

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

- Can I work ahead?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

- Can I work ahead?

Yes! All programs are available, on gradescope, 4 weeks before the deadline.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

- Can I work ahead?

Yes! All programs are available, on gradescope, 4 weeks before the deadline.

- Last lecture didn't go into details of programming. Will you in the future?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

- Can I work ahead?

Yes! All programs are available, on gradescope, 4 weeks before the deadline.

- Last lecture didn't go into details of programming. Will you in the future?

No worries— today, we'll dive into the details of for, range and string methods.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

- Can I work ahead?

Yes! All programs are available, on gradescope, 4 weeks before the deadline.

- Last lecture didn't go into details of programming. Will you in the future?

No worries— today, we'll dive into the details of for, range and string methods.

- You said “when you take second semester...” I just took this class for Pathways...

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

- Can I work ahead?

Yes! All programs are available, on gradescope, 4 weeks before the deadline.

- Last lecture didn't go into details of programming. Will you in the future?

No worries— today, we'll dive into the details of for, range and string methods.

- You said “when you take second semester...” I just took this class for Pathways...

This is Pathways, but we hope that you will be a CS major/minor.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

There is no midterm. Instead there's 14 in-class quizzes.

- When is the final?

Wednesday, 21 May, 9-11am.

- Can I submit late homework?

No. Instead we drop the 5 lowest grades.

- I missed class. Do you need documentation?

No. Missing lecture & quiz grades are replaced by your final exam score.

If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

- Why do I have to work in groups?

It's great practice to explain technical work to others.

- Can I work ahead?

Yes! All programs are available, on gradescope, 4 weeks before the deadline.

- Last lecture didn't go into details of programming. Will you in the future?

No worries— today, we'll dive into the details of for, range and string methods.

- You said “when you take second semester...” I just took this class for Pathways...

This is Pathways, but we hope that you will be a CS major/minor.

We also hope: “Get your education don't forget whence you came...”

Today's Topics



- Research Survey
- For-loops
- `range()`
- Variables: ints and strings
- Lists
- Strings

Why All the Handouts Today?

number of participants	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
number of participants (including instructors)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
number of participants (including instructors)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
number of participants (including instructors)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. What is your gender? Please mark one box next to each tick in the original table.

Male Female Other

2. How many years have you been playing chess?

0-1 year 2-5 years 6-10 years 11-15 years 16-20 years 21-25 years 26-30 years

3. How many years have you been playing chess?

0-1 year 2-5 years 6-10 years 11-15 years 16-20 years 21-25 years 26-30 years

4. Which age group do you belong to? Tick all that apply. (Max 30%)

10-14 years old 15-19 years old 20-24 years old 25-29 years old 30-34 years old 35-39 years old 40-44 years old 45-49 years old 50-54 years old 55-59 years old 60-64 years old 65-69 years old 70-74 years old 75-79 years old 80+ years old

Lecture Slip

Overview

THE VITAMIN D AND COVID-19 STUDY RECRUITMENT FORM <p>If you have questions about your rights as a research participant, or you have concerns or complaints about this study, please contact the Office of Human Research Protection (OHRP) or the Office of Compliance Administration at 314-993-6475 or via email OHRP@dfci.harvard.edu. We will respond to your concerns or complaints as quickly as possible.</p> <p>CDCI office: The Vitamind Study Coordinator Barbara G. Goss, PhD 314-993-6475</p> <p>Signatures of IRB Members:</p> <p>If you agree to participate in this research study, please sign and date below (we will give a copy of the informed consent form to you).</p> <p>Consent Form of Participant:</p> <p>Signature of the Recruit: _____ _____ Signature of Researcher Administering Consent: Protection of Individual Browsing Consent: Signature of Individual Browsing Contact: _____ _____ <small>Print Name _____ Date _____ Last Updated 06/24/2011</small></p>
--

Consent Form

Survey

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Part 1: Consists of two brief surveys completed in class.

Prof. John Ranellucci

Educational Psychology

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".

Prof. John Ranellucci

Educational Psychology

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



- Part 1: Consists of two brief surveys completed in class.
- Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".
- Part 3: Consists of two surveys available online.

Prof. John Ranellucci

Educational Psychology

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



- Part 1: Consists of two brief surveys completed in class.
- Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".
- Part 3: Consists of two surveys available online.
(Little longer and participants will be compensated with a \$20 Amazon gift certificate for completing both surveys.)

Prof. John Ranellucci

Educational Psychology

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Prof. John Ranellucci
Educational Psychology

- Part 1: Consists of two brief surveys completed in class.
- Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".
- Part 3: Consists of two surveys available online.
(Little longer and participants will be compensated with a \$20 Amazon gift certificate for completing both surveys.)

This study is not part of the class, and no individual analyses will be shared with your instructor. Survey links for the online survey will be emailed to all of you, other surveys will be distributed in class.

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)

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.

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



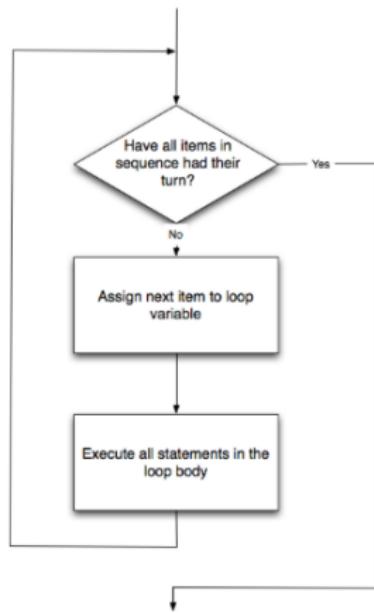
- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

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.)

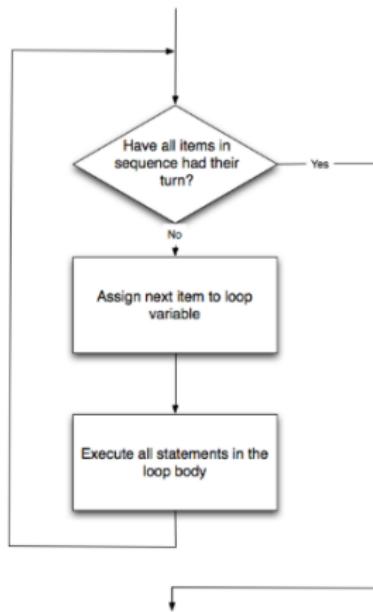
for-loop



```
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

In Pairs or Triples...

Some review and some novel challenges:

```
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(x)  
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(x)  
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,...,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)`

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.
(New version called: Unicode).

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.
(New version called: Unicode).

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	'
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	,	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	-
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

(wiki)



Converting from Character to Code:

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

ASCII TABLE

Decimal Num Char	Octal Num Char	Hex Num Char	Decimal Num Char	Octal Num Char	Hex Num Char
0	0	0	1	1	1
2	2	2	3	3	3
4	4	4	5	5	5
6	6	6	7	7	7
8	10	8	9	11	9
10	12	A	11	13	B
12	14	C	13	15	D
14	16	E	15	17	F
16	20	10	17	21	11
18	22	12	19	23	13
20	24	14	21	25	15
22	26	16	23	27	17
24	28	18	25	29	19
26	30	1A	27	31	1B
28	32	1C	29	33	1D
30	34	1E	31	35	1F
32	36	20	33	37	21
34	38	22	35	39	23
36	40	24	37	41	25
38	42	26	39	43	27
40	44	28	41	45	29
42	46	2A	43	47	2B
44	48	2C	45	49	2D
46	50	2E	47	51	2F
48	52	30	49	53	31
50	54	32	51	55	33
52	56	34	53	57	35
54	58	36	55	59	37
56	60	38	57	61	39
58	62	3A	59	63	3B
60	64	3C	61	65	3D
62	66	3E	63	67	3F
64	68	40	65	69	41
66	70	42	67	71	43
68	72	44	69	73	45
70	74	46	71	75	47
72	76	48	73	77	49
74	78	4A	75	79	4B
76	80	4C	77	81	4D
78	82	4E	79	83	4F
80	84	50	81	85	51
82	86	52	83	87	53
84	88	54	85	89	55
86	90	56	87	91	57
88	92	58	89	93	59
90	94	5A	91	95	5B
92	96	5C	93	97	5D
94	98	5E	95	99	5F
96	100	60	97	101	61
98	102	62	99	103	63
100	104	64	101	105	65
102	106	66	103	107	67
104	108	68	105	109	69
106	110	6A	107	111	6B
108	112	6C	109	113	6D
110	114	6E	111	115	6F
112	116	70	113	117	71
114	118	72	115	119	73
116	120	74	117	121	75
118	122	76	119	123	77
120	124	78	121	125	79
122	126	7A	123	127	7B
124	128	7C	125	129	7D
126	130	7E	127	131	7F
128	132	80	129	133	81
130	134	82	131	135	83
132	136	84	133	137	85
134	138	86	135	139	87
136	140	88	137	141	89
138	142	8A	139	143	8B
140	144	8C	141	145	8D
142	146	8E	143	147	8F
144	148	90	145	149	91
146	150	92	147	151	93
148	152	94	149	153	95
150	154	96	151	155	97
152	156	98	153	157	99
154	158	9A	155	159	9B
156	160	9C	157	161	9D
158	162	9E	159	163	9F
160	164	100	161	165	101
162	166	102	163	167	103
164	168	104	165	169	105
166	170	106	167	171	107
168	172	108	169	173	109
170	174	10A	171	175	10B
172	176	10C	173	177	10D
174	178	10E	175	179	10F
176	180	110	177	181	111
178	182	112	179	183	113
180	184	114	181	185	115
182	186	116	183	187	117
184	188	118	185	189	119
186	190	11A	187	191	11B
188	192	11C	189	193	11D
190	194	11E	191	195	11F
192	196	120	193	197	121
194	198	122	195	199	123
196	200	124	197	201	125
198	202	126	199	203	127
200	204	128	201	205	129
202	206	12A	203	207	12B
204	208	12C	205	209	12D
206	210	12E	207	211	12F
208	212	130	209	213	131
210	214	132	211	215	133
212	216	134	213	217	135
214	218	136	215	219	137
216	220	138	217	221	139
218	222	13A	219	223	13B
220	224	13C	221	225	13D
222	226	13E	223	227	13F
224	228	140	225	229	141
226	230	142	227	231	143
228	232	144	229	233	145
230	234	146	231	235	147
232	236	148	233	237	149
234	238	14A	235	239	14B
236	240	14C	237	241	14D
238	242	14E	239	243	14F
240	244	150	241	245	151
242	246	152	243	247	153
244	248	154	245	249	155
246	250	156	247	251	157
248	252	158	249	253	159
250	254	15A	251	255	15B
252	256	15C	253	257	15D
254	258	15E	255	259	15F
256	260	160	257	261	161
258	262	162	259	263	163
260	264	164	261	265	165
262	266	166	263	267	167
264	268	168	265	269	169
266	270	16A	267	271	16B
268	272	16C	269	273	16D
270	274	16E	271	275	16F
272	276	170	273	277	171
274	278	172	275	279	173
276	280	174	277	281	175
278	282	176	279	283	177
280	284	178	281	285	179
282	286	17A	283	287	17B
284	288	17C	285	289	17D
286	290	17E	287	291	17F
288	292	180	289	293	181
290	294	182	291	295	183
292	296	184	293	297	185
294	298	186	295	299	187
296	300	188	297	301	189
298	302	18A	299	303	18B
300	304	18C	301	305	18D
302	306	18E	303	307	18F
304	308	190	305	309	191
306	310	192	307	311	193
308	312	194	309	313	195
310	314	196	311	315	197
312	316	198	313	317	199
314	318	19A	315	319	19B
316	320	19C	317	321	19D
318	322	19E	319	323	19F
320	324	200	321	325	201
322	326	202	323	327	203
324	328	204	325	329	205
326	330	206	327	331	207
328	332	208	329	333	209
330	334	20A	331	335	20B
332	336	20C	333	337	20D
334	338	20E	335	339	20F
336	340	210	337	341	211
338	342	212	339	343	213
340	344	214	341	345	215
342	346	216	343	347	217
344	348	218	345	349	219
346	350	21A	347	351	21B
348	352	21C	349	353	21D
350	354	21E	351	355	21F
352	356	220	353	357	221
354	358	222	355	359	223
356	360	224	357	361	225
358	362	226	359	363	227
360	364	228	361	365	229
362	366	22A	363	367	22B
364	368	22C	365	369	22D
366	370	22E	367	371	22F
368	372	230	369	373	231
370	374	232	371	375	233
372	376	234	373	377	235
374	378	236	375	379	237
376	380	238	377	381	239
378	382	23A	379	383	23B
380	384	23C	381	385	23D
382	386	23E	383	387	23F
384	388	240	385	389	241
386	390	242	387	391	243
388	392	244	389	393	245
390	394	246	391	395	247
392	396	248	393	397	249
394	398	24A	395	399	24B
396	400	24C	397	401	24D
398	402	24E	399	403	24F
400	404	250	401	405	251
402	406	252	403	407	253
404	408	254	405	409	255
406	410	256	407	411	257
408	412	258	409	413	259
410	414	25A	411	415	25B
412	416	25C	413	417	25D
414	418	25E	415	419	25F
416	420	260	417	421	261
418	422	262	419	423	263
420	424	264	421	425	265
422	426	266	423	427	267
424	428	268	425	429	269
426	430	26A	427	431	26B
428	432	26C	429	433	26D
430	434	26E	431	435	26F
432	436	270	433	437	271
434	438	272	435	439	273
436	440	274	437	441	275
438	442	276	439	443	277
440	444	278	441	445	279
442	446	27A	443	447	27B
444	448	27C	445	449	27D
446	450	27E	447	451	27F
448	452	280	449	453	281
450	454	282	451	455	283
452	456	284	453	457	285
454	458	286	455	459	287
456	460	288	457	461	289
458	462	28A	459	463	28B
460	464	28C	461	465	28D
462	466	28E	463	467	28F
464	468	290	465	469	291
466	470	292	467	471	293
468	472	294	469	473	295
470	474	296	471	475	297
472	476	298	473	477	299

Converting from Character to Code:

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

Decimal New Char	Octal New Char	Hex New Char	Decimal New Char	Octal New Char	Hex New Char
'\0'	'000'	'000'	'\n'	'000'	'000'
'\t'	'009'	'009'	'\r'	'00D'	'00D'
'\v'	'012'	'012'	'\f'	'014'	'014'
'\b'	'010'	'010'	'\a'	'007'	'007'
'\f'	'014'	'014'	'\x00'	'000'	'000'
'\x01'	'001'	'001'	'\x02'	'002'	'002'
'\x03'	'003'	'003'	'\x04'	'004'	'004'
'\x05'	'005'	'005'	'\x06'	'006'	'006'
'\x07'	'007'	'007'	'\x08'	'010'	'010'
'\x09'	'011'	'011'	'\x0A'	'012'	'012'
'\x0B'	'013'	'013'	'\x0C'	'014'	'014'
'\x0D'	'015'	'015'	'\x0E'	'016'	'016'
'\x0F'	'017'	'017'	'\x10'	'020'	'020'
'\x11'	'021'	'021'	'\x12'	'022'	'022'
'\x13'	'023'	'023'	'\x14'	'024'	'024'
'\x15'	'025'	'025'	'\x16'	'026'	'026'
'\x17'	'027'	'027'	'\x18'	'030'	'030'
'\x19'	'031'	'031'	'\x1A'	'032'	'032'
'\x1B'	'033'	'033'	'\x1C'	'034'	'034'
'\x1D'	'035'	'035'	'\x1E'	'036'	'036'
'\x1F'	'037'	'037'	'\x1F'	'037'	'037'
'\x20'	'040'	'040'	'\x21'	'041'	'041'
'\x22'	'042'	'042'	'\x23'	'043'	'043'
'\x24'	'044'	'044'	'\x25'	'045'	'045'
'\x26'	'046'	'046'	'\x27'	'047'	'047'
'\x28'	'050'	'050'	'\x29'	'051'	'051'
'\x2A'	'052'	'052'	'\x2B'	'053'	'053'
'\x2C'	'054'	'054'	'\x2D'	'055'	'055'
'\x2E'	'056'	'056'	'\x2F'	'057'	'057'
'\x2E'	'056'	'056'	'\x30'	'060'	'060'
'\x31'	'061'	'061'	'\x32'	'062'	'062'
'\x33'	'063'	'063'	'\x34'	'064'	'064'
'\x35'	'065'	'065'	'\x36'	'066'	'066'
'\x37'	'067'	'067'	'\x38'	'070'	'070'
'\x39'	'071'	'071'	'\x3A'	'072'	'072'
'\x3B'	'073'	'073'	'\x3C'	'074'	'074'
'\x3D'	'075'	'075'	'\x3E'	'076'	'076'
'\x3F'	'077'	'077'	'\x3F'	'077'	'077'
'\x40'	'080'	'080'	'\x41'	'081'	'081'
'\x42'	'082'	'082'	'\x43'	'083'	'083'
'\x44'	'084'	'084'	'\x45'	'085'	'085'
'\x46'	'086'	'086'	'\x47'	'087'	'087'
'\x48'	'088'	'088'	'\x49'	'089'	'089'
'\x4A'	'090'	'090'	'\x4B'	'091'	'091'
'\x4C'	'092'	'092'	'\x4D'	'093'	'093'
'\x4E'	'094'	'094'	'\x4F'	'095'	'095'
'\x4F'	'095'	'095'	'\x50'	'096'	'096'
'\x51'	'097'	'097'	'\x52'	'098'	'098'
'\x53'	'099'	'099'	'\x54'	'100'	'0A0'
'\x55'	'101'	'0A1'	'\x56'	'102'	'0A2'
'\x57'	'103'	'0A3'	'\x58'	'104'	'0A4'
'\x59'	'105'	'0A5'	'\x5A'	'106'	'0A6'
'\x5B'	'107'	'0A7'	'\x5C'	'108'	'0A8'
'\x5D'	'109'	'0A9'	'\x5E'	'110'	'0AA'
'\x5F'	'111'	'0AB'	'\x5F'	'111'	'0AB'
'\x60'	'113'	'0AC'	'\x61'	'114'	'0AD'
'\x62'	'115'	'0AE'	'\x63'	'116'	'0AF'
'\x64'	'118'	'0B2'	'\x65'	'119'	'0B3'
'\x66'	'121'	'0B6'	'\x67'	'122'	'0B7'
'\x68'	'124'	'0B8'	'\x69'	'125'	'0B9'
'\x6A'	'127'	'0BA'	'\x6B'	'128'	'0BB'
'\x6C'	'130'	'0BC'	'\x6D'	'131'	'0BD'
'\x6E'	'133'	'0BE'	'\x6F'	'134'	'0BF'
'\x70'	'136'	'0C0'	'\x71'	'137'	'0C1'
'\x72'	'139'	'0C2'	'\x73'	'140'	'0C3'
'\x74'	'142'	'0C4'	'\x75'	'143'	'0C5'
'\x76'	'145'	'0C6'	'\x77'	'146'	'0C7'
'\x78'	'148'	'0C8'	'\x79'	'149'	'0C9'
'\x7A'	'151'	'0CA'	'\x7B'	'152'	'0CB'
'\x7C'	'154'	'0CC'	'\x7D'	'155'	'0CD'
'\x7E'	'157'	'0CE'	'\x7F'	'158'	'0CF'
'\x80'	'160'	'0D0'	'\x81'	'161'	'0D1'
'\x82'	'164'	'0D4'	'\x83'	'165'	'0D5'
'\x84'	'168'	'0D8'	'\x85'	'169'	'0D9'
'\x86'	'172'	'0DA'	'\x87'	'173'	'0DB'
'\x88'	'176'	'0DC'	'\x89'	'177'	'0DD'
'\x8A'	'180'	'0DE'	'\x8B'	'181'	'0DF'
'\x8C'	'184'	'0E0'	'\x8D'	'185'	'0E1'
'\x8E'	'188'	'0E4'	'\x8F'	'189'	'0E5'
'\x90'	'192'	'0E8'	'\x91'	'193'	'0E9'
'\x92'	'196'	'0EC'	'\x93'	'197'	'0ED'
'\x94'	'200'	'0F4'	'\x95'	'201'	'0F5'
'\x96'	'204'	'0F6'	'\x97'	'205'	'0F7'
'\x98'	'208'	'0F8'	'\x99'	'209'	'0F9'
'\x9A'	'212'	'0FA'	'\x9B'	'213'	'0FB'
'\x9C'	'216'	'0FC'	'\x9D'	'217'	'0FD'
'\x9E'	'220'	'0FE'	'\x9F'	'221'	'0FF'
'\xA0'	'224'	'0D0'	'\xA1'	'225'	'0D1'
'\xA2'	'228'	'0D4'	'\xA3'	'229'	'0D5'
'\xA4'	'232'	'0D8'	'\xA5'	'233'	'0D9'
'\xA6'	'236'	'0DA'	'\xA7'	'237'	'0DB'
'\xA8'	'240'	'0DC'	'\xA9'	'241'	'0DD'
'\xA9'	'244'	'0E0'	'\xA9'	'245'	'0E1'
'\xA9'	'248'	'0E4'	'\xA9'	'249'	'0E5'
'\xA9'	'252'	'0E8'	'\xA9'	'253'	'0E9'
'\xA9'	'256'	'0EC'	'\xA9'	'257'	'0ED'
'\xA9'	'260'	'0F4'	'\xA9'	'261'	'0F5'
'\xA9'	'264'	'0F6'	'\xA9'	'265'	'0F7'
'\xA9'	'268'	'0F8'	'\xA9'	'269'	'0F9'
'\xA9'	'272'	'0FA'	'\xA9'	'273'	'0FB'
'\xA9'	'276'	'0FC'	'\xA9'	'277'	'0FD'
'\xA9'	'280'	'0FE'	'\xA9'	'281'	'0FF'
'\xA9'	'284'	'0D0'	'\xA9'	'285'	'0D1'
'\xA9'	'288'	'0D4'	'\xA9'	'289'	'0D5'
'\xA9'	'292'	'0D8'	'\xA9'	'293'	'0D9'
'\xA9'	'296'	'0DA'	'\xA9'	'297'	'0DB'
'\xA9'	'300'	'0DC'	'\xA9'	'301'	'0DD'
'\xA9'	'304'	'0E0'	'\xA9'	'305'	'0E1'
'\xA9'	'308'	'0E4'	'\xA9'	'309'	'0E5'
'\xA9'	'312'	'0E8'	'\xA9'	'313'	'0E9'
'\xA9'	'316'	'0EC'	'\xA9'	'317'	'0ED'
'\xA9'	'320'	'0F4'	'\xA9'	'321'	'0F5'
'\xA9'	'324'	'0F6'	'\xA9'	'325'	'0F7'
'\xA9'	'328'	'0F8'	'\xA9'	'329'	'0F9'
'\xA9'	'332'	'0FA'	'\xA9'	'333'	'0FB'
'\xA9'	'336'	'0FC'	'\xA9'	'337'	'0FD'
'\xA9'	'340'	'0FE'	'\xA9'	'341'	'0FF'
'\xA9'	'344'	'0D0'	'\xA9'	'345'	'0D1'
'\xA9'	'348'	'0D4'	'\xA9'	'349'	'0D5'
'\xA9'	'352'	'0D8'	'\xA9'	'353'	'0D9'
'\xA9'	'356'	'0DA'	'\xA9'	'357'	'0DB'
'\xA9'	'360'	'0DC'	'\xA9'	'361'	'0DD'
'\xA9'	'364'	'0E0'	'\xA9'	'365'	'0E1'
'\xA9'	'368'	'0E4'	'\xA9'	'369'	'0E5'
'\xA9'	'372'	'0E8'	'\xA9'	'373'	'0E9'
'\xA9'	'376'	'0EC'	'\xA9'	'377'	'0ED'
'\xA9'	'380'	'0F4'	'\xA9'	'381'	'0F5'
'\xA9'	'384'	'0F6'	'\xA9'	'385'	'0F7'
'\xA9'	'388'	'0F8'	'\xA9'	'389'	'0F9'
'\xA9'	'392'	'0FA'	'\xA9'	'393'	'0FB'
'\xA9'	'396'	'0FC'	'\xA9'	'397'	'0FD'
'\xA9'	'3000'	'0D0'	'\xA9'	'3001'	'0D1'
'\xA9'	'3002'	'0D4'	'\xA9'	'3003'	'0D5'
'\xA9'	'3004'	'0D8'	'\xA9'	'3005'	'0D9'
'\xA9'	'3006'	'0DA'	'\xA9'	'3007'	'0DB'
'\xA9'	'3008'	'0DC'	'\xA9'	'3009'	'0DD'
'\xA9'	'300A'	'0E0'	'\xA9'	'300B'	'0E1'
'\xA9'	'300C'	'0E4'	'\xA9'	'300D'	'0E5'
'\xA9'	'300E'	'0E8'	'\xA9'	'300F'	'0E9'
'\xA9'	'3010'	'0EC'	'\xA9'	'3011'	'0ED'
'\xA9'	'3012'	'0F4'	'\xA9'	'3013'	'0F5'
'\xA9'	'3014'	'0F6'	'\xA9'	'3015'	'0F7'
'\xA9'	'3016'	'0F8'	'\xA9'	'3017'	'0F9'
'\xA9'	'3018'	'0FA'	'\xA9'	'3019'	'0FB'
'\xA9'	'301A'	'0FC'	'\xA9'	'301B'	'0FD'
'\xA9'	'301C'	'0FE'	'\xA9'	'301D'	'0FF'
'\xA9'	'301E'	'0D0'	'\xA9'	'301F'	'0D1'
'\xA9'	'3020'	'0D4'	'\xA9'	'3021'	'0D5'
'\xA9'	'3022'	'0D8'	'\xA9'	'3023'	'0D9'
'\xA9'	'3024'	'0DA'	'\xA9'	'3025'	'0DB'
'\xA9'	'3026'	'0DC'	'\xA9'	'3027'	'0DD'
'\xA9'	'3028'	'0E0'	'\xA9'	'3029'	'0E1'
'\xA9'	'302A'	'0E4'	'\xA9'	'302B'	'0E5'
'\xA9'	'302C'	'0E8'	'\xA9'	'302D'	'0E9'
'\xA9'	'302E'	'0EC'	'\xA9'	'302F'	'0ED'
'\xA9'	'3030'	'0F4'	'\xA9'	'3031'	'0F5'
'\xA9'	'3032'	'0F6'	'\xA9'	'3033'	'0F7'
'\xA9'	'3034'	'0F8'	'\xA9'	'3035'	'0F9'
'\xA9'	'3036'	'0FA'	'\xA9'	'3037'	'0FB'
'\xA9'	'3038'	'0FC'	'\xA9'	'3039'	'0FD'
'\xA9'	'303A'	'0FE'	'\xA9'	'303B'	'0FF'
'\xA9'	'303C'	'0D0'	'\xA9'	'303D'	'0D1'
'\xA9'	'303E'	'0D4'	'\xA9'	'303F'	'0D5'
'\xA9'	'3040'	'0D8'	'\xA9'	'3041'	'0D9'
'\xA9'	'3042'	'0DA'	'\xA9'	'3043'	'0DB'
'\xA9'	'3044'	'0DC'	'\xA9'	'3045'	'0DD'
'\xA9'	'3046'	'0E0'	'\xA9'	'3047'	'0E1'
'\xA9'	'3048'	'0E4'	'\xA9'	'3049'	'0E5'
'\xA9'	'304A'	'0E8'	'\xA9'	'304B'	'0E9'
'\xA9'	'304C'	'0EC'	'\xA9'	'304D'	'0ED'
'\xA9'	'304E'	'0F4'	'\xA9'	'304F'	'0F5'
'\xA9'	'3050'	'0F6'	'\xA9'	'3051'	'0F7'
'\xA9'	'3052'	'0F8'	'\xA9'	'3053'	'0F9'
'\xA9'	'3054'	'0FA'	'\xA9'	'3055'	'0FB'
'\xA9'	'3056'	'0FC'	'\xA9'	'3057'	'0FD'
'\xA9'	'3058'	'0FE'	'\xA9'	'3059'	'0FF'
'\xA9'	'305A'	'0D0'	'\xA9'	'305B'	'0D1'
'\xA9'	'305C'	'0D4'	'\xA9'	'305D'	'0D5'
'\xA9'	'305E'	'0D8'	'\xA9'	'305F'	'0D9'
'\xA9'	'3060'	'0DA'	'\xA9'	'3061'	'0DB'
'\xA9'	'3062'	'0DC'	'\xA9'	'3063'	'0DD'
'\xA9'	'3064'	'0E0'	'\xA9'	'3065'	'0E1'
'\xA9'	'3066'	'0E4'	'\xA9'	'3067'	'0E5'
'\xA9'	'3068'	'0E8'	'\xA9'	'3069'	'0E9'
'\xA9'	'306A'	'0EC'	'\xA9'	'306B'	'0ED'
'\xA9'	'306C'	'0F4'	'\xA9'	'306D'	'0F5'
'\xA9'	'306E'	'0F6'	'\xA9'	'306F'	'0F7'
'\xA9'	'3070'	'0F8'	'\xA9'	'3071'	'0F9'
'\xA9'	'3072'	'0FA'	'\xA9'	'3073'	'0FB'
'\xA9'	'3074'	'0FC'	'\xA9'	'3075'	'0FD'
'\xA9'	'3076'	'0FE'	'\xA9'	'3077'	'0FF'
'\xA9'	'3078'	'0D0'	'\xA9'	'3079'	'0D1'
'\xA9'	'307A'	'0D4'	'\xA9'	'307B'	'0D5'
'\xA9'	'307C'	'0D8'	'\xA9'	'307D'	'0D9'
'\xA9'	'307E'	'0DA'	'\xA9'	'307F'	'0DB'
'\xA9'	'3080'	'0DC'	'\xA9'	'3081'	'0DD'
'\xA9'	'3082'	'0E0'	'\xA9'	'3083'	'0E1'
'\xA9'	'3084'	'0E4'	'\xA9'	'3085'	'0E5'
'\xA9'	'3086'	'0E8'	'\xA9'	'3087'	'0E9'
'\xA9'	'3088'	'0EC'	'\xA9'	'3089'	'0ED'
'\xA9'	'308A'	'0F4'	'\xA9'	'308B'	'0F5'
'\xA9'	'308C'	'0F6'	'\xA9'	'308D'	'0F7'
'\xA9'	'308E'	'0F8'	'\xA9'	'308F'	'0F9'
'\xA9'	'3090'	'0FA'	'\xA9'	'3091'	'0FB'
'\xA9'	'3092'	'0FC'	'\xA9'	'3093'	'0FD'
'\xA9'	'3094'	'0FE'	'\xA9'	'3095'	'0FF'
'\xA9'	'3096'	'0D0'	'\xA9'	'3097'	'0D1'
'\xA9'	'3098'	'0D4'	'\xA9'	'3099'	'0D5'
'\xA9'	'309A'	'0D8'	'\xA9'	'309B'	'0D9'
'\xA9'	'309C'	'0DA'	'\xA9'	'309D'	'0DB'
'\xA9'	'309E'	'0DC'	'\xA9'	'309F'	'0DD'
'\xA9'	'30A0'	'0E0'	'\xA9'	'30A1'	'0E1'
'\xA9'	'30A2'	'0E4'	'\xA9'	'30A3'	'0E5'
'\xA9'	'30A4'	'0E8'	'\xA9'	'30A5'	'0E9'
'\xA9'	'30A6'	'0EC'	'\xA9'	'30A7'	'0ED'
'\xA9'	'30A8'	'0F4'	'\xA9'	'30A9'	'0F5'
'\xA9'	'30A0'	'0F6'	'\xA9'	'30A1'	'0F7'
'\xA9'	'30A2'	'0F8'	'\xA9'	'30A3'	'0F9'
'\xA9'	'30A4'	'0FA'	'\xA9'	'30A5'	'0FB'
'\xA9'	'30A6'	'0FC'	'\xA9'	'30A7'	'0FD'
'\xA9'	'30A8'	'0FE'	'\xA9'	'30A9'	'0FF'
'\xA9'	'30A0'	'0D0'	'\xA9'	'30A1'	'0D1'
'\xA9'	'30A2'	'0D4'	'\xA9'	'30A3'	'0D5'
'\xA9'	'30A4'	'0D8'	'\xA9'	'30A5'	'0D9'
'\xA9'	'30A6'	'0DA'	'\xA9'	'30A7'	'0DB'
'\xA9'	'30A8'	'0DC'	'\xA9'	'30A9'	'0DD'</

Converting from Character to Code:

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

ASCII TABLE														
Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char
0	00	0	NULL	\0	32	20	40	SIGKILL	\000	64	40	100	SIGPOLL	\001
1	01	1	SOH	\001	33	21	41	SIGALRM	\002	65	41	101	SIGSTOP	\003
2	02	2	STX	\002	34	22	42	SIGPOLL	\004	66	42	102	SIGCONT	\005
3	03	3	ETX	\003	35	23	43	SIGPOLL	\006	67	43	103	SIGKILL	\007
4	04	4	ENQ	\004	36	24	44	SIGPOLL	\008	68	44	104	SIGPOLL	\009
5	05	5	KSYN	\005	37	25	45	SIGPOLL	\010	69	45	105	SIGPOLL	\011
6	06	6	ACK	\006	38	26	46	SIGPOLL	\012	70	46	106	SIGPOLL	\013
7	07	7	NAK	\007	39	27	47	SIGPOLL	\014	71	47	107	SIGPOLL	\015
8	08	10	SYN	\008	40	28	48	SIGPOLL	\016	72	48	108	SIGPOLL	\017
9	09	11	EOT	\009	41	29	49	SIGPOLL	\018	73	49	109	SIGPOLL	\019
10	0A	12	EM	\00A	42	2A	4A	SIGPOLL	\01A	74	4A	110	SIGPOLL	\01B
11	0B	13	END	\00B	43	2B	4B	SIGPOLL	\01B	75	4B	111	SIGPOLL	\01C
12	0C	14	ESC	\00C	44	2C	4C	SIGPOLL	\01C	76	4C	112	SIGPOLL	\01D
13	0D	15	SUSP	\00D	45	2D	4D	SIGPOLL	\01D	77	4D	113	SIGPOLL	\01E
14	0E	16	DC1	\00E	46	2E	4E	SIGPOLL	\01E	78	4E	114	SIGPOLL	\01F
15	0F	17	DC2	\00F	47	2F	4F	SIGPOLL	\01F	79	4F	115	SIGPOLL	\020
16	10	20	DC3	\010	48	30	50	SIGPOLL	\01F	80	50	116	SIGPOLL	\021
17	11	21	DC4	\011	49	31	51	SIGPOLL	\01F	81	51	117	SIGPOLL	\022
18	12	22	DC5	\012	50	32	52	SIGPOLL	\01F	82	52	118	SIGPOLL	\023
19	13	23	DC6	\013	51	33	53	SIGPOLL	\01F	83	53	119	SIGPOLL	\024
20	14	24	DC7	\014	52	34	54	SIGPOLL	\01F	84	54	120	SIGPOLL	\025
21	15	25	DC8	\015	53	35	55	SIGPOLL	\01F	85	55	121	SIGPOLL	\026
22	16	26	DC9	\016	54	36	56	SIGPOLL	\01F	86	56	122	SIGPOLL	\027
23	17	27	DC10	\017	55	37	57	SIGPOLL	\01F	87	57	123	SIGPOLL	\028
24	18	28	DC11	\018	56	38	58	SIGPOLL	\01F	88	58	124	SIGPOLL	\029
25	19	29	DC12	\019	57	39	59	SIGPOLL	\01F	89	59	125	SIGPOLL	\02A
26	1A	2A	DC13	\01A	58	3A	5A	SIGPOLL	\01F	90	5A	126	SIGPOLL	\02B
27	1B	2B	DC14	\01B	59	3B	5B	SIGPOLL	\01F	91	5B	127	SIGPOLL	\02C
28	1C	2C	DC15	\01C	5A	3C	5C	SIGPOLL	\01F	92	5C	128	SIGPOLL	\02D
29	1D	2D	DC16	\01D	5B	3D	5D	SIGPOLL	\01F	93	5D	129	SIGPOLL	\02E
30	1E	2E	DC17	\01E	5C	3E	5E	SIGPOLL	\01F	94	5E	130	SIGPOLL	\02F
31	1F	2F	DC18	\01F	5D	3F	5F	SIGPOLL	\01F	95	5F	131	SIGPOLL	\030
32	20	30	DC19	\020	5E	40	60	SIGPOLL	\01F	96	60	132	SIGPOLL	\031
33	21	31	DC1A	\021	5F	41	61	SIGPOLL	\01F	97	61	133	SIGPOLL	\032
34	22	32	DC1B	\022	60	42	62	SIGPOLL	\01F	98	62	134	SIGPOLL	\033
35	23	33	DC1C	\023	61	43	63	SIGPOLL	\01F	99	63	135	SIGPOLL	\034
36	24	34	DC1D	\024	62	44	64	SIGPOLL	\01F	100	64	136	SIGPOLL	\035
37	25	35	DC1E	\025	63	45	65	SIGPOLL	\01F	101	65	137	SIGPOLL	\036
38	26	36	DC1F	\026	64	46	66	SIGPOLL	\01F	102	66	138	SIGPOLL	\037
39	27	37	DC20	\027	65	47	67	SIGPOLL	\01F	103	67	139	SIGPOLL	\038
40	28	38	DC21	\028	66	48	68	SIGPOLL	\01F	104	68	140	SIGPOLL	\039
41	29	39	DC22	\029	67	49	69	SIGPOLL	\01F	105	69	141	SIGPOLL	\03A
42	2A	3A	DC23	\02A	68	4A	6A	SIGPOLL	\01F	106	6A	142	SIGPOLL	\03B
43	2B	3B	DC24	\02B	69	4B	6B	SIGPOLL	\01F	107	6B	143	SIGPOLL	\03C
44	2C	3C	DC25	\02C	6A	4C	6C	SIGPOLL	\01F	108	6C	144	SIGPOLL	\03D
45	2D	3D	DC26	\02D	6B	4D	6D	SIGPOLL	\01F	109	6D	145	SIGPOLL	\03E
46	2E	3E	DC27	\02E	6C	4E	6E	SIGPOLL	\01F	110	6E	146	SIGPOLL	\03F
47	2F	3F	DC28	\02F	6D	4F	6F	SIGPOLL	\01F	111	6F	147	SIGPOLL	\040
48	30	40	DC29	\030	6E	50	70	SIGPOLL	\01F	112	70	148	SIGPOLL	\041
49	31	41	DC2A	\031	6F	51	71	SIGPOLL	\01F	113	71	149	SIGPOLL	\042
50	32	42	DC2B	\032	70	52	72	SIGPOLL	\01F	114	72	150	SIGPOLL	\043
51	33	43	DC2C	\033	71	53	73	SIGPOLL	\01F	115	73	151	SIGPOLL	\044
52	34	44	DC2D	\034	72	54	74	SIGPOLL	\01F	116	74	152	SIGPOLL	\045
53	35	45	DC2E	\035	73	55	75	SIGPOLL	\01F	117	75	153	SIGPOLL	\046
54	36	46	DC2F	\036	74	56	76	SIGPOLL	\01F	118	76	154	SIGPOLL	\047
55	37	47	DC30	\037	75	57	77	SIGPOLL	\01F	119	77	155	SIGPOLL	\048
56	38	48	DC31	\038	76	58	78	SIGPOLL	\01F	120	78	156	SIGPOLL	\049
57	39	49	DC32	\039	77	59	79	SIGPOLL	\01F	121	79	157	SIGPOLL	\04A
58	3A	4A	DC33	\03A	78	5A	7A	SIGPOLL	\01F	122	7A	158	SIGPOLL	\04B
59	3B	4B	DC34	\03B	79	5B	7B	SIGPOLL	\01F	123	7B	159	SIGPOLL	\04C
60	3C	4C	DC35	\03C	7A	5C	7C	SIGPOLL	\01F	124	7C	160	SIGPOLL	\04D
61	3D	4D	DC36	\03D	7B	5D	7D	SIGPOLL	\01F	125	7D	161	SIGPOLL	\04E
62	3E	4E	DC37	\03E	7C	5E	7E	SIGPOLL	\01F	126	7E	162	SIGPOLL	\04F
63	3F	4F	DC38	\03F	7D	5F	7F	SIGPOLL	\01F	127	7F	163	SIGPOLL	\050
64	40	50	DC39	\040	7E	60	80	SIGPOLL	\01F	128	80	164	SIGPOLL	\051
65	41	51	DC3A	\041	7F	61	81	SIGPOLL	\01F	129	81	165	SIGPOLL	\052
66	42	52	DC3B	\042	80	62	82	SIGPOLL	\01F	130	82	166	SIGPOLL	\053
67	43	53	DC3C	\043	81	63	83	SIGPOLL	\01F	131	83	167	SIGPOLL	\054
68	44	54	DC3D	\044	82	64	84	SIGPOLL	\01F	132	84	168	SIGPOLL	\055
69	45	55	DC3E	\045	83	65	85	SIGPOLL	\01F	133	85	169	SIGPOLL	\056
70	46	56	DC3F	\046	84	66	86	SIGPOLL	\01F	134	86	170	SIGPOLL	\057
71	47	57	DC40	\047	85	67	87	SIGPOLL	\01F	135	87	171	SIGPOLL	\058
72	48	58	DC41	\048	86	68	88	SIGPOLL	\01F	136	88	172	SIGPOLL	\059
73	49	59	DC42	\049	87	69	89	SIGPOLL	\01F	137	89	173	SIGPOLL	\05A
74	4A	5A	DC43	\04A	88	6A	8A	SIGPOLL	\01F	138	8A	174	SIGPOLL	\05B
75	4B	5B	DC44	\04B	89	6B	8B	SIGPOLL	\01F	139	8B	175	SIGPOLL	\05C
76	4C	5C	DC45	\04C	8A	6C	8C	SIGPOLL	\01F	140	8C	176	SIGPOLL	\05D
77	4D	5D	DC46	\04D	8B	6D	8D	SIGPOLL	\01F	141	8D	177	SIGPOLL	\05E
78	4E	5E	DC47	\04E	8C	6E	8E	SIGPOLL	\01F	142	8E	178	SIGPOLL	\05F
79	4F	5F	DC48	\04F	8D	6F	8F	SIGPOLL	\01F	143	8F	179	SIGPOLL	\060
80	50	60	DC49	\050	8E	70	90	SIGPOLL	\01F	144	90	180	SIGPOLL	\061
81	51	61	DC4A	\051	8F	71	91	SIGPOLL	\01F	145	91	181	SIGPOLL	\062
82	52	62	DC4B	\052	90	72	92	SIGPOLL	\01F	146	92	182	SIGPOLL	\063
83	53	63	DC4C	\053	91	73	93	SIGPOLL	\01F	147	93	183	SIGPOLL	\064
84	54	64	DC4D	\054	92	74	94	SIGPOLL	\01F	148	94	184	SIGPOLL	\065
85	55	65	DC4E	\055	93	75	95	SIGPOLL	\01F	149	95	185	SIGPOLL	\066
86	56	66	DC4F	\056	94	76	96	SIGPOLL	\01F	150	96	186	SIGPOLL	\067
87	57	67	DC50	\057	95	77	97	SIGPOLL	\01F	151	97	187	SIGPOLL	\068
88	58	68	DC51	\058	96	78	98	SIGPOLL	\01F	152	98	188	SIGPOLL	\069
89	59	69	DC52	\059	97	79	99	SIGPOLL	\01F	153	99	189	SIGPOLL	\06A
90	5A	6A	DC53	\05A	98	7A	9A	SIGPOLL	\01F	154	9A	190	SIGPOLL	\06B
91	5B	6B	DC54	\05B	99	7B	9B	SIGPOLL	\01F	155	9B	191	SIGPOLL	\06C
92	5C	6C	DC55	\05C	9A	7C	9C	SIGPOLL	\01F	156	9C	192	SIGPOLL	\06D
93	5D	6D	DC56	\05D	9B	7D	9D	SIGPOLL	\01F	157	9D	193	SIGPOLL	\06E
94	5E	6E	DC57	\05E	9C	7E	9E	SIGPOLL	\01F	158	9E	194	SIGPOLL	\06F
95	5F	6F	DC58	\05F	9D	7F	9F	SIGPOLL	\01F	159	9F	195	SIGPOLL	\070
96	60	70	DC59	\060	9E	80	A0	SIGPOLL	\01F	160	A0	196	SIGPOLL	\071
97	61	71	DC5A	\061	9F	81	A1	SIGPOLL	\01F	161	A1	197	SIGPOLL	\072
98	62	72	DC5B	\062	A0	82	A2	SIGPOLL	\01F	162	A2	198	SIGPOLL	\073
99	63	73	DC5C	\063	A1	83	A3	SIGPOLL	\01F	163	A3	199	SIGPOLL	\074
100	64	74	DC5D	\064	A2	84	A4	SIGPOLL	\01F	164	A4	200	SIGPOLL	\075
101	65	75	DC5E	\065	A3	85	A5	SIGPOLL	\01F	165	A5	201	SIGPOLL	\076
102	66	76	DC5F	\066	A4	86	A6	SIGPOLL	\01F	166	A6			

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

Converting from Character to Code:

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

ASCII TABLE														
Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char
0	00	0	NULL	\0	32	20	40	SIGKILL	\000	64	40	100	SIGPOLL	\001
1	01	1	SOH	\001	33	21	41	SIGALRM	\002	65	41	101	SIGSTOP	\003
2	02	2	STX	\002	34	22	42	SIGPOLL	\004	66	42	102	SIGCONT	\005
3	03	3	ETX	\003	35	23	43	SIGPOLL	\006	67	43	103	SIGKILL	\007
4	04	4	ENQ	\004	36	24	44	SIGPOLL	\008	68	44	104	SIGPOLL	\009
5	05	5	KSYN	\005	37	25	45	SIGPOLL	\010	69	45	105	SIGPOLL	\011
6	06	6	ACK	\006	38	26	46	SIGPOLL	\012	70	46	106	SIGPOLL	\013
7	07	7	NAK	\007	39	27	47	SIGPOLL	\014	71	47	107	SIGPOLL	\015
8	08	10	SYN	\008	40	28	48	SIGPOLL	\016	72	48	108	SIGPOLL	\017
9	09	11	EOT	\009	41	29	49	SIGPOLL	\018	73	49	109	SIGPOLL	\019
10	0A	12	EM	\00A	42	2A	4A	SIGPOLL	\01A	74	4A	110	SIGPOLL	\01B
11	0B	13	END	\00B	43	2B	4B	SIGPOLL	\01B	75	4B	111	SIGPOLL	\01C
12	0C	14	ESC	\00C	44	2C	4C	SIGPOLL	\01C	76	4C	112	SIGPOLL	\01D
13	0D	15	SUSP	\00D	45	2D	4D	SIGPOLL	\01D	77	4D	113	SIGPOLL	\01E
14	0E	16	DC1	\00E	46	2E	4E	SIGPOLL	\01E	78	4E	114	SIGPOLL	\01F
15	0F	17	DC2	\00F	47	2F	4F	SIGPOLL	\01F	79	4F	115	SIGPOLL	\020
16	10	20	DC3	\010	48	30	50	SIGPOLL	\01F	80	50	116	SIGPOLL	\021
17	11	21	DC4	\011	49	31	51	SIGPOLL	\01F	81	51	117	SIGPOLL	\022
18	12	22	DC5	\012	50	32	52	SIGPOLL	\01F	82	52	118	SIGPOLL	\023
19	13	23	DC6	\013	51	33	53	SIGPOLL	\01F	83	53	119	SIGPOLL	\024
20	14	24	DC7	\014	52	34	54	SIGPOLL	\01F	84	54	120	SIGPOLL	\025
21	15	25	DC8	\015	53	35	55	SIGPOLL	\01F	85	55	121	SIGPOLL	\026
22	16	26	DC9	\016	54	36	56	SIGPOLL	\01F	86	56	122	SIGPOLL	\027
23	17	27	DC10	\017	55	37	57	SIGPOLL	\01F	87	57	123	SIGPOLL	\028
24	18	28	DC11	\018	56	38	58	SIGPOLL	\01F	88	58	124	SIGPOLL	\029
25	19	29	DC12	\019	57	39	59	SIGPOLL	\01F	89	59	125	SIGPOLL	\02A
26	1A	2A	DC13	\01A	58	3A	5A	SIGPOLL	\01F	90	5A	126	SIGPOLL	\02B
27	1B	2B	DC14	\01B	59	3B	5B	SIGPOLL	\01F	91	5B	127	SIGPOLL	\02C
28	1C	2C	DC15	\01C	5A	3C	5C	SIGPOLL	\01F	92	5C	128	SIGPOLL	\02D
29	1D	2D	DC16	\01D	5B	3D	5D	SIGPOLL	\01F	93	5D	129	SIGPOLL	\02E
30	1E	2E	DC17	\01E	5C	3E	5E	SIGPOLL	\01F	94	5E	130	SIGPOLL	\02F
31	1F	2F	DC18	\01F	5D	3F	5F	SIGPOLL	\01F	95	5F	131	SIGPOLL	\030
32	20	30	DC19	\020	5E	40	60	SIGPOLL	\01F	96	60	132	SIGPOLL	\031
33	21	31	DC1A	\021	5F	41	61	SIGPOLL	\01F	97	61	133	SIGPOLL	\032
34	22	32	DC1B	\022	60	42	62	SIGPOLL	\01F	98	62	134	SIGPOLL	\033
35	23	33	DC1C	\023	61	43	63	SIGPOLL	\01F	99	63	135	SIGPOLL	\034
36	24	34	DC1D	\024	62	44	64	SIGPOLL	\01F	100	64	136	SIGPOLL	\035
37	25	35	DC1E	\025	63	45	65	SIGPOLL	\01F	101	65	137	SIGPOLL	\036
38	26	36	DC1F	\026	64	46	66	SIGPOLL	\01F	102	66	138	SIGPOLL	\037
39	27	37	DC20	\027	65	47	67	SIGPOLL	\01F	103	67	139	SIGPOLL	\038
40	28	38	DC21	\028	66	48	68	SIGPOLL	\01F	104	68	140	SIGPOLL	\039
41	29	39	DC22	\029	67	49	69	SIGPOLL	\01F	105	69	141	SIGPOLL	\03A
42	2A	3A	DC23	\02A	68	4A	6A	SIGPOLL	\01F	106	6A	142	SIGPOLL	\03B
43	2B	3B	DC24	\02B	69	4B	6B	SIGPOLL	\01F	107	6B	143	SIGPOLL	\03C
44	2C	3C	DC25	\02C	6A	4C	6C	SIGPOLL	\01F	108	6C	144	SIGPOLL	\03D
45	2D	3D	DC26	\02D	6B	4D	6D	SIGPOLL	\01F	109	6D	145	SIGPOLL	\03E
46	2E	3E	DC27	\02E	6C	4E	6E	SIGPOLL	\01F	110	6E	146	SIGPOLL	\03F
47	2F	3F	DC28	\02F	6D	4F	6F	SIGPOLL	\01F	111	6F	147	SIGPOLL	\040
48	30	40	DC29	\030	6E	50	70	SIGPOLL	\01F	112	70	148	SIGPOLL	\041
49	31	41	DC2A	\031	6F	51	71	SIGPOLL	\01F	113	71	149	SIGPOLL	\042
50	32	42	DC2B	\032	70	52	72	SIGPOLL	\01F	114	72	150	SIGPOLL	\043
51	33	43	DC2C	\033	71	53	73	SIGPOLL	\01F	115	73	151	SIGPOLL	\044
52	34	44	DC2D	\034	72	54	74	SIGPOLL	\01F	116	74	152	SIGPOLL	\045
53	35	45	DC2E	\035	73	55	75	SIGPOLL	\01F	117	75	153	SIGPOLL	\046
54	36	46	DC2F	\036	74	56	76	SIGPOLL	\01F	118	76	154	SIGPOLL	\047
55	37	47	DC30	\037	75	57	77	SIGPOLL	\01F	119	77	155	SIGPOLL	\048
56	38	48	DC31	\038	76	58	78	SIGPOLL	\01F	120	78	156	SIGPOLL	\049
57	39	49	DC32	\039	77	59	79	SIGPOLL	\01F	121	79	157	SIGPOLL	\04A
58	3A	4A	DC33	\03A	78	5A	7A	SIGPOLL	\01F	122	7A	158	SIGPOLL	\04B
59	3B	4B	DC34	\03B	79	5B	7B	SIGPOLL	\01F	123	7B	159	SIGPOLL	\04C
60	3C	4C	DC35	\03C	7A	5C	7C	SIGPOLL	\01F	124	7C	160	SIGPOLL	\04D
61	3D	4D	DC36	\03D	7B	5D	7D	SIGPOLL	\01F	125	7D	161	SIGPOLL	\04E
62	3E	4E	DC37	\03E	7C	5E	7E	SIGPOLL	\01F	126	7E	162	SIGPOLL	\04F
63	3F	4F	DC38	\03F	7D	5F	7F	SIGPOLL	\01F	127	7F	163	SIGPOLL	\050
64	40	50	DC39	\040	7E	60	80	SIGPOLL	\01F	128	80	164	SIGPOLL	\051
65	41	51	DC3A	\041	7F	61	81	SIGPOLL	\01F	129	81	165	SIGPOLL	\052
66	42	52	DC3B	\042	80	62	82	SIGPOLL	\01F	130	82	166	SIGPOLL	\053
67	43	53	DC3C	\043	81	63	83	SIGPOLL	\01F	131	83	167	SIGPOLL	\054
68	44	54	DC3D	\044	82	64	84	SIGPOLL	\01F	132	84	168	SIGPOLL	\055
69	45	55	DC3E	\045	83	65	85	SIGPOLL	\01F	133	85	169	SIGPOLL	\056
70	46	56	DC3F	\046	84	66	86	SIGPOLL	\01F	134	86	170	SIGPOLL	\057
71	47	57	DC40	\047	85	67	87	SIGPOLL	\01F	135	87	171	SIGPOLL	\058
72	48	58	DC41	\048	86	68	88	SIGPOLL	\01F	136	88	172	SIGPOLL	\059
73	49	59	DC42	\049	87	69	89	SIGPOLL	\01F	137	89	173	SIGPOLL	\05A
74	4A	5A	DC43	\04A	88	6A	8A	SIGPOLL	\01F	138	8A	174	SIGPOLL	\05B
75	4B	5B	DC44	\04B	89	6B	8B	SIGPOLL	\01F	139	8B	175	SIGPOLL	\05C
76	4C	5C	DC45	\04C	8A	6C	8C	SIGPOLL	\01F	140	8C	176	SIGPOLL	\05D
77	4D	5D	DC46	\04D	8B	6D	8D	SIGPOLL	\01F	141	8D	177	SIGPOLL	\05E
78	4E	5E	DC47	\04E	8C	6E	8E	SIGPOLL	\01F	142	8E	178	SIGPOLL	\05F
79	4F	5F	DC48	\04F	8D	6F	8F	SIGPOLL	\01F	143	8F	179	SIGPOLL	\060
80	50	60	DC49	\050	8E	70	90	SIGPOLL	\01F	144	90	180	SIGPOLL	\061
81	51	61	DC4A	\051	8F	71	91	SIGPOLL	\01F	145	91	181	SIGPOLL	\062
82	52	62	DC4B	\052	90	72	92	SIGPOLL	\01F	146	92	182	SIGPOLL	\063
83	53	63	DC4C	\053	91	73	93	SIGPOLL	\01F	147	93	183	SIGPOLL	\064
84	54	64	DC4D	\054	92	74	94	SIGPOLL	\01F	148	94	184	SIGPOLL	\065
85	55	65	DC4E	\055	93	75	95	SIGPOLL	\01F	149	95	185	SIGPOLL	\066
86	56	66	DC4F	\056	94	76	96	SIGPOLL	\01F	150	96	186	SIGPOLL	\067
87	57	67	DC50	\057	95	77	97	SIGPOLL	\01F	151	97	187	SIGPOLL	\068
88	58	68	DC51	\058	96	78	98	SIGPOLL	\01F	152	98	188	SIGPOLL	\069
89	59	69	DC52	\059	97	79	99	SIGPOLL	\01F	153	99	189	SIGPOLL	\06A
90	5A	6A	DC53	\05A	98	7A	9A	SIGPOLL	\01F	154	9A	190	SIGPOLL	\06B
91	5B	6B	DC54	\05B	99	7B	9B	SIGPOLL	\01F	155	9B	191	SIGPOLL	\06C
92	5C	6C	DC55	\05C	9A	7C	9C	SIGPOLL	\01F	156	9C	192	SIGPOLL	\06D
93	5D	6D	DC56	\05D	9B	7D	9D	SIGPOLL	\01F	157	9D	193	SIGPOLL	\06E
94	5E	6E	DC57	\05E	9C	7E	9E	SIGPOLL	\01F	158	9E	194	SIGPOLL	\06F
95	5F	6F	DC58	\05F	9D	7F	9F	SIGPOLL	\01F	159	9F	195	SIGPOLL	\070
96	60	70	DC59	\060	9E	80	A0	SIGPOLL	\01F	160	A0	196	SIGPOLL	\071
97	61	71	DC5A	\061	9F	81	A1	SIGPOLL	\01F	161	A1	197	SIGPOLL	\072
98	62	72	DC5B	\062	A0	82	A2	SIGPOLL	\01F	162	A2	198	SIGPOLL	\073
99	63	73	DC5C	\063	A1	83	A3	SIGPOLL	\01F	163	A3	199	SIGPOLL	\074
100	64	74	DC5D	\064	A2	84	A4	SIGPOLL	\01F	164	A4	200	SIGPOLL	\075
101	65	75	DC5E	\065	A3	85	A5	SIGPOLL	\01F	165	A5	201	SIGPOLL	\076
102	66	76	DC5F	\066	A4	86	A6	SIGPOLL	\01F	166	A6			

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

Converting from Character to Code:

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

Decimal Num Char	Octal Num Char	Hex Num Char	Character	Decimal Num Char	Octal Num Char	Hex Num Char
0	000	000		32	040	20
1	001	001	!	33	041	21
2	002	002	!"	34	042	22
3	003	003	"	35	043	23
4	004	004	"!"	36	044	24
5	005	005	"!"!"	37	045	25
6	006	006	"!"!"!"	38	046	26
7	007	007	"!"!"!"!"	39	047	27
8	010	008	"!"!"!"!"!"	40	050	28
9	011	009	"!"!"!"!"!"!"	41	051	29
A	020	00A	"!"!"!"!"!"!"!"	42	052	2A
B	021	00B	"!"!"!"!"!"!"!"!"	43	053	2B
C	022	00C	"!"!"!"!"!"!"!"!"!"	44	054	2C
D	023	00D	"!"!"!"!"!"!"!"!"!"!"	45	055	2D
E	024	00E	"!"!"!"!"!"!"!"!"!"!"!"	46	056	2E
F	025	00F	"!"!"!"!"!"!"!"!"!"!"!"!"	47	057	2F
G	030	010	"!"!"!"!"!"!"!"!"!"!"!"!"!"	48	060	30
H	031	011	"!"!"!"!"!"!"!"!"!"!"!"!"!"!"	49	061	31
I	032	012	"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"	50	062	32
J	033	013	"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"	51	063	33
K	034	014	"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"	52	064	34
L	035	015	"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"	53	065	35
M	036	016	"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"!"	54	066	36
N	037	017	"!"	55	067	37
O	040	018	"!"	56	070	38
P	041	019	"!"	57	071	39
Q	042	01A	"!"	58	072	3A
R	043	01B	"!"	59	073	3B
S	044	01C	"!"	60	074	3C
T	045	01D	"!"	61	075	3D
U	046	01E	"!"	62	076	3E
V	047	01F	"!"	63	077	3F
W	050	020	"!"	64	080	40
X	051	021	"!"	65	081	41
Y	052	022	"!"	66	082	42
Z	053	023	"!"	67	083	43
	054	024	"!"	68	084	44
	055	025	"!"	69	085	45
	056	026	"!"	70	086	46
	057	027	"!"	71	087	47
	058	028	"!"	72	088	48
	059	029	"!"	73	089	49
	060	02A	"!"	74	090	4A
	061	02B	"!"	75	091	4B
	062	02C	"!"	76	092	4C
	063	02D	"!"	77	093	4D
	064	02E	"!"	78	094	4E
	065	02F	"!"	79	095	4F
	066	030	"!"	80	096	50
	067	031	"!"	81	097	51
	068	032	"!"	82	098	52
	069	033	"!"	83	099	53
	070	034	"!"	84	0A0	54
	071	035	"!"	85	0A1	55
	072	036	"!"	86	0A2	56
	073	037	"!"	87	0A3	57
	074	038	"!"	88	0A4	58
	075	039	"!"	89	0A5	59
	076	03A	"!"	90	0A6	5A
	077	03B	"!"	91	0A7	5B
	078	03C	"!"	92	0A8	5C
	079	03D	"!"	93	0A9	5D
	080	03E	"!"	94	0AA	5E
	081	03F	"!"	95	0AB	5F
	082	040	"!"	96	0AC	60
	083	041	"!"	97	0AD	61
	084	042	"!"	98	0AE	62
	085	043	"!"	99	0AF	63
	086	044	"!"	100	0B0	64

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

Converting from Character to Code:

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

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

In Pairs or Triples...

Some review and some novel challenges:

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



Python Tutor

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

(Demo with pythonTutor)

User Input

Covered in detail in Lab 2:

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

(Demo with pythonTutor)

Side Note: '+' for numbers and strings

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



Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

More on Strings...

From Final Exam, Fall 2017, Version 1, #1:

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



More on Strings...

Name: _____

EmpID: _____

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:
`There are ??? fun days in a week`

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

There are ??? fun days in a week
Two of them are ???

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

There are ??? fun days in a week
Two of them are ???
My favorite ??? is Saturday.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

There are ??? fun days in a week
Two of them are ???
My favorite ??? is Saturday.
- Will get 1/3 to 1/2 points for writing down the basic structure.

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...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

There are 3 fun days in a week

Two of them are ???

My favorite ??? is Saturday.

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")
```

```
"FriXXXyXXXSaturXXXyXXXSunXXX"  
days = ['Fri', 'sSatur', 'sSun']
```

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

There are 3 fun days in a week
Two of them are Friday Sunday
My favorite ??? is Saturday.

Lecture Slip

- What is printed? Write your answer for each in the output box.

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
#Indices:   0     1     2     3     4     5     6     7     8     9     10    11
#Or:          ....      -3     -2     -1
```

Output:

```
half = months[6]
print(half.upper())
```

```
print(months[-1].lower())
```

```
start = 9
print(months[start-1])
```

```
term = 3
print(months[(start+term-1)%12])
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print()  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

► For-loops

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print()  
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()  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`

Recap

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings

Recap

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic

Recap

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

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

Recap

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

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

Recap

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

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

Recap

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10 for i in range(2):  
11     for j in range(2):  
12         print('Look around,')  
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation
 - ▶ Functions: `ord()` and `char()`
 - ▶ String Manipulation
- Pass your lecture slips to the end of the rows for the UTA's to collect.

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.
- Past exams are on the webpage ([under Final Exam Information](#)).

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.
- Past exams are on the webpage ([under Final Exam Information](#)).
- We're starting with Spring 2018, Mock Exam.

Writing Boards



- Return writing boards as you leave...