	'7F'
'\xAF'	'0006'	'0006'	'\177'	'377'	'7F'
'\xAF'	'0007'	'0007'	'\177'	'377'	'7F'

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	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	EOT	\009	41	29	49	SIGPOLL	\018	73	49	109	SIGPOLL	\019
10	0A	12	EM	\00A	42	2A	4A	SIGPOLL	\01A	74	4A	110	SIGPOLL	\01B
11	0B	13	END	\00B	43	2B	4B	SIGPOLL	\01B	75	4B	111	SIGPOLL	\01C
12	0C	14	ESC	\00C	44	2C	4C	SIGPOLL	\01C	76	4C	112	SIGPOLL	\01D
13	0D	15	SUSP	\00D	45	2D	4D	SIGPOLL	\01D	77	4D	113	SIGPOLL	\01E
14	0E	16	DC1	\00E	46	2E	4E	SIGPOLL	\01E	78	4E	114	SIGPOLL	\01F
15	0F	17	DC2	\00F	47	2F	4F	SIGPOLL	\01F	79	4F	115	SIGPOLL	\020
16	10	20	DC3	\010	48	30	50	SIGPOLL	\01F	80	50	116	SIGPOLL	\021
17	11	21	DC4	\011	49	31	51	SIGPOLL	\01F	81	51	117	SIGPOLL	\022
18	12	22	DC5	\012	50	32	52	SIGPOLL	\01F	82	52	118	SIGPOLL	\023
19	13	23	DC6	\013	51	33	53	SIGPOLL	\01F	83	53	119	SIGPOLL	\024
20	14	24	DC7	\014	52	34	54	SIGPOLL	\01F	84	54	120	SIGPOLL	\025
21	15	25	DC8	\015	53	35	55	SIGPOLL	\01F	85	55	121	SIGPOLL	\026
22	16	26	DC9	\016	54	36	56	SIGPOLL	\01F	86	56	122	SIGPOLL	\027
23	17	27	DC10	\017	55	37	57	SIGPOLL	\01F	87	57	123	SIGPOLL	\028
24	18	28	DC11	\018	56	38	58	SIGPOLL	\01F	88	58	124	SIGPOLL	\029
25	19	29	DC12	\019	57	39	59	SIGPOLL	\01F	89	59	125	SIGPOLL	\02A
26	1A	2A	DC13	\01A	58	3A	5A	SIGPOLL	\01F	90	5A	126	SIGPOLL	\02B
27	1B	2B	DC14	\01B	59	3B	5B	SIGPOLL	\01F	91	5B	127	SIGPOLL	\02C
28	1C	2C	DC15	\01C	5A	3C	5C	SIGPOLL	\01F	92	5C	128	SIGPOLL	\02D
29	1D	2D	DC16	\01D	5B	3D	5D	SIGPOLL	\01F	93	5D	129	SIGPOLL	\02E
30	1E	2E	DC17	\01E	5C	3E	5E	SIGPOLL	\01F	94	5E	130	SIGPOLL	\02F
31	1F	2F	DC18	\01F	5D	3F	5F	SIGPOLL	\01F	95	5F	131	SIGPOLL	\030
32	20	30	DC19	\020	5E	40	60	SIGPOLL	\01F	96	60	132	SIGPOLL	\031
33	21	31	DC1A	\021	5F	41	61	SIGPOLL	\01F	97	61	133	SIGPOLL	\032
34	22	32	DC1B	\022	60	42	62	SIGPOLL	\01F	98	62	134	SIGPOLL	\033
35	23	33	DC1C	\023	61	43	63	SIGPOLL	\01F	99	63	135	SIGPOLL	\034
36	24	34	DC1D	\024	62	44	64	SIGPOLL	\01F	100	64	136	SIGPOLL	\035
37	25	35	DC1E	\025	63	45	65	SIGPOLL	\01F	101	65	137	SIGPOLL	\036
38	26	36	DC1F	\026	64	46	66	SIGPOLL	\01F	102	66	138	SIGPOLL	\037
39	27	37	DC20	\027	65	47	67	SIGPOLL	\01F	103	67	139	SIGPOLL	\038
40	28	38	DC21	\028	66	48	68	SIGPOLL	\01F	104	68	140	SIGPOLL	\039
41	29	39	DC22	\029	67	49	69	SIGPOLL	\01F	105	69	141	SIGPOLL	\03A
42	2A	3A	DC23	\02A	68	4A	6A	SIGPOLL	\01F	106	6A	142	SIGPOLL	\03B
43	2B	3B	DC24	\02B	69	4B	6B	SIGPOLL	\01F	107	6B	143	SIGPOLL	\03C
44	2C	3C	DC25	\02C	6A	4C	6C	SIGPOLL	\01F	108	6C	144	SIGPOLL	\03D
45	2D	3D	DC26	\02D	6B	4D	6D	SIGPOLL	\01F	109	6D	145	SIGPOLL	\03E
46	2E	3E	DC27	\02E	6C	4E	6E	SIGPOLL	\01F	110	6E	146	SIGPOLL	\03F
47	2F	3F	DC28	\02F	6D	4F	6F	SIGPOLL	\01F	111	6F	147	SIGPOLL	\040
48	30	40	DC29	\030	6E	50	70	SIGPOLL	\01F	112	70	148	SIGPOLL	\041
49	31	41	DC2A	\031	6F	51	71	SIGPOLL	\01F	113	71	149	SIGPOLL	\042
50	32	42	DC2B	\032	70	52	72	SIGPOLL	\01F	114	72	150	SIGPOLL	\043
51	33	43	DC2C	\033	71	53	73	SIGPOLL	\01F	115	73	151	SIGPOLL	\044
52	34	44	DC2D	\034	72	54	74	SIGPOLL	\01F	116	74	152	SIGPOLL	\045
53	35	45	DC2E	\035	73	55	75	SIGPOLL	\01F	117	75	153	SIGPOLL	\046
54	36	46	DC2F	\036	74	56	76	SIGPOLL	\01F	118	76	154	SIGPOLL	\047
55	37	47	DC30	\037	75	57	77	SIGPOLL	\01F	119	77	155	SIGPOLL	\048
56	38	48	DC31	\038	76	58	78	SIGPOLL	\01F	120	78	156	SIGPOLL	\049
57	39	49	DC32	\039	77	59	79	SIGPOLL	\01F	121	79	157	SIGPOLL	\04A
58	3A	4A	DC33	\03A	78	5A	7A	SIGPOLL	\01F	122	7A	158	SIGPOLL	\04B
59	3B	4B	DC34	\03B	79	5B	7B	SIGPOLL	\01F	123	7B	159	SIGPOLL	\04C
60	3C	4C	DC35	\03C	7A	5C	7C	SIGPOLL	\01F	124	7C	160	SIGPOLL	\04D
61	3D	4D	DC36	\03D	7B	5D	7D	SIGPOLL	\01F	125	7D	161	SIGPOLL	\04E
62	3E	4E	DC37	\03E	7C	5E	7E	SIGPOLL	\01F	126	7E	162	SIGPOLL	\04F
63	3F	4F	DC38	\03F	7D	5F	7F	SIGPOLL	\01F	127	7F	163	SIGPOLL	\050
64	40	50	DC39	\040	7E	60	80	SIGPOLL	\01F	128	80	164	SIGPOLL	\051
65	41	51	DC3A	\041	7F	61	81	SIGPOLL	\01F	129	81	165	SIGPOLL	\052
66	42	52	DC3B	\042	80	62	82	SIGPOLL	\01F	130	82	166	SIGPOLL	\053
67	43	53	DC3C	\043	81	63	83	SIGPOLL	\01F	131	83	167	SIGPOLL	\054
68	44	54	DC3D	\044	82	64	84	SIGPOLL	\01F	132	84	168	SIGPOLL	\055
69	45	55	DC3E	\045	83	65	85	SIGPOLL	\01F	133	85	169	SIGPOLL	\056
70	46	56	DC3F	\046	84	66	86	SIGPOLL	\01F	134	86	170	SIGPOLL	\057
71	47	57	DC40	\047	85	67	87	SIGPOLL	\01F	135	87	171	SIGPOLL	\058
72	48	58	DC41	\048	86	68	88	SIGPOLL	\01F	136	88	172	SIGPOLL	\059
73	49	59	DC42	\049	87	69	89	SIGPOLL	\01F	137	89	173	SIGPOLL	\05A
74	4A	5A	DC43	\04A	88	6A	8A	SIGPOLL	\01F	138	8A	174	SIGPOLL	\05B
75	4B	5B	DC44	\04B	89	6B	8B	SIGPOLL	\01F	139	8B	175	SIGPOLL	\05C
76	4C	5C	DC45	\04C	8A	6C	8C	SIGPOLL	\01F	140	8C	176	SIGPOLL	\05D
77	4D	5D	DC46	\04D	8B	6D	8D	SIGPOLL	\01F	141	8D	177	SIGPOLL	\05E
78	4E	5E	DC47	\04E	8C	6E	8E	SIGPOLL	\01F	142	8E	178	SIGPOLL	\05F
79	4F	5F	DC48	\04F	8D	6F	8F	SIGPOLL	\01F	143	8F	179	SIGPOLL	\060
80	50	60	DC49	\050	8E	70	90	SIGPOLL	\01F	144	90	180	SIGPOLL	\061
81	51	61	DC4A	\051	8F	71	91	SIGPOLL	\01F	145	91	181	SIGPOLL	\062
82	52	62	DC4B	\052	90	72	92	SIGPOLL	\01F	146	92	182	SIGPOLL	\063
83	53	63	DC4C	\053	91	73	93	SIGPOLL	\01F	147	93	183	SIGPOLL	\064
84	54	64	DC4D	\054	92	74	94	SIGPOLL	\01F	148	94	184	SIGPOLL	\065
85	55	65	DC4E	\055	93	75	95	SIGPOLL	\01F	149	95	185	SIGPOLL	\066
86	56	66	DC4F	\056	94	76	96	SIGPOLL	\01F	150	96	186	SIGPOLL	\067
87	57	67	DC50	\057	95	77	97	SIGPOLL	\01F	151	97	187	SIGPOLL	\068
88	58	68	DC51	\058	96	78	98	SIGPOLL	\01F	152	98	188	SIGPOLL	\069
89	59	69	DC52	\059	97	79	99	SIGPOLL	\01F	153	99	189	SIGPOLL	\06A
90	5A	6A	DC53	\05A	98	7A	9A	SIGPOLL	\01F	154	9A	190	SIGPOLL	\06B
91	5B	6B	DC54	\05B	99	7B	9B	SIGPOLL	\01F	155	9B	191	SIGPOLL	\06C
92	5C	6C	DC55	\05C	9A	7C	9C	SIGPOLL	\01F	156	9C	192	SIGPOLL	\06D
93	5D	6D	DC56	\05D	9B	7D	9D	SIGPOLL	\01F	157	9D	193	SIGPOLL	\06E
94	5E	6E	DC57	\05E	9C	7E	9E	SIGPOLL	\01F	158	9E	194	SIGPOLL	\06F
95	5F	6F	DC58	\05F	9D	7F	9F	SIGPOLL	\01F	159	9F	195	SIGPOLL	\070
96	60	70	DC59	\060	9E	80	A0	SIGPOLL	\01F	160	A0	196	SIGPOLL	\071
97	61	71	DC5A	\061	9F	81	A1	SIGPOLL	\01F	161	A1	197	SIGPOLL	\072
98	62	72	DC5B	\062	A0	82	A2	SIGPOLL	\01F	162	A2	198	SIGPOLL	\073
99	63	73	DC5C	\063	A1	83	A3	SIGPOLL	\01F	163	A3	199	SIGPOLL	\074
100	64	74	DC5D	\064	A2	84	A4	SIGPOLL	\01F	164	A4	200	SIGPOLL	\075
101	65	75	DC5E	\065	A3	85	A5	SIGPOLL	\01F	165	A5	201	SIGPOLL	\076
102	66	76	DC5F	\066	A4	86	A6	SIGPOLL	\01F	166	A6			

- `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(ch, 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 `chr()`

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 `chr()`
- ▶ 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 `chr()`
- ▶ 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...