

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

Lecture Slip

CSsci 127 (Hunter)

Lecture 2

5 February 2019

4 / 39

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.



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".

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.

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.)

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



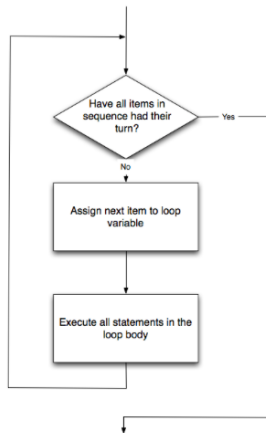
- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

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.)

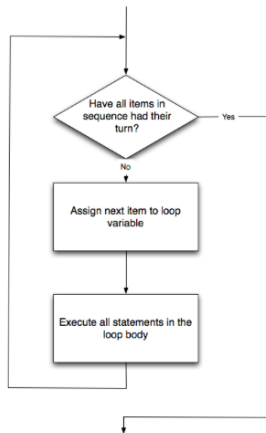
for-loop



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

How to Think Like CS, §4.5

for-loop



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 $start+k*step$ 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	ET	23	W	39	G	55	7
8	CS	24	X	40	H	56	8
9	FS	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	SOB	29]	45	M	61	=
14	STB	30	^	46	N	62	>
15	ETB	31	_	47	O	63	?
16		32	SPACE	48	0	64	0
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	0	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		64	0	80	P	96	0
49		65	1	81	Q	97	a
50		66	2	82	R	98	b
51		67	3	83	S	99	c
52		68	4	84	T	100	d
53		69	5	85	U	101	e
54		70	6	86	V	102	f
55		71	7	87	W	103	g
56		72	8	88	X	104	h
57		73	9	89	Y	105	i
58		74	:	90	Z	106	j
59		75	;	91	[107	k
60		76	<	92	\	108	l
61		77	=	93]	109	m
62		78	>	94	^	110	n
63		79	?	95	_	111	o
64		80	0	96	0	112	p
65		81	1	97	a	113	q
66		82	2	98	b	114	r
67		83	3	99	c	115	s
68		84	4	100	d	116	t
69		85	5	101	e	117	u
70		86	6	102	f	118	v
71		87	7	103	g	119	w
72		88	8	104	h	120	x
73		89	9	105	i	121	y
74		90	:	106	j	122	z
75		91	;	107	k	123	{
76		92	<	108	l	124	
77		93	=	109	m	125	}
78		94	>	110	n	126	~
79		95	?	111	o	127	DEL

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	0	32	@	48	0
1		17	1	33	!	49	1
2		18	2	34	"	50	2
3		19	3	35	#	51	3
4		20	4	36	\$	52	4
5		21	5	37	%	53	5
6		22	6	38	&	54	6
7		23	7	39	'	55	7
8		24	8	40	(56	8
9		25	9	41)	57	9
10		26	A	42	*	58	A
11		27	B	43	+	59	B
12		28	C	44	,	60	C
13		29	D	45	-	61	D
14		30	E	46	.	62	E
15		31	F	47	:	63	F
16	0	32	@	48	0	64	0
17	1	33	!	49	1	65	1
18	2	34	"	50	2	66	2
19	3	35	#	51	3	67	3
20	4	36	\$	52	4	68	4
21	5	37	%	53	5	69	5
22	6	38	&	54	6	70	6
23	7	39	'	55	7	71	7
24	8	40	(56	8	72	8
25	9	41)	57	9	73	9
26	A	42	*	58	A	74	A
27	B	43	+	59	B	75	B
28	C	44	,	60	C	76	C
29	D	45	-	61	D	77	D
30	E	46	.	62	E	78	E
31	F	47	:	63	F	79	F
32		48	0	64	0	80	
33	!	49	1	65	1	81	!
34	"	50	2	66	2	82	"
35	#	51	3	67	3	83	#
36	\$	52	4	68	4	84	\$
37	%	53	5	69	5	85	%
38	&	54	6	70	6	86	&
39	'	55	7	71	7	87	'
40	(56	8	72	8	88	(
41)	57	9	73	9	89)
42	*	58	A	74	A	90	*
43	+	59	B	75	B	91	+
44	,	60	C	76	C	92	,
45	-	61	D	77	D	93	-
46	.	62	E	78	E	94	.
47	:	63	F	79	F	95	:
48	0	64	0	80		96	0
49	1	65	1	81	!	97	1
50	2	66	2	82	"	98	2
51	3	67	3	83	#	99	3
52	4	68	4	84	\$	100	4
53	5	69	5	85	%	101	5
54	6	70	6	86	&	102	6
55	7	71	7	87	'	103	7
56	8	72	8	88	(104	8
57	9	73	9	89)	105	9
58	A	74	A	90	*	106	A
59	B	75	B	91	+	107	B
60	C	76	C	92	,	108	C
61	D	77	D	93	-	109	D
62	E	78	E	94	.	110	E
63	F	79	F	95	:	111	F
64	0	80		96	0	112	
65	1	81	!	97	1	113	!
66	2	82	"	98	2	114	"
67	3	83	#	99	3	115	#
68	4	84	\$	100	4	116	\$
69	5	85	%	101	5	117	%
70	6	86	&	102	6	118	&
71	7	87	'	103	7	119	'
72	8	88	(104	8	120	(
73	9	89)	105	9	121)
74	A	90	*	106	A	122	*
75	B	91	+	107	B	123	+
76	C	92	,	108	C	124	,
77	D	93	-	109	D	125	-
78	E	94	.	110	E	126	.
79	F	95	:	111	F	127	:
80		96	0	112		128	
81	!	97	1	113	!	129	!
82	"	98	2	114	"	130	"
83	#	99	3	115	#	131	#
84	\$	100	4	116	\$	132	\$
85	%	101	5	117	%	133	%
86	&	102	6	118	&	134	&
87	'	103	7	119	'	135	'
88	(104	8	120	(136	(
89)	105	9	121)	137)
90	*	106	A	122	*	138	*
91	+	107	B	123	+	139	+
92	,	108	C	124	,	140	,
93	-	109	D	125	-	141	-
94	.	110	E	126	.	142	.
95	:	111	F	127	:	143	:
96	0	112		128		144	
97	1	113	!	129	!	145	1
98	2	114	"	130	"	146	2
99	3	115	#	131	#	147	3
100	4	116	\$	132	\$	148	4
101	5	117	%	133	%	149	5
102	6	118	&	134	&	150	6
103	7	119	'	135	'	151	7
104	8	120	(136	(152	8
105	9	121)	137)	153	9
106	A	122	*	138	*	154	A
107	B	123	+	139	+	155	B
108	C	124	,	140	,	156	C
109	D	125	-	141	-	157	D
110	E	126	.	142	.	158	E
111	F	127	:	143	:	159	F
112		128		144		160	
113	!	129	!	145	!	161	!
114	"	130	"	146	"	162	"
115	#	131	#	147	#	163	#
116	\$	132	\$	148	\$	164	\$
117	%	133	%	149	%	165	%
118	&	134	&	150	&	166	&
119	'	135	'	151	'	167	'
120	(136	(152	(168	(
121)	137)	153)	169)
122	*	138	*	154	*	170	*
123	+	139	+	155	+	171	+
124	,	140	,	156	,	172	,
125	-	141	-	157	-	173	-
126	.	142	.	158	.	174	.
127	:	143	:	159	:	175	:
128		144		160		176	
129	!	145	!	161	!	177	!
130	"	146	"	162	"	178	"
131	#	147	#	163	#	179	#
132	\$	148	\$	164	\$	180	\$
133	%	149	%	165	%	181	%
134	&	150	&	166	&	182	&
135	'	151	'	167	'	183	'
136	(152	(168	(184	(
137)	153)	169)	185)
138	*	154	*	170	*	186	*
139	+	155	+	171	+	187	+
140	,	156	,	172	,	188	,
141	-	157	-	173	-	189	-
142	.	158	.	174	.	190	.
143	:	159	:	175	:	191	:
144		160		176		192	
145	!	161	!	177	!	193	!
146	"	162	"	178	"	194	"
147	#	163	#	179	#	195	#
148	\$	164	\$	180	\$	196	\$
149	%	165	%	181	%	197	%
150	&	166	&	182	&	198	&
151	'	167	'	183	'	199	'
152	(168	(184	(200	(
153)	169)	185)	201)
154	*	170	*	186	*	202	*
155	+	171	+	187	+	203	+
156	,	172	,	188	,	204	,
157	-	173	-	189	-	205	-
158	.	174	.	190	.	206	.
159	:	175	:	191	:	207	:
160		176		192		208	
161	!	177	!	193	!	209	!
162	"	178	"	194	"	210	"
163	#	179	#	195	#	211	#
164	\$	180	\$	196	\$	212	\$
165	%	181	%	197	%	213	%
166	&	182	&	198	&	214	&
167	'	183	'	199	'	215	'
168	(184	(200	(216	(
169)	185)	201)	217)
170	*	186	*	202	*	218	*
171	+	187	+	203	+	219	+
172	,	188	,	204	,	220	,
173	-	189	-	205	-	221	-
174	.	190	.	206	.	222	.
175	:	191	:	207	:	223	:
176		192		208		224	
177	!	193	!	209	!	225	!
178	"	194	"	210	"	226	"
179	#	195	#	211	#	227	#
180	\$	196	\$	212	\$	228	\$
181	%	197	%	213	%	229	%
182	&	198	&	214	&	230	&
183	'	199	'	215	'	231	'
184	(200	(216	(232	(
185)	201)	217)	233)
186	*	202	*	218	*	234	*
187	+	203	+	219	+	235	+
188	,	204	,	220	,	236	,
189	-	205	-	221	-	237	-
190	.	206	.	222	.	238	.
191	:	207	:	223	:	239	:
192		208		224		240	
193	!	209	!	225	!	241	!
194	"	210	"	226	"	242	"
195	#	211	#	227	#	243	#
196	\$	212	\$	228	\$	244	\$
197	%	213	%	229	%	245	%
198	&	214	&	230	&	246	&
199	'	215	'	231	'	247	'
200	(216	(232	(248	(
201)	217)	233)	249)
202	*	218	*	234	*	250	*
203	+	219	+	235	+	251	+
204	,	220	,	236	,	252	,
205	-	221	-	237	-	253	-
206	.	222	.	238	.	254	.
207	:	223	:	239	:	255	:
208		224		240			
209	!	225	!	241	!		
210	"	226	"	242	"		
211	#	227	#	243	#		
212	\$	228	\$	244	\$		
213	%	229	%	245	%		
214	&	230	&	246	&		
215	'	231	'	247	'		
216	(232	(248	(
217)	233)	249)		
218	*	234	*	250	*		
219	+	235	+	251	+		
220	,	236	,	252	,		
221	-	237	-	253	-		
222	.	238	.	254	.		
223	:	239	:	255	:</		

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		32		48	
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	!	64	@	80	P
17		33	"	65	A	81	Q
18		34	"	66	B	82	R
19		35	\$	67	C	83	S
20		36	%	68	D	84	T
21		37	&	69	E	85	U
22		38	'	70	F	86	V
23		39	(71	G	87	W
24		40)	72	H	88	X
25		41	*	73	I	89	Y
26		42	+	74	J	90	Z
27		43	,	75	K	91	[
28		44	-	76	L	92	\
29		45	.	77	M	93]
30		46	/	78	N	94	^
31		47	/	79	O	95	_
32	!	48	0	80	P	96	`
33	"	49	1	81	Q	97	a
34	"	50	2	82	R	98	b
35	\$	51	3	83	S	99	c
36	%	52	4	84	T	100	d
37	&	53	5	85	U	101	e
38	'	54	6	86	V	102	f
39	(55	7	87	W	103	g
40)	56	8	88	X	104	h
41	*	57	9	89	Y	105	i
42	+	58	:	90	Z	106	j
43	,	59	;	91	[107	k
44	-	60	<	92	\	108	l
45	.	61	=	93]	109	m
46	/	62	>	94	^	110	n
47	/	63	?	95	_	111	o
48	0	64	@	96	`	112	p
49	1	65	A	97	a	113	q
50	2	66	B	98	b	114	r
51	3	67	C	99	c	115	s
52	4	68	D	100	d	116	t
53	5	69	E	101	e	117	u
54	6	70	F	102	f	118	v
55	7	71	G	103	g	119	w
56	8	72	H	104	h	120	x
57	9	73	I	105	i	121	y
58	:	74	J	106	j	122	z
59	;	75	K	107	k	123	{
60	<	76	L	108	l	124	
61	=	77	M	109	m	125	}
62	>	78	N	110	n	126	~
63	?	79	O	111	o	127	DEL

- `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		17	11	Q	33	21	\	49	31	1
2	02		18	12	R	34	22]	50	32	2
3	03		19	13	S	35	23	^	51	33	3
4	04		20	14	T	36	24	_	52	34	4
5	05		21	15	U	37	25	`	53	35	5
6	06		22	16	V	38	26	{	54	36	6
7	07		23	17	W	39	27		55	37	7
8	08		24	18	X	40	28	~	56	38	8
9	09		25	19	Y	41	29		57	39	9
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	"
14	0E		30	1E	^	46	2E		62	3E	'
15	0F		31	1F	_	47	2F		63	3F	~
16	10	@	32	20	[48	30	0	64	40	
17	11	A	33	21	\	49	31	1	65	41	a
18	12	B	34	22]	50	32	2	66	42	b
19	13	C	35	23	^	51	33	3	67	43	c
20	14	D	36	24	_	52	34	4	68	44	d
21	15	E	37	25	`	53	35	5	69	45	e
22	16	F	38	26	{	54	36	6	70	46	f
23	17		39	27		55	37	7	71	47	g
24	18		40	28	~	56	38	8	72	48	h
25	19		41	29		57	39	9	73	49	i
26	1A		42	2A		58	3A	.	74	4A	j
27	1B		43	2B		59	3B	,	75	4B	k
28	1C		44	2C		60	3C	;	76	4C	l
29	1D		45	2D		61	3D	"	77	4D	m
30	1E		46	2E		62	3E	'	78	4E	n
31	1F		47	2F		63	3F	~	79	4F	o
32	20		48	30	0	64	40		80	50	
33	21	!	49	31	1	65	41	a	81	51	!
34	22	"	50	32	2	66	42	b	82	52	"
35	23	#	51	33	3	67	43	c	83	53	"
36	24	\$	52	34	4	68	44	d	84	54	\$
37	25	%	53	35	5	69	45	e	85	55	%
38	26	&	54	36	6	70	46	f	86	56	&
39	27	'	55	37	7	71	47	g	87	57	'
40	28	(56	38	8	72	48	h	88	58	(
41	29)	57	39	9	73	49	i	89	59)
42	2A	*	58	3A	.	74	4A	j	90	5A	*
43	2B	+	59	3B	,	75	4B	k	91	5B	+
44	2C	,	60	3C	;	76	4C	l	92	5C	,
45	2D	-	61	3D	"	77	4D	m	93	5D	-
46	2E	.	62	3E	'	78	4E	n	94	5E	.
47	2F	/	63	3F	~	79	4F	o	95	5F	/
48	30	0	64	40		80	50		96	60	0
49	31	1	65	41	a	81	51	!	97	61	1
50	32	2	66	42	b	82	52	"	98	62	2
51	33	3	67	43	c	83	53	"	99	63	3
52	34	4	68	44	d	84	54	\$	100	64	4
53	35	5	69	45	e	85	55	%			
54	36	6	70	46	f	86	56	&			
55	37	7	71	47	g	87	57	'			
56	38	8	72	48	h	88	58	(
57	39	9	73	49	i	89	59)			
58	3A	.	74	4A	j	90	5A	*			
59	3B	,	75	4B	k	91	5B	+			
60	3C	;	76	4C	l	92	5C	,			
61	3D	"	77	4D	m	93	5D	-			
62	3E	'	78	4E	n	94	5E	.			
63	3F	~	79	4F	o	95	5F	/			
64	40		80	50		96	60	0			
65	41	a	81	51	!	97	61	1			
66	42	b	82	52	"	98	62	2			
67	43	c	83	53	"	99	63	3			
68	44	d	84	54	\$						
69	45	e	85	55	%						
70	46	f	86	56	&						
71	47	g	87	57	'						
72	48	h	88	58	(
73	49	i	89	59)						
74	4A	j	90	5A	*						
75	4B	k	91	5B	+						
76	4C	l	92	5C	,						
77	4D	m	93	5D	-						
78	4E	n	94	5E	.						
79	4F	o	95	5F	/						
80	50		96	60	0						
81	51	!	97	61	1						
82	52	"	98	62	2						
83	53	"	99	63	3						
84	54	\$									
85	55	%									
86	56	&									
87	57	'									
88	58	(
89	59)									
90	5A	*									
91	5B	+									
92	5C	,									
93	5D	-									
94	5E	.									
95	5F	/									
96	60	0									
97	61	1									
98	62	2									
99	63	3									
100	64	4									

- `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		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		63	3F	c
16	10	A	32	20	R	48	30	T	64	40	d
17	11	B	33	21	S	49	31	U	65	41	e
18	12	C	34	22	T	50	32	V	66	42	f
19	13	D	35	23	U	51	33	W	67	43	g
20	14	E	36	24	V	52	34	X	68	44	h
21	15	F	37	25	W	53	35	Y	69	45	i
22	16	G	38	26	X	54	36	Z	70	46	j
23	17	H	39	27	Y	55	37	[71	47	k
24	18	I	40	28	Z	56	38	\	72	48	l
25	19	J	41	29	{	57	39]	73	49	m
26	1A	K	42	2A	}	58	3A	^	74	4A	n
27	1B	L	43	2B	~	59	3B	_	75	4B	o
28	1C	M	44	2C		60	3C		76	4C	p
29	1D	N	45	2D		61	3D	a	77	4D	q
30	1E	O	46	2E		62	3E	b	78	4E	r
31	1F	P	47	2F		63	3F	c	79	4F	s
32	20	Q	48	30	T	64	40	d	80	50	t
33	21	R	49	31	U	65	41	e	81	51	u
34	22	S	50	32	V	66	42	f	82	52	v
35	23	T	51	33	W	67	43	g	83	53	w
36	24	U	52	34	X	68	44	h	84	54	x
37	25	V	53	35	Y	69	45	i	85	55	y
38	26	W	54	36	Z	70	46	j	86	56	
39	27	X	55	37	[71	47	k	87	57	
40	28	Y	56	38	\	72	48	l	88	58	
41	29	Z	57	39]	73	49	m	89	59	
42	2A	[58	3A	^	74	4A	n	90	5A	
43	2B	}	59	3B	_	75	4B	o	91	5B	
44	2C	~	60	3C		76	4C	p	92	5C	
45	2D		61	3D	a	77	4D	q	93	5D	
46	2E		62	3E	b	78	4E	r	94	5E	
47	2F		63	3F	c	79	4F	s	95	5F	
48	30	T	64	40	d	80	50	t	96	60	
49	31	U	65	41	e	81	51	u	97	61	
50	32	V	66	42	f	82	52	v	98	62	
51	33	W	67	43	g	83	53	w	99	63	
52	34	X	68	44	h	84	54	x	100	64	
53	35	Y	69	45	i	85	55	y	101	65	
54	36	Z	70	46	j	86	56		102	66	
55	37	[71	47	k	87	57		103	67	
56	38	\	72	48	l	88	58		104	68	
57	39]	73	49	m	89	59		105	69	
58	3A	^	74	4A	n	90	5A		106	6A	
59	3B	_	75	4B	o	91	5B		107	6B	
60	3C		76	4C	p	92	5C		108	6C	
61	3D	a	77	4D	q	93	5D		109	6D	
62	3E	b	78	4E	r	94	5E		110	6E	
63	3F	c	79	4F	s	95	5F		111	6F	
64	40	d	80	50	t	96	60		112	70	
65	41	e	81	51	u	97	61		113	71	
66	42	f	82	52	v	98	62		114	72	
67	43	g	83	53	w	99	63		115	73	
68	44	h	84	54	x	100	64		116	74	
69	45	i	85	55	y	101	65		117	75	
70	46	j	86	56		102	66		118	76	
71	47	k	87	57		103	67		119	77	
72	48	l	88	58		104	68		120	78	
73	49	m	89	59		105	69		121	79	
74	4A	n	90	5A		106	6A		122	7A	
75	4B	o	91	5B		107	6B		123	7B	
76	4C	p	92	5C		108	6C		124	7C	
77	4D	q	93	5D		109	6D		125	7D	
78	4E	r	94	5E		110	6E		126	7E	
79	4F	s	95	5F		111	6F		127	7F	
80	50	t	96	60		112	70				
81	51	u	97	61		113	71				
82	52	v	98	62		114	72				
83	53	w	99	63		115	73				
84	54	x	100	64		116	74				
85	55	y	101	65		117	75				
86	56		102	66		118	76				
87	57		103	67		119	77				
88	58		104	68		120	78				
89	59		105	69		121	79				
90	5A		106	6A		122	7A				
91	5B		107	6B		123	7B				
92	5C		108	6C		124	7C				
93	5D		109	6D		125	7D				
94	5E		110	6E		126	7E				
95	5F		111	6F		127	7F				
96	60		112	70							
97	61		113	71							
98	62		114	72							
99	63		115	73							
100	64		116	74							
101	65		117	75							
102	66		118	76							
103	67		119	77							
104	68		120	78							
105	69		121	79							
106	6A		122	7A							
107	6B		123	7B							
108	6C		124	7C							
109	6D		125	7D							
110	6E		126	7E							
111	6F		127	7F							
112	70										
113	71										
114	72										
115	73										
116	74										
117	75										
118	76										
119	77										
120	78										
121	79										
122	7A										
123	7B										
124	7C										
125	7D										
126	7E										
127	7F										

- `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		16	0	32	0	48	0
1		17	1	33	!	49	1
2		18	2	34	"	50	2
3		19	3	35	#	51	3
4		20	4	36	\$	52	4
5		21	5	37	%	53	5
6		22	6	38	&	54	6
7		23	7	39	'	55	7
8		24	8	40	(56	8
9		25	9	41)	57	9
10		26	A	42	*	58	*
11		27	B	43	+	59	+
12		28	C	44	,	60	,
13		29	D	45	-	61	-
14		30	E	46	.	62	.
15		31	F	47	/	63	/
16	0	32	0	64	@	80	@
17	1	33	!	65	A	81	A
18	2	34	"	66	B	82	B
19	3	35	#	67	C	83	C
20	4	36	\$	68	D	84	D
21	5	37	%	69	E	85	E
22	6	38	&	70	F	86	F
23	7	39	'	71	G	87	G
24	8	40	(72	H	88	H
25	9	41)	73	I	89	I
26	A	42	*	74	J	90	J
27	B	43	+	75	K	91	[
28	C	44	,	76	L	92	\
29	D	45	-	77	M	93]
30	E	46	.	78	N	94	^
31	F	47	/	79	O	95	_
32	0	48	0	80	P	96	`
33	1	49	1	81	Q	97	a
34	2	50	2	82	R	98	b
35	3	51	3	83	S	99	c
36	4	52	4	84	T	100	d
37	5	53	5	85	U	101	e
38	6	54	6	86	V	102	f
39	7	55	7	87	W	103	g
40	8	56	8	88	X	104	h
41	9	57	9	89	Y	105	i
42	A	58	*	90	Z	106	j
43	B	59	+	91	[107	k
44	C	60	,	92	\	108	l
45	D	61	-	93]	109	m
46	E	62	.	94	^	110	n
47	F	63	/	95	_	111	o
48	0	64	@	96	`	112	p
49	1	65	A	97	a	113	q
50	2	66	B	98	b	114	r
51	3	67	C	99	c	115	s
52	4	68	D	100	d	116	t
53	5	69	E	101	e	117	u
54	6	70	F	102	f	118	v
55	7	71	G	103	g	119	w
56	8	72	H	104	h	120	x
57	9	73	I	105	i	121	y
58	*	74	J	106	j	122	z
59	+	75	K	107	k	123	{
60	,	76	L	108	l	124	
61	-	77	M	109	m	125	}
62	.	78	N	110	n	126	~
63	/	79	O	111	o	127	

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.
- Example: `chr(97)` returns 'a'.
- What is `chr(33)`?

In Pairs or Triples...

Some review and some novel challenges:

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

Python Tutor

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

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~~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.

```
"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.

```
"FridaysSaturdaysSunday"  
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...