

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From email

Frequently Asked Questions

From email

- I am not sure how to submit the Lab.

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

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

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

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

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

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

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

- **When is the midterm?**

Frequently Asked Questions

From email

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

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

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

- **Can I work ahead?**

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

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

- **When is the midterm?**

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

Seat Number

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

Seat Number

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

Seat Number

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

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

Seat Number

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

Today's Topics



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

Today's Topics



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

Group Work

Some review and some novel challenges:

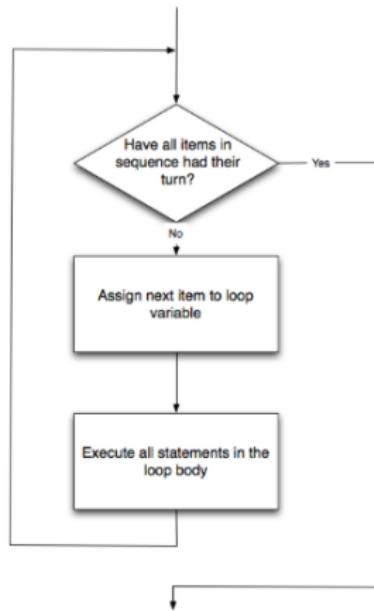
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

Python Tutor

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

(Demo with pythonTutor)

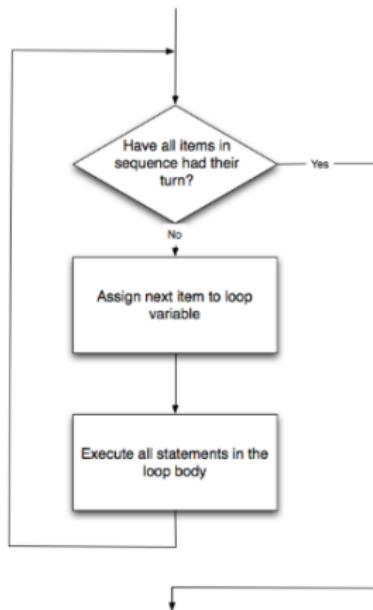
for-loop



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

How to Think Like CS, §4.5

for-loop



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

where `list` is a list of items:

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

How to Think Like CS, §4.5

Today's Topics



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

More on range():

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

Python Tutor

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

(Demo with pythonTutor)

range()

Simplest version:

- `range(stop)`



range()



Simplest version:

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

range()



Simplest version:

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

range()



Simplest version:

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

range()



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

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

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

In summary: range()



The three versions:

In summary: range()



The three versions:

- range(stop)

In summary: range()



The three versions:

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

In summary: range()



The three versions:

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

Today's Topics



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

Variables

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



Variables

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



Variables

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



Variables



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

Variables



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

Variables



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

Variables



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

Variable Names

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



Variable Names

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



Variable Names



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

Variable Names



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

Variable Names



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

Today's Topics



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

Standardized Code for Characters

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

Standardized Code for Characters

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

Standardized Code for Characters

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

ASCII TABLE

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

(wiki)



Converting from Character to Code:

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

ASCII TABLE

Converting from Character to Code:

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

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

ASCII TABLE														
Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char	Decimal	Hex	Octal	Name	Char
0	00	0	NULL	\0	32	20	40	SIGKILL	\000	64	40	100	SIGPOLL	\001
1	01	1	SOH	\001	33	21	41	SIGALRM	\002	65	41	101	SIGSTOP	\003
2	02	2	STX	\002	34	22	42	SIGPOLL	\004	66	42	102	SIGCONT	\005
3	03	3	ETX	\003	35	23	43	SIGPOLL	\006	67	43	103	SIGKILL	\007
4	04	4	EOT	\004	36	24	44	SIGPOLL	\008	68	44	104	SIGPOLL	\009
5	05	5	ENQ	\005	37	25	45	SIGPOLL	\010	69	45	105	SIGPOLL	\011
6	06	6	ACK	\006	38	26	46	SIGPOLL	\012	70	46	106	SIGPOLL	\013
7	07	7	NAK	\007	39	27	47	SIGPOLL	\014	71	47	107	SIGPOLL	\015
8	08	10	SYN	\008	40	28	48	SIGPOLL	\016	72	48	108	SIGPOLL	\017
9	09	11	BT	\009	41	29	49	SIGPOLL	\018	73	49	109	SIGPOLL	\019
10	0A	12	HT	\00A	42	2A	4A	SIGPOLL	\01A	74	4A	110	SIGPOLL	\01B
11	0B	13	LF	\00B	43	2B	4B	SIGPOLL	\01B	75	4B	111	SIGPOLL	\01C
12	0C	14	VT	\00C	44	2C	4C	SIGPOLL	\01C	76	4C	112	SIGPOLL	\01D
13	0D	15	FF	\00D	45	2D	4D	SIGPOLL	\01D	77	4D	113	SIGPOLL	\01E
14	0E	16	CR	\00E	46	2E	4E	SIGPOLL	\01E	78	4E	114	SIGPOLL	\01F
15	0F	17	SO	\00F	47	2F	4F	SIGPOLL	\01F	79	4F	115	SIGPOLL	\01F
16	10	20	SI	\010	48	30	50	SIGPOLL	\01F	80	50	116	SIGPOLL	\01F
17	11	21	DC1	\011	49	31	51	SIGPOLL	\01F	81	51	117	SIGPOLL	\01F
18	12	22	DC2	\012	50	32	52	SIGPOLL	\01F	82	52	118	SIGPOLL	\01F
19	13	23	DC3	\013	51	33	53	SIGPOLL	\01F	83	53	119	SIGPOLL	\01F
20	14	24	DC4	\014	52	34	54	SIGPOLL	\01F	84	54	120	SIGPOLL	\01F
21	15	25	NAK	\015	53	35	55	SIGPOLL	\01F	85	55	121	SIGPOLL	\01F
22	16	26	SYN	\016	54	36	56	SIGPOLL	\01F	86	56	122	SIGPOLL	\01F
23	17	27	ETX	\017	55	37	57	SIGPOLL	\01F	87	57	123	SIGPOLL	\01F
24	18	28	ENQ	\018	56	38	58	SIGPOLL	\01F	88	58	124	SIGPOLL	\01F
25	19	29	ACK	\019	57	39	59	SIGPOLL	\01F	89	59	125	SIGPOLL	\01F
26	1A	2A	BT	\01A	58	3A	5A	SIGPOLL	\01F	90	5A	126	SIGPOLL	\01F
27	1B	2B	HT	\01B	59	3B	5B	SIGPOLL	\01F	91	5B	127	SIGPOLL	\01F
28	1C	2C	LF	\01C	60	3C	5C	SIGPOLL	\01F	92	5C	128	SIGPOLL	\01F
29	1D	2D	VT	\01D	61	3D	5D	SIGPOLL	\01F	93	5D	129	SIGPOLL	\01F
30	1E	2E	FF	\01E	62	3E	5E	SIGPOLL	\01F	94	5E	130	SIGPOLL	\01F
31	1F	2F	CR	\01F	63	3F	5F	SIGPOLL	\01F	95	5F	131	SIGPOLL	\01F
32	20	30	SO	\020	64	40	60	SIGPOLL	\01F	96	60	132	SIGPOLL	\01F
33	21	31	SI	\021	65	41	61	SIGPOLL	\01F	97	61	133	SIGPOLL	\01F
34	22	32	DC1	\022	66	42	62	SIGPOLL	\01F	98	62	134	SIGPOLL	\01F
35	23	33	DC2	\023	67	43	63	SIGPOLL	\01F	99	63	135	SIGPOLL	\01F
36	24	34	DC3	\024	68	44	64	SIGPOLL	\01F	100	64	136	SIGPOLL	\01F
37	25	35	DC4	\025	69	45	65	SIGPOLL	\01F	101	65	137	SIGPOLL	\01F
38	26	36	NAK	\026	70	46	66	SIGPOLL	\01F	102	66	138	SIGPOLL	\01F
39	27	37	SYN	\027	71	47	67	SIGPOLL	\01F	103	67	139	SIGPOLL	\01F
40	28	38	ETX	\028	72	48	68	SIGPOLL	\01F	104	68	140	SIGPOLL	\01F
41	29	39	ENQ	\029	73	49	69	SIGPOLL	\01F	105	69	141	SIGPOLL	\01F
42	2A	3A	ACK	\02A	74	4A	6A	SIGPOLL	\01F	106	6A	142	SIGPOLL	\01F
43	2B	3B	BT	\02B	75	4B	6B	SIGPOLL	\01F	107	6B	143	SIGPOLL	\01F
44	2C	3C	HT	\02C	76	4C	6C	SIGPOLL	\01F	108	6C	144	SIGPOLL	\01F
45	2D	3D	LF	\02D	77	4D	6D	SIGPOLL	\01F	109	6D	145	SIGPOLL	\01F
46	2E	3E	VT	\02E	78	4E	6E	SIGPOLL	\01F	110	6E	146	SIGPOLL	\01F
47	2F	3F	FF	\02F	79	4F	6F	SIGPOLL	\01F	111	6F	147	SIGPOLL	\01F
48	30	40	CR	\030	80	50	70	SIGPOLL	\01F	112	70	148	SIGPOLL	\01F
49	31	41	SO	\031	81	51	71	SIGPOLL	\01F	113	71	149	SIGPOLL	\01F
50	32	42	SI	\032	82	52	72	SIGPOLL	\01F	114	72	150	SIGPOLL	\01F
51	33	43	DC1	\033	83	53	73	SIGPOLL	\01F	115	73	151	SIGPOLL	\01F
52	34	44	DC2	\034	84	54	74	SIGPOLL	\01F	116	74	152	SIGPOLL	\01F
53	35	45	DC3	\035	85	55	75	SIGPOLL	\01F	117	75	153	SIGPOLL	\01F
54	36	46	DC4	\036	86	56	76	SIGPOLL	\01F	118	76	154	SIGPOLL	\01F
55	37	47	NAK	\037	87	57	77	SIGPOLL	\01F	119	77	155	SIGPOLL	\01F
56	38	48	SYN	\038	88	58	78	SIGPOLL	\01F	120	78	156	SIGPOLL	\01F
57	39	49	ETX	\039	89	59	79	SIGPOLL	\01F	121	79	157	SIGPOLL	\01F
58	3A	4A	ENQ	\03A	90	5A	7A	SIGPOLL	\01F	122	7A	158	SIGPOLL	\01F
59	3B	4B	ACK	\03B	91	5B	7B	SIGPOLL	\01F	123	7B	159	SIGPOLL	\01F
60	3C	4C	BT	\03C	92	5C	7C	SIGPOLL	\01F	124	7C	160	SIGPOLL	\01F
61	3D	4D	HT	\03D	93	5D	7D	SIGPOLL	\01F	125	7D	161	SIGPOLL	\01F
62	3E	4E	LF	\03E	94	5E	7E	SIGPOLL	\01F	126	7E	162	SIGPOLL	\01F
63	3F	4F	VT	\03F	95	5F	7F	SIGPOLL	\01F	127	7F	163	SIGPOLL	\01F
64	40	50	FF	\040	96	60	80	SIGPOLL	\01F	128	80	164	SIGPOLL	\01F
65	41	51	CR	\041	97	61	81	SIGPOLL	\01F	129	81	165	SIGPOLL	\01F
66	42	52	SO	\042	98	62	82	SIGPOLL	\01F	130	82	166	SIGPOLL	\01F
67	43	53	SI	\043	99	63	83	SIGPOLL	\01F	131	83	167	SIGPOLL	\01F
68	44	54	DC1	\044	100	64	84	SIGPOLL	\01F	132	84	168	SIGPOLL	\01F
69	45	55	DC2	\045	101	65	85	SIGPOLL	\01F	133	85	169	SIGPOLL	\01F
70	46	56	DC3	\046	102	66	86	SIGPOLL	\01F	134	86	170	SIGPOLL	\01F
71	47	57	DC4	\047	103	67	87	SIGPOLL	\01F	135	87	171	SIGPOLL	\01F
72	48	58	NAK	\048	104	68	88	SIGPOLL	\01F	136	88	172	SIGPOLL	\01F
73	49	59	SYN	\049	105	69	89	SIGPOLL	\01F	137	89	173	SIGPOLL	\01F
74	4A	5A	ETX	\04A	106	6A	8A	SIGPOLL	\01F	138	8A	174	SIGPOLL	\01F
75	4B	5B	ENQ	\04B	107	6B	8B	SIGPOLL	\01F	139	8B	175	SIGPOLL	\01F
76	4C	5C	ACK	\04C	108	6C	8C	SIGPOLL	\01F	140	8C	176	SIGPOLL	\01F
77	4D	5D	BT	\04D	109	6D	8D	SIGPOLL	\01F	141	8D	177	SIGPOLL	\01F
78	4E	5E	HT	\04E	110	6E	8E	SIGPOLL	\01F	142	8E	178	SIGPOLL	\01F
79	4F	5F	LF	\04F	111	6F	8F	SIGPOLL	\01F	143	8F	179	SIGPOLL	\01F
80	50	60	VT	\050	112	70	90	SIGPOLL	\01F	144	90	180	SIGPOLL	\01F
81	51	61	FF	\051	113	71	91	SIGPOLL	\01F	145	91	181	SIGPOLL	\01F
82	52	62	CR	\052	114	72	92	SIGPOLL	\01F	146	92	182	SIGPOLL	\01F
83	53	63	SO	\053	115	73	93	SIGPOLL	\01F	147	93	183	SIGPOLL	\01F
84	54	64	SI	\054	116	74	94	SIGPOLL	\01F	148	94	184	SIGPOLL	\01F
85	55	65	DC1	\055	117	75	95	SIGPOLL	\01F	149	95	185	SIGPOLL	\01F
86	56	66	DC2	\056	118	76	96	SIGPOLL	\01F	150	96	186	SIGPOLL	\01F
87	57	67	DC3	\057	119	77	97	SIGPOLL	\01F	151	97	187	SIGPOLL	\01F
88	58	68	DC4	\058	120	78	98	SIGPOLL	\01F	152	98	188	SIGPOLL	\01F
89	59	69	NAK	\059	121	79	99	SIGPOLL	\01F	153	99	189	SIGPOLL	\01F
90	5A	6A	SYN	\05A	122	7A	9A	SIGPOLL	\01F	154	9A	190	SIGPOLL	\01F
91	5B	6B	ETX	\05B	123	7B	9B	SIGPOLL	\01F	155	9B	191	SIGPOLL	\01F
92	5C	6C	ENQ	\05C	124	7C	9C	SIGPOLL	\01F	156	9C	192	SIGPOLL	\01F
93	5D	6D	ACK	\05D	125	7D	9D	SIGPOLL	\01F	157	9D	193	SIGPOLL	\01F
94	5E	6E	BT	\05E	126	7E	9E	SIGPOLL	\01F	158	9E	194	SIGPOLL	\01F
95	5F	6F	HT	\05F	127	7F	9F	SIGPOLL	\01F	159	9F	195	SIGPOLL	\01F
96	60	70	LF	\060	128	80	A0	SIGPOLL	\01F	160	A0	196	SIGPOLL	\01F
97	61	71	VT	\061	129	81	A1	SIGPOLL	\01F	161	A1	197	SIGPOLL	\01F
98	62	72	FF	\062	130	82	A2	SIGPOLL	\01F	162	A2	198	SIGPOLL	\01F
99	63	73	CR	\063	131	83	A3	SIGPOLL	\01F	163	A3	199	SIGPOLL	\01F
100	64	74	SO	\064	132	84	A4	SIGPOLL	\01F	164	A4	200	SIGPOLL	\01F
101	65	75	SI	\065	133	85	A5	SIGPOLL	\01F	165	A5	201	SIGPOLL	\01F
102	66	76	DC1	\066	134	86	A6	SIGPOLL	\01F	166	A6	202	SIGPOLL	\01F
103	67	77	DC2	\067	135	87	A7	SIGPOLL						

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	“			
38	26	\u0026	64	46	”			
39	27	\u0027	65	47	‘			
40	28	\u0028	66	48	’			
41	29	\u0029	67	49	“			
42	2A	\u002A	68	4A	”			
43	2B	\u002B	69	4B	‘			
44	2C	\u002C	70	4C	’			
45	2D	\u002D	71	4D	“			
46	2E	\u002E	72	4E	”			
47	2F	\u002F	73	4F	‘			
48	30	\u0030	74	50	”			
49	31	\u0031	75	51	‘			
50	32	\u0032	76	52	’			
51	33	\u0033	77	53	“			
52	34	\u0034	78	54	”			
53	35	\u0035	79	55	‘			
54	36	\u0036	80	56	’			
55	37	\u0037						
56	38	\u0038						
57	39	\u0039						
58	3A	\u003A						
59	3B	\u003B						
60	3C	\u003C						
61	3D	\u003D						
62	3E	\u003E						
63	3F	\u003F						
64	40	\u0040						
65	41	\u0041						
66	42	\u0042						
67	43	\u0043						
68	44	\u0044						
69	45	\u0045						
70	46	\u0046						
71	47	\u0047						
72	48	\u0048						
73	49	\u0049						
74	4A	\u004A						
75	4B	\u004B						
76	4C	\u004C						
77	4D	\u004D						
78	4E	\u004E						
79	4F	\u004F						
80	50	\u0050						
81	51	\u0051						
82	52	\u0052						
83	53	\u0053						
84	54	\u0054						
85	55	\u0055						
86	56	\u0056						
87	57	\u0057						
88	58	\u0058						
89	59	\u0059						
90	5A	\u005A						
91	5B	\u005B						
92	5C	\u005C						
93	5D	\u005D						
94	5E	\u005E						
95	5F	\u005F						
96	60	\u0060						
97	61	\u0061						
98	62	\u0062						
99	63	\u0063						
100	64	\u0064						

- `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	\f	74	4A	\f
11	0B	\11	43	2B	\r	75	4B	\r
12	0C	\12	44	2C	\v	76	4C	\v
13	0D	\13	45	2D	\b	77	4D	\b
14	0E	\14	46	2E	\t	78	4E	\t
15	0F	\15	47	2F	\n	79	4F	\n
16	10	\16	48	30	\t	80	50	\t
17	11	\17	49	31	\n	81	51	\n
18	12	\18	4A	32	\f	82	52	\f
19	13	\19	4B	33	\r	83	53	\r
20	14	\20	4C	34	\v	84	54	\v
21	15	\21	4D	35	\b	85	55	\b
22	16	\22	4E	36	\t	86	56	\t
23	17	\23	4F	37	\n	87	57	\n
24	18	\24	50	38	\t	88	58	\t
25	19	\25	51	39	\n	89	59	\n
26	1A	\26	52	3A	\f	90	5A	\f
27	1B	\27	53	3B	\r	91	5B	\r
28	1C	\28	54	3C	\v	92	5C	\v
29	1D	\29	55	3D	\b	93	5D	\b
30	1E	\20	56	3E	\t	94	5E	\t
31	1F	\21	57	3F	\n	95	5F	\n
32	20	\22	58	40	\t	96	60	\t
33	21	\23	59	41	\n	97	61	A
34	22	\24	5A	42	\f	98	62	B
35	23	\25	5B	43	\r	99	63	F
36	24	\26	5C	44	\v	100	64	\v
37	25	\27	5D	45	\b	101	65	\b
38	26	\28	5E	46	\t	102	66	\t
39	27	\29	5F	47	\n	103	67	\n
40	28	\20	60	48	\t	104	68	\t
41	29	\21	61	49	\n	105	69	\n
42	2A	\22	62	4A	\f	106	6A	\f
43	2B	\23	63	4B	\r	107	6B	\r
44	2C	\24	64	4C	\v	108	6C	\v
45	2D	\25	65	4D	\b	109	6D	\b
46	2E	\26	66	4E	\t	110	6E	\t
47	2F	\27	67	4F	\n	111	6F	\n
48	30	\28	68	50	\t	112	70	\t
49	31	\29	69	51	\n	113	71	\n
50	32	\20	6A	52	\f	114	72	\f
51	33	\21	6B	53	\r	115	73	\r
52	34	\22	6C	54	\v	116	74	\v
53	35	\23	6D	55	\b	117	75	\b
54	36	\24	6E	56	\t	118	76	\t
55	37	\25	6F	57	\n	119	77	\n
56	38	\26	70	58	\t	120	78	\t
57	39	\27	71	59	\n	121	79	\n
58	3A	\28	72	5A	\f	122	7A	\f
59	3B	\29	73	5B	\r	123	7B	\r
60	3C	\20	74	5C	\v	124	7C	\v
61	3D	\21	75	5D	\b	125	7D	\b
62	3E	\22	76	5E	\t	126	7E	\t
63	3F	\23	77	5F	\n	127	7F	\n

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

Converting from Character to Code:

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

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00	\0	32	20	\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	C
4	04	\4	36	24	\n	68	44	D
5	05	\5	37	25	\r	69	45	E
6	06	\6	38	26	\v	70	46	F
7	07	\7	39	27	\b7	71	47	G
8	08	\8	40	28	\b8	72	48	H
9	09	\9	41	29	\b9	73	49	I
10	0A	\10	42	2A	\bA	74	4A	J
11	0B	\11	43	2B	\bB	75	4B	K
12	0C	\12	44	2C	\bC	76	4C	L
13	0D	\13	45	2D	\bD	77	4D	M
14	0E	\14	46	2E	\bE	78	4E	N
15	0F	\15	47	2F	\bF	79	4F	O
16	10	\16	48	30	\b0	80	50	P
17	11	\17	49	31	\b1	81	51	Q
18	12	\18	4A	32	\b2	82	52	R
19	13	\19	4B	33	\b3	83	53	S
20	14	\20	4C	34	\b4	84	54	T
21	15	\21	4D	35	\b5	85	55	U
22	16	\22	4E	36	\b6	86	56	V
23	17	\23	4F	37	\b7	87	57	W
24	18	\24	50	38	\b8	88	58	X
25	19	\25	51	39	\b9	89	59	Y
26	1A	\26	52	3A	\bA	90	5A	Z
27	1B	\27	53	3B	\bB	91	5B	\b7
28	1C	\28	54	3C	\bC	92	5C	\b8
29	1D	\29	55	3D	\bD	93	5D	\b9
30	1E	\2A	56	3E	\bA	94	5E	\bA
31	1F	\2B	57	3F	\bB	95	5F	\bB
32	20	\2C	58	40	\bC	96	60	\b7
33	21	\2D	59	41	\bD	97	61	\b8
34	22	\2E	5A	42	\bE	98	62	\b9
35	23	\2F	5B	43	\bF	99	63	\bA
36	24	\30	5C	44	\b0	100	64	\bB
37	25	\31	5D	45	\b1	101	65	\bC
38	26	\32	5E	46	\b2	102	66	\bD
39	27	\33	5F	47	\b3	103	67	\bE
40	28	\34	60	48	\b4	104	68	\bF
41	29	\35	61	49	\b5	105	69	\b0
42	2A	\36	62	4A	\b6	106	6A	\b1
43	2B	\37	63	4B	\b7	107	6B	\b2
44	2C	\38	64	4C	\b8	108	6C	\b3
45	2D	\39	65	4D	\b9	109	6D	\b4
46	2E	\3A	66	4E	\bA	110	6E	\b5
47	2F	\3B	67	4F	\bB	111	6F	\b6
48	30	\3C	68	50	\bC	112	70	\b7
49	31	\3D	69	51	\bD	113	71	\b8
50	32	\3E	6A	52	\bE	114	72	\b9
51	33	\3F	6B	53	\bF	115	73	\bA
52	34	\30	6C	54	\b0	116	74	\bB
53	35	\31	6D	55	\b1	117	75	\bC
54	36	\32	6E	56	\b2	118	76	\bD
55	37	\33	6F	57	\b3	119	77	\bE
56	38	\34	70	58	\b4	120	78	\bF
57	39	\35	71	59	\b5	121	79	\b0
58	3A	\36	72	5A	\b6	122	7A	\b1
59	3B	\37	73	5B	\b7	123	7B	\b2
60	3C	\38	74	5C	\b8	124	7C	\b3
61	3D	\39	75	5D	\b9	125	7D	\b4
62	3E	\3A	76	5E	\bA	126	7E	\b5
63	3F	\3B	77	5F	\bB	127	7F	\b6
64	40	\3C	78	60	\bC	128	80	\b7
65	41	\3D	79	61	\bD	129	81	\b8
66	42	\3E	7A	62	\bE	130	82	\b9
67	43	\3F	7B	63	\bF	131	83	\bA
68	44	\30	7C	64	\b0	132	84	\bB
69	45	\31	7D	65	\b1	133	85	\bC
70	46	\32	7E	66	\b2	134	86	\bD
71	47	\33	7F	67	\b3	135	87	\bE
72	48	\34	80	68	\b4	136	88	\bF
73	49	\35	81	69	\b5	137	89	\b0
74	4A	\36	82	6A	\b6	138	8A	\b1
75	4B	\37	83	6B	\b7	139	8B	\b2
76	4C	\38	84	6C	\b8	140	8C	\b3
77	4D	\39	85	6D	\b9	141	8D	\b4
78	4E	\3A	86	6E	\bA	142	8E	\b5
79	4F	\3B	87	6F	\bB	143	8F	\b6
80	50	\3C	88	70	\bC	144	90	\b7
81	51	\3D	89	71	\bD	145	91	\b8
82	52	\3E	8A	72	\bE	146	92	\b9
83	53	\3F	8B	73	\bF	147	93	\bA
84	54	\30	8C	74	\b0	148	94	\bB
85	55	\31	8D	75	\b1	149	95	\bC
86	56	\32	8E	76	\b2	150	96	\bD
87	57	\33	8F	77	\b3	151	97	\bE
88	58	\34	90	78	\b4	152	98	\bF
89	59	\35	91	79	\b5	153	99	\b0
90	5A	\36	92	7A	\b6	154	100	\b1
91	5B	\37	93	7B	\b7	155	101	\b2
92	5C	\38	94	7C	\b8	156	102	\b3
93	5D	\39	95	7D	\b9	157	103	\b4
94	5E	\3A	96	7E	\bA	158	104	\b5
95	5F	\3B	97	7F	\bB	159	105	\b6
96	60	\3C	98	80	\bC	160	106	\b7
97	61	\3D	99	81	\bD	161	107	\b8
98	62	\3E	9A	82	\bE	162	108	\b9
99	63	\3F	9B	83	\bF	163	109	\bA
100	64	\30	9C	84	\b0	164	110	\bB
101	65	\31	9D	85	\b1	165	111	\bC
102	66	\32	9E	86	\b2	166	112	\bD
103	67	\33	9F	87	\b3	167	113	\bE
104	68	\34	90	88	\b4	168	114	\bF
105	69	\35	91	89	\b5	169	115	\b0
106	6A	\36	92	90	\b6	170	116	\b1
107	6B	\37	93	91	\b7	171	117	\b2
108	6C	\38	94	92	\b8	172	118	\b3
109	6D	\39	95	93	\b9	173	119	\b4
110	6E	\3A	96	94	\bA	174	120	\b5
111	6F	\3B	97	95	\bB	175	121	\b6
112	70	\3C	98	96	\bC	176	122	\b7
113	71	\3D	99	97	\bD	177	123	\b8
114	72	\3E	9A	98	\bE	178	124	\b9
115	73	\3F	9B	99	\bF	179	125	\bA
116	74	\30	9C	100	\b0	180	126	\bB
117	75	\31	9D	101	\b1	181	127	\bC
118	76	\32	9E	102	\b2	182	128	\bD
119	77	\33	9F	103	\b3	183	129	\bE
120	78	\34	90	104	\b4	184	130	\bF
121	79	\35	91	105	\b5	185	131	\b0
122	7A	\36	92	106	\b6	186	132	\b1
123	7B	\37	93	107	\b7	187	133	\b2
124	7C	\38	94	108	\b8	188	134	\b3
125	7D	\39	95	109	\b9	189	135	\b4
126	7E	\3A	96	110	\bA	190	136	\b5
127	7F	\3B	97	111	\bB	191	137	\b6
128	80	\3C	98	112	\bC	192	138	\b7
129	81	\3D	99	113	\bD	193	139	\b8
130	82	\3E	9A	114	\bE	194	140	\b9
131	83	\3F	9B	115	\bF	195	141	\bA
132	84	\30	9C	116	\b0	196	142	\bB
133	85	\31	9D	117	\b1	197	143	\bC
134	86	\32	9E	118	\b2	198	144	\bD
135	87	\33	9F	119	\b3	199	145	\bE
136	88	\34	90	120	\b4	200	146	\bF
137	89	\35	91	121	\b5	201	147	\b0
138	8A	\36	92	122	\b6	202	148	\b1
139	8B	\37	93	123	\b7	203	149	\b2
140	8C	\38	94	124	\b8	204	150	\b3
141	8D	\39	95	125	\b9	205	151	\b4
142	8E	\3A	96	126	\bA	206	152	\b5
143	8F	\3B	97	127	\bB	207	153	\b6
144	90	\3C	98	128	\bC	208	154	\b7
145	91	\3D	99	129	\bD	209	155	\b8
146	92	\3E	9A	130	\bE	210	156	\b9
147	93	\3F	9B	131	\bF	211	157	\bA
148	94	\30	9C	132	\b0	212	158	\bB
149	95	\31	9D	133	\b1	213	159	\bC
150	96	\32	9E	134	\b2	214	160	\bD
151	97	\33	9F	135	\b3	215	161	\bE
152	98	\34	90	136	\b4	216	162	\bF
153	99	\35	91	137	\b5	217	163	\b0
154	9A	\36	92	138	\b6	218	164	\b1
155	9B	\37	93	139	\b7	219	165	\b2
156	9C	\38	94	140	\b8	220	166	\b3
157	9D	\39	95	141	\b9	221	167	\b4
158	9E	\3A	96	142	\bA	222	168	\b5
159	9F	\3B	97	143	\bB	223	169	\b6
160	90	\3C	98	144	\bC	224	170	\b7
161	91	\3D	99	145	\bD	225	171	\b8
162	92	\3E	9A	146	\bE	226	172	\b9
163	93	\3F	9B	147	\bF	227	173	\bA
164	94	\30	9C	148	\b0	228	174	\bB
165	95	\31	9D	149	\b1	229	175	\bC
166	96	\32	9E	150	\b2	230	176	\bD
167	97	\33	9F	151	\b3	231	177	\bE
168	98	\34	90	152	\b4	232	178	\bF
169	99	\35	91	153	\b5	233	179	\b0
170	9A	\36	92	154	\b6	234	180	\b1
171	9B	\37	93	155	\b7	235	181	\b2
172	9C	\38	94	156	\b8	236	182	\b3
173	9D	\39	95	157	\b9	237	183	\b4
174								

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'
'\070'	'070'	'03C'	'\071'	'071'	'03D'
'\072'	'072'	'03E'	'\073'	'073'	'03F'
'\074'	'074'	'040'	'\075'	'075'	'041'
'\077'	'077'	'042'	'\078'	'078'	'043'
'\080'	'080'	'044'	'\081'	'081'	'045'
'\082'	'082'	'046'	'\083'	'083'	'047'
'\084'	'084'	'048'	'\085'	'085'	'049'
'\086'	'086'	'04A'	'\087'	'087'	'04B'
'\088'	'088'	'04C'	'\089'	'089'	'04D'
'\090'	'090'	'04E'	'\091'	'091'	'04F'
'\092'	'092'	'050'	'\093'	'093'	'051'
'\094'	'094'	'052'	'\095'	'095'	'053'
'\096'	'096'	'054'	'\097'	'097'	'055'
'\098'	'098'	'056'	'\099'	'099'	'057'
'\0A0'	'0A0'	'058'	'\0A1'	'0A1'	'059'
'\0A2'	'0A2'	'05A'	'\0A3'	'0A3'	'05B'
'\0A4'	'0A4'	'05C'	'\0A5'	'0A5'	'05D'
'\0A6'	'0A6'	'05E'	'\0A7'	'0A7'	'05F'
'\0A8'	'0A8'	'060'	'\0A9'	'0A9'	'061'
'\0AA'	'0AA'	'062'	'\0AB'	'0AB'	'063'
'\0AC'	'0AC'	'064'	'\0AD'	'0AD'	'065'
'\0AE'	'0AE'	'066'	'\0AF'	'0AF'	'067'
'\0B0'	'0B0'	'068'	'\0B1'	'0B1'	'069'
'\0B2'	'0B2'	'06A'	'\0B3'	'0B3'	'06B'
'\0B4'	'0B4'	'06C'	'\0B5'	'0B5'	'06D'
'\0B6'	'0B6'	'06E'	'\0B7'	'0B7'	'06F'
'\0B8'	'0B8'	'070'	'\0B9'	'0B9'	'071'
'\0BA'	'0BA'	'072'	'\0BB'	'0BB'	'073'
'\0BC'	'0BC'	'074'	'\0BD'	'0BD'	'075'
'\0BE'	'0BE'	'076'	'\0BF'	'0BF'	'077'
'\0C0'	'0C0'	'078'	'\0C1'	'0C1'	'079'
'\0C2'	'0C2'	'07A'	'\0C3'	'0C3'	'07B'
'\0C4'	'0C4'	'07C'	'\0C5'	'0C5'	'07D'
'\0C6'	'0C6'	'07E'	'\0C7'	'0C7'	'07F'
'\0C8'	'0C8'	'080'	'\0C9'	'0C9'	'081'
'\0CA'	'0CA'	'082'	'\0CB'	'0CB'	'083'
'\0CC'	'0CC'	'084'	'\0CD'	'0CD'	'085'
'\0CE'	'0CE'	'086'	'\0CF'	'0CF'	'087'
'\0D0'	'0D0'	'088'	'\0D1'	'0D1'	'089'
'\0D2'	'0D2'	'08A'	'\0D3'	'0D3'	'08B'
'\0D4'	'0D4'	'08C'	'\0D5'	'0D5'	'08D'
'\0D6'	'0D6'	'08E'	'\0D7'	'0D7'	'08F'
'\0D8'	'0D8'	'090'	'\0D9'	'0D9'	'091'
'\0DA'	'0DA'	'092'	'\0DB'	'0DB'	'093'
'\0DC'	'0DC'	'094'	'\0DD'	'0DD'	'095'
'\0DE'	'0DE'	'096'	'\0DF'	'0DF'	'097'
'\0E0'	'0E0'	'098'	'\0E1'	'0E1'	'099'
'\0E2'	'0E2'	'09A'	'\0E3'	'0E3'	'09B'
'\0E4'	'0E4'	'09C'	'\0E5'	'0E5'	'09D'
'\0E6'	'0E6'	'09E'	'\0E7'	'0E7'	'09F'
'\0E8'	'0E8'	'0A0'	'\0E9'	'0E9'	'0A1'
'\0EA'	'0EA'	'0A2'	'\0EB'	'0EB'	'0A3'
'\0EC'	'0EC'	'0A4'	'\0ED'	'0ED'	'0A5'
'\0EE'	'0EE'	'0A6'	'\0EF'	'0EF'	'0A7'
'\0F0'	'0F0'	'0A8'	'\0F1'	'0F1'	'0A9'
'\0F2'	'0F2'	'0A0'	'\0F3'	'0F3'	'0A1'
'\0F4'	'0F4'	'0A2'	'\0F5'	'0F5'	'0A3'
'\0F6'	'0F6'	'0A4'	'\0F7'	'0F7'	'0A5'
'\0F8'	'0F8'	'0A6'	'\0F9'	'0F9'	'0A7'
'\0FA'	'0FA'	'0A8'	'\0FB'	'0FB'	'0A9'
'\0FC'	'0FC'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FE'	'0FE'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F00'	'0F00'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F02'	'0F02'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F04'	'0F04'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F06'	'0F06'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F08'	'0F08'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F0A'	'0F0A'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F0C'	'0F0C'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F0E'	'0F0E'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F10'	'0F10'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F12'	'0F12'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F14'	'0F14'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F16'	'0F16'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F18'	'0F18'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F1A'	'0F1A'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F1C'	'0F1C'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F1E'	'0F1E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F20'	'0F20'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F22'	'0F22'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F24'	'0F24'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F26'	'0F26'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F28'	'0F28'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F2A'	'0F2A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F2C'	'0F2C'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F2E'	'0F2E'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F30'	'0F30'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F32'	'0F32'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F34'	'0F34'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F36'	'0F36'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F38'	'0F38'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F3A'	'0F3A'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F3C'	'0F3C'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F3E'	'0F3E'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F40'	'0F40'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F42'	'0F42'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F44'	'0F44'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F46'	'0F46'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F48'	'0F48'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F4A'	'0F4A'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F4C'	'0F4C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F4E'	'0F4E'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F50'	'0F50'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F52'	'0F52'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F54'	'0F54'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F56'	'0F56'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F58'	'0F58'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F5A'	'0F5A'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F5C'	'0F5C'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F5E'	'0F5E'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F60'	'0F60'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F62'	'0F62'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F64'	'0F64'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F66'	'0F66'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F68'	'0F68'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F6A'	'0F6A'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F6C'	'0F6C'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F6E'	'0F6E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F70'	'0F70'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F72'	'0F72'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F74'	'0F74'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F76'	'0F76'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F78'	'0F78'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F7A'	'0F7A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F7C'	'0F7C'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F7E'	'0F7E'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F80'	'0F80'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F82'	'0F82'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F84'	'0F84'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F86'	'0F86'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F88'	'0F88'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F8A'	'0F8A'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F8C'	'0F8C'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F8E'	'0F8E'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F90'	'0F90'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F92'	'0F92'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F94'	'0F94'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F96'	'0F96'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F98'	'0F98'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A'	'0F9A'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9C'	'0F9C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9E'	'0F9E'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9F'	'0F9F'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FA0'	'0FA0'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FA2'	'0FA2'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FA4'	'0FA4'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FA6'	'0FA6'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FA8'	'0FA8'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FAA'	'0FAA'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FAC'	'0FAC'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FAD'	'0FAD'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FAC0'	'0FAC0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FAC2'	'0FAC2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FAC4'	'0FAC4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FAC6'	'0FAC6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FAC8'	'0FAC8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA'	'0FACA'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'

In Pairs or Triples...

Some review and some novel challenges:

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



Python Tutor

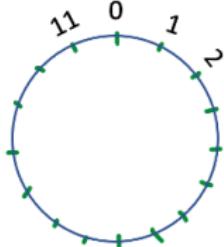
```
1 #Predict what will be printed:  
2  
3 for c in range(65,90):  
4     print(chr(c))  
5  
6 message = "I love Python"  
7 newMessage = ""  
8 for c in message:  
9     print(ord(c)) #Print the Unicode of each number  
10    print(chr(ord(c)+1)) #Print the next character  
11    newMessage = newMessage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

(Demo with pythonTutor)

Wrap

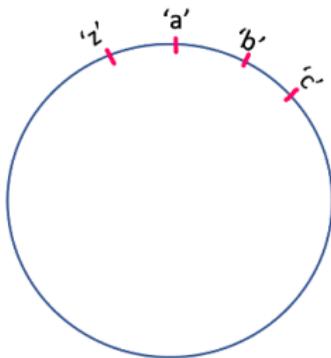
Hints for Programming Assignment 9 in

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



What is 15:00?

What is 17:00?



(1) How many scales in this shape?

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

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

User Input

Covered in detail in Lab 2:

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

(Demo with pythonTutor)

Side Note: '+' for numbers and strings

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



Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Side Note: '+' for numbers and strings



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

Today's Topics



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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: String Methods

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

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

More on Strings: Indexing & Substrings

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

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

More on Strings: Indexing & Substrings

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

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

More on Strings: Indexing & Substrings

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

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

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

More on Strings: Indexing & Substrings

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[0]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- $s[-1]$ is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- $s[-1]$ is ‘s’.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[3:6]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[:3]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[:-1]` is

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

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

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

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

Today's Topics



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

Recap

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:
 - ▶ For-loops

Recap

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

- In Python, we introduced:

- ▶ For-loops
- ▶ range()

Recap

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

- In Python, we introduced:

- ▶ For-loops
- ▶ range()
- ▶ Variables: ints and strings

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Practice Quiz & Final Questions



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

Practice Quiz & Final Questions



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

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Lecture Slips & Writing Boards



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