

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

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry!*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades.*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- Can I work ahead?
Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- 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 accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- 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 Lab Quizzes and daily programming assignments.

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- 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 Lab Quizzes and daily programming assignments.
- I missed class. Do you need documentation?

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- 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 Lab Quizzes and daily programming assignments.
- I missed class. Do you need documentation?
No, but if you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- 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 Lab Quizzes and daily programming assignments.
- I missed class. Do you need documentation?
No, but If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- I have not received any emails from this course.

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- 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 Lab Quizzes and daily programming assignments.
- I missed class. Do you need documentation?
No, but if you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- I have not received any emails from this course.
That is a big problem! *We send tons of important information through email. Please email studenthelpdesk@hunter.cuny.edu to update your email on Blackboard to one you check regularly.*

Frequently Asked Questions

From email

- I accidentally submitted the Lab Quiz before completing it. Can I retake it?
No. *Unfortunately we cannot reopen Quizzes, but don't worry! Your grade on the final exam will replace any lower quiz grades. To avoid this in the future, take the quiz after you read the Lab and are ready. You have **15 minutes** to take the quiz, after that it will automatically save your answers and close.*
- 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 Lab Quizzes and daily programming assignments.
- I missed class. Do you need documentation?
No, but if you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- I have not received any emails from this course.
That is a big problem! *We send tons of important information through email. Please email studenthelpdesk@hunter.cuny.edu to update your email on Blackboard to one you check regularly.*

Today's Topics



- **For-loops**
- `range()`
- Variables
- Characters
- Strings
- Guest: Elise Harris (Advising, Clubs, Internships and more)

In Pairs or Triples...

Some review and some novel challenges:

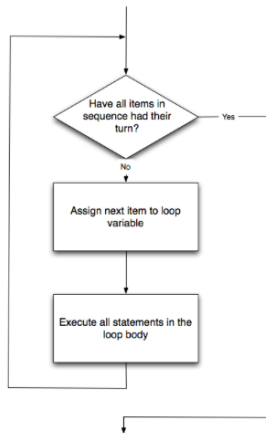
```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12   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

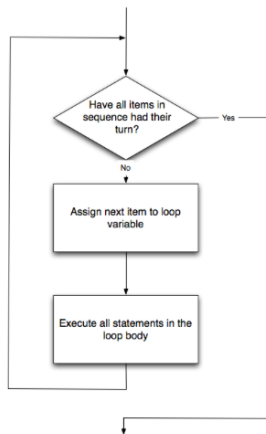
for-loop



```
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
- Guest: Elise Harris (Advising, Clubs, Internships and more)

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 $start+k*step$ 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)
```

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
- Guest: Elise Harris (Advising, Clubs, Internships and more)

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
- Guest: Elise Harris (Advising, Clubs, Internships and more)

Standardized Code for Characters

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

Standardized Code for Characters

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

Standardized Code for Characters

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

ASCII TABLE

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

(wiki)

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	0x10	32	0x20	48	0x30
1		17	0x11	33	0x21	49	0x31
2		18	0x12	34	0x22	50	0x32
3		19	0x13	35	0x23	51	0x33
4		20	0x14	36	0x24	52	0x34
5		21	0x15	37	0x25	53	0x35
6		22	0x16	38	0x26	54	0x36
7		23	0x17	39	0x27	55	0x37
8		24	0x18	40	0x28	56	0x38
9		25	0x19	41	0x29	57	0x39
10	0x0A	26	0x1A	42	0x2A	58	0x3A
11	0x0B	27	0x1B	43	0x2B	59	0x3B
12	0x0C	28	0x1C	44	0x2C	60	0x3C
13	0x0D	29	0x1D	45	0x2D	61	0x3D
14	0x0E	30	0x1E	46	0x2E	62	0x3E
15	0x0F	31	0x1F	47	0x2F	63	0x3F
16	0x10	32	0x20	48	0x30	64	0x40
17	0x11	33	0x21	49	0x31	65	0x41
18	0x12	34	0x22	50	0x32	66	0x42
19	0x13	35	0x23	51	0x33	67	0x43
20	0x14	36	0x24	52	0x34	68	0x44
21	0x15	37	0x25	53	0x35	69	0x45
22	0x16	38	0x26	54	0x36	70	0x46
23	0x17	39	0x27	55	0x37	71	0x47
24	0x18	40	0x28	56	0x38	72	0x48
25	0x19	41	0x29	57	0x39	73	0x49
26	0x1A	42	0x2A	58	0x3A	74	0x4A
27	0x1B	43	0x2B	59	0x3B	75	0x4B
28	0x1C	44	0x2C	60	0x3C	76	0x4C
29	0x1D	45	0x2D	61	0x3D	77	0x4D
30	0x1E	46	0x2E	62	0x3E	78	0x4E
31	0x1F	47	0x2F	63	0x3F	79	0x4F
32	0x20	48	0x30	64	0x40	80	0x50
33	0x21	49	0x31	65	0x41	81	0x51
34	0x22	50	0x32	66	0x42	82	0x52
35	0x23	51	0x33	67	0x43	83	0x53
36	0x24	52	0x34	68	0x44	84	0x54
37	0x25	53	0x35	69	0x45	85	0x55
38	0x26	54	0x36	70	0x46	86	0x56
39	0x27	55	0x37	71	0x47	87	0x57
40	0x28	56	0x38	72	0x48	88	0x58
41	0x29	57	0x39	73	0x49	89	0x59
42	0x2A	58	0x3A	74	0x4A	90	0x5A
43	0x2B	59	0x3B	75	0x4B	91	0x5B
44	0x2C	60	0x3C	76	0x4C	92	0x5C
45	0x2D	61	0x3D	77	0x4D	93	0x5D
46	0x2E	62	0x3E	78	0x4E	94	0x5E
47	0x2F	63	0x3F	79	0x4F	95	0x5F
48	0x30	64	0x40	80	0x50	96	0x60
49	0x31	65	0x41	81	0x51	97	0x61
50	0x32	66	0x42	82	0x52	98	0x62
51	0x33	67	0x43	83	0x53	99	0x63
52	0x34	68	0x44	84	0x54	100	0x64
53	0x35	69	0x45	85	0x55	101	0x65
54	0x36	70	0x46	86	0x56	102	0x66
55	0x37	71	0x47	87	0x57	103	0x67
56	0x38	72	0x48	88	0x58	104	0x68
57	0x39	73	0x49	89	0x59	105	0x69
58	0x3A	74	0x4A	90	0x5A	106	0x6A
59	0x3B	75	0x4B	91	0x5B	107	0x6B
60	0x3C	76	0x4C	92	0x5C	108	0x6C
61	0x3D	77	0x4D	93	0x5D	109	0x6D
62	0x3E	78	0x4E	94	0x5E	110	0x6E
63	0x3F	79	0x4F	95	0x5F	111	0x6F
64	0x40	80	0x50	96	0x60	112	0x70
65	0x41	81	0x51	97	0x61	113	0x71
66	0x42	82	0x52	98	0x62	114	0x72
67	0x43	83	0x53	99	0x63	115	0x73
68	0x44	84	0x54	100	0x64	116	0x74
69	0x45	85	0x55	101	0x65	117	0x75
70	0x46	86	0x56	102	0x66	118	0x76
71	0x47	87	0x57	103	0x67	119	0x77
72	0x48	88	0x58	104	0x68	120	0x78
73	0x49	89	0x59	105	0x69	121	0x79
74	0x4A	90	0x5A	106	0x6A	122	0x7A
75	0x4B	91	0x5B	107	0x6B	123	0x7B
76	0x4C	92	0x5C	108	0x6C	124	0x7C
77	0x4D	93	0x5D	109	0x6D	125	0x7D
78	0x4E	94	0x5E	110	0x6E	126	0x7E
79	0x4F	95	0x5F	111	0x6F	127	0x7F
80	0x50	96	0x60	112	0x70	128	0x80
81	0x51	97	0x61	113	0x71	129	0x81
82	0x52	98	0x62	114	0x72	130	0x82
83	0x53	99	0x63	115	0x73	131	0x83
84	0x54	100	0x64	116	0x74	132	0x84
85	0x55	101	0x65	117	0x75	133	0x85
86	0x56	102	0x66	118	0x76	134	0x86
87	0x57	103	0x67	119	0x77	135	0x87
88	0x58	104	0x68	120	0x78	136	0x88
89	0x59	105	0x69	121	0x79	137	0x89
90	0x5A	106	0x6A	122	0x7A	138	0x8A
91	0x5B	107	0x6B	123	0x7B	139	0x8B
92	0x5C	108	0x6C	124	0x7C	140	0x8C
93	0x5D	109	0x6D	125	0x7D	141	0x8D
94	0x5E	110	0x6E	126	0x7E	142	0x8E
95	0x5F	111	0x6F	127	0x7F	143	0x8F
96	0x60	112	0x70	128	0x80	144	0x90
97	0x61	113	0x71	129	0x81	145	0x91
98	0x62	114	0x72	130	0x82	146	0x92
99	0x63	115	0x73	131	0x83	147	0x93
100	0x64	116	0x74	132	0x84	148	0x94
101	0x65	117	0x75	133	0x85	149	0x95
102	0x66	118	0x76	134	0x86	150	0x96
103	0x67	119	0x77	135	0x87	151	0x97
104	0x68	120	0x78	136	0x88	152	0x98
105	0x69	121	0x79	137	0x89	153	0x99
106	0x6A	122	0x7A	138	0x8A	154	0x9A
107	0x6B	123	0x7B	139	0x8B	155	0x9B
108	0x6C	124	0x7C	140	0x8C	156	0x9C
109	0x6D	125	0x7D	141	0x8D	157	0x9D
110	0x6E	126	0x7E	142	0x8E	158	0x9E
111	0x6F	127	0x7F	143	0x8F	159	0x9F
112	0x70	128	0x80	144	0x90	160	0xA0
113	0x71	129	0x81	145	0x91	161	0xA1
114	0x72	130	0x82	146	0x92	162	0xA2
115	0x73	131	0x83	147	0x93	163	0xA3
116	0x74	132	0x84	148	0x94	164	0xA4
117	0x75	133	0x85	149	0x95	165	0xA5
118	0x76	134	0x86	150	0x96	166	0xA6
119	0x77	135	0x87	151	0x97	167	0xA7
120	0x78	136	0x88	152	0x98	168	0xA8
121	0x79	137	0x89	153	0x99	169	0xA9
122	0x7A	138	0x8A	154	0x9A	170	0xAA
123	0x7B	139	0x8B	155	0x9B	171	0xAB
124	0x7C	140	0x8C	156	0x9C	172	0xAC
125	0x7D	141	0x8D	157	0x9D	173	0xAD
126	0x7E	142	0x8E	158	0x9E	174	0xAE
127	0x7F	143	0x8F	159	0x9F	175	0xAF
128	0x80	144	0x90	160	0xA0	176	0xB0
129	0x81	145	0x91	161	0xA1	177	0xB1
130	0x82	146	0x92	162	0xA2	178	0xB2
131	0x83	147	0x93	163	0xA3	179	0xB3
132	0x84	148	0x94	164	0xA4	180	0xB4
133	0x85	149	0x95	165	0xA5	181	0xB5
134	0x86	150	0x96	166	0xA6	182	0xB6
135	0x87	151	0x97	167	0xA7	183	0xB7
136	0x88	152	0x98	168	0xA8	184	0xB8
137	0x89	153	0x99	169	0xA9	185	0xB9
138	0x8A	154	0x9A	170	0xAA	186	0xBA
139	0x8B	155	0x9B	171	0xAB	187	0xBB
140	0x8C	156	0x9C	172	0xAC	188	0xBC
141	0x8D	157	0x9D	173	0xAD	189	0xBD
142	0x8E	158	0x9E	174	0xAE	190	0xBE
143	0x8F	159	0x9F	175	0xAF	191	0xBF
144	0x90	160	0xA0	176	0xB0	192	0xC0
145	0x91	161	0xA1	177	0xB1	193	0xC1
146	0x92	162	0xA2	178	0xB2	194	0xC2
147	0x93	163	0xA3	179	0xB3	195	0xC3
148	0x94	164	0xA4	180	0xB4	196	0xC4
149	0x95	165	0xA5	181	0xB5	197	0xC5
150	0x96	166	0xA6	182	0xB6	198	0xC6
151	0x97	167	0xA7	183	0xB7	199	0xC7
152	0x98	168	0xA8	184	0xB8	200	0xC8
153	0x99	169	0xA9	185	0xB9	201	0xC9
154	0x9A	170	0xAA	186	0xBA	202	0xCA
155	0x9B	171	0xAB	187	0xBB	203	0xCB
156	0x9C	172	0xAC	188	0xBC	204	0xCC
157	0x9D	173	0xAD	189	0xBD	205	0xCD
158	0x9E	174	0xAE	190	0xBE	206	0xCE
159	0x9F	175	0xAF	191	0xBF	207	0xCF
160	0xA0	176	0xB0	192	0xC0	208	0xD0
161	0xA1	177	0xB1	193	0xC1	209	0xD1
162	0xA2	178	0xB2	194	0xC2	210	0xD2
163	0xA3	179	0xB3	195	0xC3	211	0xD3
164	0xA4	180	0xB4	196	0xC4	212	0xD4
165	0xA5	181	0xB5	197	0xC5	213	0xD5
166	0xA6	182	0xB6	198	0xC6	214	0xD6
167	0xA7	183	0xB7	199	0xC7	215	0xD7
168	0xA8	184	0xB8	200	0xC8	216	0xD8
169	0xA9	185	0xB9	201	0xC9	217	0xD9
170	0xAA	186	0xBA	202	0xCA	218	0xDA
171	0xAB	187	0xBB	203	0xCB	219	0xDB
172	0xAC	188	0xBC	204	0xCC	220	0xDC
173	0xAD	189	0xBD	205	0xCD	221	0xDD
174	0xAE	190	0xBE	206	0xCE	222	0xDE
175	0xAF	191	0xBF	207	0xCF	223	0xDF
176	0xB0	192	0xC0	208	0xD0	224	0xE0

Converting from Character to Code:

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

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

ASCII TABLE

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

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

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	0	32	@	48	X
1		17	1	33	A	49	Y
2		18	2	34	B	50	Z
3		19	3	35	C	51	[
4		20	4	36	D	52	\
5		21	5	37	E	53]
6		22	6	38	F	54	^
7		23	7	39	G	55	_
8		24	8	40	H	56	`
9		25	9	41	I	57	{
10		26	A	42	J	58	
11		27	B	43	K	59	~
12		28	C	44	L	60	
13		29	D	45	M	61	
14		30	E	46	N	62	
15		31	F	47	O	63	
16	0	32	@	48	X	64	
17	1	33	A	49	Y	65	
18	2	34	B	50	Z	66	
19	3	35	C	51	[67	
20	4	36	D	52	\	68	
21	5	37	E	53]	69	
22	6	38	F	54	^	70	
23	7	39	G	55	_	71	
24	8	40	H	56	`	72	
25	9	41	I	57	{	73	
26	A	42	J	58		74	
27	B	43	K	59	~	75	
28	C	44	L	60		76	
29	D	45	M	61		77	
30	E	46	N	62		78	
31	F	47	O	63		79	
32	0	48	X	64		80	
33	1	49	Y	65		81	
34	2	50	Z	66		82	
35	3	51	[67		83	
36	4	52	\	68		84	
37	5	53]	69		85	
38	6	54	^	70		86	
39	7	55	_	71		87	
40	8	56	`	72		88	
41	9	57	{	73		89	
42	A	58		74		90	
43	B	59	~	75		91	
44	C	60		76		92	
45	D	61		77		93	
46	E	62		78		94	
47	F	63		79		95	
48	0	64		80		96	
49	1	65		81		97	
50	2	66		82		98	
51	3	67		83		99	
52	4	68		84		100	
53	5	69		85		101	
54	6	70		86		102	
55	7	71		87		103	
56	8	72		88		104	
57	9	73		89		105	
58	A	74		90		106	
59	B	75		91		107	
60	C	76		92		108	
61	D	77		93		109	
62	E	78		94		110	
63	F	79		95		111	
64	0	80		96		112	
65	1	81		97		113	
66	2	82		98		114	
67	3	83		99		115	
68	4	84		100		116	
69	5	85		101		117	
70	6	86		102		118	
71	7	87		103		119	
72	8	88		104		120	
73	9	89		105		121	
74	A	90		106		122	
75	B	91		107		123	
76	C	92		108		124	
77	D	93		109		125	
78	E	94		110		126	
79	F	95		111		127	
80	0	96		112		128	
81	1	97		113		129	
82	2	98		114		130	
83	3	99		115		131	
84	4	100		116		132	
85	5	101		117		133	
86	6	102		118		134	
87	7	103		119		135	
88	8	104		120		136	
89	9	105		121		137	
90	A	106		122		138	
91	B	107		123		139	
92	C	108		124		140	
93	D	109		125		141	
94	E	110		126		142	
95	F	111		127		143	
96	0	112		128		144	
97	1	113		129		145	
98	2	114		130		146	
99	3	115		131		147	
100	4	116		132		148	
101	5	117		133		149	
102	6	118		134		150	
103	7	119		135		151	
104	8	120		136		152	
105	9	121		137		153	
106	A	122		138		154	
107	B	123		139		155	
108	C	124		140		156	
109	D	125		141		157	
110	E	126		142		158	
111	F	127		143		159	
112	0	128		144		160	
113	1	129		145		161	
114	2	130		146		162	
115	3	131		147		163	
116	4	132		148		164	
117	5	133		149		165	
118	6	134		150		166	
119	7	135		151		167	
120	8	136		152		168	
121	9	137		153		169	
122	A	138		154		170	
123	B	139		155		171	
124	C	140		156		172	
125	D	141		157		173	
126	E	142		158		174	
127	F	143		159		175	
128	0	144		160		176	
129	1	145		161		177	
130	2	146		162		178	
131	3	147		163		179	
132	4	148		164		180	
133	5	149		165		181	
134	6	150		166		182	
135	7	151		167		183	
136	8	152		168		184	
137	9	153		169		185	
138	A	154		170		186	
139	B	155		171		187	
140	C	156		172		188	
141	D	157		173		189	
142	E	158		174		190	
143	F	159		175		191	
144	0	160		176		192	
145	1	161		177		193	
146	2	162		178		194	
147	3	163		179		195	
148	4	164		180		196	
149	5	165		181		197	
150	6	166		182		198	
151	7	167		183		199	
152	8	168		184		200	
153	9	169		185		201	
154	A	170		186		202	
155	B	171		187		203	
156	C	172		188		204	

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

Converting from Character to Code:

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

ASCII TABLE

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

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

In Pairs or Triples...

Some review and some novel challenges:

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

Python Tutor

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

(Demo with pythonTutor)

User Input

Covered in detail in Lab 2:

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

(Demo with pythonTutor)

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

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**
- Guest: Elise Harris (Advising, Clubs, Internships and more)

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
- **Guest: Elise Harris (Advising, Clubs, Internships and more)**

Guest Speaker: Elise Harris CS Opportunities

- Announcement on Blackboard:
 - ▶ 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

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

● In Python, we introduced:

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

Recap

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

● In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Practice Quiz & Final Questions



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

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.

Practice Quiz & Final Questions



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

Practice Quiz & Final Questions



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

See you next week!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments