

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From email

Frequently Asked Questions

From email

- I am not sure how to submit the Lab.

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

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

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

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

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

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

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

- **When is the midterm?**

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

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

- **When is the midterm?**

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

Seat Number

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

Seat Number

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

Seat Number

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

https://docs.google.com/spreadsheets/d/11enjiMGP GT1uLF7AG_r8dzYEIqsMFSd81Y5mIdITQwg/edit?usp=sharing

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**
- Please, **write down the row and seat number** you are seating in and continue to seat there for the rest of the semester.
- **Submit your row and seat number** using this link:
https://docs.google.com/spreadsheets/d/11enjiMGP GT1uLF7AG_r8dzYEIqsMFSd81Y5mIdITQwg/edit?usp=sharing
- The link to the form can also be found on Blackboard under Announcements.

Today's Topics



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

Today's Topics



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

Group Work

Some review and some novel challenges:

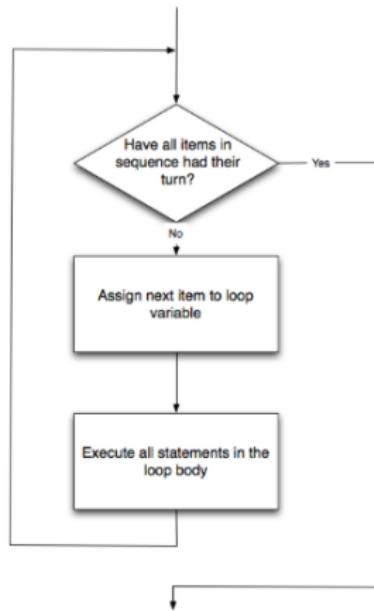
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

Python Tutor

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

(Demo with pythonTutor)

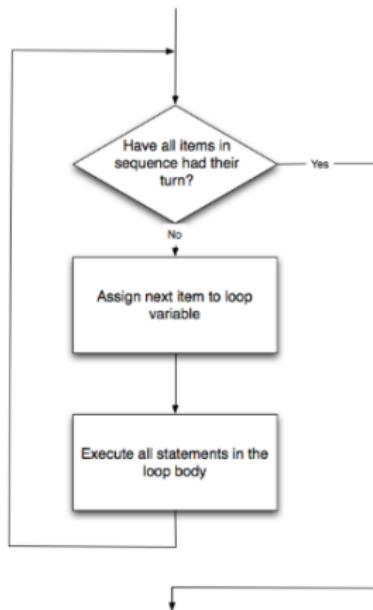
for-loop



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

How to Think Like CS, §4.5

for-loop



```
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()`.

How to Think Like CS, §4.5

Today's Topics



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

More on range():

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

Python Tutor

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

(Demo with pythonTutor)

range()

Simplest version:

- `range(stop)`



range()



Simplest version:

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

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the 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 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 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 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:
 $[start, start+step, start+2*step\dots, last]$
(where last is the largest $start+k*step$ less than stop)
- For example, if you want 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:
 $[start, start+step, start+2*step\dots, last]$
(where last is the largest $start+k*step$ less than stop)
- For example, if you want the list
[5,10,...,50]
you would write:

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

In summary: range()



The three versions:

In summary: range()



The three versions:

- range(stop)

In summary: range()



The three versions:

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

In summary: range()



The three versions:

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

Today's Topics



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

Variables

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



Variables

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



Variables

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



Variables



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

Variables



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

Variables



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

Variables



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

Variable Names

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



Variable Names

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



Variable Names



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

Variable Names



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

Variable Names



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

Today's Topics



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

Standardized Code for Characters

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

Standardized Code for Characters

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

Standardized Code for Characters

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

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	'
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	,	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END 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

Octal	Hex	Char									
000	00		010	0A	\t	020	14	\n	030	24	\v
001	01	\f	011	0B	\r	021	15	\b	031	25	\f
002	02	\n	012	0C	\t	022	16	\a	032	26	\n
003	03	\v	013	0D	\r\n	023	17	\b	033	27	\v
004	04	\b	014	0E	\t	024	18	\a	034	28	\b
005	05	\f	015	0F	\r\n	025	19	\b	035	29	\f
006	06	\n	016	10	\t	026	1A	\a	036	2A	\n
007	07	\v	017	11	\r\n	027	1B	\b	037	2B	\v
010	0A	\t	018	12	\t	028	1C	\a	038	2C	\t
011	0B	\r	019	13	\r\n	029	1D	\b	039	2D	\r
012	0C	\b	01A	14	\t	02A	1E	\a	03A	2E	\b
013	0D	\f	01B	15	\r\n	02B	1F	\b	03B	2F	\f
014	0E	\n	01C	16	\t	02C	20	\a	03C	30	\n
015	0F	\v	01D	17	\r\n	02D	21	\b	03D	31	\v
016	10	\b	01E	18	\t	02E	22	\a	03E	32	\b
017	11	\f	01F	19	\r\n	02F	23	\b	03F	33	\f
020	14	\n	020	1A	\t	030	24	\a	040	40	\n
021	15	\v	021	1B	\r\n	031	25	\b	041	41	\v
022	16	\b	022	1C	\t	032	26	\a	042	42	\b
023	17	\f	023	1D	\r\n	033	27	\b	043	43	\f
024	18	\n	024	1E	\t	034	28	\a	044	44	\n
025	19	\v	025	1F	\r\n	035	29	\b	045	45	\v
026	1A	\b	026	20	\t	036	2A	\a	046	46	\b
027	1B	\a	027	21	\r\n	037	2B	\b	047	47	\a
028	1C	\b	028	22	\t	038	2C	\a	048	48	\b
029	1D	\f	029	23	\r\n	039	2D	\b	049	49	\f
02A	1E	\n	02A	24	\t	03A	2E	\a	04A	4A	\n
02B	1F	\v	02B	25	\r\n	03B	2F	\b	04B	4B	\v
02C	20	\b	02C	26	\t	03C	20	\a	04C	4C	\b
02D	21	\f	02D	27	\r\n	03D	21	\b	04D	4D	\f
02E	22	\n	02E	28	\t	03E	22	\a	04E	4E	\n
02F	23	\v	02F	29	\r\n	03F	23	\b	04F	4F	\v
030	24	\b	030	2A	\t	040	40	\a	050	50	\n
031	25	\a	031	2B	\r\n	041	41	\b	051	51	\v
032	26	\b	032	2C	\t	042	42	\a	052	52	\b
033	27	\f	033	2D	\r\n	043	43	\b	053	53	\f
034	28	\n	034	20	\t	044	44	\a	054	54	\n
035	29	\v	035	21	\r\n	045	45	\b	055	55	\v
036	2A	\b	036	22	\t	046	46	\a	056	56	\b
037	2B	\a	037	23	\r\n	047	47	\b	057	57	\a
038	2C	\b	038	24	\t	048	48	\a	058	58	\b
039	2D	\f	039	25	\r\n	049	49	\b	059	59	\f
03A	20	\n	03A	26	\t	040	40	\a	060	60	\n
03B	21	\v	03B	27	\r\n	041	41	\b	061	61	\v
03C	22	\b	03C	28	\t	042	42	\a	062	62	\b
03D	23	\a	03D	29	\r\n	043	43	\b	063	63	\a
03E	24	\b	03E	20	\t	044	44	\a	064	64	\b
03F	25	\f	03F	21	\r\n	045	45	\b	065	65	\f
040	26	\n	040	22	\t	046	46	\a	066	66	\n
041	27	\v	041	23	\r\n	047	47	\b	067	67	\v
042	28	\b	042	24	\t	048	48	\a	068	68	\b
043	29	\a	043	25	\r\n	049	49	\b	069	69	\a
044	20	\b	044	26	\t	040	40	\a	070	70	\n
045	21	\f	045	27	\r\n	041	41	\b	071	71	\v
046	22	\n	046	28	\t	042	42	\a	072	72	\b
047	23	\v	047	29	\r\n	043	43	\b	073	73	\v
048	24	\b	048	20	\t	044	44	\a	074	74	\n
049	25	\a	049	21	\r\n	045	45	\b	075	75	\a
04A	26	\b	04A	22	\t	040	40	\a	076	76	\n
04B	27	\f	04B	23	\r\n	041	41	\b	077	77	\v
04C	28	\n	04C	24	\t	042	42	\a	078	78	\b
04D	29	\v	04D	25	\r\n	043	43	\b	079	79	\v
04E	20	\b	04E	26	\t	044	44	\a	080	80	\n
04F	21	\a	04F	27	\r\n	045	45	\b	081	81	\a

Converting from Character to Code:

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

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0000		32	0020		64	0040	
1	0001	!>	33	0021	!	65	0041	A
2	0002	"	34	0022	"	66	0042	B
3	0003	#	35	0023	#	67	0043	C
4	0004	\$	36	0024	\$	68	0044	D
5	0005	%	37	0025	%	69	0045	E
6	0006	&	38	0026	&	70	0046	F
7	0007	^	39	0027	^	71	0047	G
8	0008	_	40	0028	_	72	0048	H
9	0009	:	41	0029	:	73	0049	I
10	000A	,	42	002A	,	74	004A	J
11	000B	.	43	002B	.	75	004B	K
12	000C	;	44	002C	;	76	004C	L
13	000D	<	45	002D	<	77	004D	M
14	000E	=	46	002E	=	78	004E	N
15	000F	+	47	002F	+	79	004F	O
16	0010	-	48	0030	-	80	0050	P
17	0011	*	49	0031	*	81	0051	Q
18	0012	/	50	0032	/	82	0052	R
19	0013	\	51	0033	\	83	0053	S
20	0014	~	52	0034	~	84	0054	T
21	0015	`	53	0035	`	85	0055	U
22	0016	~	54	0036	~	86	0056	V
23	0017	`	55	0037	`	87	0057	W
24	0018	~	56	0038	~	88	0058	X
25	0019	`	57	0039	~	89	0059	Y
26	001A	~	58	003A	`	90	005A	Z
27	001B	~	59	003B	~	91	005B	[\
28	001C	~	60	003C	~	92	005C]
29	001D	~	61	003D	~	93	005D	^
30	001E	~	62	003E	~	94	005E	_
31	001F	~	63	003F	~	95	005F	:
32	0020		64	0040		96	0060	;
33	0021	!	65	0041	A	97	0061	,
34	0022	"	66	0042	B	98	0062	.
35	0023	#	67	0043	C	99	0063	=
36	0024	\$	68	0044	D	100	0064	+
37	0025	%	69	0045	E			
38	0026	&	70	0046	F			
39	0027	^	71	0047	G			
40	0028	_	72	0048	H			
41	0029	:	73	0049	I			
42	002A	,	74	004A	J			
43	002B	.	75	004B	K			
44	002C	;	76	004C	L			
45	002D	<	77	004D	M			
46	002E	=	78	004E	N			
47	002F	+	79	004F	O			
48	0030	-	80	0050	P			
49	0031	*	81	0051	Q			
50	0032	/	82	0052	R			
51	0033	\	83	0053	S			
52	0034	~	84	0054	T			
53	0035	`	85	0055	U			
54	0036	~	86	0056	V			
55	0037	`	87	0057	W			
56	0038	~	88	0058	X			
57	0039	~	89	0059	Y			
58	003A	~	90	005A	Z			
59	003B	~	91	005B	[
60	003C	~	92	005C]			
61	003D	~	93	005D	^			
62	003E	~	94	005E	_			
63	003F	~	95	005F	:			
64	0040		96	0060	;			
65	0041	A	97	0061	,			
66	0042	B	98	0062	.			
67	0043	C	99	0063	=			
68	0044	D	100	0064	+			

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

Converting from Character to Code:

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

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00	\0	32	20		64	40	!
1	01	\1	33	21	!	65	41	A
2	02	\2	34	22	”	66	42	B
3	03	\3	35	23	”	67	43	C
4	04	\4	36	24	”	68	44	D
5	05	\5	37	25	”	69	45	E
6	06	\6	38	26	”	70	46	F
7	07	\7	39	27	”	71	47	G
8	08	\8	40	28	”	72	48	H
9	09	\9	41	29	”	73	49	I
10	0A	\n	42	2A	”	74	4A	J
11	0B	\r	43	2B	”	75	4B	K
12	0C	\t	44	2C	”	76	4C	L
13	0D	\v	45	2D	”	77	4D	M
14	0E	\f	46	2E	”	78	4E	N
15	0F	\u000F	47	2F	”	79	4F	O
16	10	\u0010	48	30	”	80	50	P
17	11	\u0011	49	31	”	81	51	Q
18	12	\u0012	4A	32	”	82	52	R
19	13	\u0013	4B	33	”	83	53	S
20	14	\u0014	4C	34	”	84	54	T
21	15	\u0015	4D	35	”	85	55	U
22	16	\u0016	4E	36	”	86	56	V
23	17	\u0017	4F	37	”	87	57	W
24	18	\u0018	50	38	”	88	58	X
25	19	\u0019	51	39	”	89	59	Y
26	1A	\u001A	52	3A	”	90	5A	Z
27	1B	\u001B	53	3B	”	91	5B	[\u001B]
28	1C	\u001C	54	3C	”	92	5C	\u001C
29	1D	\u001D	55	3D	”	93	5D	\u001D
30	1E	\u001E	56	3E	”	94	5E	\u001E
31	1F	\u001F	57	3F	”	95	5F	\u001F
32	20	\u0020	58	40		96	60	\u0020
33	21	\u0021	59	41	!	97	61	‘a’
34	22	\u0022	60	42	”	98	62	‘b’
35	23	\u0023	61	43	”	99	63	‘c’
36	24	\u0024	62	44	”	100	64	‘d’
37	25	\u0025	63	45	”			
38	26	\u0026	64	46	”			
39	27	\u0027	65	47	”			
40	28	\u0028	66	48	”			
41	29	\u0029	67	49	”			
42	2A	\u002A	68	4A	”			
43	2B	\u002B	69	4B	”			
44	2C	\u002C	70	4C	”			
45	2D	\u002D	71	4D	”			
46	2E	\u002E	72	4E	”			
47	2F	\u002F	73	4F	”			
48	30	\u0030	74	50	”			
49	31	\u0031	75	51	”			
50	32	\u0032	76	52	”			
51	33	\u0033	77	53	”			
52	34	\u0034	78	54	”			
53	35	\u0035	79	55	”			
54	36	\u0036	80	56	”			
55	37	\u0037	81	57	”			
56	38	\u0038	82	58	”			
57	39	\u0039	83	59	”			
58	3A	\u003A	84	5A	”			
59	3B	\u003B	85	5B	”			
60	3C	\u003C	86	5C	”			
61	3D	\u003D	87	5D	”			
62	3E	\u003E	88	5E	”			
63	3F	\u003F	89	5F	”			
64	40	\u0040	90	60	”			
65	41	\u0041	91	61	‘a’			
66	42	\u0042	92	62	‘b’			
67	43	\u0043	93	63	‘c’			
68	44	\u0044	94	64	‘d’			
69	45	\u0045	95	65	\u0045			
70	46	\u0046	96	66	\u0046			
71	47	\u0047	97	67	\u0047			
72	48	\u0048	98	68	\u0048			
73	49	\u0049	99	69	\u0049			
74	4A	\u004A	100	6A	\u004A			

- `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'.)

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00	\0	32	20	Space	128	80	~
1	01	\1	33	21	!	129	81	!
2	02	\2	34	22	“	130	82	“
3	03	\3	35	23	”	131	83	”
4	04	\4	36	24	#	132	84	#
5	05	\5	37	25	%	133	85	%
6	06	\6	38	26	&	134	86	&
7	07	\7	39	27	*	135	87	*
8	08	\8	40	28	(136	88	(
9	09	\9	41	29)	137	89)
10	0A	\n	42	2A	,	138	8A	,
11	0B	\v	43	2B	-	139	8B	-
12	0C	\f	44	2C	_	140	8C	_
13	0D	\r	45	2D	.	141	8D	.
14	0E	\t	46	2E	;	142	8E	;
15	0F	\n	47	2F	=	143	8F	=
16	10	\010	48	30	:	144	80	:
17	11	\011	49	31	,	145	81	,
18	12	\012	50	32	:	146	82	:
19	13	\013	51	33	,	147	83	,
20	14	\014	52	34	:	148	84	:
21	15	\015	53	35	,	149	85	,
22	16	\016	54	36	:	150	86	:
23	17	\017	55	37	,	151	87	,
24	18	\018	56	38	:	152	88	:
25	19	\019	57	39	,	153	89	,
26	1A	\01A	58	3A	:	154	8A	:
27	1B	\01B	59	3B	,	155	8B	,
28	1C	\01C	60	3C	:	156	8C	:
29	1D	\01D	61	3D	,	157	8D	,
30	1E	\01E	62	3E	:	158	8E	:
31	1F	\01F	63	3F	,	159	8F	,
32	20	\020	64	40	(`	160	90	(`
33	21	\021	65	41)`	161	91)`
34	22	\022	66	42	(`	162	92	(`
35	23	\023	67	43)`	163	93)`
36	24	\024	68	44	(`	164	94	(`
37	25	\025	69	45)`	165	95)`
38	26	\026	70	46	(`	166	96	(`
39	27	\027	71	47)`	167	97)`
40	28	\028	72	48	(`	168	98	(`
41	29	\029	73	49)`	169	99)`
42	2A	\02A	74	4A	(`	170	9A	(`
43	2B	\02B	75	4B)`	171	9B)`
44	2C	\02C	76	4C	(`	172	9C	(`
45	2D	\02D	77	4D)`	173	9D)`
46	2E	\02E	78	4E	(`	174	9E	(`
47	2F	\02F	79	4F)`	175	9F)`
48	30	\030	80	50	(`	176	90	(`
49	31	\031	81	51)`	177	91)`
50	32	\032	82	52	(`	178	92	(`
51	33	\033	83	53)`	179	93)`
52	34	\034	84	54	(`	180	94	(`
53	35	\035	85	55)`	181	95)`
54	36	\036	86	56	(`	182	96	(`
55	37	\037	87	57)`	183	97)`
56	38	\038	88	58	(`	184	98	(`
57	39	\039	89	59)`	185	99)`
58	3A	\03A	90	5A	(`	186	9A	(`
59	3B	\03B	91	5B)`	187	9B)`
60	3C	\03C	92	5C	(`	188	9C	(`
61	3D	\03D	93	5D)`	189	9D)`
62	3E	\03E	94	5E	(`	190	9E	(`
63	3F	\03F	95	5F)`	191	9F)`
64	40	\040	96	60	(`	192	A0	(`
65	41	\041	97	61)`	193	A1)`
66	42	\042	98	62	(`	194	A2	(`
67	43	\043	99	63)`	195	A3)`
68	44	\044	100	64	(`	196	A4	(`
69	45	\045	101	65)`	197	A5)`
70	46	\046	102	66	(`	198	A6	(`
71	47	\047	103	67)`	199	A7)`
72	48	\048	104	68	(`	200	A8	(`
73	49	\049	105	69)`	201	A9)`
74	4A	\04A	106	6A	(`	202	AA	(`
75	4B	\04B	107	6B)`	203	AB)`
76	4C	\04C	108	6C	(`	204	AC	(`
77	4D	\04D	109	6D)`	205	AD)`
78	4E	\04E	110	6E	(`	206	AE	(`
79	4F	\04F	111	6F)`	207	AF)`
80	50	\050	112	70	(`	208	A0	(`
81	51	\051	113	71)`	209	A1)`
82	52	\052	114	72	(`	210	A2	(`
83	53	\053	115	73)`	211	A3)`
84	54	\054	116	74	(`	212	A4	(`
85	55	\055	117	75)`	213	A5)`
86	56	\056	118	76	(`	214	A6	(`
87	57	\057	119	77)`	215	A7)`
88	58	\058	120	78	(`	216	A8	(`
89	59	\059	121	79)`	217	A9)`
90	5A	\05A	122	7A	(`	218	AA	(`
91	5B	\05B	123	7B)`	219	AB)`
92	5C	\05C	124	7C	(`	220	AC	(`
93	5D	\05D	125	7D)`	221	AD)`
94	5E	\05E	126	7E	(`	222	AE	(`
95	5F	\05F	127	7F)`	223	AF)`
96	60	\060	128	80	(`	224	A0	(`
97	61	\061	129	81)`	225	A1)`
98	62	\062	130	82	(`	226	A2	(`
99	63	\063	131	83)`	227	A3)`
100	64	\064	132	84	(`	228	A4	(`
101	65	\065	133	85)`	229	A5)`
102	66	\066	134	86	(`	230	A6	(`
103	67	\067	135	87)`	231	A7)`
104	68	\068	136	88	(`	232	A8	(`
105	69	\069	137	89)`	233	A9)`
106	6A	\06A	138	8A	(`	234	AA	(`
107	6B	\06B	139	8B)`	235	AB)`
108	6C	\06C	140	8C	(`	236	AC	(`
109	6D	\06D	141	8D)`	237	AD)`
110	6E	\06E	142	8E	(`	238	AE	(`
111	6F	\06F	143	8F)`	239	AF)`

- `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

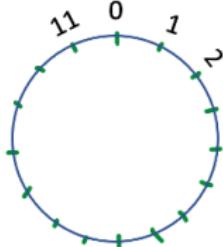
```
1 #Predict what will be printed:  
2  
3 for c in range(65,90):  
4     print(chr(c))  
5  
6 message = "I love Python"  
7 newMessage = ""  
8 for c in message:  
9     print(ord(c)) #Print the Unicode of each number  
10    print(chr(ord(c)+1)) #Print the next character  
11    newMessage = newMessage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

(Demo with pythonTutor)

Wrap

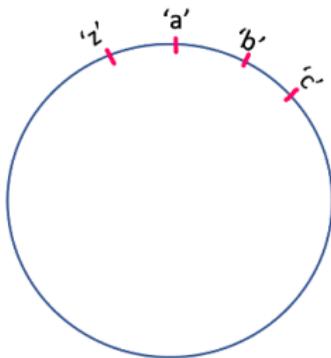
Hints for Programming Assignment 9 in

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



What is 15:00?

What is 17:00?



(1) How many scales in this shape?

(2) How to map 'a' to 0, 'b' to 1, ...

(3) How to map 0 back to 'a', 1 back to 'b', ...

User Input

Covered in detail in Lab 2:

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

(Demo with pythonTutor)

Side Note: '+' for numbers and strings

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



Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Today's Topics



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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

- The first line creates a variable, called `s`, that stores the string:
`"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.
 - ▶ What would `print(s.count("sS"))` output?
 - ▶ What about:
`mess = "10 20 21 9 101 35"
mults = mess.count("0 ")
print(mults)`

More on Strings: Indexing & Substrings

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

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

More on Strings: Indexing & Substrings

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

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

More on Strings: Indexing & Substrings

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

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

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

More on Strings: Indexing & Substrings

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[0]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- $s[-1]$ is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- $s[-1]$ is ‘s’.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[3:6]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[:3]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[:-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

Today's Topics



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

Recap

- In Python, we introduced:

```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print()  
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()  
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

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()  
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()

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()  
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

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()  
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

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Practice Quiz & Final Questions



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

Practice Quiz & Final Questions



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

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Lecture Slips & Writing Boards



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