

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.

- Who is Henriette Avram?

Frequently Asked Questions

From lecture slips & recitation sections.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?

Frequently Asked Questions

From lecture slips & recitation sections.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.

Frequently Asked Questions

From lecture slips & recitation sections.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- 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.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 9-11am.

Frequently Asked Questions

From lecture slips & recitation sections.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 9-11am.
- Can I submit late homework?

Frequently Asked Questions

From lecture slips & recitation sections.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.

Frequently Asked Questions

From lecture slips & recitation sections.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 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.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 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.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 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.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 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.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 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.

- Who is Henriette Avram?
Pioneering computer programmer & Hunter Alumna.
- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Wednesday, 19 December, 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.

Today's Topics



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

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



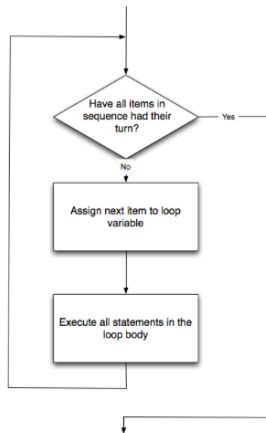
- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

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.)

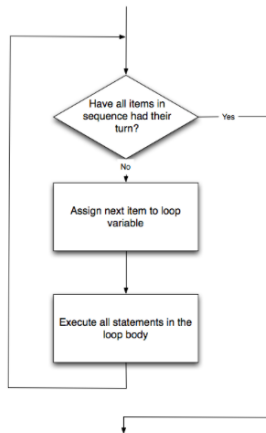
for-loop



```
for i in list:  
    statement1  
    statement2  
    statement3
```

How to Think Like CS, §4.5

for-loop



How to Think Like CS, §4.5

```
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()`.

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 the list `[0,1,2,3,...,100]`, you would write:

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the the list `[0,1,2,3,...,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 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 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 $\text{start} + k * \text{step}$ less than stop)
- For example, if you want the 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..., last]`
(where last is the largest $\text{start} + k * \text{step}$ less than stop)
- For example, if you want the 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	[ENG 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	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	SOH	17	Q	33	A	49	1
2	STX	18	R	34	B	50	2
3	ETX	19	S	35	C	51	3
4	END	20	T	36	D	52	4
5	SO	21	U	37	E	53	5
6	ST	22	V	38	F	54	6
7	ACK	23	W	39	G	55	7
8	BS	24	X	40	H	56	8
9	HT	25	Y	41	I	57	9
10	LF	26	Z	42	J	58	:
11	VT	27	[43	K	59	;
12	FF	28	\	44	L	60	<
13	SOH	29]	45	M	61	=
14	STX	30	^	46	N	62	>
15	ETX	31	_	47	O	63	?
16	END	32	SPACE	48	P	64	0
17	SO	33	!	49	Q	65	a
18	ST	34	"	50	R	66	b
19	ACK	35	#	51	S	67	c
20	BS	36	\$	52	T	68	d
21	HT	37	%	53	U	69	e
22	LF	38	&	54	V	70	f
23	VT	39	'	55	W	71	g
24	FF	40	(56	X	72	h
25	SOH	41)	57	Y	73	i
26	STX	42	*	58	Z	74	j
27	ETX	43	+	59	[75	k
28	END	44	,	60	\	76	l
29	SO	45	-	61]	77	m
30	ST	46	.	62	^	78	n
31	ACK	47	/	63	_	79	o
32	BS	48	0	64	SPACE	80	0
33	HT	49	1	65	a	81	A
34	LF	50	2	66	b	82	B
35	VT	51	3	67	c	83	C
36	FF	52	4	68	d	84	D
37	SOH	53	5	69	e	85	E
38	STX	54	6	70	f	86	F
39	ETX	55	7	71	g	87	G
40	END	56	8	72	h	88	H
41	SO	57	9	73	i	89	I
42	ST	58	:	74	j	90	J
43	ACK	59	;	75	k	91	K
44	BS	60	<	76	l	92	L
45	HT	61	=	77	m	93	M
46	LF	62	>	78	n	94	N
47	VT	63	?	79	o	95	O
48	FF	64	0	80	0	96	P
49	SOH	65	a	81	A	97	Q
50	STX	66	b	82	B	98	R
51	ETX	67	c	83	C	99	S
52	END	68	d	84	D	100	T
53	SO	69	e	85	E	101	U
54	ST	70	f	86	F	102	V
55	ACK	71	g	87	G	103	W
56	BS	72	h	88	H	104	X
57	HT	73	i	89	I	105	Y
58	LF	74	j	90	J	106	Z
59	VT	75	k	91	K	107	[
60	FF	76	l	92	L	108	\
61	SOH	77	m	93	M	109]
62	STX	78	n	94	N	110	^
63	ETX	79	o	95	O	111	_
64	END	80	0	96	P	112	0
65	SO	81	A	97	Q	113	1
66	ST	82	B	98	R	114	2
67	ACK	83	C	99	S	115	3
68	BS	84	D	100	T	116	4
69	HT	85	E	101	U	117	5
70	LF	86	F	102	V	118	6
71	VT	87	G	103	W	119	7
72	FF	88	H	104	X	120	8
73	SOH	89	I	105	Y	121	9
74	STX	90	J	106	Z	122	:
75	ETX	91	K	107	[123	;
76	END	92	L	108	\	124	<
77	SO	93	M	109]	125	=
78	ST	94	N	110	^	126	>
79	ACK	95	O	111	_	127	?
80	BS	96	P	112	0	128	0
81	HT	97	Q	113	1	129	1
82	LF	98	R	114	2	130	2
83	VT	99	S	115	3	131	3
84	FF	100	T	116	4	132	4
85	SOH	101	U	117	5	133	5
86	STX	102	V	118	6	134	6
87	ETX	103	W	119	7	135	7
88	END	104	X	120	8	136	8
89	SO	105	Y	121	9	137	9
90	ST	106	Z	122	:	138	:
91	ACK	107	[123	;	139	;
92	BS	108	\	124	<	140	<
93	HT	109]	125	=	141	=
94	LF	110	^	126	>	142	>
95	VT	111	_	127	?	143	?
96	FF	112	0	128	0	144	0
97	SOH	113	1	129	1	145	1
98	STX	114	2	130	2	146	2
99	ETX	115	3	131	3	147	3
100	END	116	4	132	4	148	4
101	SO	117	5	133	5	149	5
102	ST	118	6	134	6	150	6
103	ACK	119	7	135	7	151	7
104	BS	120	8	136	8	152	8
105	HT	121	9	137	9	153	9
106	LF	122	:	138	:	154	:
107	VT	123	;	139	;	155	;
108	FF	124	<	140	<	156	<
109	SOH	125	=	141	=	157	=
110	STX	126	>	142	>	158	>
111	ETX	127	?	143	?	159	?
112	END	128	0	144	0	160	0
113	SO	129	1	145	1	161	1
114	ST	130	2	146	2	162	2
115	ACK	131	3	147	3	163	3
116	BS	132	4	148	4	164	4
117	HT	133	5	149	5	165	5
118	LF	134	6	150	6	166	6
119	VT	135	7	151	7	167	7
120	FF	136	8	152	8	168	8
121	SOH	137	9	153	9	169	9
122	STX	138	:	154	:	170	:
123	ETX	139	;	155	;	171	;
124	END	140	<	156	<	172	<
125	SO	141	=	157	=	173	=
126	ST	142	>	158	>	174	>
127	ACK	143	?	159	?	175	?
128	BS	144	0	160	0	176	0
129	HT	145	1	161	1	177	1
130	LF	146	2	162	2	178	2
131	VT	147	3	163	3	179	3
132	FF	148	4	164	4	180	4
133	SOH	149	5	165	5	181	5
134	STX	150	6	166	6	182	6
135	ETX	151	7	167	7	183	7
136	END	152	8	168	8	184	8
137	SO	153	9	169	9	185	9
138	ST	154	:	170	:	186	:
139	ACK	155	;	171	;	187	;
140	BS	156	<	172	<	188	<
141	HT	157	=	173	=	189	=
142	LF	158	>	174	>	190	>
143	VT	159	?	175	?	191	?
144	FF	160	0	176	0	192	0
145	SOH	161	1	177	1	193	1
146	STX	162	2	178	2	194	2
147	ETX	163	3	179	3	195	3
148	END	164	4	180	4	196	4
149	SO	165	5	181	5	197	5
150	ST	166	6	182	6	198	6
151	ACK	167	7	183	7	199	7
152	BS	168	8	184	8	200	8
153	HT	169	9	185	9	201	9
154	LF	170	:	186	:	202	:
155	VT	171	;	187	;	203	;
156	FF	172	<	188	<	204	<
157	SOH	173	=	189	=	205	=
158	STX	174	>	190	>	206	>
159	ETX	175	?	191	?	207	?
160	END	176	0	192	0	208	0
161	SO	177	1	193	1	209	1
162	ST	178	2	194	2	210	2
163	ACK	179	3	195	3	211	3
164	BS	180	4	196	4	212	4
165	HT	181	5	197	5	213	5
166	LF	182	6	198	6	214	6
167	VT	183	7	199	7	215	7
168	FF	184	8	200	8	216	8
169	SOH	185	9	201	9	217	9
170	STX	186	:	202	:	218	:
171	ETX	187	;	203	;	219	;
172	END	188	<	204	<	220	<
173	SO	189	=	205	=	221	=
174	ST	190	>	206	>	222	>
175	ACK	191	?	207	?	223	?
176	BS	192	0	208	0	224	0
177	HT	193	1	209	1	225	1
178	LF	194	2	210	2	226	2
179	VT	195	3	211	3	227	3
180	FF	196	4	212	4	228	4
181	SOH	197	5	213	5	229	5
182	STX	198	6	214	6	230	6
183	ETX	199	7	215	7	231	7
184	END	200	8	216	8	232	8
185	SO	201	9	217	9	233	9
186	ST	202	:	218	:	234	:
187	ACK	203	;	219	;	235	;
188	BS	204	<	220	<	236	<
189	HT	205	=	221	=	237	=
190	LF	206	>	222	>	238	>
191	VT	207	?	223	?	239	?
192	FF	208	0	224	0	240	0
193	SOH	209	1	225	1	241	1
194	STX	210	2	226	2	242	2
195	ETX	211	3	227	3	243	3
196	END	212	4	228	4	244	4
197	SO	213	5	229	5	245	5
198	ST	214	6	230	6	246	6
199	ACK	215	7	231	7	247	7
200	BS	216	8	232	8	248	8
201	HT	217	9	233	9	249	9
202	LF	218	:	234	:	250	:
203	VT	219	;	235	;	251	;
204	FF	220	<	236	<	252	<
205	SOH	221	=	237	=	253	=
206	STX	222	>	238	>	254	>
207	ETX	223	?	239	?	255	?
208	END	224	0	240	0		
209	SO	225	1	241	1		
210	ST	226	2	242	2		
211	ACK	227	3	243	3		
212	BS	228	4	24			

Converting from Character to Code:

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

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

ASCII TABLE

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	SOH	17	Q	33	A	49	1
2	STX	18	R	34	B	50	2
3	ETX	19	S	35	C	51	3
4	END	20	T	36	D	52	4
5	SO	21	U	37	E	53	5
6	SI	22	V	38	F	54	6
7	DEL	23	W	39	G	55	7
8		24	X	40	H	56	8
9		25	Y	41	I	57	9
10	LF	26	Z	42	J	58	:
11		27	[43	K	59	;
12	FF	28	\	44	L	60	<
13	CR	29]	45	M	61	=
14		30	^	46	N	62	>
15		31	_	47	O	63	?

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		16	10	P	32	20	[48	30	0
1	01	SOH	17	11	Q	33	21	\	49	31	1
2	02	STX	18	12	R	34	22]	50	32	2
3	03	ETX	19	13	S	35	23	^	51	33	3
4	04	END	20	14	T	36	24	_	52	34	4
5	05	SO	21	15	U	37	25	`	53	35	5
6	06	SI	22	16	V	38	26	{	54	36	6
7	07	BS	23	17	W	39	27		55	37	7
8	08	HT	24	18	X	40	28	~	56	38	8
9	09	LF	25	19	Y	41	29	DEL	57	39	9
10	0A	VT	26	1A	Z	42	2A				
11	0B	FF	27	1B	[43	2B				
12	0C	DEL	28	1C	\	44	2C				
13	0D		29	1D]	45	2D				
14	0E		30	1E	^	46	2E				
15	0F		31	1F	_	47	2F				

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00	NUL	128	80	DEL	129	81		130	82	
1	01		129	81		130	82		131	83	
2	02		130	82		131	83		132	84	
3	03		131	83		132	84		133	85	
4	04		132	84		133	85		134	86	
5	05		133	85		134	86		135	87	
6	06		134	86		135	87		136	88	
7	07		135	87		136	88		137	89	
8	08		136	88		137	89		138	8A	
9	09		137	89		138	8A		139	8B	
10	0A		138	8A		139	8B		140	8C	
11	0B		139	8B		140	8C		141	8D	
12	0C		140	8C		141	8D		142	8E	
13	0D		141	8D		142	8E		143	8F	
14	0E		142	8E		143	8F		144	90	
15	0F		143	8F		144	90		145	91	
16	10		144	90		145	91		146	92	
17	11		145	91		146	92		147	93	
18	12		146	92		147	93		148	94	
19	13		147	93		148	94		149	95	
20	14		148	94		149	95		150	96	
21	15		149	95		150	96		151	97	
22	16		150	96		151	97		152	98	
23	17		151	97		152	98		153	99	
24	18		152	98		153	99		154	9A	
25	19		153	99		154	9A		155	9B	
26	1A		154	9A		155	9B		156	9C	
27	1B		155	9B		156	9C		157	9D	
28	1C		156	9C		157	9D		158	9E	
29	1D		157	9D		158	9E		159	9F	
30	1E		158	9E		159	9F		160	A0	
31	1F		159	9F		160	A0		161	A1	
32	20		160	A0		161	A1		162	A2	
33	21		161	A1		162	A2		163	A3	
34	22		162	A2		163	A3		164	A4	
35	23		163	A3		164	A4		165	A5	
36	24		164	A4		165	A5		166	A6	
37	25		165	A5		166	A6		167	A7	
38	26		166	A6		167	A7		168	A8	
39	27		167	A7		168	A8		169	A9	
40	28		168	A8		169	A9		170	AA	
41	29		169	A9		170	AA		171	AB	
42	2A		170	AA		171	AB		172	AC	
43	2B		171	AB		172	AC		173	AD	
44	2C		172	AC		173	AD		174	AE	
45	2D		173	AD		174	AE		175	AF	
46	2E		174	AE		175	AF		176	B0	
47	2F		175	AF		176	B0		177	B1	
48	30		176	B0		177	B1		178	B2	
49	31		177	B1		178	B2		179	B3	
50	32		178	B2		179	B3		180	B4	
51	33		179	B3		180	B4		181	B5	
52	34		180	B4		181	B5		182	B6	
53	35		181	B5		182	B6		183	B7	
54	36		182	B6		183	B7		184	B8	
55	37		183	B7		184	B8		185	B9	
56	38		184	B8		185	B9		186	BA	
57	39		185	B9		186	BA		187	BB	
58	3A		186	BA		187	BB		188	BC	
59	3B		187	BB		188	BC		189	BD	
60	3C		188	BC		189	BD		190	BE	
61	3D		189	BD		190	BE		191	BF	
62	3E		190	BE		191	BF		192	C0	
63	3F		191	BF		192	C0		193	C1	
64	40		192	C0		193	C1		194	C2	
65	41		193	C1		194	C2		195	C3	
66	42		194	C2		195	C3		196	C4	
67	43		195	C3		196	C4		197	C5	
68	44		196	C4		197	C5		198	C6	
69	45		197	C5		198	C6		199	C7	
70	46		198	C6		199	C7		200	C8	
71	47		199	C7		200	C8		201	C9	
72	48		200	C8		201	C9		202	CA	
73	49		201	C9		202	CA		203	CB	
74	4A		202	CA		203	CB		204	CC	
75	4B		203	CB		204	CC		205	CD	
76	4C		204	CC		205	CD		206	CE	
77	4D		205	CD		206	CE		207	CF	
78	4E		206	CE		207	CF		208	D0	
79	4F		207	CF		208	D0		209	D1	
80	50		208	D0		209	D1		210	D2	
81	51		209	D1		210	D2		211	D3	
82	52		210	D2		211	D3		212	D4	
83	53		211	D3		212	D4		213	D5	
84	54		212	D4		213	D5		214	D6	
85	55		213	D5		214	D6		215	D7	
86	56		214	D6		215	D7		216	D8	
87	57		215	D7		216	D8		217	D9	
88	58		216	D8		217	D9		218	DA	
89	59		217	D9		218	DA		219	DB	
90	5A		218	DA		219	DB		220	DC	
91	5B		219	DB		220	DC		221	DD	
92	5C		220	DC		221	DD		222	DE	
93	5D		221	DD		222	DE		223	DF	
94	5E		222	DE		223	DF		224	E0	
95	5F		223	DF		224	E0		225	E1	
96	60		224	E0		225	E1		226	E2	
97	61		225	E1		226	E2		227	E3	
98	62		226	E2		227	E3		228	E4	
99	63		227	E3		228	E4		229	E5	
100	64		228	E4		229	E5		230	E6	
101	65		229	E5		230	E6		231	E7	
102	66		230	E6		231	E7		232	E8	
103	67		231	E7		232	E8		233	E9	
104	68		232	E8		233	E9		234	EA	
105	69		233	E9		234	EA		235	EB	
106	6A		234	EA		235	EB		236	EC	
107	6B		235	EB		236	EC		237	ED	
108	6C		236	EC		237	ED		238	EE	
109	6D		237	ED		238	EE		239	EF	
110	6E		238	EE		239	EF		240	F0	
111	6F		239	EF		240	F0		241	F1	
112	70		240	F0		241	F1		242	F2	
113	71		241	F1		242	F2		243	F3	
114	72		242	F2		243	F3		244	F4	
115	73		243	F3		244	F4		245	F5	
116	74		244	F4		245	F5		246	F6	
117	75		245	F5		246	F6		247	F7	
118	76		246	F6		247	F7		248	F8	
119	77		247	F7		248	F8		249	F9	
120	78		248	F8		249	F9		250	FA	
121	79		249	F9		250	FA		251	FB	
122	7A		250	FA		251	FB		252	FC	
123	7B		251	FB		252	FC		253	FD	
124	7C		252	FC		253	FD		254	FE	
125	7D		253	FD		254	FE		255	FF	

- `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.)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		16	10	P	32	20	R	48	30	T
1	01		17	11	Q	33	21	S	49	31	U
2	02		18	12	R	34	22	T	50	32	V
3	03		19	13	S	35	23	U	51	33	W
4	04		20	14	T	36	24	V	52	34	X
5	05		21	15	U	37	25	W	53	35	Y
6	06		22	16	V	38	26	X	54	36	Z
7	07		23	17	W	39	27	Y	55	37	[
8	08		24	18	X	40	28	Z	56	38	\
9	09		25	19	Y	41	29	[57	39]
10	0A		26	1A	Z	42	2A	\	58	3A	^
11	0B		27	1B	[43	2B]	59	3B	_
12	0C		28	1C	\	44	2C	^	60	3C	`
13	0D		29	1D]	45	2D	_	61	3D	a
14	0E		30	1E	^	46	2E	`	62	3E	b
15	0F		31	1F	_	47	2F	a	63	3F	c
16	10	A	32	20	b	48	30	n	64	40	d
17	11	B	33	21	c	49	31	o	65	41	e
18	12	C	34	22	d	50	32	p	66	42	f
19	13	D	35	23	e	51	33	q	67	43	g
20	14	E	36	24	f	52	34	r	68	44	h
21	15	F	37	25	g	53	35	s	69	45	i
22	16	G	38	26	h	54	36	t	70	46	j
23	17	H	39	27	i	55	37	u	71	47	k
24	18	I	40	28	j	56	38	v	72	48	l
25	19	J	41	29	k	57	39	w	73	49	m
26	1A	K	42	2A	l	58	3A	x	74	4A	n
27	1B	L	43	2B	m	59	3B	y	75	4B	o
28	1C	M	44	2C	n	60	3C	z	76	4C	p
29	1D	N	45	2D	o	61	3D	{	77	4D	q
30	1E	O	46	2E	p	62	3E		78	4E	r
31	1F	P	47	2F	q	63	3F	}	79	4F	s
32	20	Q	48	30	r	64	40	~	80	50	t
33	21	R	49	31	s	65	41		81	51	u
34	22	S	50	32	t	66	42		82	52	v
35	23	T	51	33	u	67	43		83	53	w
36	24	U	52	34	v	68	44		84	54	x
37	25	V	53	35	w	69	45		85	55	y
38	26	W	54	36	x	70	46		86	56	z
39	27	X	55	37	y	71	47		87	57	[
40	28	Y	56	38	z	72	48		88	58	\
41	29	Z	57	39	[73	49		89	59]
42	2A		58	3A	\	74	4A		90	5A	^
43	2B		59	3B]	75	4B		91	5B	_
44	2C		60	3C	^	76	4C		92	5C	`
45	2D		61	3D	_	77	4D		93	5D	a
46	2E		62	3E	`	78	4E		94	5E	b
47	2F		63	3F	a	79	4F		95	5F	c
48	30	A	64	40	d	80	50		96	60	
49	31	B	65	41	e	81	51		97	61	
50	32	C	66	42	f	82	52		98	62	
51	33	D	67	43	g	83	53		99	63	
52	34	E	68	44	h	84	54		100	64	
53	35	F	69	45	i	85	55		101	65	
54	36	G	70	46	j	86	56		102	66	
55	37	H	71	47	k	87	57		103	67	
56	38	I	72	48	l	88	58		104	68	
57	39	J	73	49	m	89	59		105	69	
58	3A	K	74	4A	n	90	5A		106	6A	
59	3B	L	75	4B	o	91	5B		107	6B	
60	3C	M	76	4C	p	92	5C		108	6C	
61	3D	N	77	4D	q	93	5D		109	6D	
62	3E	O	78	4E	r	94	5E		110	6E	
63	3F	P	79	4F	s	95	5F		111	6F	
64	40	Q	80	50	t	96	60		112	70	
65	41	R	81	51	u	97	61		113	71	
66	42	S	82	52	v	98	62		114	72	
67	43	T	83	53	w	99	63		115	73	
68	44	U	84	54	x	100	64		116	74	
69	45	V	85	55	y	101	65		117	75	
70	46	W	86	56	z	102	66		118	76	
71	47	X	87	57	[103	67		119	77	
72	48	Y	88	58	\	104	68		120	78	
73	49	Z	89	59]	105	69		121	79	
74	4A		90	5A	^	106	6A		122	7A	
75	4B		91	5B	_	107	6B		123	7B	
76	4C		92	5C	`	108	6C		124	7C	
77	4D		93	5D	a	109	6D		125	7D	
78	4E		94	5E	b	110	6E		126	7E	
79	4F		95	5F	c	111	6F		127	7F	
80	50		96	60		112	70		128	80	
81	51		97	61		113	71		129	81	
82	52		98	62		114	72		130	82	
83	53		99	63		115	73		131	83	
84	54		100	64		116	74		132	84	
85	55		101	65		117	75		133	85	
86	56		102	66		118	76		134	86	
87	57		103	67		119	77		135	87	
88	58		104	68		120	78		136	88	
89	59		105	69		121	79		137	89	
90	5A		106	6A		122	7A		138	8A	
91	5B		107	6B		123	7B		139	8B	
92	5C		108	6C		124	7C		140	8C	
93	5D		109	6D		125	7D		141	8D	
94	5E		110	6E		126	7E		142	8E	
95	5F		111	6F		127	7F		143	8F	
96	60		112	70		128	80		144	90	
97	61		113	71		129	81		145	91	
98	62		114	72		130	82		146	92	
99	63		115	73		131	83		147	93	
100	64		116	74		132	84		148	94	
101	65		117	75		133	85		149	95	
102	66		118	76		134	86		150	96	
103	67		119	77		135	87		151	97	
104	68		120	78		136	88		152	98	
105	69		121	79		137	89		153	99	
106	6A		122	7A		138	8A		154	9A	
107	6B		123	7B		139	8B		155	9B	
108	6C		124	7C		140	8C		156	9C	
109	6D		125	7D		141	8D		157	9D	
110	6E		126	7E		142	8E		158	9E	
111	6F		127	7F		143	8F		159	9F	
112	70		128	80		144	90		160	A0	
113	71		129	81		145	91		161	A1	
114	72		130	82		146	92		162	A2	
115	73		131	83		147	93		163	A3	
116	74		132	84		148	94		164	A4	
117	75		133	85		149	95		165	A5	
118	76		134	86		150	96		166	A6	
119	77		135	87		151	97		167	A7	
120	78		136	88		152	98		168	A8	
121	79		137	89		153	99		169	A9	
122	7A		138	8A		154	9A		170	AA	
123	7B		139	8B		155	9B		171	AB	
124	7C		140	8C		156	9C		172	AC	
125	7D		141	8D		157	9D		173	AD	
126	7E		142	8E		158	9E		174	AE	
127	7F		143	8F		159	9F		175	AF	
128	80		144	90		160	A0		176	B0	
129	81		145	91		161	A1		177	B1	
130	82		146	92		162	A2		178	B2	
131	83		147	93		163	A3		179	B3	
132	84		148	94		164	A4		180	B4	
133	85		149	95		165	A5		181	B5	
134	86		150	96		166	A6		182	B6	
135	87		151	97		167	A7		183	B7	
136	88		152	98		168	A8		184	B8	
137	89		153	99		169	A9		185	B9	
138	8A		154	9A		170	AA		186	BA	
139	8B		155	9B		171	AB		187	BB	
140	8C		156	9C		172	AC		188	BC	
141	8D		157	9D		173	AD		189	BD	
142	8E		158	9E		174	AE		190	BE	
143	8F		159	9F		175	AF		191	BF	
144	90		160	A0		176	B0		192	C0	
145	91		161	A1		177	B1		193	C1	
146	92		162	A2		178	B2		194	C2	
147	93		163	A3		179	B3		195	C3	
148	94		164	A4		180	B4		196	C4	
149	95		165	A5		181	B5		197	C5	
150	96		166	A6		182	B6		198	C6	
151	97		167	A7		183	B7		199	C7	
152	98		168	A8		184	B8		200	C8	
153	99		169	A9							

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 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)
```

(Demo with pythonTutor)

User Input

Covered in detail in Lab 2:

```
→ 1 mess = input('Please enter a message: ')\n   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...

Ended last class with 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:

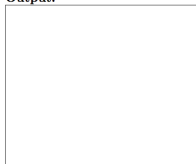
EmpID:

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~~s~~Saturday~~s~~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.

```
"FridaysSaturdaysSunday"  
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.

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

- Different delimiters give different lists:

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

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.

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

- Different delimiters give different lists:

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

1. 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(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

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

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


Recap

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

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

Recap

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

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

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ range()
 - ▶ Variables: ints and strings

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

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic

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

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation

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

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation
 - ▶ Functions: `ord()` and `char()`

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

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation
 - ▶ Functions: `ord()` and `char()`
 - ▶ String Manipulation

```
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!')
```

- 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



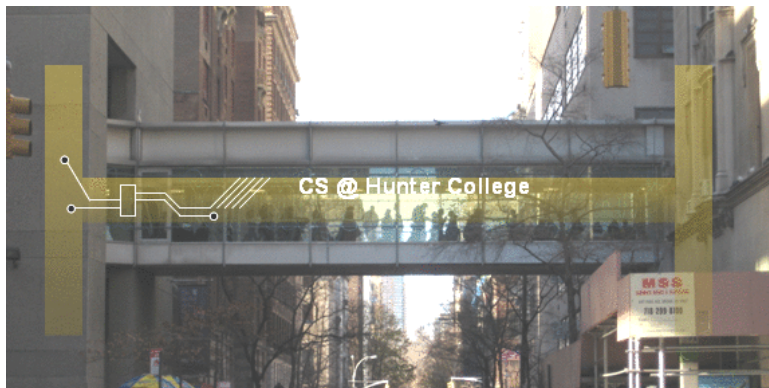
- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.
- Past exams are on the webpage (under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