

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

- This lecture will be recorded

Frequently Asked Questions

From email

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope and thake the Lab Quiz, then submit this week's 5 programming assignments.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
You don't submit the lab, you read the lab.
When you are done, got to Gradescope and thake the Lab Quiz, then submit this week's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
You don't submit the lab, you read the lab.
When you are done, got to Gradescope and thake the Lab Quiz, then submit this week's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**
Lab Quiz (Gradescope) can be submitted only once.

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope and thake the Lab Quiz, then submit this week's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope and thake the Lab Quiz, then submit this week's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope and thake the Lab Quiz, then submit this week's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **Can I work ahead?**

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope and thake the Lab Quiz, then submit this week's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **Can I work ahead?**

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

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **Can I work ahead?**

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

- **When is the midterm?**

Frequently Asked Questions

From email

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

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

When you are done, go to Gradescope and take the Lab Quiz, then submit this week's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Lecture Previews (Blackboard) can be submitted multiple times

- **Can I work ahead?**

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

- **When is the midterm?**

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

Today's Topics



- **For-loops**
- `range()`
- Variables
- Characters
- Strings
- Guests: Internships, Advising & Clubs

In Pairs or Triples...

Some review and some novel challenges:

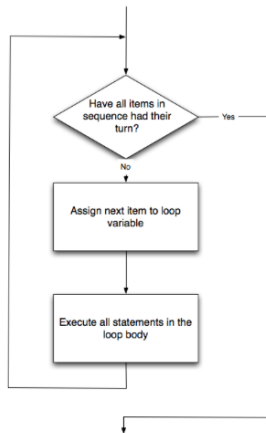
```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12   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

Python Tutor

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

(Demo with pythonTutor)

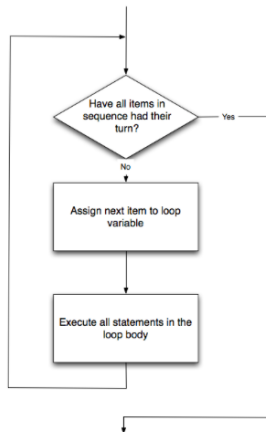
for-loop



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

How to Think Like CS, §4.5

for-loop



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
- Guests: Internships, Advising & Clubs

More on range():

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


Python Tutor

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

(Demo with pythonTutor)

range()



Simplest version:

- `range(stop)`

range()



Simplest version:

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

range()



Simplest version:

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

Variables

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



Variables

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



Variables

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



Variables

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



Variables

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



Variables



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

Variables



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

Variable Names

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



Variable Names



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

Variable Names



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

Variable Names



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

Variable Names



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

Today's Topics



- For-loops
- `range()`
- Variables
- **Characters**
- Strings
- Guests: Internships, Advising & Clubs

Standardized Code for Characters

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

Standardized Code for Characters

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

Standardized Code for Characters

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

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[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	0x10	32	@	48	0x30
1	!	17	0x11	33	A	49	0x31
2	"	18	0x12	34	B	50	0x32
3	#	19	0x13	35	C	51	0x33
4	\$	20	0x14	36	D	52	0x34
5	%	21	0x15	37	E	53	0x35
6	&	22	0x16	38	F	54	0x36
7	'	23	0x17	39	G	55	0x37
8	(24	0x18	40	H	56	0x38
9)	25	0x19	41	I	57	0x39
10	*	26	0x1A	42	J	58	0x3A
11	+	27	0x1B	43	K	59	0x3B
12	,	28	0x1C	44	L	60	0x3C
13	-	29	0x1D	45	M	61	0x3D
14	.	30	0x1E	46	N	62	0x3E
15	/	31	0x1F	47	O	63	0x3F
16		32	0x20	48	P	64	0x40
17	0	33	0x21	49	Q	65	0x41
18	1	34	0x22	50	R	66	0x42
19	2	35	0x23	51	S	67	0x43
20	3	36	0x24	52	T	68	0x44
21	4	37	0x25	53	U	69	0x45
22	5	38	0x26	54	V	70	0x46
23	6	39	0x27	55	W	71	0x47
24	7	40	0x28	56	X	72	0x48
25	8	41	0x29	57	Y	73	0x49
26	9	42	0x2A	58	Z	74	0x4A
27	:	43	0x2B	59	[75	0x4B
28	;	44	0x2C	60	\	76	0x4C
29	<	45	0x2D	61]	77	0x4D
30	=	46	0x2E	62	^	78	0x4E
31	>	47	0x2F	63	_	79	0x4F
32	?	48	0x30	64	`	80	0x50
33	@	49	0x31	65	a	81	0x51
34	A	50	0x32	66	b	82	0x52
35	B	51	0x33	67	c	83	0x53
36	C	52	0x34	68	d	84	0x54
37	D	53	0x35	69	e	85	0x55
38	E	54	0x36	70	f	86	0x56
39	F	55	0x37	71	g	87	0x57
40	G	56	0x38	72	h	88	0x58
41	H	57	0x39	73	i	89	0x59
42	I	58	0x3A	74	j	90	0x5A
43	J	59	0x3B	75	k	91	0x5B
44	K	60	0x3C	76	l	92	0x5C
45	L	61	0x3D	77	m	93	0x5D
46	M	62	0x3E	78	n	94	0x5E
47	N	63	0x3F	79	o	95	0x5F
48	O	64	0x40	80	p	96	0x60
49	P	65	0x41	81	q	97	0x61
50	Q	66	0x42	82	r	98	0x62
51	R	67	0x43	83	s	99	0x63
52	S	68	0x44	84	t	100	0x64
53	T	69	0x45	85	u	101	0x65
54	U	70	0x46	86	v	102	0x66
55	V	71	0x47	87	w	103	0x67
56	W	72	0x48	88	x	104	0x68
57	X	73	0x49	89	y	105	0x69
58	Y	74	0x4A	90	z	106	0x6A
59	Z	75	0x4B	91	{	107	0x6B
60	[76	0x4C	92		108	0x6C
61	\	77	0x4D	93	}	109	0x6D
62]	78	0x4E	94	~	110	0x6E
63	^	79	0x4F	95		111	0x6F
64	_	80	0x50	96		112	0x70
65	`	81	0x51	97		113	0x71
66	a	82	0x52	98		114	0x72
67	b	83	0x53	99		115	0x73
68	c	84	0x54	100		116	0x74
69	d	85	0x55	101		117	0x75
70	e	86	0x56	102		118	0x76
71	f	87	0x57	103		119	0x77
72	g	88	0x58	104		120	0x78
73	h	89	0x59	105		121	0x79
74	i	90	0x5A	106		122	0x7A
75	j	91	0x5B	107		123	0x7B
76	k	92	0x5C	108		124	0x7C
77	l	93	0x5D	109		125	0x7D
78	m	94	0x5E	110		126	0x7E
79	n	95	0x5F	111		127	0x7F
80	o	96	0x60	112		128	0x80
81	p	97	0x61	113		129	0x81
82	q	98	0x62	114		130	0x82
83	r	99	0x63	115		131	0x83
84	s	100	0x64	116		132	0x84
85	t	101	0x65	117		133	0x85
86	u	102	0x66	118		134	0x86
87	v	103	0x67	119		135	0x87
88	w	104	0x68	120		136	0x88
89	x	105	0x69	121		137	0x89
90	y	106	0x6A	122		138	0x8A
91	z	107	0x6B	123		139	0x8B
92	{	108	0x6C	124		140	0x8C
93		109	0x6D	125		141	0x8D
94	}	110	0x6E	126		142	0x8E
95	~	111	0x6F	127		143	0x8F
96		112	0x70	128		144	0x90
97		113	0x71	129		145	0x91
98		114	0x72	130		146	0x92
99		115	0x73	131		147	0x93
100		116	0x74	132		148	0x94
101		117	0x75	133		149	0x95
102		118	0x76	134		150	0x96
103		119	0x77	135		151	0x97
104		120	0x78	136		152	0x98
105		121	0x79	137		153	0x99
106		122	0x7A	138		154	0x9A
107		123	0x7B	139		155	0x9B
108		124	0x7C	140		156	0x9C
109		125	0x7D	141		157	0x9D
110		126	0x7E	142		158	0x9E
111		127	0x7F	143		159	0x9F
112		128	0x80	144		160	0xA0
113		129	0x81	145		161	0xA1
114		130	0x82	146		162	0xA2
115		131	0x83	147		163	0xA3
116		132	0x84	148		164	0xA4
117		133	0x85	149		165	0xA5
118		134	0x86	150		166	0xA6
119		135	0x87	151		167	0xA7
120		136	0x88	152		168	0xA8
121		137	0x89	153		169	0xA9
122		138	0x8A	154		170	0xAA
123		139	0x8B	155		171	0xAB
124		140	0x8C	156		172	0xAC
125		141	0x8D	157		173	0xAD
126		142	0x8E	158		174	0xAE
127		143	0x8F	159		175	0xAF
128		144	0x90	160		176	0xB0
129		145	0x91	161		177	0xB1
130		146	0x92	162		178	0xB2
131		147	0x93	163		179	0xB3
132		148	0x94	164		180	0xB4
133		149	0x95	165		181	0xB5
134		150	0x96	166		182	0xB6
135		151	0x97	167		183	0xB7
136		152	0x98	168		184	0xB8
137		153	0x99	169		185	0xB9
138		154	0x9A	170		186	0xBA
139		155	0x9B	171		187	0xBB
140		156	0x9C	172		188	0xBC
141		157	0x9D	173		189	0xBD
142		158	0x9E	174		190	0xBE
143		159	0x9F	175		191	0xBF
144		160	0xA0	176		192	0xC0
145		161	0xA1	177		193	0xC1
146		162	0xA2	178		194	0xC2
147		163	0xA3	179		195	0xC3
148		164	0xA4	180		196	0xC4
149		165	0xA5	181		197	0xC5
150		166	0xA6	182		198	0xC6
151		167	0xA7	183		199	0xC7
152		168	0xA8	184		200	0xC8
153		169	0xA9	185		201	0xC9
154		170	0xAA	186		202	0xCA
155		171	0xAB	187		203	0xCB
156		172	0xAC	188		204	0xCC
157		173	0xAD	189		205	0xCD
158		174	0xAE	190		206	0xCE
159		175	0xAF	191		207	0xCF
160		176	0xB0	192		208	0xD0
161		177	0xB1	193		209	0xD1
162		178	0xB2	194		210	0xD2
163		179	0xB3	195		211	0xD3
164		180	0xB4	196		212	0xD4
165		181	0xB5	197		213	0xD5
166		182	0xB6	198		214	0xD6
167		183	0xB7	199		215	0xD7
168		184	0xB8	200		216	0xD8
169		185	0xB9	201		217	0xD9
170		186	0xBA	202		218	0xDA
171		187	0xBB	203		219	0xDB
172		188	0xBC	204		220	0xDC
173		189	0xBD	205		221	0xDD
174		190	0xBE	206		222	0xDE
175		191	0xBF	207		223	0xDF
176		192	0xC0	208		224	0xE0
177		193	0xC1	209		225	0xE1
178		194	0xC2	210		226	0xE2
179		195	0xC3	211		227	0xE3
180		196	0xC4	212		228	0xE4
181		197	0xC5	213		229	0xE5
182		198	0xC6	214		230	0xE6
183		199	0xC7	215		231	0xE7
184		200	0xC8	216		232	0xE8
185		201	0xC9	217		233	0xE9
186		202	0xCA	218		234	0xEA
187		203	0xCB	219		235	0xEB
188		204	0xCC	220		236	0xEC
189		205	0xCD	221		237	0xED
190		206	0xCE	222		238	0xEE
191		207	0xCF	223		239	0xEF
192		208	0xD0	224		240	0xF0
193		209	0xD1	225		241	0xF1
194		210	0xD2	226		242	0xF2
195		211	0xD3	227		243	0xF3
196		212	0xD4	228		244	0xF4
197		213	0xD5	229		245	0xF5
198		214	0xD6	230		246	0xF6
199		215	0xD7	231		247	0xF7
200		216	0xD8	232		248	0xF8
201		217					

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	
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	
64	@	80	P	112	p	128	
65	A	81	Q	113	q	129	
66	B	82	R	114	r	130	
67	C	83	S	115	s	131	
68	D	84	T	116	t	132	
69	E	85	U	117	u	133	
70	F	86	V	118	v	134	
71	G	87	W	119	w	135	
72	H	88	X	120	x	136	
73	I	89	Y	121	y	137	
74	J	90	Z	122	z	138	
75	K	91	[123	{	139	
76	L	92	\	124		140	
77	M	93]	125	}	141	
78	N	94	^	126	~	142	
79	O	95	_	127		143	
80	P	96	`	128		144	
81	Q	97	a	129		145	
82	R	98	b	130		146	
83	S	99	c	131		147	
84	T	100	d	132		148	
85	U	101	e	133		149	
86	V	102	f	134		150	
87	W	103	g	135		151	
88	X	104	h	136		152	
89	Y	105	i	137		153	
90	Z	106	j	138		154	
91	[107	k	139		155	
92	\	108	l	140		156	
93]	109	m	141		157	
94	^	110	n	142		158	
95	_	111	o	143		159	
96	`	112	p	144		160	
97	a	113	q	145		161	
98	b	114	r	146		162	
99	c	115	s	147		163	
100	d	116	t	148		164	
101	e	117	u	149		165	
102	f	118	v	150		166	
103	g	119	w	151		167	
104	h	120	x	152		168	
105	i	121	y	153		169	
106	j	122	z	154		170	
107	k	123	{	155		171	
108	l	124		156		172	
109	m	125	}	157		173	
110	n	126	~	158		174	
111	o	127		159		175	
112	p	128		160		176	
113	q	129		161		177	
114	r	130		162		178	
115	s	131		163		179	
116	t	132		164		180	
117	u	133		165		181	
118	v	134		166		182	
119	w	135		167		183	
120	x	136		168		184	
121	y	137		169		185	
122	z	138		170		186	
123	{	139		171		187	
124		140		172		188	
125	}	141		173		189	
126	~	142		174		190	
127		143		175		191	
128		144		176		192	
129		145		177		193	
130		146		178		194	
131		147		179		195	
132		148		180		196	
133		149		181		197	
134		150		182		198	
135		151		183		199	
136		152		184		200	
137		153		185		201	
138		154		186		202	
139		155		187		203	
140		156		188		204	
141		157		189		205	
142		158		190		206	
143		159		191		207	
144		160		192		208	
145		161		193		209	
146		162		194		210	
147		163		195		211	
148		164		196		212	
149		165		197		213	
150		166		198		214	
151		167		199		215	
152		168		200		216	
153		169		201		217	
154		170		202		218	
155		171		203		219	
156		172		204		220	
157		173		205		221	
158		174		206		222	
159		175		207		223	
160		176		208		224	
161		177		209		225	
162		178		210		226	
163		179		211		227	
164		180		212		228	
165		181		213		229	
166		182		214		230	
167		183		215		231	
168		184		216		232	
169		185		217		233	
170		186		218		234	
171		187		219		235	
172		188		220		236	
173		189		221		237	
174		190		222		238	
175		191		223		239	
176		192		224		240	
177		193		225		241	
178		194		226		242	
179		195		227		243	
180		196		228		244	
181		197		229		245	
182		198		230		246	
183		199		231		247	
184		200		232		248	
185		201		233		249	
186		202		234		250	
187		203		235		251	
188		204		236		252	
189		205		237		253	
190		206		238		254	
191		207		239		255	
192		208		240		256	
193		209		241		257	
194		210		242		258	
195		211		243		259	
196		212		244		260	
197		213		245		261	
198		214		246		262	
199		215		247		263	
200		216		248		264	
201		217		249		265	
202		218		250		266	
203		219		251		267	
204		220		252		268	
205		221		253		269	
206		222		254		270	
207		223		255		271	
208		224		256		272	
209		225		257		273	
210		226		258		274	
211		227		259		275	
212		228		260		276	
213		229		261		277	
214		230		262		278	
215		231		263		279	
216		232		264		280	
217		233		265		281	
218		234		266		282	
219		235		267		283	
220		236		268		284	
221		237		269		285	
222		238		270		286	
223		239		271		287	
224		240		272		288	
225		241		273		289	
226		242		274		290	
227		243		275		291	
228		244		276		292	
229		245		277		293	
230		246		2			

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

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

- `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	ST	22	V	38	F	54	6
7	ACK	23	W	39	G	55	7
8	BS	24	X	40	H	56	8
9	HT	25	Y	41	I	57	9
10	LF	26	Z	42	J	58	:
11	VT	27	[43	K	59	;
12	FF	28	\	44	L	60	<
13	CR	29]	45	M	61	=
14	DEL	30	^	46	N	62	>
15		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	A	81	Q	97	a
50		66	B	82	R	98	b
51		67	C	83	S	99	c
52		68	D	84	T	100	d
53		69	E	85	U	101	e
54		70	F	86	V	102	f
55		71	G	87	W	103	g
56		72	H	88	X	104	h
57		73	I	89	Y	105	i
58		74	J	90	Z	106	j
59		75	K	91	[107	k
60		76	L	92	\	108	l
61		77	M	93]	109	m
62		78	N	94	^	110	n
63		79	O	95	_	111	o
64		80	P	96	0	112	p
65		81	Q	97	a	113	q
66		82	R	98	b	114	r
67		83	S	99	c	115	s
68		84	T	100	d	116	t
69		85	U	101	e	117	u
70		86	V	102	f	118	v
71		87	W	103	g	119	w
72		88	X	104	h	120	x
73		89	Y	105	i	121	y
74		90	Z	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

- `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		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	HT	23	W	39	G	55	7
8	LF	24	X	40	H	56	8
9	VT	25	Y	41	I	57	9
10	FF	26	Z	42	J	58	:
11		27	[43	K	59	;
12		28	\	44	L	60	<
13	CR	29]	45	M	61	=
14		30	^	46	N	62	>
15		31	_	47	O	63	?

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

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

offset: how many letters past 'a'?

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

User Input

Covered in detail in Lab 2:

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

Lecture Quiz

- Log-in to Gradescope
- Find LECTURE 2 Quiz
- Take the quiz
- **You have 3 minutes**

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- **Strings**
- Guests: Internships, Advising & Clubs

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

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

More on Strings: Indexing & Substrings

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

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

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is 'F'.

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is `'r'`.

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is 's'.

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is 'day'.

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is 'Fri'.

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is

More on Strings: Indexing & Substrings

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

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is 'FridaysSaturdaysSunday'.
(no trailing 's' at the end)

More on Strings: Splits

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

- `split()` divides a string into a list.

More on Strings: Splits

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

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

More on Strings: Splits

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

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"

More on Strings: Splits

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

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

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

More on Strings: Splits

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

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"  
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")  
"FridaysSaturdaysSunday"
```

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")  
"FridaysSaturdaysSunday"  
days = ['Fri', 'sSatur', 'sSun']
```

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- Strings
- **Guests: Internships, Advising & Clubs**

Guest Speakers

- Announcement on Blackboard:
 - ▶ Advising
 - ▶ Programs and Clubs Handout
 - ▶ Internships Handout
 - ▶ Hunter CS Handbook
 - ▶ PreTech Center (formerly CUNY2X) Newsletter

Recap

- In Python, we introduced:

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

Recap

- 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

- 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

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Practice Quiz & Final Questions



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

Practice Quiz & Final Questions



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

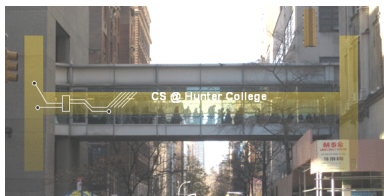
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

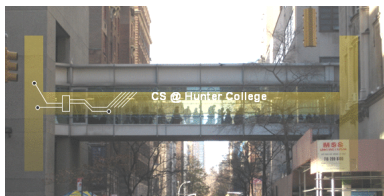
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Synchronous Meetings)

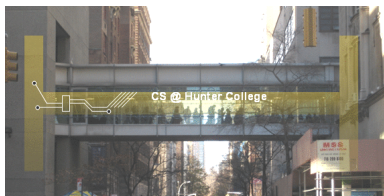
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Synchronous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Synchronous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 6-10)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Synchronous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 6-10)
- At any point, visit our [Drop-In Tutoring 11am-5pm](#) for help!!!

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend a Lab Review (Zoom links on Blackboard / Synchronous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 6-10)
- At any point, visit our [Drop-In Tutoring 11am-5pm](#) for help!!!
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)