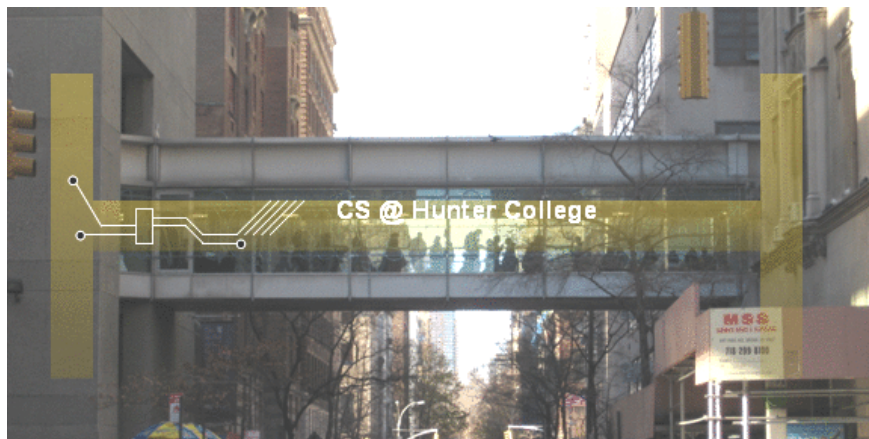


CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

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

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
*No. Missing lecture slip & quiz grades are replaced by your final exam score.
If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.*

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
*No. Missing lecture slip & quiz grades are replaced by your final exam score.
If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.*
- Why do I have to work in groups?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture slip & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Why do I have to work in groups?
It's great practice to explain technical work to others.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
*No. Missing lecture slip & quiz grades are replaced by your final exam score.
If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.*
- Why do I have to work in groups?
It's great practice to explain technical work to others.
- Can I work ahead?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture slip & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Why do I have to work in groups?
It's great practice to explain technical work to others.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture slip & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Why do I have to work in groups?
It's great practice to explain technical work to others.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- You said “when you take second semester...” I just took this class for Pathways...

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture slip & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Why do I have to work in groups?
It's great practice to explain technical work to others.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- You said “when you take second semester...” I just took this class for Pathways...
This is Pathways, but we hope that you will be a CS major/minor.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
Tuesday, 22 May, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture slip & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Why do I have to work in groups?
It's great practice to explain technical work to others.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- You said “when you take second semester...” I just took this class for Pathways...
This is Pathways, but we hope that you will be a CS major/minor. We also hope: “Get your education don't forget whence you came...”

Today's Topics



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

In Pairs or Triples...

Some review and some novel challenges:

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

Python Tutor

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

(Demo with pythonTutor)

Variables

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



Variables



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

Variables



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

Variables



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

Variables



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

Variables



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

Variables



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

Variable Names

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



Variable Names



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

Variable Names



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

Variable Names



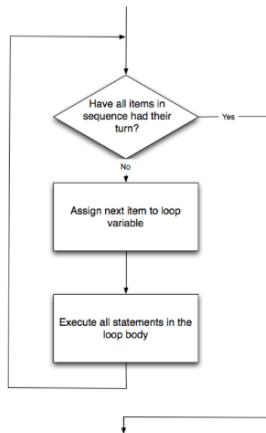
- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

Variable Names



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

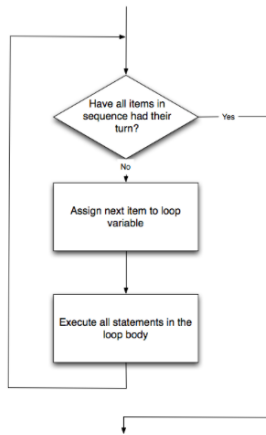
for-loop



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

How to Think Like CS, §4.5

for-loop



How to Think Like CS, §4.5

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

where `list` is a list of items:

- stated explicitly (e.g. `[1,2,3]`) or
- generated by a function, e.g. `range()`.

In Pairs or Triples...

Some review and some novel challenges:

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

Python Tutor

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

(Demo with pythonTutor)

range()

Simplest version:



range()

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()



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(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 start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start,start+1,...,stop-1]`
- For example, if you want the the list
`[10,11,...,20]`
you would write:

```
range(10,21)
```

`range()`

What if you wanted to count by twos, or some other number:



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
[start, start+step, start+2*step..., last]
(where last is the largest start+k*step less than stop)



range()



What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest `start+k*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 `start+k*step` less than stop)
- For example, if you want the the list `[5,10,...,50]` you would write:

```
range(5, 51, 5)
```

range()



What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest $start+k*step$ less than stop)
- For example, if you want the 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 $start+k*step$ less than stop)
- For example, if you want the the list `[5,10,...,50]` you would write:

```
range(5, 51, 5)
```


In summary: `range()`



The three versions:

In summary: `range()`



The three versions:

- `range(stop)`

In summary: `range()`



The three versions:

- `range(stop)`
- `range(start, stop)`

In summary: `range()`



The three versions:

- `range(stop)`
- `range(start, stop)`
- `range(start, stop, step)`

Standardized Code for Characters

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

Standardized Code for Characters

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

Standardized Code for Characters

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

ASCII TABLE

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Character	Decimal	Hex Char	Character	Decimal	Hex Char	Character	Decimal	Hex Char	Character
0	00		10	0A	LF	20	14	SP	30	1E	0
1	01	SOH	11	0B	VT	21	15	!	31	1F	1
2	02	STX	12	0C	FF	22	16	"	32	20	2
3	03	ETX	13	0D	CR	23	17	#	33	21	3
4	04	END	14	0E	SH	24	18	\$	34	22	4
5	05	SO	15	0F	SI	25	19	%	35	23	5
6	06	ACK	16	10	DL	26	1A	&	36	24	6
7	07	BEL	17	11	EQ	27	1B	'	37	25	7
8	08	BS	18	12	LR	28	1C	(38	26	8
9	09	HT	19	13	LS	29	1D)	39	27	9
10	0A	LF	20	14	SP	30	1E	0	40	28	[
11	0B	VT	21	15	!	31	1F	1	41	29	\
12	0C	FF	22	16	"	32	20	2	42	2A]
13	0D	CR	23	17	#	33	21	3	43	2B	^
14	0E	SH	24	18	\$	34	22	4	44	2C	_
15	0F	SI	25	19	%	35	23	5	45	2D	`
16	10	DL	26	1A	&	36	24	6	46	2E	a
17	11	EQ	27	1B	'	37	25	7	47	2F	b
18	12	LR	28	1C	(38	26	8	48	30	c
19	13	LS	29	1D)	39	27	9	49	31	d
20	14	SP	30	1E	0	40	28	[50	32	0
21	15	!	31	1F	1	41	29	\	51	33	1
22	16	"	32	20	2	42	2A]	52	34	2
23	17	#	33	21	3	43	2B	^	53	35	3
24	18	\$	34	22	4	44	2C	_	54	36	4
25	19	%	35	23	5	45	2D	`	55	37	5
26	1A	&	36	24	6	46	2E	a	56	38	6
27	1B	'	37	25	7	47	2F	b	57	39	7
28	1C	(38	26	8	48	30	c	58	3A	8
29	1D)	39	27	9	49	31	d	59	3B	9
30	1E	0	40	28	[50	32	0	60	3C	<
31	1F	1	41	29	\	51	33	1	61	3D	>
32	20	2	42	2A]	52	2B	2	62	3E	@
33	21	3	43	2B	^	53	2C	3	63	3F	A
34	22	4	44	2C	_	54	2D	4	64	40	B
35	23	5	45	2D	`	55	2E	5	65	41	C
36	24	6	46	2E	a	56	2F	6	66	42	D
37	25	7	47	2F	b	57	30	7	67	43	E
38	26	8	48	30	c	58	31	8	68	44	F
39	27	9	49	31	d	59	32	9	69	45	G
40	28	[50	32	0	60	33	0	70	46	H
41	29	\	51	33	1	61	34	1	71	47	I
42	2A]	52	2B	2	62	35	2	72	48	J
43	2B	^	53	2C	3	63	36	3	73	49	K
44	2C	_	54	2D	4	64	37	4	74	4A	L
45	2D	`	55	2E	5	65	38	5	75	4B	M
46	2E	a	56	2F	6	66	39	6	76	4C	N
47	2F	b	57	30	7	67	3A	7	77	4D	O
48	30	c	58	31	8	68	3B	8	78	4E	P
49	31	d	59	32	9	69	3C	9	79	4F	Q
50	32	0	60	33	0	70	3D	0	80	50	R
51	33	1	61	34	1	71	3E	1	81	51	S
52	34	2	62	35	2	72	3F	2	82	52	T
53	35	3	63	36	3	73	40	3	83	53	U
54	36	4	64	37	4	74	41	4	84	54	V
55	37	5	65	38	5	75	42	5	85	55	W
56	38	6	66	39	6	76	43	6	86	56	X
57	39	7	67	3A	7	77	44	7	87	57	Y
58	3A	8	68	3B	8	78	45	8	88	58	Z
59	3B	9	69	3C	9	79	46	9	89	59	[
60	3C	<	60	3D	>	80	47	0	90	5A	\
61	3D	>	61	3E	@	81	48	1	91	5B	_
62	3E	@	62	3F	A	82	49	2	92	5C	`
63	3F	A	63	40	B	83	4A	3	93	5D	a
64	40	B	64	41	C	84	4B	4	94	5E	b
65	41	C	65	42	D	85	4C	5	95	5F	c
66	42	D	66	43	E	86	4D	6	96	60	d
67	43	E	67	44	F	87	4E	7	97	61	e
68	44	F	68	45	G	88	4F	8	98	62	f
69	45	G	69	46	H	89	50	9	99	63	g
70	46	H	70	47	I	90	51	0	100	64	h
71	47	I	71	48	J	91	52	1	101	65	i
72	48	J	72	49	K	92	53	2	102	66	j
73	49	K	73	4A	L	93	54	3	103	67	k
74	4A	L	74	4B	M	94	55	4	104	68	l
75	4B	M	75	4C	N	95	56	5	105	69	m
76	4C	N	76	4D	O	96	57	6	106	6A	n
77	4D	O	77	4E	P	97	58	7	107	6B	o
78	4E	P	78	4F	Q	98	59	8	108	6C	p
79	4F	Q	79	50	R	99	5A	9	109	6D	q
80	50	R	80	51	S	100	5B	0	110	6E	r
81	51	S	81	52	T	101	5C	1	111	6F	s
82	52	T	82	53	U	102	5D	2	112	70	t
83	53	U	83	54	V	103	5E	3	113	71	u
84	54	V	84	55	W	104	5F	4	114	72	v
85	55	W	85	56	X	105	60	5	115	73	w
86	56	X	86	57	Y	106	61	6	116	74	x
87	57	Y	87	58	Z	107	62	7	117	75	y
88	58	Z	88	59	[108	63	8	118	76	z
89	59	[89	5A	\	109	64	9	119	77	{
90	5A	\	90	5B	_	110	65	0	120	78	
91	5B	_	91	5C	`	111	66	1	121	79	}
92	5C	`	92	5D	a	112	67	2	122	7A	~
93	5D	a	93	5E	b	113	68	3	123	7B	
94	5E	b	94	5F	c	114	69	4	124	7C	
95	5F	c	95	60	d	115	6A	5	125	7D	
96	60	d	96	61	e	116	6B	6	126	7E	
97	61	e	97	62	f	117	6C	7	127	7F	
98	62	f	98	63	g	118	6D	8	128	80	
99	63	g	99	64	h	119	6E	9	129	81	
100	64	h	100	65	i	120	6F	0	130	82	
101	65	i	101	66	j	121	70	1	131	83	
102	66	j	102	67	k	122	71	2	132	84	
103	67	k	103	68	l	123	72	3	133	85	
104	68	l	104	69	m	124	73	4	134	86	
105	69	m	105	6A	n	125	74	5	135	87	
106	6A	n	106	6B	o	126	75	6	136	88	
107	6B	o	107	6C	p	127	76	7	137	89	
108	6C	p	108	6D	q	128	77	8	138	8A	
109	6D	q	109	6E	r	129	78	9	139	8B	
110	6E	r	110	6F	s	130	79	0	140	8C	
111	6F	s	111	70	t	131	7A	1	141	8D	
112	70	t	112	71	u	132	7B	2	142	8E	
113	71	u	113	72	v	133	7C	3	143	8F	
114	72	v	114	73	w	134	7D	4	144	90	
115	73	w	115	74	x	135	7E	5	145	91	
116	74	x	116	75	y	136	7F	6	146	92	
117	75	y	117	76	z	137	80	7	147	93	
118	76	z	118	77	{	138	81	8	148	94	
119	77	{	119	78		139	82	9	149	95	
120	78		120	79	}	140	83	0	150	96	
121	79	}	121	7A	~	141	84	1	151	97	
122	7A	~	122	7B		142	85	2	152	98	
123	7B		123	7C		143	86	3	153	99	
124	7C		124	7D		144	87	4	154	9A	
125	7D		125	7E		145	88	5	155	9B	
126	7E		126	7F		146	89	6	156	9C	
127	7F		127	80		147	8A	7	157	9D	
128	80		128	81		148	8B	8	158	9E	
129	81		129	82		149	8C	9	159	9F	
130	82		130	83		150	8D	0	160	A0	
131	83		131	84		151	8E	1	161	A1	
132	84		132	85		152	8F	2	162	A2	
133	85		133	86		153	90	3	163	A3	
134	86		134	87		154	91	4	164	A4	
135	87		135	88		155	92	5	165	A5	
136	88		136	89		156	93	6	166	A6	
137	89		137	8A		157	94	7	167	A7	
138	8A		138	8B		158	95	8	168	A8	
139	8B		139	8C		159	96	9	169	A9	
140	8C		140	8D		160	97	0	170	AA	
141	8D		141	8E		161	98	1	171	AB	
142	8E		142	8F		162	99	2	172	AC	
143	8F		143	90		163	9A				

Converting from Character to Code:

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

ASCII TABLE

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

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

Converting from Character to Code:

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

ASCII TABLE

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

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

Converting from Character to Code:

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

ASCII TABLE



Decimal	Hex	Octal	Char	Decimal	Hex	Octal	Char	Decimal	Hex	Octal	Char
0	00	00	NUL	128	80	170	SOH	129	81	171	STX
1	01	01	SOH	130	82	172	STX	131	83	173	ETX
2	02	02	STX	132	84	174	ETX	133	85	175	END
3	03	03	ETX	134	86	176	END	135	87	177	START
4	04	04	END	136	88	178	START	137	89	179	ONE
5	05	05	START	138	8A	180	ONE	139	8B	181	TWO
6	06	06	ONE	140	8C	182	TWO	141	8D	183	THREE
7	07	07	TWO	142	8E	184	THREE	143	8F	185	FOUR
8	08	10	THREE	144	90	186	FOUR	145	91	187	FIVE
9	09	11	FOUR	146	92	188	FIVE	147	93	189	SIX
10	0A	12	FIVE	148	94	190	SIX	149	95	191	SEVEN
11	0B	13	SIX	150	96	192	SEVEN	151	97	193	EIGHT
12	0C	14	SEVEN	152	98	194	EIGHT	153	99	195	NINE
13	0D	15	EIGHT	154	9A	196	NINE	155	9B	197	TEN
14	0E	16	NINE	156	9C	198	TEN	157	9D	199	SPACE
15	0F	17	SPACE	158	9E	200	SPACE	159	9F	201	!
16	10	20	!	160	A0	240	!	161	A1	241	@"
17	11	21	@"	162	A2	242	@"	163	A3	243	#"
18	12	22	#"	164	A4	244	#"	165	A5	245	\$"
19	13	23	\$"	166	A6	246	\$"	167	A7	247	%"
20	14	24	%"	168	A8	248	%"	169	A9	249	&"
21	15	25	&"	170	AA	250	&"	171	AB	251	'"
22	16	26	'"	172	AC	252	'"	173	AD	253	(",
23	17	27	(,)	174	AE	254	(,)	175	AF	255	.)
24	18	30	.)	176	B0	270	.)	177	B1	271	/*
25	19	31	/*	178	B2	272	/*	179	B3	273	+
26	1A	32	+	180	B4	274	+	181	B5	275	,
27	1B	33	,	182	B6	276	,	183	B7	277	-
28	1C	34	-	184	B8	278	-	185	B9	279	.
29	1D	35	.	186	BA	280	.	187	BB	281	:"
30	1E	36	:"	188	BC	282	:"	189	BD	283	;
31	1F	37	;	190	BE	284	;	191	BF	285	:"
32	20	40	:"	192	C0	300	:"	193	C1	301	["
33	21	41	["	194	C2	302	["	195	C3	303]
34	22	42]	196	C4	304]	197	C5	305	^"
35	23	43	^"	198	C6	306	^"	199	C7	307	_"
36	24	44	_"	200	C8	308	_"	201	C9	309	~"
37	25	45	~"	202	CA	310	~"	203	CB	311	DEL
38	26	46	DEL	204	CC	312	DEL	205	CD	313	DEL
39	27	47	DEL	206	CE	314	DEL	207	CF	315	DEL
40	28	50	DEL	208	CD	316	DEL	209	CE	317	DEL
41	29	51	DEL	210	CE	318	DEL	211	CF	319	DEL
42	2A	52	DEL	212	CF	320	DEL	213	CF	321	DEL
43	2B	53	DEL	214	CF	322	DEL	215	CF	323	DEL
44	2C	54	DEL	216	CF	324	DEL	217	CF	325	DEL
45	2D	55	DEL	218	CF	326	DEL	219	CF	327	DEL
46	2E	56	DEL	220	CF	328	DEL	221	CF	329	DEL
47	2F	57	DEL	222	CF	330	DEL	223	CF	331	DEL
48	30	60	DEL	224	CF	332	DEL	225	CF	333	DEL
49	31	61	DEL	226	CF	334	DEL	227	CF	335	DEL
50	32	62	DEL	228	CF	336	DEL	229	CF	337	DEL
51	33	63	DEL	230	CF	338	DEL	231	CF	339	DEL
52	34	64	DEL	232	CF	340	DEL	233	CF	341	DEL
53	35	65	DEL	234	CF	342	DEL	235	CF	343	DEL
54	36	66	DEL	236	CF	344	DEL	237	CF	345	DEL
55	37	67	DEL	238	CF	346	DEL	239	CF	347	DEL
56	38	68	DEL	240	CF	348	DEL	241	CF	349	DEL
57	39	69	DEL	242	CF	350	DEL	243	CF	351	DEL
58	3A	70	DEL	244	CF	352	DEL	245	CF	353	DEL
59	3B	71	DEL	246	CF	354	DEL	247	CF	355	DEL
60	3C	72	DEL	248	CF	356	DEL	249	CF	357	DEL
61	3D	73	DEL	250	CF	358	DEL	251	CF	359	DEL
62	3E	74	DEL	252	CF	360	DEL	253	CF	361	DEL
63	3F	75	DEL	254	CF	362	DEL	255	CF	363	DEL
64	40	76	DEL	256	CF	364	DEL	257	CF	365	DEL
65	41	77	DEL	258	CF	366	DEL	259	CF	367	DEL
66	42	78	DEL	260	CF	368	DEL	261	CF	369	DEL
67	43	79	DEL	262	CF	370	DEL	263	CF	371	DEL
68	44	80	DEL	264	CF	372	DEL	265	CF	373	DEL
69	45	81	DEL	266	CF	374	DEL	267	CF	375	DEL
70	46	82	DEL	268	CF	376	DEL	269	CF	377	DEL
71	47	83	DEL	270	CF	378	DEL	271	CF	379	DEL
72	48	84	DEL	272	CF	380	DEL	273	CF	381	DEL
73	49	85	DEL	274	CF	382	DEL	275	CF	383	DEL
74	4A	86	DEL	276	CF	384	DEL	277	CF	385	DEL
75	4B	87	DEL	278	CF	386	DEL	279	CF	387	DEL
76	4C	88	DEL	280	CF	388	DEL	281	CF	389	DEL
77	4D	89	DEL	282	CF	390	DEL	283	CF	391	DEL
78	4E	90	DEL	284	CF	392	DEL	285	CF	393	DEL
79	4F	91	DEL	286	CF	394	DEL	287	CF	395	DEL
80	50	92	DEL	288	CF	396	DEL	289	CF	397	DEL
81	51	93	DEL	290	CF	398	DEL	291	CF	399	DEL
82	52	94	DEL	292	CF	400	DEL	293	CF	401	DEL
83	53	95	DEL	294	CF	402	DEL	295	CF	403	DEL
84	54	96	DEL	296	CF	404	DEL	297	CF	405	DEL
85	55	97	DEL	298	CF	406	DEL	299	CF	407	DEL
86	56	98	DEL	300	CF	408	DEL	301	CF	409	DEL
87	57	99	DEL	302	CF	410	DEL	303	CF	411	DEL
88	58	100	DEL	304	CF	412	DEL	305	CF	413	DEL
89	59	101	DEL	306	CF	414	DEL	307	CF	415	DEL
90	5A	102	DEL	308	CF	416	DEL	309	CF	417	DEL
91	5B	103	DEL	310	CF	418	DEL	311	CF	419	DEL
92	5C	104	DEL	312	CF	420	DEL	313	CF	421	DEL
93	5D	105	DEL	314	CF	422	DEL	315	CF	423	DEL
94	5E	106	DEL	316	CF	424	DEL	317	CF	425	DEL
95	5F	107	DEL	318	CF	426	DEL	319	CF	427	DEL
96	60	110	DEL	320	CF	428	DEL	321	CF	429	DEL
97	61	111	DEL	322	CF	430	DEL	323	CF	431	DEL
98	62	112	DEL	324	CF	432	DEL	325	CF	433	DEL
99	63	113	DEL	326	CF	434	DEL	327	CF	435	DEL
100	64	114	DEL	328	CF	436	DEL	329	CF	437	DEL
101	65	115	DEL	330	CF	438	DEL	331	CF	439	DEL
102	66	116	DEL	332	CF	440	DEL	333	CF	441	DEL
103	67	117	DEL	334	CF	442	DEL	335	CF	443	DEL
104	68	120	DEL	336	CF	444	DEL	337	CF	445	DEL
105	69	121	DEL	338	CF	446	DEL	339	CF	447	DEL
106	6A	122	DEL	340	CF	448	DEL	341	CF	449	DEL
107	6B	123	DEL	342	CF	450	DEL	343	CF	451	DEL
108	6C	124	DEL	344	CF	452	DEL	345	CF	453	DEL
109	6D	125	DEL	346	CF	454	DEL	347	CF	455	DEL
110	6E	130	DEL	348	CF	456	DEL	349	CF	457	DEL
111	6F	131	DEL	350	CF	458	DEL	351	CF	459	DEL
112	70	132	DEL	352	CF	460	DEL	353	CF	461	DEL
113	71	133	DEL	354	CF	462	DEL	355	CF	463	DEL
114	72	134	DEL	356	CF	464	DEL	357	CF	465	DEL
115	73	135	DEL	358	CF	466	DEL	359	CF	467	DEL
116	74	140	DEL	360	CF	468	DEL	361	CF	469	DEL
117	75	141	DEL	362	CF	470	DEL	363	CF	471	DEL
118	76	142	DEL	364	CF	472	DEL	365	CF	473	DEL
119	77	143	DEL	366	CF	474	DEL	367	CF	475	DEL
120	78	144	DEL	368	CF	476	DEL	369	CF	477	DEL
121	79	145	DEL	370	CF	478	DEL	371	CF	479	DEL
122	7A	150	DEL	372	CF	480	DEL	373	CF	481	DEL
123	7B	151	DEL	374	CF	482	DEL	375	CF	483	DEL
124	7C	152	DEL	376	CF	484	DEL	377	CF	485	DEL
125	7D	153	DEL	378	CF	486	DEL	379	CF	487	DEL
126	7E	154	DEL	380	CF	488	DEL	381	CF	489	DEL
127	7F	155	DEL	382	CF	490	DEL	383	CF	491	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 an ASCII table on the back of today's lecture slip.)

ASCII TABLE



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

Converting from Character to Code:

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

ASCII TABLE



Decimal	Hex	Octal	Char
0	00	00	NUL
1	01	01	SOH
2	02	02	STX
3	03	03	ETX
4	04	04	EOT
5	05	05	ENQ
6	06	06	ACK
7	07	07	BEL
8	08	10	BS
9	09	11	HT
10	0A	12	LF
11	0B	13	VT
12	0C	14	FF
13	0D	15	CR
14	0E	16	SO
15	0F	17	SI
16	10	20	SP
17	11	21	!
18	12	22	"
19	13	23	#
20	14	24	\$
21	15	25	%
22	16	26	&
23	17	27	'
24	18	30	(
25	19	31)
26	1A	32	*
27	1B	33	+
28	1C	34	,
29	1D	35	-
30	1E	36	.
31	1F	37	/
32	20	40	0
33	21	41	1
34	22	42	2
35	23	43	3
36	24	44	4
37	25	45	5
38	26	46	6
39	27	47	7
40	28	48	8
41	29	49	9
42	2A	50	:
43	2B	51	;
44	2C	52	<
45	2D	53	=
46	2E	54	>
47	2F	55	?
48	30	60	@
49	31	61	A
50	32	62	B
51	33	63	C
52	34	64	D
53	35	65	E
54	36	66	F
55	37	67	G
56	38	68	H
57	39	69	I
58	3A	70	J
59	3B	71	K
60	3C	72	L
61	3D	73	M
62	3E	74	N
63	3F	75	O
64	40	80	P
65	41	81	Q
66	42	82	R
67	43	83	S
68	44	84	T
69	45	85	U
70	46	86	V
71	47	87	W
72	48	88	X
73	49	89	Y
74	4A	90	Z
75	4B	91	[
76	4C	92	\
77	4D	93]
78	4E	94	^
79	4F	95	_
80	50	100	`
81	51	101	a
82	52	102	b
83	53	103	c
84	54	104	d
85	55	105	e
86	56	106	f
87	57	107	g
88	58	110	h
89	59	111	i
90	5A	112	j
91	5B	113	k
92	5C	114	l
93	5D	115	m
94	5E	116	n
95	5F	117	o
96	60	120	p
97	61	121	q
98	62	122	r
99	63	123	s
100	64	124	t
101	65	125	u
102	66	126	v
103	67	127	w
104	68	130	x
105	69	131	y
106	6A	132	z
107	6B	133	{
108	6C	134	
109	6D	135	}
110	6E	136	~
111	6F	137	_

- `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'.

In Pairs or Triples...

Some review and some novel challenges:

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

Python Tutor

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

(Demo with pythonTutor)

User Input

Covered in detail in Lab 2:

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

(Demo with pythonTutor)

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Lecture Slip

Name:

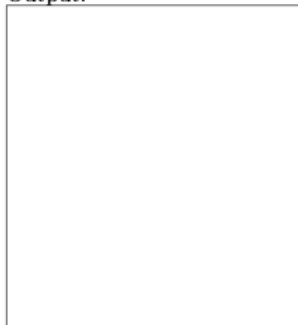
EmpID:

CSci 127 Sample Final, F17

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

```
months = ["Jan", "Feb", "Mar", "Apr", "May", \
"Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
half = months[6]
print(half.upper())
print(half[0])
print(months[-1].lower())
print(months[2:4])
start = 9
print(months[start-1])
term = 3
print(months[(start+term-1)%12])
```

Output:



Recap

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

Name: _____ Email: _____ CSci 127 Sample Final #11

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

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
year = 2015
month = "May"
week = 2
weekdays = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
weekdays = weekdays[0:week]
print(weekdays)
print(month)
print(months.index(month))
print(months[2:4])
week = 3
print(months[month:month+1])
week = 3
print(months[month:month+week])
```

Output:

Recap

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

```
Name: _____ Email: _____ CSci 127 Sample Final #11
```

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

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
year = 2015
month = "May"
week = 2
weekdays = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
weekdays.append("Mon")
print(weekdays)
print(months[2])
print(months[2:4])
print(months[2:4] * 2)
month = 3
print(months[month-1])
week = 3
print(weekdays[week*7-1:week*7])
```

Output:

Recap

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

Name: _____ Email: _____ CSci 127 Sample Final #11

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

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
year = 2015
month = "May"
year = year + 1
month = month.lower()
print(month)
print(months.index(month))
print(months[2])
print(months[2:4])
month = months[2:4]
print(months[month])
month = 3
print(months[month+month])
```

Output:

Recap

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

Name: _____ Email: _____ CSci 127 Sample Final #11

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

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
year = "2018"
msg = "Msg: " + year + " " + months[0]
last = months[-1]
print(msg.append())
print(msg[0])
print(months[-1].lower())
print(months[2:4])
month = 0
print(months[month-1])
month = 3
print(months[month+month-1])
```

Output:

Lecture Slips & Writing Boards



- Turn in lecture slips & writing boards as you leave...