

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 form: bit.ly/127_seat_number

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 form: bit.ly/127_seat_number
- The link to the form can also be found on Blackboard under Announcements.

Today's Topics



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

Today's Topics



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

Group Work

Some review and some novel challenges:

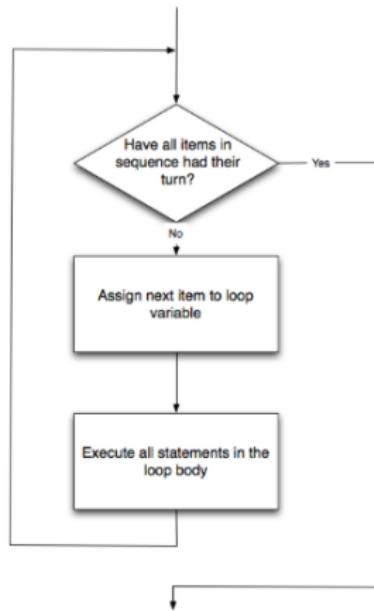
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

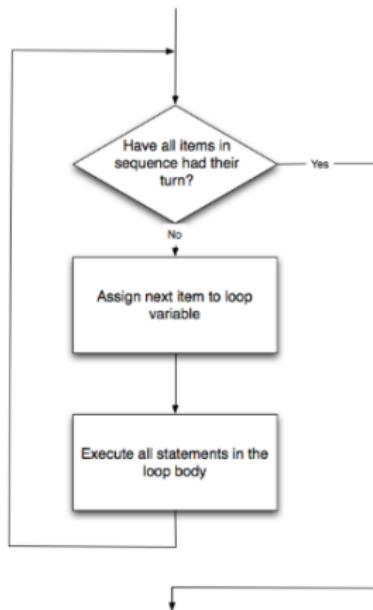
for-loop



```
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
- Guests: Internships & Clubs

More on range():

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

Python Tutor

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

(Demo with pythonTutor)

range()

Simplest version:

- `range(stop)`



range()



Simplest version:

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

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the list `[0,1,2,3,...,100]`, you would write:

range()



Simplest version:

- `range(stop)`
- Produces a list: $[0,1,2,3,\dots,stop-1]$
- For example, if you want the list $[0,1,2,3,\dots,100]$, 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...,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
 $[5, 10, \dots, 50]$
you would write:

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



In summary: range()



The three versions:

In summary: range()



The three versions:

- range(stop)

In summary: range()



The three versions:

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

In summary: range()



The three versions:

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

Today's Topics



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

Variables

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



Variables

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



Variables

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



Variables



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

Variables



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

Variables



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

Variables



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

Variable Names

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



Variable Names

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



Variable Names



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

Variable Names



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

Variable Names



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

Today's Topics



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

Standardized Code for Characters

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

Standardized Code for Characters

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

Standardized Code for Characters

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

ASCII TABLE

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

Decimal Value	Hex Char	Octal Char	Binary Char	Decimal Value	Hex Char	Octal Char	Binary Char
0	00	0	00000000	32	20	40	00100000
1	01	1	00000001	33	21	41	00100001
2	02	2	00000010	34	22	42	00100010
3	03	3	00000011	35	23	43	00100011
4	04	4	00000100	36	24	44	00100100
5	05	5	00000101	37	25	45	00100101
6	06	6	00000110	38	26	46	00100110
7	07	7	00000111	39	27	47	00100111
8	08	10	00001000	40	28	50	00101000
9	09	11	00001001	41	29	51	00101001
10	0A	12	00001010	42	2A	52	00101010
11	0B	13	00001011	43	2B	53	00101011
12	0C	14	00001100	44	2C	54	00101100
13	0D	15	00001101	45	2D	55	00101101
14	0E	16	00001110	46	2E	56	00101110
15	0F	17	00001111	47	2F	57	00101111
16	10	20	00010000	48	30	60	001010000
17	11	21	00010001	49	31	61	001010001
18	12	22	00010010	4A	32	62	001010010
19	13	23	00010011	4B	33	63	001010011
20	14	24	00010100	4C	34	64	001010100
21	15	25	00010101	4D	35	65	001010101
22	16	26	00010110	4E	36	66	001010110
23	17	27	00010111	4F	37	67	001010111
24	18	28	00011000	50	38	68	001011000
25	19	29	00011001	51	39	69	001011001
26	1A	2A	00011010	52	3A	6A	001011010
27	1B	2B	00011011	53	3B	6B	001011011
28	1C	2C	00011100	54	3C	6C	001011100
29	1D	2D	00011101	55	3D	6D	001011101
30	1E	2E	00011110	56	3E	6E	001011110
31	1F	2F	00011111	57	3F	6F	001011111
32	20	30	001000000	64	40	100	01000000
33	21	31	001000001	65	41	101	01000001
34	22	32	001000010	66	42	110	01000010
35	23	33	001000011	67	43	111	01000011
36	24	34	001000100	68	44	120	01000100
37	25	35	001000101	69	45	121	01000101
38	26	36	001000110	6A	46	122	01000110
39	27	37	001000111	6B	47	123	01000111
40	28	38	001001000	6C	48	124	010001000
41	29	39	001001001	6D	49	125	010001001
42	2A	3A	001001010	6E	4A	126	010001010
43	2B	3B	001001011	6F	4B	127	010001011
44	2C	3C	001001100	70	4C	130	010001100
45	2D	3D	001001101	71	4D	131	010001101
46	2E	3E	001001110	72	4E	132	010001110
47	2F	3F	001001111	73	4F	133	010001111
48	30	40	0010100000	74	40	140	01001000
49	31	41	0010100001	75	41	141	010010001
50	32	42	0010100010	76	42	142	010010010
51	33	43	0010100011	77	43	143	010010011
52	34	44	0010100100	78	44	144	010010100
53	35	45	0010100101	79	45	145	010010101
54	36	46	0010100110	7A	46	146	010010110
55	37	47	0010100111	7B	47	147	010010111
56	38	48	0010101000	7C	48	148	010011000
57	39	49	0010101001	7D	49	149	010011001
58	40	4A	0010101010	7E	4A	150	010011010
59	41	4B	0010101011	7F	4B	151	010011011
60	42	4C	0010101100	80	4C	152	010011100
61	43	4D	0010101101	81	4D	153	010011101
62	44	4E	0010101110	82	4E	154	010011110
63	45	4F	0010101111	83	4F	155	010011111
64	46	50	0010110000	84	40	160	010011000
65	47	51	0010110001	85	41	161	010011001
66	48	52	0010110010	86	42	162	010011010
67	49	53	0010110011	87	43	163	010011011
68	4A	54	0010110100	88	44	164	010011100
69	4B	55	0010110101	89	45	165	010011101
70	4C	56	0010110110	8A	46	166	010011110
71	4D	57	0010110111	8B	47	167	010011111
72	4E	58	0010111000	8C	48	168	010011100
73	4F	59	0010111001	8D	49	169	010011101
74	50	5A	0010111010	8E	4A	170	010011110
75	51	5B	0010111011	8F	4B	171	010011111
76	52	5C	0010111100	90	4C	172	010011100
77	53	5D	0010111101	91	4D	173	010011101
78	54	5E	0010111110	92	4E	174	010011110
79	55	5F	0010111111	93	4F	175	010011111
80	56	60	0011000000	94	50	180	010100000
81	57	61	0011000001	95	51	181	010100001
82	58	62	0011000010	96	52	182	010100010
83	59	63	0011000011	97	53	183	010100011
84	5A	64	0011000100	98	54	184	010100100
85	5B	65	0011000101	99	55	185	010100101
86	5C	66	0011000110	9A	56	186	010100110
87	5D	67	0011000111	9B	57	187	010100111
88	5E	68	0011001000	9C	58	188	010101000
89	5F	69	0011001001	9D	59	189	010101001
90	60	6A	0011001010	9E	5A	190	010101010
91	61	6B	0011001011	9F	5B	191	010101011
92	62	6C	0011001100	90	40	192	010101100
93	63	6D	0011001101	91	41	193	010101101
94	64	6E	0011001110	92	42	194	010101110
95	65	6F	0011001111	93	43	195	010101111
96	66	70	0011010000	94	44	196	010101100
97	67	71	0011010001	95	45	197	010101101
98	68	72	0011010010	96	46	198	010101110
99	69	73	0011010011	97	47	199	010101111
100	6A	74	0011010100	98	48	200	010101100

Converting from Character to Code:

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

Decimal Num Char	Octal Num Char	Hex Num Char	Decimal Num Char	Octal Num Char	Hex Num Char
' '	40	20	'A'	41	25
'A'	65	41	'B'	66	42
'B'	66	42	'C'	67	43
'C'	67	43	'D'	68	44
'D'	68	44	'E'	69	45
'E'	69	45	'F'	70	46
'F'	70	46	'G'	71	47
'G'	71	47	'H'	72	48
'H'	72	48	'I'	73	49
'I'	73	49	'J'	74	4A
'J'	74	4A	'K'	75	4B
'K'	75	4B	'L'	76	4C
'L'	76	4C	'M'	77	4D
'M'	77	4D	'N'	78	4E
'N'	78	4E	'O'	79	4F
'O'	79	4F	'P'	80	50
'P'	80	50	'Q'	81	51
'Q'	81	51	'R'	82	52
'R'	82	52	'S'	83	53
'S'	83	53	'T'	84	54
'T'	84	54	'U'	85	55
'U'	85	55	'V'	86	56
'V'	86	56	'W'	87	57
'W'	87	57	'X'	88	58
'X'	88	58	'Y'	89	59
'Y'	89	59	'Z'	90	5A
'Z'	90	5A	'[space]'	91	5B
'[space]'	91	5B	'`'	92	5C
'`'	92	5C	'`'	93	5D
'`'	93	5D	'`'	94	5E
'`'	94	5E	'`'	95	5F
'`'	95	5F	'`'	96	60
'`'	96	60	'`'	97	61
'`'	97	61	'`'	98	62
'`'	98	62	'`'	99	63
'`'	99	63	'`'	100	64
'`'	100	64	'`'	101	65
'`'	101	65	'`'	102	66
'`'	102	66	'`'	103	67
'`'	103	67	'`'	104	68
'`'	104	68	'`'	105	69
'`'	105	69	'`'	106	6A
'`'	106	6A	'`'	107	6B
'`'	107	6B	'`'	108	6C
'`'	108	6C	'`'	109	6D
'`'	109	6D	'`'	110	6E
'`'	110	6E	'`'	111	6F
'`'	111	6F	'`'	112	70
'`'	112	70	'`'	113	71
'`'	113	71	'`'	114	72
'`'	114	72	'`'	115	73
'`'	115	73	'`'	116	74
'`'	116	74	'`'	117	75
'`'	117	75	'`'	118	76
'`'	118	76	'`'	119	77
'`'	119	77	'`'	120	78
'`'	120	78	'`'	121	79
'`'	121	79	'`'	122	7A
'`'	122	7A	'`'	123	7B
'`'	123	7B	'`'	124	7C
'`'	124	7C	'`'	125	7D
'`'	125	7D	'`'	126	7E
'`'	126	7E	'`'	127	7F
'`'	127	7F	'`'	128	80
'`'	128	80	'`'	129	81
'`'	129	81	'`'	130	82
'`'	130	82	'`'	131	83
'`'	131	83	'`'	132	84
'`'	132	84	'`'	133	85
'`'	133	85	'`'	134	86
'`'	134	86	'`'	135	87
'`'	135	87	'`'	136	88
'`'	136	88	'`'	137	89
'`'	137	89	'`'	138	8A
'`'	138	8A	'`'	139	8B
'`'	139	8B	'`'	140	8C
'`'	140	8C	'`'	141	8D
'`'	141	8D	'`'	142	8E
'`'	142	8E	'`'	143	8F
'`'	143	8F	'`'	144	90
'`'	144	90	'`'	145	91
'`'	145	91	'`'	146	92
'`'	146	92	'`'	147	93
'`'	147	93	'`'	148	94
'`'	148	94	'`'	149	95
'`'	149	95	'`'	150	96
'`'	150	96	'`'	151	97
'`'	151	97	'`'	152	98
'`'	152	98	'`'	153	99
'`'	153	99	'`'	154	9A
'`'	154	9A	'`'	155	9B
'`'	155	9B	'`'	156	9C
'`'	156	9C	'`'	157	9D
'`'	157	9D	'`'	158	9E
'`'	158	9E	'`'	159	9F
'`'	159	9F	'`'	160	A0
'`'	160	A0	'`'	161	A1
'`'	161	A1	'`'	162	A2
'`'	162	A2	'`'	163	A3
'`'	163	A3	'`'	164	A4
'`'	164	A4	'`'	165	A5
'`'	165	A5	'`'	166	A6
'`'	166	A6	'`'	167	A7
'`'	167	A7	'`'	168	A8
'`'	168	A8	'`'	169	A9
'`'	169	A9	'`'	170	A0
'`'	170	A0	'`'	171	A1
'`'	171	A1	'`'	172	A2
'`'	172	A2	'`'	173	A3
'`'	173	A3	'`'	174	A4
'`'	174	A4	'`'	175	A5
'`'	175	A5	'`'	176	A6
'`'	176	A6	'`'	177	A7
'`'	177	A7	'`'	178	A8
'`'	178	A8	'`'	179	A9
'`'	179	A9	'`'	180	A0
'`'	180	A0	'`'	181	A1
'`'	181	A1	'`'	182	A2
'`'	182	A2	'`'	183	A3
'`'	183	A3	'`'	184	A4
'`'	184	A4	'`'	185	A5
'`'	185	A5	'`'	186	A6
'`'	186	A6	'`'	187	A7
'`'	187	A7	'`'	188	A8
'`'	188	A8	'`'	189	A9
'`'	189	A9	'`'	190	A0
'`'	190	A0	'`'	191	A1
'`'	191	A1	'`'	192	A2
'`'	192	A2	'`'	193	A3
'`'	193	A3	'`'	194	A4
'`'	194	A4	'`'	195	A5
'`'	195	A5	'`'	196	A6
'`'	196	A6	'`'	197	A7
'`'	197	A7	'`'	198	A8
'`'	198	A8	'`'	199	A9
'`'	199	A9	'`'	200	A0

- `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	[\u0021]
34	22	\u0022	60	42	”	98	62	[\u0022]
35	23	\u0023	61	43	‘	99	63	[\u0023]
36	24	\u0024	62	44	’	100	64	[\u0024]
37	25	\u0025	63	45	“	101	65	[\u0025]
38	26	\u0026	64	46	”	102	66	[\u0026]
39	27	\u0027	65	47	‘	103	67	[\u0027]
40	28	\u0028	66	48	’	104	68	[\u0028]
41	29	\u0029	67	49	“	105	69	[\u0029]
42	2A	\u002A	68	4A	”	106	6A	[\u002A]
43	2B	\u002B	69	4B	‘	107	6B	[\u002B]
44	2C	\u002C	70	4C	’	108	6C	[\u002C]
45	2D	\u002D	71	4D	“	109	6D	[\u002D]
46	2E	\u002E	72	4E	”	110	6E	[\u002E]
47	2F	\u002F	73	4F	‘	111	6F	[\u002F]
48	30	\u0030	74	50	”	112	70	[\u0030]
49	31	\u0031	75	51	‘	113	71	[\u0031]
50	32	\u0032	76	52	’	114	72	[\u0032]
51	33	\u0033	77	53	“	115	73	[\u0033]
52	34	\u0034	78	54	”	116	74	[\u0034]
53	35	\u0035	79	55	‘	117	75	[\u0035]
54	36	\u0036	80	56	’	118	76	[\u0036]
55	37	\u0037	81	57	“	119	77	[\u0037]
56	38	\u0038	82	58	”	120	78	[\u0038]
57	39	\u0039	83	59	‘	121	79	[\u0039]
58	3A	\u003A	84	5A	’	122	7A	[\u003A]
59	3B	\u003B	85	5B	“	123	7B	[\u003B]
60	3C	\u003C	86	5C	”	124	7C	[\u003C]
61	3D	\u003D	87	5D	‘	125	7D	[\u003D]
62	3E	\u003E	88	5E	’	126	7E	[\u003E]
63	3F	\u003F	89	5F	“	127	7F	[\u003F]
64	40	\u0040	90	60		128	80	[\u0040]
65	41	\u0041	91	61	!	129	81	[\u0041]
66	42	\u0042	92	62	”	130	82	[\u0042]
67	43	\u0043	93	63	‘	131	83	[\u0043]
68	44	\u0044	94	64	’	132	84	[\u0044]
69	45	\u0045	95	65	“	133	85	[\u0045]
70	46	\u0046	96	66	”	134	86	[\u0046]
71	47	\u0047	97	67	‘	135	87	[\u0047]
72	48	\u0048	98	68	’	136	88	[\u0048]
73	49	\u0049	99	69	“	137	89	[\u0049]
74	4A	\u004A	100	6A	”	138	8A	[\u004A]
75	4B	\u004B	101	6B	‘	139	8B	[\u004B]
76	4C	\u004C	102	6C	’	140	8C	[\u004C]
77	4D	\u004D	103	6D	“	141	8D	[\u004D]
78	4E	\u004E	104	6E	”	142	8E	[\u004E]
79	4F	\u004F	105	6F	‘	143	8F	[\u004F]
80	50	\u0050	106	70	”	144	90	[\u0050]
81	51	\u0051	107	71	‘	145	91	[\u0051]
82	52	\u0052	108	72	’	146	92	[\u0052]
83	53	\u0053	109	73	“	147	93	[\u0053]
84	54	\u0054	110	74	”	148	94	[\u0054]
85	55	\u0055	111	75	‘	149	95	[\u0055]
86	56	\u0056	112	76	’	150	96	[\u0056]
87	57	\u0057	113	77	“	151	97	[\u0057]
88	58	\u0058	114	78	”	152	98	[\u0058]
89	59	\u0059	115	79	‘	153	99	[\u0059]
90	5A	\u005A	116	7A	”	154	9A	[\u005A]
91	5B	\u005B	117	7B	‘	155	9B	[\u005B]
92	5C	\u005C	118	7C	”	156	9C	[\u005C]
93	5D	\u005D	119	7D	‘	157	9D	[\u005D]
94	5E	\u005E	120	7E	”	158	9E	[\u005E]
95	5F	\u005F	121	7F	‘	159	9F	[\u005F]

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

Converting from Character to Code:

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

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00	\0	32	20	\t	64	40	\n
1	01	\1	33	21	\a	65	41	A
2	02	\2	34	22	\b	66	42	B
3	03	\3	35	23	\f	67	43	F
4	04	\4	36	24	\n	68	44	\n
5	05	\5	37	25	\r	69	45	\r
6	06	\6	38	26	\v	70	46	\v
7	07	\7	39	27	\b	71	47	\b
8	08	\8	40	28	\t	72	48	\t
9	09	\9	41	29	\n	73	49	\n
10	0A	\10	42	2A	\r	74	4A	\r
11	0B	\11	43	2B	\v	75	4B	\v
12	0C	\12	44	2C	\b	76	4C	\b
13	0D	\13	45	2D	\t	77	4D	\t
14	0E	\14	46	2E	\n	78	4E	\n
15	0F	\15	47	2F	\r	79	4F	\r
16	10	\16	48	30	\t	80	50	\t
17	11	\17	49	31	\n	81	51	\n
18	12	\18	4A	32	\r	82	52	\r
19	13	\19	4B	33	\v	83	53	\v
20	14	\20	4C	34	\b	84	54	\b
21	15	\21	4D	35	\t	85	55	\t
22	16	\22	4E	36	\n	86	56	\n
23	17	\23	4F	37	\r	87	57	\r
24	18	\24	50	38	\v	88	58	\v
25	19	\25	51	39	\b	89	59	\b
26	1A	\26	52	3A	\t	90	5A	\t
27	1B	\27	53	3B	\n	91	5B	\n
28	1C	\28	54	3C	\r	92	5C	\r
29	1D	\29	55	3D	\v	93	5D	\v
30	1E	\2A	56	3E	\b	94	5E	\b
31	1F	\2B	57	3F	\t	95	5F	\t
32	20	\2C	58	40	\n	96	60	\n
33	21	\2D	59	41	\r	97	61	\r
34	22	\2E	5A	42	\v	98	62	\v
35	23	\2F	5B	43	\b	99	63	\b
36	24	\30	5C	44	\t	100	64	\t
37	25	\31	5D	45	\n	101	65	\n
38	26	\32	5E	46	\r	102	66	\r
39	27	\33	5F	47	\v	103	67	\v
40	28	\34	60	48	\b	104	68	\b
41	29	\35	61	49	\t	105	69	\t
42	2A	\36	62	4A	\n	106	6A	\n
43	2B	\37	63	4B	\r	107	6B	\r
44	2C	\38	64	4C	\v	108	6C	\v
45	2D	\39	65	4D	\b	109	6D	\b
46	2E	\3A	66	4E	\t	110	6E	\t
47	2F	\3B	67	4F	\n	111	6F	\n
48	30	\3C	68	50	\r	112	70	\r
49	31	\3D	69	51	\v	113	71	\v
50	32	\3E	6A	52	\b	114	72	\b
51	33	\3F	6B	53	\t	115	73	\t
52	34	\30	6C	54	\n	116	74	\n
53	35	\31	6D	55	\r	117	75	\r
54	36	\32	6E	56	\v	118	76	\v
55	37	\33	6F	57	\b	119	77	\b
56	38	\34	70	58	\t	120	78	\t
57	39	\35	71	59	\n	121	79	\n
58	3A	\36	72	5A	\r	122	7A	\r
59	3B	\37	73	5B	\v	123	7B	\v
60	3C	\38	74	5C	\b	124	7C	\b
61	3D	\39	75	5D	\t	125	7D	\t
62	3E	\3A	76	5E	\n	126	7E	\n
63	3F	\3B	77	5F	\r	127	7F	\r
64	40	\3C	78	60	\v	128	80	\v
65	41	\3D	79	61	\b	129	81	\b
66	42	\3E	7A	62	\t	130	82	\t
67	43	\3F	7B	63	\n	131	83	\n
68	44	\30	7C	64	\r	132	84	\r
69	45	\31	7D	65	\v	133	85	\v
70	46	\32	7E	66	\b	134	86	\b
71	47	\33	7F	67	\t	135	87	\t
72	48	\34	80	68	\n	136	88	\n
73	49	\35	81	69	\r	137	89	\r
74	4A	\36	82	6A	\v	138	8A	\v
75	4B	\37	83	6B	\b	139	8B	\b
76	4C	\38	84	6C	\t	140	8C	\t
77	4D	\39	85	6D	\n	141	8D	\n
78	4E	\3A	86	6E	\r	142	8E	\r
79	4F	\3B	87	6F	\v	143	8F	\v
80	50	\3C	88	70	\b	144	90	\b
81	51	\3D	89	71	\t	145	91	\t
82	52	\3E	8A	72	\n	146	92	\n
83	53	\3F	8B	73	\r	147	93	\r
84	54	\30	8C	74	\v	148	94	\v
85	55	\31	8D	75	\b	149	95	\b
86	56	\32	8E	76	\t	150	96	\t
87	57	\33	8F	77	\n	151	97	\n
88	58	\34	90	78	\r	152	98	\r
89	59	\35	91	79	\v	153	99	\v
90	5A	\36	92	7A	\b	154	9A	\b
91	5B	\37	93	7B	\t	155	9B	\t
92	5C	\38	94	7C	\n	156	9C	\n
93	5D	\39	95	7D	\r	157	9D	\r
94	5E	\3A	96	7E	\v	158	9E	\v
95	5F	\3B	97	7F	\b	159	9F	\b
96	60	\3C	98	80	\t	160	100	\t
97	61	\3D	99	81	\n	161	101	\n
98	62	\3E	9A	82	\r	162	102	\r
99	63	\3F	9B	83	\v	163	103	\v
100	64	\30	9C	84	\b	164	104	\b
101	65	\31	9D	85	\t	165	105	\t
102	66	\32	9E	86	\n	166	106	\n
103	67	\33	9F	87	\r	167	107	\r
104	68	\34	100	88	\v	168	108	\v
105	69	\35	101	89	\b	169	109	\b
106	6A	\36	102	8A	\t	170	10A	\t
107	6B	\37	103	8B	\n	171	10B	\n
108	6C	\38	104	8C	\r	172	10C	\r
109	6D	\39	105	8D	\v	173	10D	\v
110	6E	\3A	106	8E	\b	174	10E	\b
111	6F	\3B	107	8F	\t	175	10F	\t
112	70	\3C	108	90	\n	176	110	\n
113	71	\3D	109	91	\r	177	111	\r
114	72	\3E	110	92	\v	178	112	\v
115	73	\3F	111	93	\b	179	113	\b
116	74	\30	112	94	\t	180	114	\t
117	75	\31	113	95	\n	181	115	\n
118	76	\32	114	96	\r	182	116	\r
119	77	\33	115	97	\v	183	117	\v
120	78	\34	116	98	\b	184	118	\b
121	79	\35	117	99	\t	185	119	\t
122	7A	\36	118	100	\n	186	120	\n
123	7B	\37	119	101	\r	187	121	\r
124	7C	\38	120	102	\v	188	122	\v
125	7D	\39	121	103	\b	189	123	\b
126	7E	\3A	122	104	\t	190	124	\t
127	7F	\3B	123	105	\n	191	125	\n
128	80	\3C	124	106	\r	192	126	\r
129	81	\3D	125	107	\v	193	127	\v
130	82	\3E	126	108	\b	194	128	\b
131	83	\3F	127	109	\t	195	129	\t
132	84	\30	128	110	\n	196	130	\n
133	85	\31	129	111	\r	197	131	\r
134	86	\32	130	112	\v	198	132	\v
135	87	\33	131	113	\b	199	133	\b
136	88	\34	132	114	\t	200	134	\t
137	89	\35	133	115	\n	201	135	\n
138	8A	\36	134	116	\r	202	136	\r
139	8B	\37	135	117	\v	203	137	\v
140	8C	\38	136	118	\b	204	138	\b
141	8D	\39	137	119	\t	205	139	\t
142	8E	\3A	138	120	\n	206	140	\n
143	8F	\3B	139	121	\r	207	141	\r
144	90	\3C	140	122	\v	208	142	\v
145	91	\3D	141	123	\b	209	143	\b
146	92	\3E	142	124	\t	210	144	\t
147	93	\3F	143	125	\n	211	145	\n
148	94	\30	144	126	\r	212	146	\r
149	95	\31	145	127	\v	213	147	\v
150	96	\32	146	128	\b	214	148	\b
151	97	\33	147	129	\t	215	149	\t
152	98	\34	148	130	\n	216	150	\n
153	99	\35	149	131	\r	217	151	\r
154	9A	\36	150	132	\v	218	152	\v
155	9B	\37	151	133	\b	219	153	\b
156	9C	\38	152	134	\t	220	154	\t
157	9D	\39	153	135	\n	221	155	\n
158	9E	\3A	154	136	\r	222	156	\r
159	9F	\3B	155	137	\v	223	157	\v
160	100	\3C	156	138	\b	224	158	\b
161	101	\3D	157	139	\t	225	159	\t
162	102	\3E	158	140	\n	226	160	\n
163	103	\3F	159	141	\r	227	161	\r
164	104	\30	160	142	\v	228	162	\v
165	105	\31	161	143	\b	229	163	\b
166	106	\32	162	144	\t	230	164	\t
167	107	\33	163	145	\n	231	165	\n
168	108	\34	164	146	\r	232	166	\r
169	109	\35	165	147	\v	233	167	\v
170	110	\36	166	148	\b	234	168	\b
171	111	\37	167	149	\t	235	169	\t
172	112	\38	168	150	\n	236	170	\n
173	113	\39	169	151	\r	237	171	\r
174	114	\3A	170	152	\v	238	172	\v
175	115	\3B	171	153	\b	239	173	\b
176	116	\3C	172	154	\t	240	174	\t
177	117	\3D</						

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	\t	64	40	\n
1	01	\1	33	21	\a	65	41	A
2	02	\2	34	22	\b	66	42	B
3	03	\3	35	23	\f	67	43	F
4	04	\4	36	24	\n	68	44	\n
5	05	\5	37	25	\r	69	45	\r
6	06	\6	38	26	\v	70	46	\v
7	07	\7	39	27	\b	71	47	\b
8	08	\8	40	28	\t	72	48	\t
9	09	\9	41	29	\n	73	49	\n
10	0A	\10	42	2A	\r	74	4A	\r
11	0B	\11	43	2B	\v	75	4B	\v
12	0C	\12	44	2C	\b	76	4C	\b
13	0D	\13	45	2D	\t	77	4D	\t
14	0E	\14	46	2E	\n	78	4E	\n
15	0F	\15	47	2F	\r	79	4F	\r
16	10	\16	48	30	\t	80	50	\t
17	11	\17	49	31	\n	81	51	\n
18	12	\18	4A	32	\r	82	52	\r
19	13	\19	4B	33	\v	83	53	\v
20	14	\20	4C	34	\b	84	54	\b
21	15	\21	4D	35	\t	85	55	\t
22	16	\22	4E	36	\n	86	56	\n
23	17	\23	4F	37	\r	87	57	\r
24	18	\24	50	38	\v	88	58	\v
25	19	\25	51	39	\b	89	59	\b
26	1A	\26	52	3A	\t	90	5A	\t
27	1B	\27	53	3B	\n	91	5B	\n
28	1C	\28	54	3C	\r	92	5C	\r
29	1D	\29	55	3D	\v	93	5D	\v
30	1E	\2A	56	3E	\b	94	5E	\b
31	1F	\2B	57	3F	\t	95	5F	\t
32	20	\2C	58	40	\n	96	60	\n
33	21	\2D	59	41	\r	97	61	\r
34	22	\2E	5A	42	\v	98	62	\v
35	23	\2F	5B	43	\b	99	63	\b
36	24	\30	5C	44	\t	100	64	\t
37	25	\31	5D	45	\n	101	65	\n
38	26	\32	5E	46	\r	102	66	\r
39	27	\33	5F	47	\v	103	67	\v
40	28	\34	60	48	\b	104	68	\b
41	29	\35	61	49	\t	105	69	\t
42	2A	\36	62	4A	\n	106	6A	\n
43	2B	\37	63	4B	\r	107	6B	\r
44	2C	\38	64	4C	\v	108	6C	\v
45	2D	\39	65	4D	\b	109	6D	\b
46	2E	\3A	66	4E	\t	110	6E	\t
47	2F	\3B	67	4F	\n	111	6F	\n
48	30	\3C	68	50	\r	112	70	\r
49	31	\3D	69	51	\v	113	71	\v
50	32	\3E	6A	52	\b	114	72	\b
51	33	\3F	6B	53	\t	115	73	\t
52	34	\30	6C	54	\n	116	74	\n
53	35	\31	6D	55	\r	117	75	\r
54	36	\32	6E	56	\v	118	76	\v
55	37	\33	6F	57	\b	119	77	\b
56	38	\34	70	58	\t	120	78	\t
57	39	\35	71	59	\n	121	79	\n
58	3A	\36	72	5A	\r	122	7A	\r
59	3B	\37	73	5B	\v	123	7B	\v
60	3C	\38	74	5C	\b	124	7C	\b
61	3D	\39	75	5D	\t	125	7D	\t
62	3E	\3A	76	5E	\n	126	7E	\n
63	3F	\3B	77	5F	\r	127	7F	\r
64	40	\3C	78	60	\v	128	80	\v
65	41	\3D	79	61	\b	129	81	\b
66	42	\3E	7A	62	\t	130	82	\t
67	43	\3F	7B	63	\n	131	83	\n
68	44	\30	7C	64	\r	132	84	\r
69	45	\31	7D	65	\v	133	85	\v
70	46	\32	7E	66	\b	134	86	\b
71	47	\33	7F	67	\t	135	87	\t
72	48	\34	80	68	\n	136	88	\n
73	49	\35	81	69	\r	137	89	\r
74	4A	\36	82	6A	\v	138	8A	\v
75	4B	\37	83	6B	\b	139	8B	\b
76	4C	\38	84	6C	\t	140	8C	\t
77	4D	\39	85	6D	\n	141	8D	\n
78	4E	\3A	86	6E	\r	142	8E	\r
79	4F	\3B	87	6F	\v	143	8F	\v
80	50	\3C	88	70	\b	144	90	\b
81	51	\3D	89	71	\t	145	91	\t
82	52	\3E	8A	72	\n	146	92	\n
83	53	\3F	8B	73	\r	147	93	\r
84	54	\30	8C	74	\v	148	94	\v
85	55	\31	8D	75	\b	149	95	\b
86	56	\32	8E	76	\t	150	96	\t
87	57	\33	8F	77	\n	151	97	\n
88	58	\34	90	78	\r	152	98	\r
89	59	\35	91	79	\v	153	99	\v
90	5A	\36	92	7A	\b	154	9A	\b
91	5B	\37	93	7B	\t	155	9B	\t
92	5C	\38	94	7C	\n	156	9C	\n
93	5D	\39	95	7D	\r	157	9D	\r
94	5E	\3A	96	7E	\v	158	9E	\v
95	5F	\3B	97	7F	\b	159	9F	\b
96	60	\3C	98	80	\t	160	100	\t
97	61	\3D	99	81	\n	161	101	\n
98	62	\3E	9A	82	\r	162	102	\r
99	63	\3F	9B	83	\v	163	103	\v
100	64	\30	9C	84	\b	164	104	\b
101	65	\31	9D	85	\t	165	105	\t
102	66	\32	9E	86	\n	166	106	\n
103	67	\33	9F	87	\r	167	107	\r
104	68	\34	100	88	\v	168	108	\v
105	69	\35	101	89	\b	169	109	\b
106	6A	\36	102	8A	\t	170	10A	\t
107	6B	\37	103	8B	\n	171	10B	\n
108	6C	\38	104	8C	\r	172	10C	\r
109	6D	\39	105	8D	\v	173	10D	\v
110	6E	\3A	106	8E	\b	174	10E	\b
111	6F	\3B	107	8F	\t	175	10F	\t
112	70	\3C	108	90	\n	176	110	\n
113	71	\3D	109	91	\r	177	111	\r
114	72	\3E	110	92	\v	178	112	\v
115	73	\3F	111	93	\b	179	113	\b
116	74	\30	112	94	\t	180	114	\t
117	75	\31	113	95	\n	181	115	\n
118	76	\32	114	96	\r	182	116	\r
119	77	\33	115	97	\v	183	117	\v
120	78	\34	116	98	\b	184	118	\b
121	79	\35	117	99	\t	185	119	\t
122	7A	\36	118	100	\n	186	120	\n
123	7B	\37	119	101	\r	187	121	\r
124	7C	\38	120	102	\v	188	122	\v
125	7D	\39	121	103	\b	189	123	\b
126	7E	\3A	122	104	\t	190	124	\t
127	7F	\3B	123	105	\n	191	125	\n
128	80	\3C	124	106	\r	192	126	\r
129	81	\3D	125	107	\v	193	127	\v
130	82	\3E	126	108	\b	194	128	\b
131	83	\3F	127	109	\t	195	129	\t
132	84	\30	128	110	\n	196	130	\n
133	85	\31	129	111	\r	197	131	\r
134	86	\32	130	112	\v	198	132	\v
135	87	\33	131	113	\b	199	133	\b
136	88	\34	132	114	\t	200	134	\t
137	89	\35	133	115	\n	201	135	\n
138	8A	\36	134	116	\r	202	136	\r
139	8B	\37	135	117	\v	203	137	\v
140	8C	\38	136	118	\b	204	138	\b
141	8D	\39	137	119	\t	205	139	\t
142	8E	\3A	138	120	\n	206	140	\n
143	8F	\3B	139	121	\r	207	141	\r
144	90	\3C	140	122	\v	208	142	\v
145	91	\3D	141	123	\b	209	143	\b
146	92	\3E	142	124	\t	210	144	\t
147	93	\3F	143	125	\n	211	145	\n
148	94	\30	144	126	\r	212	146	\r
149	95	\31	145	127	\v	213	147	\v
150	96	\32	146	128	\b	214	148	\b
151	97	\33	147	129	\t	215	149	\t
152	98	\34	148	130	\n	216	150	\n
153	99	\35	149	131	\r	217	151	\r
154	9A	\36	150	132	\v	218	152	\v
155	9B	\37	151	133	\b	219	153	\b
156	9C	\38	152	134	\t	220	154	\t
157	9D	\39	153	135	\n	221	155	\n
158	9E	\3A	154	136	\r	222	156	\r
159	9F	\3B	155	137	\v	223	157	\v
160	100	\3C	156	138	\b	224	158	\b
161	101	\3D	157	139	\t	225	159	\t
162	102	\3E	158	140	\n	226	160	\n
163	103	\3F	159	141	\r	227	161	\r
164	104	\30	160	142	\v	228	162	\v
165	105	\31	161	143	\b	229	163	\b
166	106	\32	162	144	\t	230	164	\t
167	107	\33	163	145	\n	231	165	\n
168	108	\34	164	146	\r	232	166	\r
169	109	\35	165	147	\v	233	167	\v
170	110	\36	166	148	\b	234	168	\b
171	111	\37	167	149	\t	235	169	\t
172	112	\38	168	150	\n	236	170	\n
173	113	\39	169	151	\r	237	171	\r
174	114	\3A	170	152	\v	238	172	\v
175	115	\3B	171	153	\b	239	173	\b
176	116	\3C	172	154	\t	240	174	\t
177	117	\3D</						

Converting from Character to Code:

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

Decimal Num Char	Octal Num Char	Hex Num Char	Decimal Num Char	Octal Num Char	Hex Num Char
'\000'	'000'	'000'	'\001'	'001'	'001'
'\002'	'002'	'002'	'\003'	'003'	'003'
'\004'	'004'	'004'	'\005'	'005'	'005'
'\006'	'006'	'006'	'\007'	'007'	'007'
'\010'	'010'	'00A'	'\011'	'011'	'00B'
'\012'	'012'	'00C'	'\013'	'013'	'00D'
'\014'	'014'	'00E'	'\015'	'015'	'00F'
'\016'	'016'	'010'	'\017'	'017'	'011'
'\020'	'020'	'012'	'\021'	'021'	'013'
'\022'	'022'	'014'	'\023'	'023'	'015'
'\024'	'024'	'016'	'\025'	'025'	'017'
'\026'	'026'	'018'	'\027'	'027'	'019'
'\030'	'030'	'01A'	'\031'	'031'	'01B'
'\032'	'032'	'01C'	'\033'	'033'	'01D'
'\034'	'034'	'01E'	'\035'	'035'	'01F'
'\036'	'036'	'020'	'\037'	'037'	'021'
'\040'	'040'	'022'	'\041'	'041'	'023'
'\042'	'042'	'024'	'\043'	'043'	'025'
'\044'	'044'	'026'	'\045'	'045'	'027'
'\046'	'046'	'028'	'\047'	'047'	'029'
'\048'	'048'	'02A'	'\049'	'049'	'02B'
'\050'	'050'	'02C'	'\051'	'051'	'02D'
'\052'	'052'	'02E'	'\053'	'053'	'02F'
'\054'	'054'	'030'	'\055'	'055'	'031'
'\056'	'056'	'032'	'\057'	'057'	'033'
'\060'	'060'	'034'	'\061'	'061'	'035'
'\062'	'062'	'036'	'\063'	'063'	'037'
'\064'	'064'	'038'	'\065'	'065'	'039'
'\066'	'066'	'03A'	'\067'	'067'	'03B'
'\068'	'068'	'03C'	'\069'	'069'	'03D'
'\070'	'070'	'03E'	'\071'	'071'	'03F'
'\072'	'072'	'040'	'\073'	'073'	'041'
'\074'	'074'	'042'	'\075'	'075'	'043'
'\076'	'076'	'044'	'\077'	'077'	'045'
'\080'	'080'	'048'	'\081'	'081'	'049'
'\082'	'082'	'04A'	'\083'	'083'	'04B'
'\084'	'084'	'04C'	'\085'	'085'	'04D'
'\086'	'086'	'04E'	'\087'	'087'	'04F'
'\088'	'088'	'050'	'\089'	'089'	'051'
'\090'	'090'	'052'	'\091'	'091'	'053'
'\092'	'092'	'054'	'\093'	'093'	'055'
'\094'	'094'	'056'	'\095'	'095'	'057'
'\096'	'096'	'058'	'\097'	'097'	'059'
'\098'	'098'	'05A'	'\099'	'099'	'05B'
'\09A'	'09A'	'05C'	'\09B'	'09B'	'05D'
'\09C'	'09C'	'05E'	'\09D'	'09D'	'05F'
'\09E'	'09E'	'060'	'\09F'	'09F'	'061'
'\0A0'	'0A0'	'062'	'\0A1'	'0A1'	'063'
'\0A2'	'0A2'	'064'	'\0A3'	'0A3'	'065'
'\0A4'	'0A4'	'066'	'\0A5'	'0A5'	'067'
'\0A6'	'0A6'	'068'	'\0A7'	'0A7'	'069'
'\0A8'	'0A8'	'06A'	'\0A9'	'0A9'	'06B'
'\0AA'	'0AA'	'06C'	'\0AB'	'0AB'	'06D'
'\0AC'	'0AC'	'06E'	'\0AD'	'0AD'	'06F'
'\0AE'	'0AE'	'070'	'\0AF'	'0AF'	'071'
'\0B0'	'0B0'	'072'	'\0B1'	'0B1'	'073'
'\0B2'	'0B2'	'074'	'\0B3'	'0B3'	'075'
'\0B4'	'0B4'	'076'	'\0B5'	'0B5'	'077'
'\0B6'	'0B6'	'078'	'\0B7'	'0B7'	'079'
'\0B8'	'0B8'	'07A'	'\0B9'	'0B9'	'07B'
'\0BA'	'0BA'	'07C'	'\0BB'	'0BB'	'07D'
'\0BC'	'0BC'	'07E'	'\0BD'	'0BD'	'07F'
'\0BE'	'0BE'	'080'	'\0BF'	'0BF'	'081'
'\0C0'	'0C0'	'082'	'\0C1'	'0C1'	'083'
'\0C2'	'0C2'	'084'	'\0C3'	'0C3'	'085'
'\0C4'	'0C4'	'086'	'\0C5'	'0C5'	'087'
'\0C6'	'0C6'	'088'	'\0C7'	'0C7'	'089'
'\0C8'	'0C8'	'08A'	'\0C9'	'0C9'	'08B'
'\0CA'	'0CA'	'08C'	'\0CB'	'0CB'	'08D'
'\0CC'	'0CC'	'08E'	'\0CD'	'0CD'	'08F'
'\0CE'	'0CE'	'090'	'\0CF'	'0CF'	'091'
'\0D0'	'0D0'	'092'	'\0D1'	'0D1'	'093'
'\0D2'	'0D2'	'094'	'\0D3'	'0D3'	'095'
'\0D4'	'0D4'	'096'	'\0D5'	'0D5'	'097'
'\0D6'	'0D6'	'098'	'\0D7'	'0D7'	'099'
'\0D8'	'0D8'	'09A'	'\0D9'	'0D9'	'09B'
'\0DA'	'0DA'	'09C'	'\0DB'	'0DB'	'09D'
'\0DC'	'0DC'	'09E'	'\0DD'	'0DD'	'09F'
'\0DE'	'0DE'	'0A0'	'\0EF'	'0EF'	'0A1'
'\0F0'	'0F0'	'0A2'	'\0F1'	'0F1'	'0A3'
'\0F2'	'0F2'	'0A4'	'\0F3'	'0F3'	'0A5'
'\0F4'	'0F4'	'0A6'	'\0F5'	'0F5'	'0A7'
'\0F6'	'0F6'	'0A8'	'\0F7'	'0F7'	'0A9'
'\0F8'	'0F8'	'0AA'	'\0F9'	'0F9'	'0AB'
'\0FA'	'0FA'	'0AC'	'\0FB'	'0FB'	'0AD'
'\0FC'	'0FC'	'0AE'	'\0FD'	'0FD'	'0BD'
'\0FE'	'0FE'	'0C0'	'\0FF'	'0FF'	'0C1'

- `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 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



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



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

User Input

Covered in detail in Lab 2:

```
→ 1 mess = input('Please enter a message: ')
  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**
- Guests: Internships & Clubs

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[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)

Lecture Slip

LECTURE 2, CSci 127
SPRING 2022

Name:								
EmpID:								

- **Introducing Design Challenges:** these are "think up an Algorithm"-type exercises. We introduce a topic in lecture, and then we ask you to apply it to solve a problem. Here we are asking you to come up with a sequence of steps (short English sentences) that describe the **process** – i.e. your Algorithm.
You should also identify the **input** and **output**.

1. Design a program that **counts** the number of plural nouns provided as a string containing only the nouns separated by spaces. Think about what the input is, what the output is, and how you can determine if a noun is plural.
Note: To simplify the problem, assume all plural nouns end in "s".

Input:

Output:

Process:

Today's Topics



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

Guest Speakers

- Hunter staff
 - ▶ Elise Harris, Internship Manager, Cooperman Business Center and Computer Science
- Club officers
 - ▶ Asad Rafique, Hunter Association of Computing Machinery (ACM)
 - ▶ David Arcos Mawyn, Esports and Game Design Collective (EGD)
 - ▶ Kelly Camacho, Women in Computer Science (WiCS)
 - ▶ Isabel Abonitalla, Google Developers Student Club (DSC)
- See Announcement on Blackboard for links to important resources.

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

```
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

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()`

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

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Practice Quiz & Final Questions



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

Practice Quiz & Final Questions



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

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

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

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