

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 am not sure how to submit the Lab.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
You don't submit the lab, you read the lab.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Lab Quiz (Gradescope) can be submitted only once.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **Can I work ahead?**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **Can I work ahead?**

Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **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 am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

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

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **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 quizzes and programming assignments.

Today's Topics



- **For-loops**
- `range()`
- Variables
- Characters
- Strings
- Guests: Internships, Advising & Clubs

In Pairs or Triples...

Some review and some novel challenges:

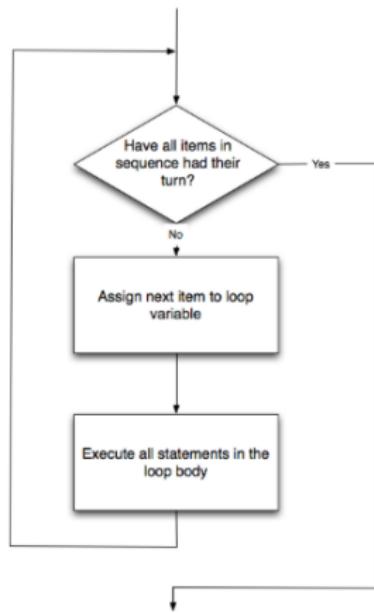
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

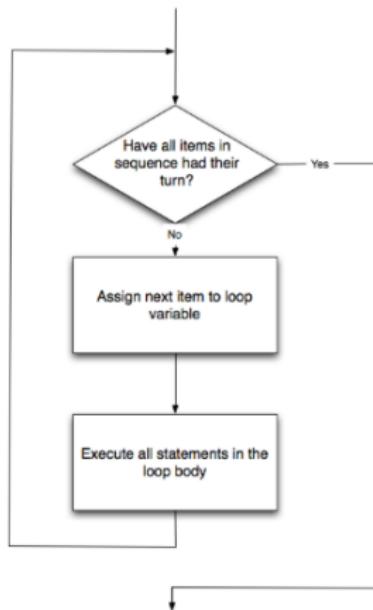
for-loop



```
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
- Guests: Internships, Advising & Clubs

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,...,stop-1]`
- For example, if you want the list `[0,1,2,3,...,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
 $[5, 10, \dots, 50]$
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,...,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
- Guests: Internships, Advising & Clubs

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
- Guests: Internships, Advising & Clubs

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 a link to the ASCII table on the course webpage, under 'Useful Links'.)

ASCII TABLE

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

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

ASCII TABLE											
Decimal	Hex	Char	Octal	Decimal	Hex	Char	Octal	Decimal	Hex	Char	Octal
0	00	\0	000	1	01	\1	001	2	02	\2	002
3	03	\3	003	4	04	\4	004	5	05	\5	005
6	06	\6	006	7	07	\7	007	8	08	\8	008
9	09	\9	009	10	0A	\A	010	11	0B	\B	011
12	0C	\C	014	13	0D	\D	015	14	0E	\E	016
15	0F	\F	017	16	10	\10	020	17	11	\11	021
18	12	\12	022	19	13	\13	023	20	14	\14	024
21	15	\15	025	22	16	\16	026	23	17	\17	027
24	18	\18	030	25	19	\19	031	26	1A	\1A	032
27	1B	\1B	035	28	1C	\1C	036	29	1D	\1D	037
30	1E	\1E	042	31	1F	\1F	043	32	20	\20	044
33	23	\23	051	34	24	\24	052	35	25	\25	053
36	26	\26	060	37	27	\27	061	38	28	\28	062
39	2A	\2A	066	40	2B	\2B	067	41	2C	\2C	070
42	2D	\2D	072	43	2E	\2E	073	44	2F	\2F	074
45	31	\31	081	46	32	\32	082	47	33	\33	083
48	35	\35	090	49	36	\36	091	50	37	\37	092
51	39	\39	099	52	3A	\3A	100	53	3B	\3B	101
55	3D	\3D	104	56	3E	\3E	105	57	3F	\3F	106
59	43	\43	111	60	44	\44	112	61	45	\45	113
63	47	\47	117	64	48	\48	118	65	49	\49	119
67	4B	\4B	124	68	4C	\4C	125	69	4D	\4D	126
71	4F	\4F	131	72	50	\50	132	73	51	\51	133
75	53	\53	139	76	54	\54	140	77	55	\55	141
79	57	\57	147	80	58	\58	148	81	59	\59	149
83	5B	\5B	155	84	5C	\5C	156	85	5D	\5D	157
87	5F	\5F	163	88	60	\60	164	89	61	\61	165
91	63	\63	171	92	64	\64	172	93	65	\65	173
95	67	\67	179	96	68	\68	180	97	69	\69	181
99	6B	\6B	187	100	6C	\6C	188	101	6D	\6D	189
103	6F	\6F	195	104	70	\70	196	105	71	\71	197
107	73	\73	203	108	74	\74	204	109	75	\75	205
111	77	\77	211	112	78	\78	212	113	79	\79	213
115	7B	\7B	219	116	7C	\7C	220	117	7D	\7D	221
119	7F	\7F	227	120	80	\80	228	121	81	\81	229
123	83	\83	235	124	84	\84	236	125	85	\85	237
127	87	\87	243	128	88	\88	244	129	89	\89	245
131	8B	\8B	251	132	8C	\8C	252	133	8D	\8D	253
135	8F	\8F	259	136	90	\90	260	137	91	\91	261
139	93	\93	267	140	94	\94	268	141	95	\95	269
143	97	\97	275	144	98	\98	276	145	99	\99	277
147	9B	\9B	283	148	9C	\9C	284	149	9D	\9D	285
151	9F	\9F	291	152	A0	\A0	292	153	A1	\A1	293
155	A3	\A3	299	156	A4	\A4	300	157	A5	\A5	301
159	A7	\A7	307	160	A8	\A8	308	161	A9	\A9	309
163	AB	\AB	315	164	AC	\AC	316	165	AD	\AD	317
167	AF	\AF	323	168	BA	\BA	324	169	CB	\CB	325
171	B3	\B3	331	172	B4	\B4	332	173	B5	\B5	333
175	B7	\B7	340	176	B8	\B8	341	177	B9	\B9	342
179	BB	\BB	349	180	BC	\BC	350	181	BD	\BD	351
183	BF	\BF	358	184	CA	\CA	359	185	CB	\CB	360
187	C3	\C3	367	188	C4	\C4	368	189	C5	\C5	369
191	C7	\C7	376	192	C8	\C8	377	193	C9	\C9	378
195	CB	\CB	385	196	CC	\CC	386	197	CD	\CD	387
199	CF	\CF	394	200	CA	\CA	395	201	CB	\CB	396
203	D3	\D3	403	204	D4	\D4	404	205	D5	\D5	405
207	D7	\D7	412	208	D8	\D8	413	209	D9	\D9	414
211	DB	\DB	421	212	DC	\DC	422	213	DD	\DD	423
215	DF	\DF	430	216	EA	\EA	431	217	EB	\EB	432
219	E3	\E3	439	220	E4	\E4	440	221	E5	\E5	441
223	E7	\E7	448	224	E8	\E8	449	225	E9	\E9	450
227	EB	\EB	457	228	EC	\EC	458	229	ED	\ED	459
231	EF	\EF	466	232	FA	\FA	467	233	FB	\FB	468
235	F3	\F3	475	236	F4	\F4	476	237	F5	\F5	477
239	F7	\F7	484	240	F8	\F8	485	241	F9	\F9	486
243	FB	\FB	493	244	FC	\FC	494	245	FD	\FD	495
247	FF	\FF	502	248	FE	\FE	503	249	FD	\FD	504

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ASCII TABLE	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
	0	00		1	01	!>	2	02	!>
	2	02		3	03	!>	4	04	!>
	3	03		5	05	!>	6	06	!>
	5	05		7	07	!>	8	08	!>
	7	07		9	09	!>	10	0A	!>
	9	09		11	0B	!>	12	0C	!>
	11	0B		13	0D	!>	14	0E	!>
	13	0D		15	0F	!>	16	10	!>
	15	0F		17	11	!>	18	12	!>
	17	11		19	13	!>	20	14	!>
	19	13		21	15	!>	22	16	!>
	21	15		23	17	!>	24	18	!>
	23	17		25	1A	!>	26	1B	!>
	25	1A		27	1C	!>	28	1D	!>
	27	1C		29	1E	!>	30	1F	!>
	29	1E		31	20	!>	32	21	!>
	31	20		33	22	!>	34	23	!>
	33	22		35	25	!>	36	26	!>
	35	25		37	27	!>	38	28	!>
	37	27		39	2A	!>	40	2B	!>
	39	2A		41	2D	!>	42	2E	!>
	41	2D		43	2F	!>	44	30	!>
	43	2F		45	31	!>	46	32	!>
	45	31		47	33	!>	48	34	!>
	47	33		49	36	!>	50	37	!>
	49	36		51	39	!>	52	3A	!>
	51	39		53	3C	!>	54	3D	!>
	53	3C		55	3F	!>	56	40	!>
	55	3F		57	41	!>	58	42	!>
	57	41		59	44	!>	60	45	!>
	59	44		61	47	!>	62	48	!>
	61	47		63	4A	!>	64	4B	!>
	63	4A		65	4D	!>	66	4E	!>
	65	4D		67	4F	!>	68	50	!>
	67	4F		69	51	!>	70	52	!>
	69	51		71	54	!>	72	55	!>
	71	54		73	57	!>	74	58	!>
	73	57		75	5A	!>	76	5B	!>
	75	5A		77	5D	!>	78	5E	!>
	77	5D		79	5F	!>	80	60	!>
	79	5F		81	61	!>	82	62	!>
	81	61		83	64	!>	84	65	!>
	83	64		85	67	!>	86	68	!>
	85	67		87	6A	!>	88	6B	!>
	87	6A		89	6D	!>	90	6E	!>
	89	6D		91	6F	!>	92	70	!>
	91	6F		93	71	!>	94	72	!>
	93	71		95	74	!>	96	75	!>
	95	74		97	77	!>	98	78	!>
	97	77		99	7A	!>	100	7B	!>
	99	7A		101	7D	!>	102	7E	!>
	101	7D		103	7F	!>	104	80	!>
	103	7F		105	81	!>	106	82	!>
	105	81		107	84	!>	108	85	!>
	107	84		109	87	!>	110	88	!>
	109	87		111	8A	!>	112	8B	!>
	111	8A		113	8D	!>	114	8E	!>
	113	8D		115	8F	!>	116	90	!>
	115	8F		117	91	!>	118	92	!>
	117	91		119	94	!>	120	95	!>
	119	94		121	97	!>	122	98	!>
	121	97		123	9A	!>	124	9B	!>
	123	9A		125	9D	!>	126	9E	!>
	125	9D		127	9F	!>	128	A0	!>
	127	9F		129	A1	!>	130	A2	!>
	129	A1		131	A4	!>	132	A5	!>
	131	A4		133	A7	!>	134	A8	!>
	133	A7		135	AA	!>	136	AB	!>
	135	A9		137	AD	!>	138	AE	!>
	137	AD		139	AF	!>	140	B0	!>
	139	AF		141	B1	!>	142	B2	!>
	141	B1		143	B4	!>	144	B5	!>
	143	B4		145	B7	!>	146	B8	!>
	145	B7		147	B9	!>	148	BA	!>
	147	B9		149	BD	!>	150	BE	!>
	149	BD		151	BF	!>	152	C0	!>
	151	BF		153	C1	!>	154	C2	!>
	153	C1		155	C4	!>	156	C5	!>
	155	C4		157	C7	!>	158	C8	!>
	157	C7		159	CA	!>	160	CB	!>
	159	CA		161	CD	!>	162	CE	!>
	161	CD		163	CF	!>	164	D0	!>
	163	CF		165	D1	!>	166	D2	!>
	165	D1		167	D4	!>	168	D5	!>
	167	D4		169	D7	!>	170	D8	!>
	169	D7		171	DA	!>	172	DB	!>
	171	DA		173	DC	!>	174	DD	!>
	173	DC		175	DF	!>	176	E0	!>
	175	DF		177	E1	!>	178	E2	!>
	177	E1		179	E4	!>	180	E5	!>
	179	E4		181	E7	!>	182	E8	!>
	181	E7		183	EA	!>	184	EB	!>
	183	EA		185	ED	!>	186	EE	!>
	185	ED		187	FF	!>	188	00	!>
	187	FF		189	01	!>	190	02	!>
	189	01		191	04	!>	192	05	!>
	191	04		193	07	!>	194	08	!>
	193	07		195	0A	!>	196	0B	!>
	195	0A		197	0D	!>	198	0E	!>
	197	0D		199	0F	!>	200	10	!>
	199	0F		201	11	!>	202	12	!>
	201	11		203	13	!>	204	14	!>
	203	13		205	15	!>	206	16	!>
	205	15		207	17	!>	208	18	!>
	207	17		209	1A	!>	210	1B	!>
	209	1A		211	1D	!>	212	1E	!>
	211	1D		213	1F	!>	214	20	!>
	213	1F		215	21	!>	216	22	!>
	215	21		217	24	!>	218	25	!>
	217	24		219	27	!>	220	28	!>
	219	27		221	2A	!>	222	2B	!>
	221	2A		223	2D	!>	224	2E	!>
	223	2D		225	2F	!>	226	30	!>
	225	2F		227	31	!>	228	32	!>
	227	31		229	34	!>	230	35	!>
	229	34		231	37	!>	232	38	!>
	231	37		233	3A	!>	234	3B	!>
	233	3A		235	3D	!>	236	3E	!>
	235	3D		237	3F	!>	238	40	!>
	237	3F		239	41	!>	240	42	!>
	239	41		241	44	!>	242	45	!>
	241	44		243	47	!>	244	48	!>
	243	47		245	4A	!>	246	4B	!>
	245	4A		247	4D	!>	248	4E	!>
	247	4D		249	4F	!>	250	50	!>
	249	4F		251	51	!>	252	52	!>
	251	51		253	54	!>	254	55	!>
	253	54		255	57	!>	256	58	!>
	255	57		257	5A	!>	258	5B	!>
	257	5A		259	5D	!>	260	5E	!>
	259	5D		261	5F	!>	262	60	!>
	261	5F		263	61	!>	264	62	!>
	263	61		265	64	!>	266	65	!>
	265	64		267	67	!>	268	68	!>
	267	67		269	6A	!>	270	6B	!>
	269	6A		271	6D	!>	272	6E	!>
	271	6D		273	6F	!>	274	70	!>
	273	6F		275	71	!>	276	72	!>
	275	71		277	74	!>	278	75	!>
	277	74		279	77	!>	280	78	!>
	279	77		281	7A	!>	282	7B	!>
	281	7A		283	7D	!>	284	7E	!>
	283	7D		285	7F	!>	286	80	!>
	285	7F		287	81	!>	288	82	!>
	287	81		289	84	!>	290	85	!>
	289	84		291	87	!>	292	88	!>
	291	87		293	8A	!>	294	8B	!>
	293	8A		295	8D	!>	296	8E	!>
	295	8D		297	8F	!>	298	90	!>
	297	8F		299	91	!>	300	92	!>
	299	91		301	94	!>	302	95	!>
	301	94		303	97	!>	304	98	!>
	303	97		305	9A	!>	306	9B	!>
	305	9A		307	9D	!>	308	9E	!>
	307	9D		309	9F	!>	310	A0	!>
	309	9F		311	A1	!>	312	A2	!>
	311	A1		313	A4	!>	314	A5	!>
	313	A4		315	A7	!>	316	A8	!>
	315	A7							

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00	\0	32	20	\t	64	40	\n
1	01	\1	33	21	\a	65	41	A
2	02	\2	34	22	\b	66	42	B
3	03	\3	35	23	\f	67	43	F
4	04	\4	36	24	\n	68	44	\n
5	05	\5	37	25	\r	69	45	\r
6	06	\6	38	26	\v	70	46	\v
7	07	\7	39	27	\b	71	47	\b
8	08	\8	40	28	\t	72	48	\t
9	09	\9	41	29	\n	73	49	\n
10	0A	\10	42	2A	\f	74	4A	\f
11	0B	\11	43	2B	\r	75	4B	\r
12	0C	\12	44	2C	\v	76	4C	\v
13	0D	\13	45	2D	\b	77	4D	\b
14	0E	\14	46	2E	\t	78	4E	\t
15	0F	\15	47	2F	\n	79	4F	\n
16	10	\16	48	30	\t	80	50	\t
17	11	\17	49	31	\n	81	51	\n
18	12	\18	4A	32	\f	82	52	\f
19	13	\19	4B	33	\r	83	53	\r
20	14	\20	4C	34	\v	84	54	\v
21	15	\21	4D	35	\b	85	55	\b
22	16	\22	4E	36	\t	86	56	\t
23	17	\23	4F	37	\n	87	57	\n
24	18	\24	50	38	\t	88	58	\t
25	19	\25	51	39	\n	89	59	\n
26	1A	\26	52	3A	\f	90	5A	\f
27	1B	\27	53	3B	\r	91	5B	\r
28	1C	\28	54	3C	\v	92	5C	\v
29	1D	\29	55	3D	\b	93	5D	\b
30	1E	\20	56	3E	\t	94	5E	\t
31	1F	\21	57	3F	\n	95	5F	\n
32	20	\22	58	40	\t	96	60	\t
33	21	\23	59	41	\n	97	61	A
34	22	\24	5A	42	\f	98	62	B
35	23	\25	5B	43	\r	99	63	F
36	24	\26	5C	44	\v	100	64	\v
37	25	\27	5D	45	\b	101	65	\b
38	26	\28	5E	46	\t	102	66	\t
39	27	\29	5F	47	\n	103	67	\n
40	28	\20	60	48	\t	104	68	\t
41	29	\21	61	49	\n	105	69	\n
42	2A	\22	62	4A	\f	106	6A	\f
43	2B	\23	63	4B	\r	107	6B	\r
44	2C	\24	64	4C	\v	108	6C	\v
45	2D	\25	65	4D	\b	109	6D	\b
46	2E	\26	66	4E	\t	110	6E	\t
47	2F	\27	67	4F	\n	111	6F	\n
48	30	\28	68	50	\t	112	70	\t
49	31	\29	69	51	\n	113	71	\n
50	32	\20	6A	52	\f	114	72	\f
51	33	\21	6B	53	\r	115	73	\r
52	34	\22	6C	54	\v	116	74	\v
53	35	\23	6D	55	\b	117	75	\b
54	36	\24	6E	56	\t	118	76	\t
55	37	\25	6F	57	\n	119	77	\n
56	38	\26	70	58	\t	120	78	\t
57	39	\27	71	59	\n	121	79	\n
58	3A	\28	72	5A	\f	122	7A	\f
59	3B	\29	73	5B	\r	123	7B	\r
60	3C	\20	74	5C	\v	124	7C	\v
61	3D	\21	75	5D	\b	125	7D	\b
62	3E	\22	76	5E	\t	126	7E	\t
63	3F	\23	77	5F	\n	127	7F	\n

- `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 a link to the ASCII table on the course webpage, under 'Useful Links'.)

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00	\0	32	20	\t	64	40	\n
1	01	\1	33	21	\a	65	41	A
2	02	\2	34	22	\b	66	42	B
3	03	\3	35	23	\f	67	43	F
4	04	\4	36	24	\n	68	44	\n
5	05	\5	37	25	\r	69	45	\r
6	06	\6	38	26	\v	70	46	\v
7	07	\7	39	27	\b	71	47	\b
8	08	\8	40	28	\t	72	48	\t
9	09	\9	41	29	\n	73	49	\n
10	0A	\10	42	2A	\r	74	4A	\r
11	0B	\11	43	2B	\v	75	4B	\v
12	0C	\12	44	2C	\b	76	4C	\b
13	0D	\13	45	2D	\t	77	4D	\t
14	0E	\14	46	2E	\n	78	4E	\n
15	0F	\15	47	2F	\r	79	4F	\r
16	10	\16	48	30	\t	80	50	\t
17	11	\17	49	31	\n	81	51	\n
18	12	\18	4A	32	\r	82	52	\r
19	13	\19	4B	33	\v	83	53	\v
20	14	\20	4C	34	\b	84	54	\b
21	15	\21	4D	35	\t	85	55	\t
22	16	\22	4E	36	\n	86	56	\n
23	17	\23	4F	37	\r	87	57	\r
24	18	\24	50	38	\v	88	58	\v
25	19	\25	51	39	\b	89	59	\b
26	1A	\26	52	3A	\t	90	5A	\t
27	1B	\27	53	3B	\n	91	5B	\n
28	1C	\28	54	3C	\r	92	5C	\r
29	1D	\29	55	3D	\v	93	5D	\v
30	1E	\20	56	3E	\b	94	5E	\b
31	1F	\21	57	3F	\t	95	5F	\t
32	20	\22	58	40	\n	96	60	\n
33	21	\23	59	41	\r	97	61	\r
34	22	\24	5A	42	\v	98	62	\v
35	23	\25	5B	43	\b	99	63	\b
36	24	\26	5C	44	\t	100	64	\t
37	25	\27	5D	45	\n	101	65	\n
38	26	\28	5E	46	\r	102	66	\r
39	27	\29	5F	47	\v	103	67	\v
40	28	\20	60	48	\b	104	68	\b
41	29	\21	61	49	\t	105	69	\t
42	2A	\22	62	4A	\n	106	6A	\n
43	2B	\23	63	4B	\r	107	6B	\r
44	2C	\24	64	4C	\v	108	6C	\v
45	2D	\25	65	4D	\b	109	6D	\b
46	2E	\26	66	4E	\t	110	6E	\t
47	2F	\27	67	4F	\n	111	6F	\n
48	30	\28	68	50	\r	112	70	\r
49	31	\29	69	51	\v	113	71	\v
50	32	\20	6A	52	\b	114	72	\b
51	33	\21	6B	53	\t	115	73	\t
52	34	\22	6C	54	\n	116	74	\n
53	35	\23	6D	55	\r	117	75	\r
54	36	\24	6E	56	\v	118	76	\v
55	37	\25	6F	57	\b	119	77	\b
56	38	\26	70	58	\t	120	78	\t
57	39	\27	71	59	\n	121	79	\n
58	3A	\28	72	5A	\r	122	7A	\r
59	3B	\29	73	5B	\v	123	7B	\v
60	3C	\20	74	5C	\b	124	7C	\b
61	3D	\21	75	5D	\t	125	7D	\t
62	3E	\22	76	5E	\n	126	7E	\n
63	3F	\23	77	5F	\r	127	7F	\r
64	40	\24	78	60	\v	128	80	\v
65	41	\25	79	61	\b	129	81	\b
66	42	\26	7A	62	\t	130	82	\t
67	43	\27	7B	63	\n	131	83	\n
68	44	\28	7C	64	\r	132	84	\r
69	45	\29	7D	65	\v	133	85	\v
70	46	\20	7E	66	\b	134	86	\b
71	47	\21	7F	67	\t	135	87	\t
72	48	\22	80	68	\n	136	88	\n
73	49	\23	81	69	\r	137	89	\r
74	4A	\24	82	6A	\v	138	8A	\v
75	4B	\25	83	6B	\b	139	8B	\b
76	4C	\26	84	6C	\t	140	8C	\t
77	4D	\27	85	6D	\n	141	8D	\n
78	4E	\28	86	6E	\r	142	8E	\r
79	4F	\29	87	6F	\v	143	8F	\v
80	50	\20	88	70	\b	144	90	\b
81	51	\21	89	71	\t	145	91	\t
82	52	\22	8A	72	\n	146	92	\n
83	53	\23	8B	73	\r	147	93	\r
84	54	\24	8C	74	\v	148	94	\v
85	55	\25	8D	75	\b	149	95	\b
86	56	\26	8E	76	\t	150	96	\t
87	57	\27	8F	77	\n	151	97	\n
88	58	\28	90	78	\r	152	98	\r
89	59	\29	91	79	\v	153	99	\v
90	5A	\20	92	7A	\b	154	9A	\b
91	5B	\21	93	7B	\t	155	9B	\t
92	5C	\22	94	7C	\n	156	9C	\n
93	5D	\23	95	7D	\r	157	9D	\r
94	5E	\24	96	7E	\v	158	9E	\v
95	5F	\25	97	7F	\b	159	9F	\b
96	60	\26	98	80	\t	160	100	\t
97	61	\27	99	81	\n	161	101	\n
98	62	\28	9A	82	\r	162	102	\r
99	63	\29	9B	83	\v	163	103	\v
100	64	\20	9C	84	\b	164	104	\b
101	65	\21	9D	85	\t	165	105	\t
102	66	\22	9E	86	\n	166	106	\n
103	67	\23	9F	87	\r	167	107	\r
104	68	\24	100	88	\v	168	108	\v
105	69	\25	101	89	\b	169	109	\b
106	6A	\26	102	8A	\t	170	10A	\t
107	6B	\27	103	8B	\n	171	10B	\n
108	6C	\28	104	8C	\r	172	10C	\r
109	6D	\29	105	8D	\v	173	10D	\v
110	6E	\20	106	8E	\b	174	10E	\b
111	6F	\21	107	8F	\t	175	10F	\t
112	70	\22	108	90	\n	176	110	\n
113	71	\23	109	91	\r	177	111	\r
114	72	\24	110	92	\v	178	112	\v
115	73	\25	111	93	\b	179	113	\b
116	74	\26	112	94	\t	180	114	\t
117	75	\27	113	95	\n	181	115	\n
118	76	\28	114	96	\r	182	116	\r
119	77	\29	115	97	\v	183	117	\v
120	78	\20	116	98	\b	184	118	\b
121	79	\21	117	99	\t	185	119	\t
122	7A	\22	118	9A	\n	186	11A	\n
123	7B	\23	119	9B	\r	187	11B	\r
124	7C	\24	120	9C	\v	188	11C	\v
125	7D	\25	121	9D	\b	189	11D	\b
126	7E	\26	122	9E	\t	190	11E	\t
127	7F	\27	123	9F	\n	191	11F	\n
128	80	\28	124	100	\r	192	120	\r
129	81	\29	125	101	\v	193	121	\v
130	82	\20	126	102	\b	194	122	\b
131	83	\21	127	103	\t	195	123	\t
132	84	\22	128	104	\n	196	124	\n
133	85	\23	129	105	\r	197	125	\r
134	86	\24	130	106	\v	198	126	\v
135	87	\25	131	107	\b	199	127	\b
136	88	\26	132	108	\t	200	128	\t
137	89	\27	133	109	\n	201	129	\n
138	8A	\28	134	110	\r	202	12A	\r
139	8B	\29	135	111	\v	203	12B	\v
140	8C	\20	136	112	\b	204	12C	\b
141	8D	\21	137	113	\t	205	12D	\t
142	8E	\22	138	114	\n	206	12E	\n
143	8F	\23	139	115	\r	207	12F	\r
144	90	\24	140	116	\v	208	130	\v
145	91	\25	141	117	\b	209	131	\b
146	92	\26	142	118	\t	210	132	\t
147	93	\27	143	119	\n	211	133	\n
148	94	\28	144	120	\r	212	134	\r
149	95	\29	145	121	\v	213	135	\v
150	96	\20	146	122	\b	214	136	\b
151	97	\21	147	123	\t	215	137	\t
152	98	\22	148	124	\n	216	138	\n
153	99	\23	149	125	\r	217	139	\r
154	9A	\24	150	126	\v	218	13A	\v
155	9B	\25	151	127	\b	219	13B	\b
156	9C	\26	152	128	\t	220	13C	\t
157	9D	\27	153	129	\n	221	13D	\n
158	9E	\28	154	130	\r	222	13E	\r
159	9F	\29	155	131	\v	223	13F	\v
160	100	\20	156	132	\b	224	140	\b
161	101	\21	157	133	\t	225	141	\t
162	102	\22	158	134	\n	226	142	\n
163	103	\23	159	135	\r	227	143	\r
164	104	\24	160	136	\v	228	144	\v
165	105	\25	161	137	\b	229	145	\b
166	106	\26	162	138	\t	230	146	\t
167	107	\27	163	139	\n	231	147	\n
168	108	\28	164	140	\r	232	148	\r
169	109	\29	165	141	\v	233	149	\v
170	10A	\20	166	142	\b	234	14A	\b
171	10B	\21	167	143	\t	235	14B	\t
172	10C	\22	168	144	\n	236	14C	\n
173	10D	\23	169	145	\r	237	14D	\r
174	10E	\24	170	146	\v	238	14E	\v
175	10F	\25	171	147	\b	239	14F	\b
176	110	\26	172	148	\t	240	150	\t
177	111	\27	17					

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

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 the offset is 26, wrap back to 0  
19     newChar = chr(ord('a') + wrap) #compute the new letter  
20     print(wrap, chr(ord('a') + wrap)) #print the wrap & new lett  
21     codedWord = codedWord + newChar #add the newChar to the coded w  
22  
23 print("The coded word (with wrap) is", codedWord)
```

(Demo with pythonTutor)

Wrap

chr()	a	b	c			...			x	y	z
ord()	97	98	99			...			120	121	122



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**
- Guests: Internships, Advising & Clubs

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
- **Guests: Internships, Advising & Clubs**

Guest Speakers

- Announcement on Blackboard:
 - ▶ Advising
 - ▶ 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).
- 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.

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 6-10)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 6-10)
- At any point, visit our [Drop-In Tutoring 11am-5pm](#) for help!!!

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 6-10)
- At any point, visit our [Drop-In Tutoring 11am-5pm](#) for help!!!
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)