

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:

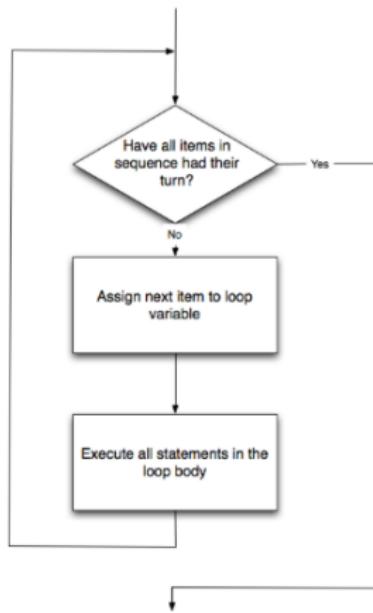
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

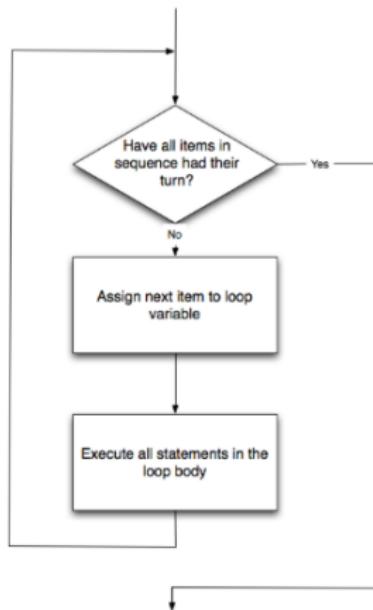
for-loop



```
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...,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
- 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											
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	16	\16	026	22	17	\17	027	23	18	\18	028
24	1C	\1C	032	25	1D	\1D	033	26	1E	\1E	034
27	1F	\1F	037	28	20	\20	040	29	21	\21	041
30	22	\22	042	31	23	\23	043	32	24	\24	044
33	26	\26	046	34	27	\27	047	35	28	\28	048
36	2C	\2C	052	37	2D	\2D	053	38	2E	\2E	054
39	2F	\2F	057	40	30	\30	060	41	31	\31	061
42	32	\32	062	43	33	\33	063	44	34	\34	064
46	36	\36	066	47	37	\37	067	48	38	\38	070
49	39	\39	071	50	3A	\3A	072	51	3B	\3B	073
52	3C	\3C	074	53	3D	\3D	075	54	3E	\3E	076
56	3F	\3F	077	57	40	\40	080	58	41	\41	081
59	42	\42	082	60	43	\43	083	61	44	\44	084
63	46	\46	086	64	47	\47	087	65	48	\48	088
66	4C	\4C	092	67	4D	\4D	093	68	4E	\4E	094
69	4F	\4F	097	70	50	\50	100	71	51	\51	101
72	52	\52	102	73	53	\53	103	74	54	\54	104
76	56	\56	106	77	57	\57	107	78	58	\58	108
79	59	\59	109	80	5A	\5A	110	81	5B	\5B	111
82	5C	\5C	112	83	5D	\5D	113	84	5E	\5E	114
86	5F	\5F	117	87	60	\60	120	88	61	\61	121
89	62	\62	122	90	63	\63	123	91	64	\64	124
93	66	\66	126	94	67	\67	127	95	68	\68	130
97	6F	\6F	135	98	70	\70	136	99	71	\71	137
101	72	\72	140	102	73	\73	141	103	74	\74	142
105	76	\76	144	106	77	\77	145	107	78	\78	146
109	79	\79	148	110	7A	\7A	149	111	7B	\7B	150
113	7C	\7C	152	114	7D	\7D	153	115	7E	\7E	154
117	7F	\7F	157	118	80	\80	160	119	81	\81	161
121	82	\82	164	122	83	\83	165	123	84	\84	166
125	86	\86	170	126	87	\87	171	127	88	\88	172
129	8F	\8F	177	130	90	\90	180	131	91	\91	181
133	92	\92	184	134	93	\93	185	135	94	\94	186
137	96	\96	192	138	97	\97	193	139	98	\98	194
141	99	\99	198	142	9A	\9A	200	143	9B	\9B	201
145	9C	\9C	204	146	9D	\9D	205	147	9E	\9E	206
149	9F	\9F	211	150	A0	\A0	216	151	A1	\A1	217
153	A2	\A2	224	154	A3	\A3	225	155	A4	\A4	226
157	A6	\A6	232	158	A7	\A7	233	159	A8	\A8	234
161	A9	\A9	240	162	AA	\AA	241	163	AB	\AB	242
165	AC	\AC	248	166	AD	\AD	249	167	AE	\AE	250
169	AF	\AF	256	170	B0	\B0	260	171	B1	\B1	261
173	B2	\B2	264	174	B3	\B3	265	175	B4	\B4	266
177	B6	\B6	272	178	B7	\B7	273	179	B8	\B8	274
181	B9	\B9	280	182	BA	\BA	281	183	BB	\BB	282
185	BC	\BC	288	186	BD	\BD	289	187	BE	\BE	290
189	BF	\BF	296	190	C0	\C0	300	191	C1	\C1	301
193	C2	\C2	304	194	C3	\C3	305	195	C4	\C4	306
197	C6	\C6	312	198	C7	\C7	313	199	C8	\C8	314
201	C9	\C9	320	202	CA	\CA	321	203	CB	\CB	322
205	CC	\CC	328	206	CD	\CD	329	207	CE	\CE	330
209	CF	\CF	336	210	D0	\D0	340	211	D1	\D1	341
213	D2	\D2	344	214	D3	\D3	345	215	D4	\D4	346
217	D6	\D6	352	218	D7	\D7	353	219	D8	\D8	354
221	D9	\D9	360	222	DA	\DA	361	223	DB	\DB	362
225	DC	\DC	368	226	DD	\DD	369	227	DE	\DE	370
229	DF	\DF	376	230	E0	\E0	380	231	E1	\E1	381
233	E2	\E2	384	234	E3	\E3	385	235	E4	\E4	386
237	E6	\E6	392	238	E7	\E7	393	239	E8	\E8	394
241	E9	\E9	396	242	EA	\EA	397	243	EB	\EB	398
245	EC	\EC	398	246	ED	\ED	399	247	EE	\EE	400
249	EF	\EF	400	250	F0	\F0	401	251	F1	\F1	402
253	F2	\F2	404	254	F3	\F3	405	255	F4	\F4	406
257	F6	\F6	408	258	F7	\F7	409	259	F8	\F8	410
261	F9	\F9	412	262	FA	\FA	413	263	FB	\FB	414
265	FC	\FC	416	266	FD	\FD	417	267	FE	\FE	418
269	FF	\FF	419	270	00	\00	420	271	01	\01	421

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	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			

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		64	40	!
1	01	\1	33	21	!	65	41	A
2	02	\2	34	22	“	66	42	B
3	03	\3	35	23	”	67	43	C
4	04	\4	36	24	‘	68	44	D
5	05	\5	37	25	’	69	45	E
6	06	\6	38	26	“	70	46	F
7	07	\7	39	27	”	71	47	G
8	08	\8	40	28	‘	72	48	H
9	09	\9	41	29	’	73	49	I
10	0A	\n	42	2A	“	74	4A	J
11	0B	\r	43	2B	”	75	4B	K
12	0C	\t	44	2C	‘	76	4C	L
13	0D	\v	45	2D	’	77	4D	M
14	0E	\f	46	2E	“	78	4E	N
15	0F	\u000F	47	2F	”	79	4F	O
16	10	\u0010	48	30	“	80	50	P
17	11	\u0011	49	31	”	81	51	Q
18	12	\u0012	4A	32	‘	82	52	R
19	13	\u0013	4B	33	’	83	53	S
20	14	\u0014	4C	34	“	84	54	T
21	15	\u0015	4D	35	”	85	55	U
22	16	\u0016	4E	36	‘	86	56	V
23	17	\u0017	4F	37	’	87	57	W
24	18	\u0018	50	38	“	88	58	X
25	19	\u0019	51	39	”	89	59	Y
26	1A	\u001A	52	3A	‘	90	5A	Z
27	1B	\u001B	53	3B	’	91	5B	[\u001B]
28	1C	\u001C	54	3C	“	92	5C	[\u001C]
29	1D	\u001D	55	3D	”	93	5D	[\u001D]
30	1E	\u001E	56	3E	‘	94	5E	[\u001E]
31	1F	\u001F	57	3F	’	95	5F	[\u001F]
32	20	\u0020	58	40		96	60	[\u0020]
33	21	\u0021	59	41	!	97	61	[\u0021]
34	22	\u0022	60	42	”	98	62	[\u0022]
35	23	\u0023	61	43	‘	99	63	[\u0023]
36	24	\u0024	62	44	’	100	64	[\u0024]
37	25	\u0025	63	45	“	101	65	[\u0025]
38	26	\u0026	64	46	”	102	66	[\u0026]
39	27	\u0027	65	47	‘	103	67	[\u0027]
40	28	\u0028	66	48	’	104	68	[\u0028]
41	29	\u0029	67	49	“	105	69	[\u0029]
42	2A	\u002A	68	4A	”	106	6A	[\u002A]
43	2B	\u002B	69	4B	‘	107	6B	[\u002B]
44	2C	\u002C	70	4C	’	108	6C	[\u002C]
45	2D	\u002D	71	4D	“	109	6D	[\u002D]
46	2E	\u002E	72	4E	”	110	6E	[\u002E]
47	2F	\u002F	73	4F	‘	111	6F	[\u002F]
48	30	\u0030	74	50	”	112	70	[\u0030]
49	31	\u0031	75	51	‘	113	71	[\u0031]
50	32	\u0032	76	52	’	114	72	[\u0032]
51	33	\u0033	77	53	“	115	73	[\u0033]
52	34	\u0034	78	54	”	116	74	[\u0034]
53	35	\u0035	79	55	‘	117	75	[\u0035]
54	36	\u0036	80	56	’	118	76	[\u0036]
55	37	\u0037	81	57	“	119	77	[\u0037]
56	38	\u0038	82	58	”	120	78	[\u0038]
57	39	\u0039	83	59	‘	121	79	[\u0039]
58	3A	\u003A	84	5A	’	122	7A	[\u003A]
59	3B	\u003B	85	5B	“	123	7B	[\u003B]
60	3C	\u003C	86	5C	”	124	7C	[\u003C]
61	3D	\u003D	87	5D	‘	125	7D	[\u003D]
62	3E	\u003E	88	5E	’	126	7E	[\u003E]
63	3F	\u003F	89	5F	“	127	7F	[\u003F]
64	40	\u0040	90	60		128	80	[\u0040]
65	41	\u0041	91	61	!	129	81	[\u0041]
66	42	\u0042	92	62	”	130	82	[\u0042]
67	43	\u0043	93	63	‘	131	83	[\u0043]
68	44	\u0044	94	64	’	132	84	[\u0044]
69	45	\u0045	95	65	“	133	85	[\u0045]
70	46	\u0046	96	66	”	134	86	[\u0046]
71	47	\u0047	97	67	‘	135	87	[\u0047]
72	48	\u0048	98	68	’	136	88	[\u0048]
73	49	\u0049	99	69	“	137	89	[\u0049]
74	4A	\u004A	100	6A	”	138	8A	[\u004A]
75	4B	\u004B	101	6B	‘	139	8B	[\u004B]
76	4C	\u004C	102	6C	’	140	8C	[\u004C]
77	4D	\u004D	103	6D	“	141	8D	[\u004D]
78	4E	\u004E	104	6E	”	142	8E	[\u004E]
79	4F	\u004F	105	6F	‘	143	8F	[\u004F]
80	50	\u0050	106	70	”	144	90	[\u0050]
81	51	\u0051	107	71	‘	145	91	[\u0051]
82	52	\u0052	108	72	’	146	92	[\u0052]
83	53	\u0053	109	73	“	147	93	[\u0053]
84	54	\u0054	110	74	”	148	94	[\u0054]
85	55	\u0055	111	75	‘	149	95	[\u0055]
86	56	\u0056	112	76	’	150	96	[\u0056]
87	57	\u0057	113	77	“	151	97	[\u0057]
88	58	\u0058	114	78	”	152	98	[\u0058]
89	59	\u0059	115	79	‘	153	99	[\u0059]
90	5A	\u005A	116	7A	”	154	9A	[\u005A]
91	5B	\u005B	117	7B	‘	155	9B	[\u005B]
92	5C	\u005C	118	7C	”	156	9C	[\u005C]
93	5D	\u005D	119	7D	‘	157	9D	[\u005D]
94	5E	\u005E	120	7E	”	158	9E	[\u005E]
95	5F	\u005F	121	7F	‘	159	9F	[\u005F]

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

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

Converting from Character to Code:

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

Decimal Num Char	Octal Num Char	Hex Num Char	Decimal Num Char	Octal Num Char	Hex Num Char
'\000'	'000'	'000'	'\001'	'001'	'001'
'\002'	'002'	'002'	'\003'	'003'	'003'
'\004'	'004'	'004'	'\005'	'005'	'005'
'\006'	'006'	'006'	'\007'	'007'	'007'
'\010'	'010'	'00A'	'\011'	'011'	'00B'
'\012'	'012'	'00C'	'\013'	'013'	'00D'
'\014'	'014'	'00E'	'\015'	'015'	'00F'
'\016'	'016'	'010'	'\017'	'017'	'011'
'\020'	'020'	'012'	'\021'	'021'	'013'
'\022'	'022'	'014'	'\023'	'023'	'015'
'\024'	'024'	'016'	'\025'	'025'	'017'
'\026'	'026'	'018'	'\027'	'027'	'019'
'\030'	'030'	'01A'	'\031'	'031'	'01B'
'\032'	'032'	'01C'	'\033'	'033'	'01D'
'\034'	'034'	'01E'	'\035'	'035'	'01F'
'\036'	'036'	'020'	'\037'	'037'	'021'
'\040'	'040'	'022'	'\041'	'041'	'023'
'\042'	'042'	'024'	'\043'	'043'	'025'
'\044'	'044'	'026'	'\045'	'045'	'027'
'\046'	'046'	'028'	'\047'	'047'	'029'
'\048'	'048'	'02A'	'\049'	'049'	'02B'
'\050'	'050'	'02C'	'\051'	'051'	'02D'
'\052'	'052'	'02E'	'\053'	'053'	'02F'
'\054'	'054'	'030'	'\055'	'055'	'031'
'\056'	'056'	'032'	'\057'	'057'	'033'
'\060'	'060'	'034'	'\061'	'061'	'035'
'\062'	'062'	'036'	'\063'	'063'	'037'
'\064'	'064'	'038'	'\065'	'065'	'039'
'\066'	'066'	'03A'	'\067'	'067'	'03B'
'\070'	'070'	'03C'	'\071'	'071'	'03D'
'\072'	'072'	'03E'	'\073'	'073'	'03F'
'\074'	'074'	'040'	'\075'	'075'	'041'
'\077'	'077'	'042'	'\078'	'078'	'043'
'\080'	'080'	'044'	'\081'	'081'	'045'
'\082'	'082'	'046'	'\083'	'083'	'047'
'\084'	'084'	'048'	'\085'	'085'	'049'
'\086'	'086'	'04A'	'\087'	'087'	'04B'
'\088'	'088'	'04C'	'\089'	'089'	'04D'
'\090'	'090'	'04E'	'\091'	'091'	'04F'
'\092'	'092'	'050'	'\093'	'093'	'051'
'\094'	'094'	'052'	'\095'	'095'	'053'
'\096'	'096'	'054'	'\097'	'097'	'055'
'\098'	'098'	'056'	'\099'	'099'	'057'
'\0A0'	'0A0'	'058'	'\0A1'	'0A1'	'059'
'\0A2'	'0A2'	'05A'	'\0A3'	'0A3'	'05B'
'\0A4'	'0A4'	'05C'	'\0A5'	'0A5'	'05D'
'\0A6'	'0A6'	'05E'	'\0A7'	'0A7'	'05F'
'\0A8'	'0A8'	'060'	'\0A9'	'0A9'	'061'
'\0AA'	'0AA'	'062'	'\0AB'	'0AB'	'063'
'\0AC'	'0AC'	'064'	'\0AD'	'0AD'	'065'
'\0AE'	'0AE'	'066'	'\0AF'	'0AF'	'067'
'\0B0'	'0B0'	'068'	'\0B1'	'0B1'	'069'
'\0B2'	'0B2'	'06A'	'\0B3'	'0B3'	'06B'
'\0B4'	'0B4'	'06C'	'\0B5'	'0B5'	'06D'
'\0B6'	'0B6'	'06E'	'\0B7'	'0B7'	'06F'
'\0B8'	'0B8'	'070'	'\0B9'	'0B9'	'071'
'\0BA'	'0BA'	'072'	'\0BB'	'0BB'	'073'
'\0BC'	'0BC'	'074'	'\0BD'	'0BD'	'075'
'\0BE'	'0BE'	'076'	'\0BF'	'0BF'	'077'
'\0C0'	'0C0'	'078'	'\0C1'	'0C1'	'079'
'\0C2'	'0C2'	'07A'	'\0C3'	'0C3'	'07B'
'\0C4'	'0C4'	'07C'	'\0C5'	'0C5'	'07D'
'\0C6'	'0C6'	'07E'	'\0C7'	'0C7'	'07F'
'\0C8'	'0C8'	'080'	'\0C9'	'0C9'	'081'
'\0CA'	'0CA'	'082'	'\0CB'	'0CB'	'083'
'\0CC'	'0CC'	'084'	'\0CD'	'0CD'	'085'
'\0CE'	'0CE'	'086'	'\0CF'	'0CF'	'087'
'\0D0'	'0D0'	'088'	'\0D1'	'0D1'	'089'
'\0D2'	'0D2'	'08A'	'\0D3'	'0D3'	'08B'
'\0D4'	'0D4'	'08C'	'\0D5'	'0D5'	'08D'
'\0D6'	'0D6'	'08E'	'\0D7'	'0D7'	'08F'
'\0D8'	'0D8'	'090'	'\0D9'	'0D9'	'091'
'\0DA'	'0DA'	'092'	'\0DB'	'0DB'	'093'
'\0DC'	'0DC'	'094'	'\0DD'	'0DD'	'095'
'\0DE'	'0DE'	'096'	'\0DF'	'0DF'	'097'
'\0E0'	'0E0'	'098'	'\0E1'	'0E1'	'099'
'\0E2'	'0E2'	'09A'	'\0E3'	'0E3'	'09B'
'\0E4'	'0E4'	'09C'	'\0E5'	'0E5'	'09D'
'\0E6'	'0E6'	'09E'	'\0E7'	'0E7'	'09F'
'\0E8'	'0E8'	'0A0'	'\0E9'	'0E9'	'0A1'
'\0EA'	'0EA'	'0A2'	'\0EB'	'0EB'	'0A3'
'\0EC'	'0EC'	'0A4'	'\0ED'	'0ED'	'0A5'
'\0EE'	'0EE'	'0A6'	'\0EF'	'0EF'	'0A7'
'\0F0'	'0F0'	'0A8'	'\0F1'	'0F1'	'0A9'
'\0F2'	'0F2'	'0A0'	'\0F3'	'0F3'	'0A1'
'\0F4'	'0F4'	'0A2'	'\0F5'	'0F5'	'0A3'
'\0F6'	'0F6'	'0A4'	'\0F7'	'0F7'	'0A5'
'\0F8'	'0F8'	'0A6'	'\0F9'	'0F9'	'0A7'
'\0FA'	'0FA'	'0A8'	'\0FB'	'0FB'	'0A9'
'\0FC'	'0FC'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FE'	'0FE'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F00'	'0F00'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F02'	'0F02'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F04'	'0F04'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F06'	'0F06'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F08'	'0F08'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F0A'	'0F0A'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F0C'	'0F0C'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F0E'	'0F0E'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F10'	'0F10'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F12'	'0F12'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F14'	'0F14'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F16'	'0F16'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F18'	'0F18'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F1A'	'0F1A'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F1C'	'0F1C'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F1E'	'0F1E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F20'	'0F20'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F22'	'0F22'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F24'	'0F24'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F26'	'0F26'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F28'	'0F28'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F2A'	'0F2A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F2C'	'0F2C'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F2E'	'0F2E'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F30'	'0F30'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F32'	'0F32'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F34'	'0F34'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F36'	'0F36'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F38'	'0F38'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F3A'	'0F3A'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F3C'	'0F3C'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F3E'	'0F3E'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F40'	'0F40'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F42'	'0F42'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F44'	'0F44'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F46'	'0F46'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F48'	'0F48'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F4A'	'0F4A'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F4C'	'0F4C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F4E'	'0F4E'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F50'	'0F50'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F52'	'0F52'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F54'	'0F54'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F56'	'0F56'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F58'	'0F58'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F5A'	'0F5A'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F5C'	'0F5C'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F5E'	'0F5E'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F60'	'0F60'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F62'	'0F62'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F64'	'0F64'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F66'	'0F66'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F68'	'0F68'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F6A'	'0F6A'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F6C'	'0F6C'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F6E'	'0F6E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F70'	'0F70'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F72'	'0F72'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F74'	'0F74'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F76'	'0F76'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F78'	'0F78'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F7A'	'0F7A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F7C'	'0F7C'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F7E'	'0F7E'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F80'	'0F80'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F82'	'0F82'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F84'	'0F84'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F86'	'0F86'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F88'	'0F88'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F8A'	'0F8A'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F8C'	'0F8C'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F8E'	'0F8E'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F90'	'0F90'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F92'	'0F92'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F94'	'0F94'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F96'	'0F96'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F98'	'0F98'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A'	'0F9A'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9C'	'0F9C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9E'	'0F9E'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9F'	'0F9F'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A0'	'0F9A0'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A2'	'0F9A2'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A4'	'0F9A4'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A6'	'0F9A6'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A8'	'0F9A8'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A9'	'0F9A9'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A00'	'0F9A00'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A02'	'0F9A02'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A04'	'0F9A04'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A06'	'0F9A06'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A08'	'0F9A08'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A09'	'0F9A09'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A0A'	'0F9A0A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A0B'	'0F9A0B'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A0C'	'0F9A0C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A0D'	'0F9A0D'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A0E'	'0F9A0E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A0F'	'0F9A0F'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A10'	'0F9A10'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A11'	'0F9A11'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A12'	'0F9A12'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A13'	'0F9A13'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A14'	'0F9A14'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A15'	'0F9A15'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A16'	'0F9A16'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A17'	'0F9A17'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A18'	'0F9A18'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A19'	'0F9A19'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A1A'	'0F9A1A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A1B'	'0F9A1B'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A1C'	'0F9A1C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A1D'	'0F9A1D'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A1E'	'0F9A1E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A1F'	'0F9A1F'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A20'	'0F9A20'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A21'	'0F9A21'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A22'	'0F9A22'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A23'	'0F9A23'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A24'	'0F9A24'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A25'	'0F9A25'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A26'	'0F9A26'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A27'	'0F9A27'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A28'	'0F9A28'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A29'	'0F9A29'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A2A'	'0F9A2A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A2B'	'0F9A2B'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A2C'	'0F9A2C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A2D'	'0F9A2D'				

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

(Demo with pythonTutor)

Wrap

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



wrap: if offset > 26 then wrap around
% is the remainder
 $27 \% 26 = 1$

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)