

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

- This lecture will be recorded

Frequently Asked Questions

From email

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry!*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades.*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.
- When is the midterm?

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.
- When is the midterm?
There is no midterm. Instead there's required weekly Lab Quizzes and daily programming assignments.

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.
- When is the midterm?
There is no midterm. Instead there's required weekly Lab Quizzes and daily programming assignments.
- I missed class. Do you need documentation?

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.
- When is the midterm?
There is no midterm. Instead there's required weekly Lab 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 email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.
- When is the midterm?
There is no midterm. Instead there's required weekly Lab 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 email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.
- When is the midterm?
There is no midterm. Instead there's required weekly Lab 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 email studenthelpdesk@hunter.cuny.edu to update your email on Blackboard to one you check regularly.*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizze, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have 15 minutes to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.
- When is the midterm?
There is no midterm. Instead there's required weekly Lab 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 email studenthelpdesk@hunter.cuny.edu to update your email on Blackboard to one you check regularly.*

Today's Topics



- **For-loops**
- `range()`
- Variables
- Characters
- Strings
- Guest: Elise Harris (Advising, Clubs, Internships and more)

In Pairs or Triples...

Some review and some novel challenges:

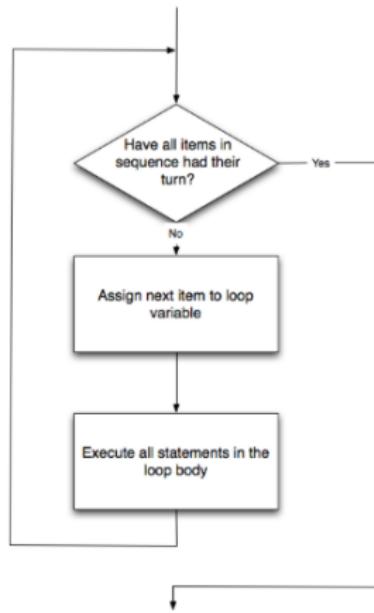
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

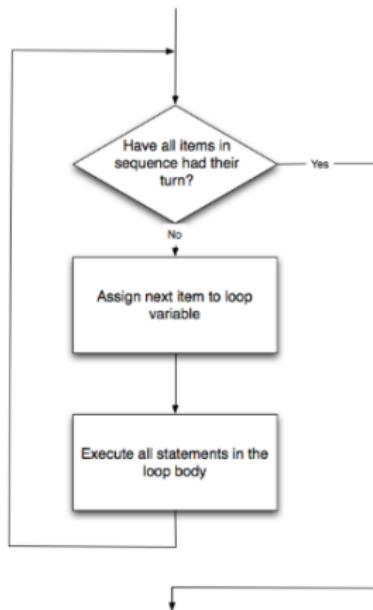
for-loop



```
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
- Guest: Elise Harris (Advising, Clubs, Internships and more)

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:
 $[start, start+step, start+2*step\dots, last]$
(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
- Guest: Elise Harris (Advising, Clubs, Internships and more)

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
- Guest: Elise Harris (Advising, Clubs, Internships and more)

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.)

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

ASCII TABLE												
Decimal	Hex	Char	Octal	Binary	Hex	Char	Octal	Binary	Hex	Char	Octal	Binary
0	00	\0	000	0000000000000000	00	\0	000	0000000000000000	00	\0	000	0000000000000000
1	01	\1	001	0000000000000001	01	\1	001	0000000000000001	01	\1	001	0000000000000001
2	02	\2	010	0000000000000010	02	\2	010	0000000000000010	02	\2	010	0000000000000010
3	03	\3	011	0000000000000011	03	\3	011	0000000000000011	03	\3	011	0000000000000011
4	04	\4	100	0000000000000100	04	\4	100	0000000000000100	04	\4	100	0000000000000100
5	05	\5	101	0000000000000101	05	\5	101	0000000000000101	05	\5	101	0000000000000101
6	06	\6	110	0000000000000110	06	\6	110	0000000000000110	06	\6	110	0000000000000110
7	07	\7	111	0000000000000111	07	\7	111	0000000000000111	07	\7	111	0000000000000111
8	08	\8	1000	0000000000001000	08	\8	1000	0000000000001000	08	\8	1000	0000000000001000
9	09	\9	1001	0000000000001001	09	\9	1001	0000000000001001	09	\9	1001	0000000000001001
10	0A	\a	1010	0000000000001010	0A	\a	1010	0000000000001010	0A	\a	1010	0000000000001010
11	0B	\b	1011	0000000000001011	0B	\b	1011	0000000000001011	0B	\b	1011	0000000000001011
12	0C	\c	1100	0000000000001100	0C	\c	1100	0000000000001100	0C	\c	1100	0000000000001100
13	0D	\d	1101	0000000000001101	0D	\d	1101	0000000000001101	0D	\d	1101	0000000000001101
14	0E	\e	1110	0000000000001110	0E	\e	1110	0000000000001110	0E	\e	1110	0000000000001110
15	0F	\f	1111	0000000000001111	0F	\f	1111	0000000000001111	0F	\f	1111	0000000000001111
16	10	\n	10000	0000000000010000	10	\n	10000	0000000000010000	10	\n	10000	0000000000010000
17	11	\v	10001	0000000000010001	11	\v	10001	0000000000010001	11	\v	10001	0000000000010001
18	12	\t	10010	0000000000010010	12	\t	10010	0000000000010010	12	\t	10010	0000000000010010
19	13	\r	10011	0000000000010011	13	\r	10011	0000000000010011	13	\r	10011	0000000000010011
20	14	\012	10100	0000000000010100	14	\012	10100	0000000000010100	14	\012	10100	0000000000010100
21	15	\015	10101	0000000000010101	15	\015	10101	0000000000010101	15	\015	10101	0000000000010101
22	16	\016	10110	0000000000010110	16	\016	10110	0000000000010110	16	\016	10110	0000000000010110
23	17	\017	10111	0000000000010111	17	\017	10111	0000000000010111	17	\017	10111	0000000000010111
24	18	\020	11000	0000000000011000	18	\020	11000	0000000000011000	18	\020	11000	0000000000011000
25	19	\021	11001	0000000000011001	19	\021	11001	0000000000011001	19	\021	11001	0000000000011001
26	1A	\022	11010	0000000000011010	1A	\022	11010	0000000000011010	1A	\022	11010	0000000000011010
27	1B	\023	11011	0000000000011011	1B	\023	11011	0000000000011011	1B	\023	11011	0000000000011011
28	1C	\024	11100	0000000000011100	1C	\024	11100	0000000000011100	1C	\024	11100	0000000000011100
29	1D	\025	11101	0000000000011101	1D	\025	11101	0000000000011101	1D	\025	11101	0000000000011101
30	1E	\026	11110	0000000000011110	1E	\026	11110	0000000000011110	1E	\026	11110	0000000000011110
31	1F	\027	11111	0000000000011111	1F	\027	11111	0000000000011111	1F	\027	11111	0000000000011111
32	20	\040	100000	0000000000100000	20	\040	100000	0000000000100000	20	\040	100000	0000000000100000
33	21	\041	100001	0000000000100001	21	\041	100001	0000000000100001	21	\041	100001	0000000000100001
34	22	\042	100010	0000000000100010	22	\042	100010	0000000000100010	22	\042	100010	0000000000100010
35	23	\043	100011	0000000000100011	23	\043	100011	0000000000100011	23	\043	100011	0000000000100011
36	24	\044	100100	0000000000100100	24	\044	100100	0000000000100100	24	\044	100100	0000000000100100
37	25	\045	100101	0000000000100101	25	\045	100101	0000000000100101	25	\045	100101	0000000000100101
38	26	\046	100110	0000000000100110	26	\046	100110	0000000000100110	26	\046	100110	0000000000100110
39	27	\047	100111	0000000000100111	27	\047	100111	0000000000100111	27	\047	100111	0000000000100111
40	28	\050	101000	0000000000101000	28	\050	101000	0000000000101000	28	\050	101000	0000000000101000
41	29	\051	101001	0000000000101001	29	\051	101001	0000000000101001	29	\051	101001	0000000000101001
42	2A	\052	101010	0000000000101010	2A	\052	101010	0000000000101010	2A	\052	101010	0000000000101010
43	2B	\053	101011	0000000000101011	2B	\053	101011	0000000000101011	2B	\053	101011	0000000000101011
44	2C	\054	101100	0000000000101100	2C	\054	101100	0000000000101100	2C	\054	101100	0000000000101100
45	2D	\055	101101	0000000000101101	2D	\055	101101	0000000000101101	2D	\055	101101	0000000000101101
46	2E	\056	101110	0000000000101110	2E	\056	101110	0000000000101110	2E	\056	101110	0000000000101110
47	2F	\057	101111	0000000000101111	2F	\057	101111	0000000000101111	2F	\057	101111	0000000000101111
48	30	\060	110000	0000000001000000	30	\060	110000	0000000001000000	30	\060	110000	0000000001000000
49	31	\061	110001	0000000001000001	31	\061	110001	0000000001000001	31	\061	110001	0000000001000001
50	32	\062	110010	0000000001000010	32	\062	110010	0000000001000010	32	\062	110010	0000000001000010
51	33	\063	110011	0000000001000011	33	\063	110011	0000000001000011	33	\063	110011	0000000001000011
52	34	\064	110100	0000000001000100	34	\064	110100	0000000001000100	34	\064	110100	0000000001000100
53	35	\065	110101	0000000001000101	35	\065	110101	0000000001000101	35	\065	110101	0000000001000101
54	36	\066	110110	0000000001000110	36	\066	110110	0000000001000110	36	\066	110110	0000000001000110
55	37	\067	110111	0000000001000111	37	\067	110111	0000000001000111	37	\067	110111	0000000001000111
56	38	\070	111000	0000000001001000	38	\070	111000	0000000001001000	38	\070	111000	0000000001001000
57	39	\071	111001	0000000001001001	39	\071	111001	0000000001001001	39	\071	111001	0000000001001001
58	3A	\072	111010	0000000001001010	3A	\072	111010	0000000001001010	3A	\072	111010	0000000001001010
59	3B	\073	111011	0000000001001011	3B	\073	111011	0000000001001011	3B	\073	111011	0000000001001011
60	3C	\074	111100	0000000001001100	3C	\074	111100	0000000001001100	3C	\074	111100	0000000001001100
61	3D	\075	111101	0000000001001101	3D	\075	111101	0000000001001101	3D	\075	111101	0000000001001101
62	3E	\076	111110	0000000001001110	3E	\076	111110	0000000001001110	3E	\076	111110	0000000001001110
63	3F	\077	111111	0000000001001111	3F	\077	111111	0000000001001111	3F	\077	111111	0000000001001111
64	40	\080	1000000	0000000010000000	40	\080	1000000	0000000010000000	40	\080	1000000	0000000010000000
65	41	\081	1000001	0000000010000001	41	\081	1000001	0000000010000001	41	\081	1000001	0000000010000001
66	42	\082	1000010	0000000010000010	42	\082	1000010	0000000010000010	42	\082	1000010	0000000010000010
67	43	\083	1000011	0000000010000011	43	\083	1000011	0000000010000011	43	\083	1000011	0000000010000011
68	44	\084	1000100	0000000010000100	44	\084	1000100	0000000010000100	44	\084	1000100	0000000010000100
69	45	\085	1000101	0000000010000101	45	\085	1000101	0000000010000101	45	\085	1000101	0000000010000101
70	46	\086	1000110	0000000010000110	46	\086	1000110	0000000010000110	46	\086	1000110	0000000010000110
71	47	\087	1000111	0000000010000111	47	\087	1000111	0000000010000111	47	\087	1000111	0000000010000111
72	48	\090	1001000	0000000010001000	48	\090	1001000	0000000010001000	48	\090	1001000	0000000010001000
73	49	\091	1001001	0000000010001001	49	\091	1001001	0000000010001001	49	\091	1001001	0000000010001001
74	4A	\092	1001010	0000000010001010	4A	\092	1001010	0000000010001010	4A	\092	1001010	0000000010001010
75	4B	\093	1001011	0000000010001011	4B	\093	1001011	0000000010001011	4B	\093	1001011	0000000010001011
76	4C	\094	1001100	0000000010001100	4C	\094	1001100	0000000010001100	4C	\094	1001100	0000000010001100
77	4D	\095	1001101	0000000010001101	4D	\095	1001101	0000000010001101	4D	\095	1001101	0000000010001101
78	4E	\096	1001110	0000000010001110	4E	\096	1001110	0000000010001110	4E	\096	1001110	0000000010001110
79	4F	\097	1001111	0000000010001111	4F	\097	1001111	0000000010001111	4F	\097	1001111	0000000010001111
80	50	\100	101									

Converting from Character to Code:

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

ASCII TABLE								
Decimal	Hex	Char	Octal	Hex	Char	Octal	Hex	Char
0	00	\0	000	00	\0	000	00	\0
1	01	\1	001	01	\1	001	01	\1
2	02	\2	002	02	\2	002	02	\2
3	03	\3	003	03	\3	003	03	\3
4	04	\4	004	04	\4	004	04	\4
5	05	\5	005	05	\5	005	05	\5
6	06	\6	006	06	\6	006	06	\6
7	07	\7	007	07	\7	007	07	\7
8	08	\8	010	08	\8	010	08	\8
9	09	\9	011	09	\9	011	09	\9
10	0A	\n	012	0A	\n	012	0A	\n
11	0B	\v	013	0B	\v	013	0B	\v
12	0C	\f	014	0C	\f	014	0C	\f
13	0D	\r	015	0D	\r	015	0D	\r
14	0E	\t	016	0E	\t	016	0E	\t
15	0F	\b	017	0F	\b	017	0F	\b
16	10	\012	020	10	\012	020	10	\012
17	11	\013	021	11	\013	021	11	\013
18	12	\014	022	12	\014	022	12	\014
19	13	\015	023	13	\015	023	13	\015
20	14	\016	024	14	\016	024	14	\016
21	15	\017	025	15	\017	025	15	\017
22	16	\020	026	16	\020	026	16	\020
23	17	\021	027	17	\021	027	17	\021
24	18	\022	028	18	\022	028	18	\022
25	19	\023	029	19	\023	029	19	\023
26	1A	\024	02A	1A	\024	02A	1A	\024
27	1B	\025	02B	1B	\025	02B	1B	\025
28	1C	\026	02C	1C	\026	02C	1C	\026
29	1D	\027	02D	1D	\027	02D	1D	\027
30	1E	\030	02E	1E	\030	02E	1E	\030
31	1F	\031	02F	1F	\031	02F	1F	\031
32	20	\032	030	20	\032	030	20	\032
33	21	\033	031	21	\033	031	21	\033
34	22	\034	032	22	\034	032	22	\034
35	23	\035	033	23	\035	033	23	\035
36	24	\036	034	24	\036	034	24	\036
37	25	\037	035	25	\037	035	25	\037
38	26	\040	036	26	\040	036	26	\040
39	27	\041	037	27	\041	037	27	\041
40	28	\042	038	28	\042	038	28	\042
41	29	\043	039	29	\043	039	29	\043
42	2A	\044	03A	2A	\044	03A	2A	\044
43	2B	\045	03B	2B	\045	03B	2B	\045
44	2C	\046	03C	2C	\046	03C	2C	\046
45	2D	\047	03D	2D	\047	03D	2D	\047
46	2E	\050	03E	2E	\050	03E	2E	\050
47	2F	\051	03F	2F	\051	03F	2F	\051
48	30	\052	040	30	\052	040	30	\052
49	31	\053	041	31	\053	041	31	\053
50	32	\054	042	32	\054	042	32	\054
51	33	\055	043	33	\055	043	33	\055
52	34	\056	044	34	\056	044	34	\056
53	35	\057	045	35	\057	045	35	\057
54	36	\060	046	36	\060	046	36	\060
55	37	\061	047	37	\061	047	37	\061
56	38	\062	048	38	\062	048	38	\062
57	39	\063	049	39	\063	049	39	\063
58	3A	\064	04A	3A	\064	04A	3A	\064
59	3B	\065	04B	3B	\065	04B	3B	\065
60	3C	\066	04C	3C	\066	04C	3C	\066
61	3D	\067	04D	3D	\067	04D	3D	\067
62	3E	\070	04E	3E	\070	04E	3E	\070
63	3F	\071	04F	3F	\071	04F	3F	\071
64	40	\072	050	40	\072	050	40	\072
65	41	\073	051	41	\073	051	41	\073
66	42	\074	052	42	\074	052	42	\074
67	43	\075	053	43	\075	053	43	\075
68	44	\076	054	44	\076	054	44	\076
69	45	\077	055	45	\077	055	45	\077
70	46	\080	056	46	\080	056	46	\080
71	47	\081	057	47	\081	057	47	\081
72	48	\082	058	48	\082	058	48	\082
73	49	\083	059	49	\083	059	49	\083
74	4A	\084	05A	4A	\084	05A	4A	\084
75	4B	\085	05B	4B	\085	05B	4B	\085
76	4C	\086	05C	4C	\086	05C	4C	\086
77	4D	\087	05D	4D	\087	05D	4D	\087
78	4E	\090	05E	4E	\090	05E	4E	\090
79	4F	\091	05F	4F	\091	05F	4F	\091
80	50	\092	060	50	\092	060	50	\092
81	51	\093	061	51	\093	061	51	\093
82	52	\094	062	52	\094	062	52	\094
83	53	\095	063	53	\095	063	53	\095
84	54	\096	064	54	\096	064	54	\096
85	55	\097	065	55	\097	065	55	\097
86	56	\0A0	066	56	\0A0	066	56	\0A0
87	57	\0A1	067	57	\0A1	067	57	\0A1
88	58	\0A2	068	58	\0A2	068	58	\0A2
89	59	\0A3	069	59	\0A3	069	59	\0A3
90	5A	\0A4	06A	5A	\0A4	06A	5A	\0A4
91	5B	\0A5	06B	5B	\0A5	06B	5B	\0A5
92	5C	\0A6	06C	5C	\0A6	06C	5C	\0A6
93	5D	\0A7	06D	5D	\0A7	06D	5D	\0A7
94	5E	\0B0	06E	5E	\0B0	06E	5E	\0B0
95	5F	\0B1	06F	5F	\0B1	06F	5F	\0B1
96	60	\0B2	070	60	\0B2	070	60	\0B2
97	61	\0B3	071	61	\0B3	071	61	\0B3
98	62	\0B4	072	62	\0B4	072	62	\0B4
99	63	\0B5	073	63	\0B5	073	63	\0B5
100	64	\0B6	074	64	\0B6	074	64	\0B6
101	65	\0B7	075	65	\0B7	075	65	\0B7
102	66	\0C0	076	66	\0C0	076	66	\0C0
103	67	\0C1	077	67	\0C1	077	67	\0C1
104	68	\0C2	078	68	\0C2	078	68	\0C2
105	69	\0C3	079	69	\0C3	079	69	\0C3
106	6A	\0C4	07A	6A	\0C4	07A	6A	\0C4
107	6B	\0C5	07B	6B	\0C5	07B	6B	\0C5
108	6C	\0C6	07C	6C	\0C6	07C	6C	\0C6
109	6D	\0C7	07D	6D	\0C7	07D	6D	\0C7
110	6E	\0D0	07E	6E	\0D0	07E	6E	\0D0
111	6F	\0D1	07F	6F	\0D1	07F	6F	\0D1
112	70	\0D2	080	70	\0D2	080	70	\0D2
113	71	\0D3	081	71	\0D3	081	71	\0D3
114	72	\0D4	082	72	\0D4	082	72	\0D4
115	73	\0D5	083	73	\0D5	083	73	\0D5
116	74	\0D6	084	74	\0D6	084	74	\0D6
117	75	\0D7	085	75	\0D7	085	75	\0D7
118	76	\0E0	086	76	\0E0	086	76	\0E0
119	77	\0E1	087	77	\0E1	087	77	\0E1
120	78	\0E2	088	78	\0E2	088	78	\0E2
121	79	\0E3	089	79	\0E3	089	79	\0E3
122	7A	\0E4	08A	7A	\0E4	08A	7A	\0E4
123	7B	\0E5	08B	7B	\0E5	08B	7B	\0E5
124	7C	\0E6	08C	7C	\0E6	08C	7C	\0E6
125	7D	\0E7	08D	7D	\0E7	08D	7D	\0E7
126	7E	\0F0	08E	7E	\0F0	08E	7E	\0F0
127	7F	\0F1	08F	7F	\0F1	08F	7F	\0F1
128	80	\0F2	090	80	\0F2	090	80	\0F2
129	81	\0F3	091	81	\0F3	091	81	\0F3
130	82	\0F4	092	82	\0F4	092	82	\0F4
131	83	\0F5	093	83	\0F5	093	83	\0F5
132	84	\0F6	094	84	\0F6	094	84	\0F6
133	85	\0F7	095	85	\0F7	095	85	\0F7
134	86	\0F8	096	86	\0F8	096	86	\0F8
135	87	\0F9	097	87	\0F9	097	87	\0F9
136	88	\0FA	098	88	\0FA	098	88	\0FA
137	89	\0FB	099	89	\0FB	099	89	\0FB
138	8A	\0FC	09A	8A	\0FC	09A	8A	\0FC
139	8B	\0FD	09B	8B	\0FD	09B	8B	\0FD
140	8C	\0FE	09C	8C	\0FE	09C	8C	\0FE
141	8D	\0FF	09D	8D	\0FF	09D	8D	\0FF
142	8E	\00	09E	8E	\00	09E	8E	\00
143	8F	\01	09F	8F	\01	09F	8F	\01
144	90	\02	0A0	90	\02	0A0	90	\02
145	91	\03	0A1	91	\03	0A1	91	\03
146	92	\04	0A2	92	\04	0A2	92	\04
147	93	\05	0A3	93	\05	0A3	93	\05
148	94	\06	0A4	94	\06	0A4	94	\06
149	95	\07	0A5	95	\07	0A5	95	\07
150	96	\08	0A6	96	\08	0A6	96	\08
151	97	\09	0A7	97	\09	0A7	97	\09
152	98	\0A	0A8	98	\0A	0A8	98	\0A
153	99	\0B	0A9	99	\0B	0A9	99	\0B
154	9A	\0C	0AA	9A	\0C	0AA	9A	\0C
155	9B	\0D	0AB	9B	\0D	0AB	9B	\0D
156	9C	\0E	0AC	9C	\0E	0AC	9C	\0E
157	9D	\0F	0AD	9D	\0F	0AD	9D	\0F
158	9E	\00	0AE	9E	\00	0AE	9E	\00
159	9F	\01	0AF	9F	\01	0AF	9F	\01
160	9A0	\02	0A0	9A0	\02	0A0	9A0	\02
161	9A1	\03	0A1	9A1	\03	0A1	9A1	\03
162	9A2	\04	0A2	9A2	\04	0A2	9A2	\04
163	9A3	\05	0A3	9A3	\05	0A3	9A3	\05
164	9A4	\06	0A4	9A4	\06	0A4	9A4	\06
165	9A5	\07	0A5	9A5	\07	0A5	9A5	\07
166	9A6	\08	0A6	9A6	\08	0A6	9A6	\08
167	9A7	\09	0A7	9A7	\09	0A7	9A7	\09
168	9A8	\0A						

- `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.)

ASCII TABLE									
Decimal	Hex	Char	Octal	Binary	Decimal	Hex	Char	Octal	Binary
0	00	\0	000	00000000	0	00	\0	000	00000000
1	01	\1	001	00000001	1	01	\1	001	00000001
2	02	\2	010	00000010	2	02	\2	010	00000010
3	03	\3	011	00000011	3	03	\3	011	00000011
4	04	\4	020	00000100	4	04	\4	020	00000100
5	05	\5	021	00000101	5	05	\5	021	00000101
6	06	\6	030	00000110	6	06	\6	030	00000110
7	07	\7	031	00000111	7	07	\7	031	00000111
8	08	\8	040	00001000	8	08	\8	040	00001000
9	09	\9	041	00001001	9	09	\9	041	00001001
10	0A	\n	050	00001010	10	0A	\n	050	00001010
11	0B	\v	051	00001011	11	0B	\v	051	00001011
12	0C	\f	060	00001100	12	0C	\f	060	00001100
13	0D	\r	061	00001101	13	0D	\r	061	00001101
14	0E	\t	070	00001110	14	0E	\t	070	00001110
15	0F	\b	071	00001111	15	0F	\b	071	00001111
16	10	\012	080	00010000	16	10	\012	080	00010000
17	11	\013	081	00010001	17	11	\013	081	00010001
18	12	\014	082	00010010	18	12	\014	082	00010010
19	13	\015	083	00010011	19	13	\015	083	00010011
20	14	\016	090	00010100	20	14	\016	090	00010100
21	15	\017	091	00010101	21	15	\017	091	00010101
22	16	\020	092	00010110	22	16	\020	092	00010110
23	17	\021	093	00010111	23	17	\021	093	00010111
24	18	\022	100	00011000	24	18	\022	100	00011000
25	19	\023	101	00011001	25	19	\023	101	00011001
26	1A	\024	102	00011010	26	1A	\024	102	00011010
27	1B	\025	103	00011011	27	1B	\025	103	00011011
28	1C	\026	110	00011100	28	1C	\026	110	00011100
29	1D	\027	111	00011101	29	1D	\027	111	00011101
30	1E	\030	120	00011110	30	1E	\030	120	00011110
31	1F	\031	121	00011111	31	1F	\031	121	00011111
32	20	\040	140	00100000	32	20	\040	140	00100000
33	21	\041	141	00100001	33	21	\041	141	00100001
34	22	\042	142	00100010	34	22	\042	142	00100010
35	23	\043	143	00100011	35	23	\043	143	00100011
36	24	\044	150	00100100	36	24	\044	150	00100100
37	25	\045	151	00100101	37	25	\045	151	00100101
38	26	\046	152	00100110	38	26	\046	152	00100110
39	27	\047	153	00100111	39	27	\047	153	00100111
40	28	\050	160	00101000	40	28	\050	160	00101000
41	29	\051	161	00101001	41	29	\051	161	00101001
42	2A	\052	162	00101010	42	2A	\052	162	00101010
43	2B	\053	163	00101011	43	2B	\053	163	00101011
44	2C	\054	170	00101100	44	2C	\054	170	00101100
45	2D	\055	171	00101101	45	2D	\055	171	00101101
46	2E	\056	172	00101110	46	2E	\056	172	00101110
47	2F	\057	173	00101111	47	2F	\057	173	00101111
48	30	\060	180	00110000	48	30	\060	180	00110000
49	31	\061	181	00110001	49	31	\061	181	00110001
50	32	\062	182	00110010	50	32	\062	182	00110010
51	33	\063	183	00110011	51	33	\063	183	00110011
52	34	\064	190	00110100	52	34	\064	190	00110100
53	35	\065	191	00110101	53	35	\065	191	00110101
54	36	\066	192	00110110	54	36	\066	192	00110110
55	37	\067	193	00110111	55	37	\067	193	00110111
56	38	\070	200	00111000	56	38	\070	200	00111000
57	39	\071	201	00111001	57	39	\071	201	00111001
58	3A	\072	202	00111010	58	3A	\072	202	00111010
59	3B	\073	203	00111011	59	3B	\073	203	00111011
60	3C	\074	210	00111100	60	3C	\074	210	00111100
61	3D	\075	211	00111101	61	3D	\075	211	00111101
62	3E	\076	212	00111110	62	3E	\076	212	00111110
63	3F	\077	213	00111111	63	3F	\077	213	00111111
64	40	\080	220	01000000	64	40	\080	220	01000000
65	41	\081	221	01000001	65	41	\081	221	01000001
66	42	\082	222	01000010	66	42	\082	222	01000010
67	43	\083	223	01000011	67	43	\083	223	01000011
68	44	\084	230	01000100	68	44	\084	230	01000100
69	45	\085	231	01000101	69	45	\085	231	01000101
70	46	\086	232	01000110	70	46	\086	232	01000110
71	47	\087	233	01000111	71	47	\087	233	01000111
72	48	\090	240	01001000	72	48	\090	240	01001000
73	49	\091	241	01001001	73	49	\091	241	01001001
74	4A	\092	242	01001010	74	4A	\092	242	01001010
75	4B	\093	243	01001011	75	4B	\093	243	01001011
76	4C	\094	250	01001100	76	4C	\094	250	01001100
77	4D	\095	251	01001101	77	4D	\095	251	01001101
78	4E	\096	252	01001110	78	4E	\096	252	01001110
79	4F	\097	253	01001111	79	4F	\097	253	01001111
80	50	\0A0	260	01010000	80	50	\0A0	260	01010000
81	51	\0A1	261	01010001	81	51	\0A1	261	01010001
82	52	\0A2	262	01010010	82	52	\0A2	262	01010010
83	53	\0A3	263	01010011	83	53	\0A3	263	01010011
84	54	\0A4	270	01010100	84	54	\0A4	270	01010100
85	55	\0A5	271	01010101	85	55	\0A5	271	01010101
86	56	\0A6	272	01010110	86	56	\0A6	272	01010110
87	57	\0A7	273	01010111	87	57	\0A7	273	01010111
88	58	\0B0	280	01011000	88	58	\0B0	280	01011000
89	59	\0B1	281	01011001	89	59	\0B1	281	01011001
90	5A	\0B2	282	01011010	90	5A	\0B2	282	01011010
91	5B	\0B3	283	01011011	91	5B	\0B3	283	01011011
92	5C	\0B4	290	01011100	92	5C	\0B4	290	01011100
93	5D	\0B5	291	01011101	93	5D	\0B5	291	01011101
94	5E	\0B6	292	01011110	94	5E	\0B6	292	01011110
95	5F	\0B7	293	01011111	95	5F	\0B7	293	01011111
96	60	\0C0	300	01100000	96	60	\0C0	300	01100000
97	61	\0C1	301	01100001	97	61	\0C1	301	01100001
98	62	\0C2	302	01100010	98	62	\0C2	302	01100010
99	63	\0C3	303	01100011	99	63	\0C3	303	01100011
100	64	\0C4	310	01100100	100	64	\0C4	310	01100100
101	65	\0C5	311	01100101	101	65	\0C5	311	01100101
102	66	\0C6	312	01100110	102	66	\0C6	312	01100110
103	67	\0C7	313	01100111	103	67	\0C7	313	01100111
104	68	\0D0	320	01101000	104	68	\0D0	320	01101000
105	69	\0D1	321	01101001	105	69	\0D1	321	01101001
106	6A	\0D2	322	01101010	106	6A	\0D2	322	01101010
107	6B	\0D3	323	01101011	107	6B	\0D3	323	01101011
108	6C	\0D4	330	01101100	108	6C	\0D4	330	01101100
109	6D	\0D5	331	01101101	109	6D	\0D5	331	01101101
110	6E	\0D6	332	01101110	110	6E	\0D6	332	01101110
111	6F	\0D7	333	01101111	111	6F	\0D7	333	01101111
112	70	\0E0	340	01110000	112	70	\0E0	340	01110000
113	71	\0E1	341	01110001	113	71	\0E1	341	01110001
114	72	\0E2	342	01110010	114	72	\0E2	342	01110010
115	73	\0E3	343	01110011	115	73	\0E3	343	01110011
116	74	\0E4	350	01110100	116	74	\0E4	350	01110100
117	75	\0E5	351	01110101	117	75	\0E5	351	01110101
118	76	\0E6	352	01110110	118	76	\0E6	352	01110110
119	77	\0E7	353	01110111	119	77	\0E7	353	01110111
120	78	\0F0	360	01111000	120	78	\0F0	360	01111000
121	79	\0F1	361	01111001	121	79	\0F1	361	01111001
122	7A	\0F2	362	01111010	122	7A	\0F2	362	01111010
123	7B	\0F3	363	01111011	123	7B	\0F3	363	01111011
124	7C	\0F4	370	01111100	124	7C	\0F4	370	01111100
125	7D	\0F5	371	01111101	125	7D	\0F5	371	01111101
126	7E	\0F6	372	01111110	126	7E	\0F6	372	01111110
127	7F	\0F7	373	01111111	127	7F	\0F7	373	01111111

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

Converting from Character to Code:

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

Decimal Num Char	Octal Num Char	Hex Num Char	Char	Decimal Num Char	Octal Num Char	Hex Num Char	Char
0	000	000	\0	32	040	20	@
1	001	001	\1	33	041	21	#
2	002	002	\2	34	042	22	\$
3	003	003	\3	35	043	23	%
4	004	004	\4	36	044	24	&
5	005	005	\5	37	045	25	*
6	006	006	\6	38	046	26	+
7	007	007	\7	39	047	27	-
8	010	008	\8	40	050	30	=
9	011	009	\9	41	051	31	,
10	012	00A	\A	42	052	32	.
11	013	00B	\B	43	053	33	\t
12	014	00C	\C	44	054	34	\n
13	015	00D	\D	45	055	35	\r
14	016	00E	\E	46	056	36	\f
15	017	00F	\F	47	057	37	\v
16	020	010	\016	48	060	40	\b
17	021	011	\017	49	061	41	\a
18	022	012	\020	50	062	42	\x0d
19	023	013	\021	51	063	43	\x0a
20	024	014	\022	52	064	44	\x0c
21	025	015	\023	53	065	45	\x0b
22	026	016	\024	54	066	46	\x0e
23	027	017	\025	55	067	47	\x0f
24	030	020	\030	56	070	50	\x0d\x0a
25	031	021	\031	57	071	51	\x0d\x0b
26	032	022	\032	58	072	52	\x0d\x0c
27	033	023	\033	59	073	53	\x0d\x0b\x0c
28	034	024	\034	60	074	54	
29	035	025	\035	61	075	55	
30	036	026	\036	62	076	56	
31	037	027	\037	63	077	57	
32	040	030	\040	64	080	60	\x00
33	041	031	\041	65	081	61	\x01
34	042	032	\042	66	082	62	\x02
35	043	033	\043	67	083	63	\x03
36	044	034	\044	68	084	64	\x04
37	045	035	\045	69	085	65	\x05
38	046	036	\046	70	086	66	\x06
39	047	037	\047	71	087	67	\x07
40	050	040	\050	72	088	68	\x08
41	051	041	\051	73	089	69	\x09
42	052	042	\052	74	090	6A	\x0A
43	053	043	\053	75	091	6B	\x0B
44	054	044	\054	76	092	6C	\x0C
45	055	045	\055	77	093	6D	\x0D
46	056	046	\056	78	094	6E	\x0E
47	057	047	\057	79	095	6F	\x0F
48	060	050	\060	80	096	70	\x00\x0d
49	061	051	\061	81	097	71	\x00\x0a
50	062	052	\062	82	098	72	\x00\x0d\x0a
51	063	053	\063	83	099	73	
52	064	054	\064	84	100	74	
53	065	055	\065	85	101	75	
54	066	056	\066	86	102	76	
55	067	057	\067	87	103	77	
56	070	060	\070	88	104	78	
57	071	061	\071	89	105	79	
58	072	062	\072	90	106	7A	
59	073	063	\073	91	107	7B	
60	074	064	\074	92	108	7C	
61	075	065	\075	93	109	7D	
62	076	066	\076	94	110	7E	
63	077	067	\077	95	111	7F	
64	080	070	\080	96	112	80	
65	081	071	\081	97	113	81	
66	082	072	\082	98	114	82	
67	083	073	\083	99	115	83	
68	084	074	\084	100	116	84	
69	085	075	\085	101	117	85	
70	086	076	\086	102	118	86	
71	087	077	\087	103	119	87	
72	088	080	\088	104	120	88	
73	089	081	\089	105	121	89	
74	090	082	\090	106	122	8A	
75	091	083	\091	107	123	8B	
76	092	084	\092	108	124	8C	
77	093	085	\093	109	125	8D	
78	094	086	\094	110	126	8E	
79	095	087	\095	111	127	8F	
80	096	090	\096	112	128	90	
81	097	091	\097	113	129	91	
82	098	092	\098	114	130	92	
83	099	093	\099	115	131	93	
84	100	094	\100	116	132	94	
85	101	095	\101	117	133	95	
86	102	096	\102	118	134	96	
87	103	097	\103	119	135	97	
88	104	098	\104	120	136	98	
89	105	099	\105	121	137	99	
90	106	09A	\106	122	138	9A	
91	107	09B	\107	123	139	9B	
92	108	09C	\108	124	140	9C	
93	109	09D	\109	125	141	9D	
94	110	09E	\110	126	142	9E	
95	111	09F	\111	127	143	9F	
96	112	0A0	\112	128	144	90	
97	113	0A1	\113	129	145	91	
98	114	0A2	\114	130	146	92	
99	115	0A3	\115	131	147	93	
100	116	0A4	\116	132	148	94	
101	117	0A5	\117	133	149	95	
102	118	0A6	\118	134	150	96	
103	119	0A7	\119	135	151	97	
104	120	0A8	\120	136	152	98	
105	121	0A9	\121	137	153	99	
106	122	0AA	\122	138	154	9A	
107	123	0AB	\123	139	155	9B	
108	124	0AC	\124	140	156	9C	
109	125	0AD	\125	141	157	9D	
110	126	0AE	\126	142	158	9E	
111	127	0AF	\127	143	159	9F	
112	128	0B0	\128	144	160	90	
113	129	0B1	\129	145	161	91	
114	130	0B2	\130	146	162	92	
115	131	0B3	\131	147	163	93	
116	132	0B4	\132	148	164	94	
117	133	0B5	\133	149	165	95	
118	134	0B6	\134	150	166	96	
119	135	0B7	\135	151	167	97	
120	136	0B8	\136	152	168	98	
121	137	0B9	\137	153	169	99	
122	138	0BA	\138	154	170	9A	
123	139	0BB	\139	155	171	9B	
124	140	0BC	\140	156	172	9C	
125	141	0BD	\141	157	173	9D	
126	142	0BE	\142	158	174	9E	
127	143	0BF	\143	159	175	9F	
128	144	0C0	\144	160	176	90	
129	145	0C1	\145	161	177	91	
130	146	0C2	\146	162	178	92	
131	147	0C3	\147	163	179	93	
132	148	0C4	\148	164	180	94	
133	149	0C5	\149	165	181	95	
134	150	0C6	\150	166	182	96	
135	151	0C7	\151	167	183	97	
136	152	0C8	\152	168	184	98	
137	153	0C9	\153	169	185	99	
138	154	0CA	\154	170	186	9A	
139	155	0CB	\155	171	187	9B	
140	156	0CC	\156	172	188	9C	
141	157	0CD	\157	173	189	9D	
142	158	0CE	\158	174	190	9E	
143	159	0CF	\159	175	191	9F	
144	160	0D0	\160	176	192	90	
145	161	0D1	\161	177	193	91	
146	162	0D2	\162	178	194	92	
147	163	0D3	\163	179	195	93	
148	164	0D4	\164	180	196	94	
149	165	0D5	\165	181	197	95	
150	166	0D6	\166	182	198	96	
151	167	0D7	\167	183	199	97	
152	168	0D8	\168	184	200	98	
153	169	0D9	\169	185	201	99	
154	170	0DA	\170	186	202	9A	
155	171	0DB	\171	187	203	9B	
156	172	0DC	\172	188	204	9C	
157	173	0DD	\173	189	205	9D	
158	174	0DE	\174	190	206	9E	
159	175	0DF	\175	191	207	9F	
160	176	0E0	\176	192	208	90	
161	177	0E1	\177	193	209	91	
162	178	0E2	\178	194	210	92	
163	179	0E3	\179	195	211	93	
164	180	0E4	\180	196	212	94	
165	181	0E5	\181	197	213	95	
166	182	0E6	\182	198	214	96	
167	183	0E7	\183	199	215	97	
168	184	0E8	\184	200	216	98	
169	185	0E9	\185	201	217	99	
170	186	0EA	\186	202	218	9A	
171	187	0EB	\187	203	219	9B	
172	188	0EC	\188	204	220	9C	
173	189	0ED	\189	205	221	9D	
174	190	0EE	\190	206	222	9E	
175	191	0EF	\191	207	223	9F	
176	192	0E0	\192	208	224	90	
177	193	0E1	\193	209	225	91	
178	194	0E2	\194	210	226	92	
179	195	0E3	\195	211	227	93	
180	196	0E4	\196	212	228	94	
181	197	0E5	\197	213	229	95	
182	198	0E6	\198	214	230	96	
183	199	0E7	\199	215	231	97	
184	200	0E8	\200	216	232	98	
185	201	0E9	\201	217	233	99	
186	202	0EA	\202	218	234	9A	
187	203	0EB	\203	219	235	9B	
188	204	0EC	\204	220	236	9C	
189	205	0ED	\205	221	237	9D	
190	206	0EE	\206	222	238		

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'.
 - What is `chr(33)`?

In Pairs or Triples...

Some review and some novel challenges:

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



Python Tutor

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

(Demo with pythonTutor)

User Input

Covered in detail in Lab 2:

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

(Demo with pythonTutor)

Side Note: '+' for numbers and strings

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



Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Lecture Quiz

- Log-in to Gradescope
- Find LECTURE 2 Quiz
- Take the quiz
- **You have 3 minutes**

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- **Strings**
- Guest: Elise Harris (Advising, Clubs, Internships and more)

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.

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

- Different delimiters give different lists:

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

```
"FridayXXXXsaturdayXXXXsunday"
```

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
- **Guest: Elise Harris (Advising, Clubs, Internships and more)**

Guest Speaker: Elise Harris CS Opportunities

- Announcement on Blackboard:
 - ▶ Programs and Clubs Handout
 - ▶ Internships Handout
 - ▶ Hunter CS Handbook
 - ▶ PreTech Center (formerly CUNY2X) Newsletter

Recap

- 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

```
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!')
```

- In Python, we introduced:
 - ▶ For-loops

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!')
```

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`

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!')
```

- In Python, we introduced:

- ▶ 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!')
```

- 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!')
```

- 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!')
```

- 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!')
```

- 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!')
```

- In Python, we introduced:

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

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.

See you next week!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments