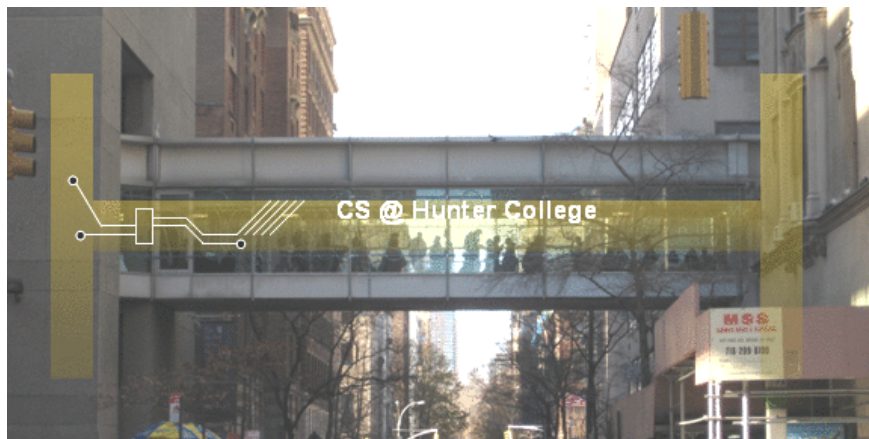


CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From email

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

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

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

Frequently Asked Questions

From email

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

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

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

Frequently Asked Questions

From email

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

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

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

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

Frequently Asked Questions

From email

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

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

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

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

IMPORTANT: Students who work on the due dates in this class tend to miss deadlines and fall behind. If, instead, you work on programs the week of the associated lecture, you will have time to ask for help if you get stuck and still make the deadline.

Frequently Asked Questions

From email

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

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

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

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

IMPORTANT: Students who work on the due dates in this class tend to miss deadlines and fall behind. If, instead, you work on programs the week of the associated lecture, you will have time to ask for help if you get stuck and still make the deadline.

- **When is the midterm?**

Frequently Asked Questions

From email

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

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

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

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

IMPORTANT: Students who work on the due dates in this class tend to miss deadlines and fall behind. If, instead, you work on programs the week of the associated lecture, you will have time to ask for help if you get stuck and still make the deadline.

- **When is the midterm?**

There is no midterm. Instead there's required weekly quizzes, code reviews and programming assignments.

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**
- Please, **write down the row and seat number** you are seating in and continue to seat there for the rest of the semester.

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**
- Please, **write down the row and seat number** you are seating in and continue to seat there for the rest of the semester.
- **Submit your row and seat number** using this link:
https://docs.google.com/spreadsheets/d/11enjiMGPGT1uLF7AG_r8dzYElqsMFSd81Y5mIdITQwg/edit?usp=sharing

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**
- Please, **write down the row and seat number** you are seating in and continue to seat there for the rest of the semester.
- **Submit your row and seat number** using this link:
https://docs.google.com/spreadsheets/d/11enjiMGPGT1uLF7AG_r8dzYElqsMFSd81Y5mIdITQwg/edit?usp=sharing
- The link to the form can also be found on Blackboard under Announcements.

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- Strings

Today's Topics



- **For-loops**
- `range()`
- Variables
- Characters
- Strings

Group Work

Some review and some novel challenges:

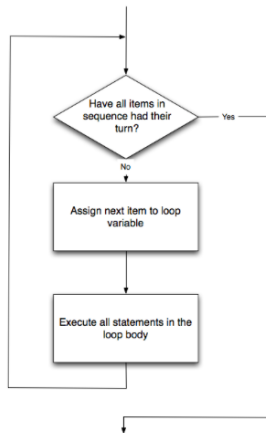
```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12   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

Python Tutor

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

(Demo with pythonTutor)

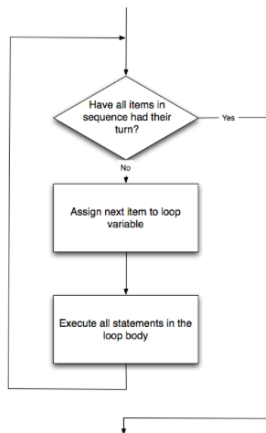
for-loop



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

How to Think Like CS, §4.5

for-loop



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()`.

Today's Topics



- For-loops
- **range()**
- Variables
- Characters
- Strings

More on range():

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(sum)
12
13 for c in "ABCD":
14     print(c)
```

Python Tutor

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(sum)
12
13 for c in "ABCD":
14     print(c)
```

(Demo with pythonTutor)

range()



Simplest version:

- `range(stop)`

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the 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)`

Today's Topics



- For-loops
- `range()`
- **Variables**
- Characters
- Strings

Variables

- A **variable** is a reserved memory location for storing a value.



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items



Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. `[3, 1, 4, 5, 9]` or
`['violet', 'purple', 'indigo']`

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. [3, 1, 4, 5, 9] or
['violet', 'purple', 'indigo']
 - ▶ **class variables**: for complex objects, like turtles.
- In Python (unlike other languages) you don't need to specify the type; it is deduced by its value.

Variable Names

- There's some rules about valid names for variables.



Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.

Variable Names



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

Variable Names



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

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.
- Can't use some words that Python has reserved for itself (e.g. `for`).
(List of reserved words in *Think CS*, §2.5.)

Today's Topics



- For-loops
- `range()`
- Variables
- **Characters**
- Strings

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

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	ACK	22	V	38	F	54	6
7	NAK	23	W	39	G	55	7
8	SYN	24	X	40	H	56	8
9	ETB	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	ST	30	^	46	N	62	>
15	DEL	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 a link to the ASCII table on the course webpage, under 'Useful Links'.)

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

ASCII TABLE

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	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		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		
63	:	79	O	111	o		
64	@	80	P	112	p		
65	A	81	Q	113	q		
66	B	82	R	114	r		
67	C	83	S	115	s		
68	D	84	T	116	t		
69	E	85	U	117	u		
70	F	86	V	118	v		
71	G	87	W	119	w		
72	H	88	X	120	x		
73	I	89	Y	121	y		
74	J	90	Z	122	z		
75	K	91	[
76	L	92]				
77	M	93	^				
78	N	94	_				
79	O	95	`				
80	P	96	{				
81	Q	97	a				
82	R	98	b				
83	S	99	c				
84	T	100	d				
85	U	101	e				
86	V	102	f				
87	W	103	g				
88	X	104	h				
89	Y	105	i				
90	Z	106	j				
91	[107	k				
92]	108	l				
93	^	109	m				
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						
108	l						
109	m						
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	~						

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

Converting from Character to Code:

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

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	SOH	17	11	Q	33	21	S	49	31	U
2	02	STX	18	12	R	34	22	T	50	32	V
3	03	ETX	19	13	S	35	23	U	51	33	W
4	04	END	20	14	T	36	24	V	52	34	X
5	05	SO	21	15	U	37	25	W	53	35	Y
6	06	SI	22	16	V	38	26	X	54	36	Z
7	07	BS	23	17	W	39	27	Y	55	37	[
8	08	HT	24	18	X	40	28	Z	56	38	\
9	09	LF	25	19	Y	41	29	[57	39]
10	0A	VT	26	1A	Z	42	2A	\	58	3A	^
11	0B	FF	27	1B	[43	2B]	59	3B	_
12	0C	DEL	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		32	20	R	48	30	T	64	40	d
17	11	P	33	21	S	49	31	U	65	41	e
18	12	Q	34	22	T	50	32	V	66	42	f
19	13	R	35	23	U	51	33	W	67	43	g
20	14	S	36	24	V	52	34	X	68	44	h
21	15	T	37	25	W	53	35	Y	69	45	i
22	16	U	38	26	X	54	36	Z	70	46	j
23	17	V	39	27	Y	55	37	[71	47	k
24	18	W	40	28	Z	56	38	\	72	48	l
25	19	X	41	29	[57	39]	73	49	m
26	1A	Z	42	2A	\	58	3A	^	74	4A	n
27	1B	[43	2B]	59	3B	_	75	4B	o
28	1C	\	44	2C	^	60	3C	`	76	4C	p
29	1D]	45	2D	_	61	3D	a	77	4D	q
30	1E	^	46	2E	`	62	3E	b	78	4E	r
31	1F	_	47	2F	a	63	3F	c	79	4F	s
32	20	R	48	30	T	64	40	d	80	50	t
33	21	S	49	31	U	65	41	e	81	51	u
34	22	T	50	32	V	66	42	f	82	52	v
35	23	U	51	33	W	67	43	g	83	53	w
36	24	V	52	34	X	68	44	h	84	54	x
37	25	W	53	35	Y	69	45	i	85	55	y
38	26	X	54	36	Z	70	46	j	86	56	
39	27	Y	55	37	[71	47	k	87	57	
40	28	Z	56	38	\	72	48	l	88	58	
41	29	[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	a	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.

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	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	BS	23	W	39	G	55	7
8	HT	24	X	40	H	56	8
9	LF	25	Y	41	I	57	9
10	VT	26	Z	42	J	58	:
11	FF	27	[43	K	59	;
12		28	\	44	L	60	<
13	CR	29]	45	M	61	=
14	SO	30	^	46	N	62	>
15	SI	31	_	47	O	63	?
16	P	32	@	48	0	64	SP
17	Q	33	A	49	1	65	a
18	R	34	B	50	2	66	b
19	S	35	C	51	3	67	c
20	T	36	D	52	4	68	d
21	U	37	E	53	5	69	e
22	V	38	F	54	6	70	f
23	W	39	G	55	7	71	g
24	X	40	H	56	8	72	h
25	Y	41	I	57	9	73	i
26	Z	42	J	58	:	74	j
27	[43	K	59	;	75	k
28	\	44	L	60	<	76	l
29]	45	M	61	=	77	m
30	^	46	N	62	>	78	n
31	_	47	O	63	?	79	o
32	@	48	0	64	SP	80	p
33	A	49	1	65	a	81	q
34	B	50	2	66	b	82	r
35	C	51	3	67	c	83	s
36	D	52	4	68	d	84	t
37	E	53	5	69	e	85	u
38	F	54	6	70	f	86	v
39	G	55	7	71	g	87	w
40	H	56	8	72	h	88	x
41	I	57	9	73	i	89	y
42	J	58	:	74	j	90	z
43	K	59	;	75	k	91	{
44	L	60	<	76	l	92	
45	M	61	=	77	m	93	}
46	N	62	>	78	n	94	~
47	O	63	?	79	o	95	
48	0	64	SP	80	p	96	
49	1	65	a	81	q	97	
50	2	66	b	82	r	98	
51	3	67	c	83	s	99	
52	4	68	d	84	t	100	
53	5	69	e	85	u	101	
54	6	70	f	86	v	102	
55	7	71	g	87	w	103	
56	8	72	h	88	x	104	
57	9	73	i	89	y	105	
58	:	74	j	90	z	106	
59	;	75	k	91	{	107	
60	<	76	l	92		108	
61	=	77	m	93	}	109	
62	>	78	n	94	~	110	
63	?	79	o	95		111	
64	SP	80	p	96		112	
65	a	81	q	97		113	
66	b	82	r	98		114	
67	c	83	s	99		115	
68	d	84	t	100		116	
69	e	85	u	101		117	
70	f	86	v	102		118	
71	g	87	w	103		119	
72	h	88	x	104		120	
73	i	89	y	105		121	
74	j	90	z	106		122	
75	k	91	{	107		123	
76	l	92		108		124	
77	m	93	}	109		125	
78	n	94	~	110		126	
79	o	95		111		127	DEL

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

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		128	80	DEL	129	81		130	82	
2	02		128	80	DEL	129	81		130	82	
3	03		128	80	DEL	129	81		130	82	
4	04		128	80	DEL	129	81		130	82	
5	05		128	80	DEL	129	81		130	82	
6	06		128	80	DEL	129	81		130	82	
7	07		128	80	DEL	129	81		130	82	
8	08		128	80	DEL	129	81		130	82	
9	09		128	80	DEL	129	81		130	82	
10	0A		128	80	DEL	129	81		130	82	
11	0B		128	80	DEL	129	81		130	82	
12	0C		128	80	DEL	129	81		130	82	
13	0D		128	80	DEL	129	81		130	82	
14	0E		128	80	DEL	129	81		130	82	
15	0F		128	80	DEL	129	81		130	82	
16	10		128	80	DEL	129	81		130	82	
17	11		128	80	DEL	129	81		130	82	
18	12		128	80	DEL	129	81		130	82	
19	13		128	80	DEL	129	81		130	82	
20	14		128	80	DEL	129	81		130	82	
21	15		128	80	DEL	129	81		130	82	
22	16		128	80	DEL	129	81		130	82	
23	17		128	80	DEL	129	81		130	82	
24	18		128	80	DEL	129	81		130	82	
25	19		128	80	DEL	129	81		130	82	
26	1A		128	80	DEL	129	81		130	82	
27	1B		128	80	DEL	129	81		130	82	
28	1C		128	80	DEL	129	81		130	82	
29	1D		128	80	DEL	129	81		130	82	
30	1E		128	80	DEL	129	81		130	82	
31	1F		128	80	DEL	129	81		130	82	
32	20	SP	128	80	DEL	129	81		130	82	
33	21	!	128	80	DEL	129	81		130	82	
34	22	"	128	80	DEL	129	81		130	82	
35	23	#	128	80	DEL	129	81		130	82	
36	24	\$	128	80	DEL	129	81		130	82	
37	25	%	128	80	DEL	129	81		130	82	
38	26	&	128	80	DEL	129	81		130	82	
39	27	'	128	80	DEL	129	81		130	82	
40	28	(128	80	DEL	129	81		130	82	
41	29)	128	80	DEL	129	81		130	82	
42	2A	*	128	80	DEL	129	81		130	82	
43	2B	+	128	80	DEL	129	81		130	82	
44	2C	,	128	80	DEL	129	81		130	82	
45	2D	-	128	80	DEL	129	81		130	82	
46	2E	.	128	80	DEL	129	81		130	82	
47	2F	/	128	80	DEL	129	81		130	82	
48	30	0	128	80	DEL	129	81		130	82	
49	31	1	128	80	DEL	129	81		130	82	
50	32	2	128	80	DEL	129	81		130	82	
51	33	3	128	80	DEL	129	81		130	82	
52	34	4	128	80	DEL	129	81		130	82	
53	35	5	128	80	DEL	129	81		130	82	
54	36	6	128	80	DEL	129	81		130	82	
55	37	7	128	80	DEL	129	81		130	82	
56	38	8	128	80	DEL	129	81		130	82	
57	39	9	128	80	DEL	129	81		130	82	
58	3A	:	128	80	DEL	129	81		130	82	
59	3B	;	128	80	DEL	129	81		130	82	
60	3C	<	128	80	DEL	129	81		130	82	
61	3D	=	128	80	DEL	129	81		130	82	
62	3E	>	128	80	DEL	129	81		130	82	
63	3F	?	128	80	DEL	129	81		130	82	
64	40	@	128	80	DEL	129	81		130	82	
65	41	A	128	80	DEL	129	81		130	82	
66	42	B	128	80	DEL	129	81		130	82	
67	43	C	128	80	DEL	129	81		130	82	
68	44	D	128	80	DEL	129	81		130	82	
69	45	E	128	80	DEL	129	81		130	82	
70	46	F	128	80	DEL	129	81		130	82	
71	47		128	80	DEL	129	81		130	82	
72	48		128	80	DEL	129	81		130	82	
73	49		128	80	DEL	129	81		130	82	
74	4A		128	80	DEL	129	81		130	82	
75	4B		128	80	DEL	129	81		130	82	
76	4C		128	80	DEL	129	81		130	82	
77	4D		128	80	DEL	129	81		130	82	
78	4E		128	80	DEL	129	81		130	82	
79	4F		128	80	DEL	129	81		130	82	
80	50		128	80	DEL	129	81		130	82	
81	51		128	80	DEL	129	81		130	82	
82	52		128	80	DEL	129	81		130	82	
83	53		128	80	DEL	129	81		130	82	
84	54		128	80	DEL	129	81		130	82	
85	55		128	80	DEL	129	81		130	82	
86	56		128	80	DEL	129	81		130	82	
87	57		128	80	DEL	129	81		130	82	
88	58		128	80	DEL	129	81		130	82	
89	59		128	80	DEL	129	81		130	82	
90	5A		128	80	DEL	129	81		130	82	
91	5B		128	80	DEL	129	81		130	82	
92	5C		128	80	DEL	129	81		130	82	
93	5D		128	80	DEL	129	81		130	82	
94	5E		128	80	DEL	129	81		130	82	
95	5F		128	80	DEL	129	81		130	82	
96	60		128	80	DEL	129	81		130	82	
97	61	a	128	80	DEL	129	81		130	82	
98	62	b	128	80	DEL	129	81		130	82	
99	63	c	128	80	DEL	129	81		130	82	
100	64	d	128	80	DEL	129	81		130	82	
101	65	e	128	80	DEL	129	81		130	82	
102	66	f	128	80	DEL	129	81		130	82	
103	67	g	128	80	DEL	129	81		130	82	
104	68	h	128	80	DEL	129	81		130	82	
105	69	i	128	80	DEL	129	81		130	82	
106	6A	j	128	80	DEL	129	81		130	82	
107	6B	k	128	80	DEL	129	81		130	82	
108	6C	l	128	80	DEL	129	81		130	82	
109	6D	m	128	80	DEL	129	81		130	82	
110	6E	n	128	80	DEL	129	81		130	82	
111	6F	o	128	80	DEL	129	81		130	82	
112	70	p	128	80	DEL	129	81		130	82	
113	71	q	128	80	DEL	129	81		130	82	
114	72	r	128	80	DEL	129	81		130	82	
115	73	s	128	80	DEL	129	81		130	82	
116	74	t	128	80	DEL	129	81		130	82	
117	75	u	128	80	DEL	129	81		130	82	
118	76	v	128	80	DEL	129	81		130	82	
119	77	w	128	80	DEL	129	81		130	82	
120	78	x	128	80	DEL	129	81		130	82	
121	79	y	128	80	DEL	129	81		130	82	
122	7A	z	128	80	DEL	129	81		130	82	
123	7B	[128	80	DEL	129	81		130	82	
124	7C	\	128	80	DEL	129	81		130	82	
125	7D]	128	80	DEL	129	81		130	82	
126	7E	~	128	80	DEL	129	81		130	82	
127	7F		128	80	DEL	129	81		130	82	

- `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

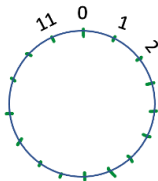
```
1 #Predict what will be printed:
2
3 for c in range(65,90):
4     print(chr(c))
5
6 message = "I love Python"
7 newMessage = ""
8 for c in message:
9     print(ord(c))    #Print the Unicode of each number
10    print(chr(ord(c)+1))    #Print the next character
11    newMessage = newMessage + chr(ord(c)+1) #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)

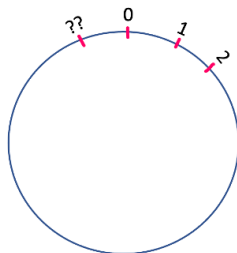
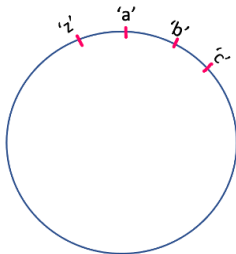
Wrap

Hints for Programming Assignment 9 in

<https://huntercsci127.github.io/f22/ps.html>. Given a string with only small letters, shift each letter by 2, get an encrypted message within the same alphabet. For example, original message is “abyz”, the encrypted message should be “cdab”.



What is 15:00?
What is 17:00?



- (1) How many scales in this shape?
- (2) How to map 'a' to 0, 'b' to 1, ...
- (3) How to map 0 back to 'a', 1 back to 'b', ...

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`.

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- **Strings**

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.
 - ▶ What would `print(s.count("sS"))` output?

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.
 - ▶ What would `print(s.count("sS"))` output?
 - ▶ What about:

```
mess = "10 20 21 9 101 35"  
mults = mess.count("0 ")  
print(mults)
```

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[7]  
days = s[7:15]  
days = s[:-1]
```

- Strings are made up of individual characters (letters, numbers, etc.)

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[7]  
days = s[7:15]  
days = s[:-1]
```

- 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[7]  
days = s[7:15]  
days = s[:-1]
```

- 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[7]  
days = s[7:15]  
days = s[:-1]
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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"
```

- 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)

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- Strings

Recap

- In Python, we introduced:

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

Recap

- 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

- 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

- 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

- 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

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

● In Python, we introduced:

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

Recap

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

● In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Practice Quiz & Final Questions



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

Practice Quiz & Final Questions



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

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North

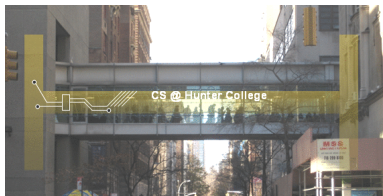
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (**programs 6-10**)

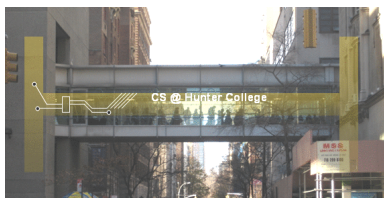
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (**programs 6-10**)
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5:30pm

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (**programs 6-10**)
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5:30pm
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)

Lecture Slips & Writing Boards



- Hand your lecture slip to a UTA.
- Return writing boards as you leave.