

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

- This lecture will be recorded

Frequently Asked Questions

From email

Frequently Asked Questions

From email

- **I 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.

Today's Topics



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

In Pairs or Triples...

Some review and some novel challenges:

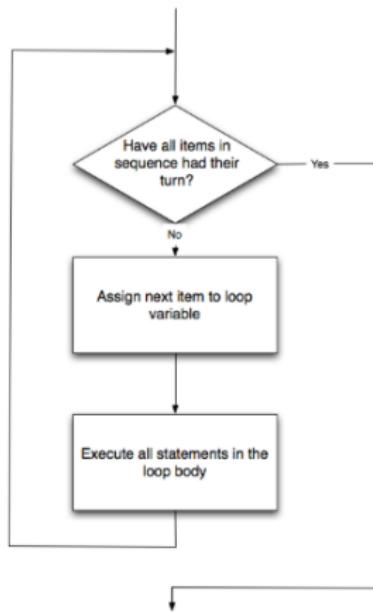
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

Python Tutor

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

(Demo with pythonTutor)

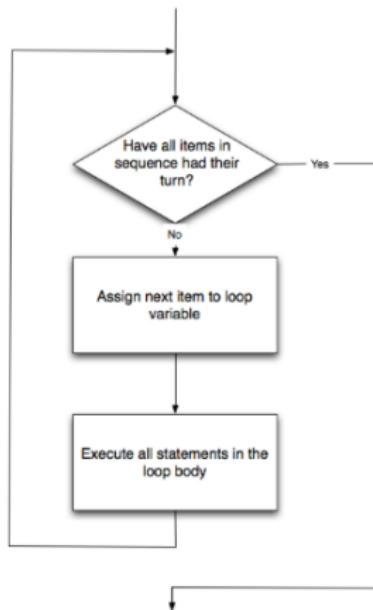
for-loop



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

How to Think Like CS, §4.5

for-loop



```
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, Advising & 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:
 $[start, start+step, start+2*step\dots, last]$
(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()

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
- Guests: Internships, Advising & 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, Advising & 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'.)

ASCII TABLE											
Decimal	Hex	Char	Octal	Decimal	Hex	Char	Octal	Decimal	Hex	Char	Octal
0	00	\0	000	1	01	\1	001	2	02	\2	002
3	03	\3	003	4	04	\4	004	5	05	\5	005
6	06	\6	006	7	07	\7	007	8	08	\8	008
9	09	\9	009	10	0A	\A	010	11	0B	\B	011
12	0C	\C	014	13	0D	\D	015	14	0E	\E	016
15	0F	\F	017	16	10	\10	020	17	11	\11	021
18	12	\12	022	19	13	\13	023	20	14	\14	024
21	16	\16	026	22	17	\17	027	23	18	\18	028
24	1C	\1C	032	25	1D	\1D	033	26	1E	\1E	034
27	1F	\1F	037	28	20	\20	040	29	21	\21	041
30	22	\22	042	31	23	\23	043	32	24	\24	044
33	26	\26	046	34	27	\27	047	35	28	\28	048
36	2C	\2C	052	37	2D	\2D	053	38	2E	\2E	054
39	2F	\2F	057	40	30	\30	060	41	31	\31	061
42	32	\32	062	43	33	\33	063	44	34	\34	064
46	36	\36	066	47	37	\37	067	48	38	\38	070
49	39	\39	071	50	3A	\3A	072	51	3B	\3B	073
52	3C	\3C	074	53	3D	\3D	075	54	3E	\3E	076
56	3F	\3F	077	57	40	\40	080	58	41	\41	081
59	42	\42	082	60	43	\43	083	61	44	\44	084
62	46	\46	086	63	47	\47	087	64	48	\48	088
65	49	\49	089	66	4A	\4A	090	67	4B	\4B	091
68	4C	\4C	092	69	4D	\4D	093	70	4E	\4E	094
71	4F	\4F	095	72	50	\50	096	73	51	\51	097
74	52	\52	098	75	53	\53	099	76	54	\54	100
77	56	\56	101	78	57	\57	102	79	58	\58	103
80	59	\59	104	81	5A	\5A	105	82	5B	\5B	106
83	5C	\5C	108	84	5D	\5D	109	85	5E	\5E	110
86	5F	\5F	111	87	60	\60	112	88	61	\61	113
89	62	\62	114	90	63	\63	115	91	64	\64	116
92	66	\66	118	93	67	\67	119	94	68	\68	120
95	69	\69	121	96	6A	\6A	122	97	6B	\6B	123
98	6C	\6C	125	99	6D	\6D	126	100	6E	\6E	127

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'.)

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	\020
16	10	20	SI	\010	48	30	50	SIGPOLL	\01F	80	50	116	SIGPOLL	\021
17	11	21	DC1	\011	49	31	51	SIGPOLL	\01F	81	51	117	SIGPOLL	\022
18	12	22	DC2	\012	50	32	52	SIGPOLL	\01F	82	52	118	SIGPOLL	\023
19	13	23	DC3	\013	51	33	53	SIGPOLL	\01F	83	53	119	SIGPOLL	\024
20	14	24	DC4	\014	52	34	54	SIGPOLL	\01F	84	54	120	SIGPOLL	\025
21	15	25	NAK	\015	53	35	55	SIGPOLL	\01F	85	55	121	SIGPOLL	\026
22	16	26	SYN	\016	54	36	56	SIGPOLL	\01F	86	56	122	SIGPOLL	\027
23	17	27	ETX	\017	55	37	57	SIGPOLL	\01F	87	57	123	SIGPOLL	\028
24	18	28	ENQ	\018	56	38	58	SIGPOLL	\01F	88	58	124	SIGPOLL	\029
25	19	29	ACK	\019	57	39	59	SIGPOLL	\01F	89	59	125	SIGPOLL	\02A
26	1A	2A	BT	\01A	58	3A	5A	SIGPOLL	\01F	90	5A	126	SIGPOLL	\02B
27	1B	2B	HT	\01B	59	3B	5B	SIGPOLL	\01F	91	5B	127	SIGPOLL	\02C
28	1C	2C	LF	\01C	60	3C	5C	SIGPOLL	\01F	92	5C	128	SIGPOLL	\02D
29	1D	2D	VT	\01D	61	3D	5D	SIGPOLL	\01F	93	5D	129	SIGPOLL	\02E
30	1E	2E	FF	\01E	62	3E	5E	SIGPOLL	\01F	94	5E	130	SIGPOLL	\02F
31	1F	2F	CR	\01F	63	3F	5F	SIGPOLL	\01F	95	5F	131	SIGPOLL	\030
32	20	30	SO	\020	64	40	60	SIGPOLL	\01F	96	60	132	SIGPOLL	\031
33	21	31	SI	\021	65	41	61	SIGPOLL	\01F	97	61	133	SIGPOLL	\032
34	22	32	DC1	\022	66	42	62	SIGPOLL	\01F	98	62	134	SIGPOLL	\033
35	23	33	DC2	\023	67	43	63	SIGPOLL	\01F	99	63	135	SIGPOLL	\034
36	24	34	DC3	\024	68	44	64	SIGPOLL	\01F	100	64	136	SIGPOLL	\035
37	25	35	DC4	\025	69	45	65	SIGPOLL	\01F	101	65	137	SIGPOLL	\036
38	26	36	NAK	\026	70	46	66	SIGPOLL	\01F	102	66	138	SIGPOLL	\037
39	27	37	SYN	\027	71	47	67	SIGPOLL	\01F	103	67	139	SIGPOLL	\038
40	28	38	ETX	\028	72	48	68	SIGPOLL	\01F	104	68	140	SIGPOLL	\039
41	29	39	ENQ	\029	73	49	69	SIGPOLL	\01F	105	69	141	SIGPOLL	\03A
42	2A	3A	ACK	\02A	74	4A	6A	SIGPOLL	\01F	106	6A	142	SIGPOLL	\03B
43	2B	3B	BT	\02B	75	4B	6B	SIGPOLL	\01F	107	6B	143	SIGPOLL	\03C
44	2C	3C	HT	\02C	76	4C	6C	SIGPOLL	\01F	108	6C	144	SIGPOLL	\03D
45	2D	3D	LF	\02D	77	4D	6D	SIGPOLL	\01F	109	6D	145	SIGPOLL	\03E
46	2E	3E	VT	\02E	78	4E	6E	SIGPOLL	\01F	110	6E	146	SIGPOLL	\03F
47	2F	3F	FF	\02F	79	4F	6F	SIGPOLL	\01F	111	6F	147	SIGPOLL	\040
48	30	40	CR	\030	80	50	70	SIGPOLL	\01F	112	70	148	SIGPOLL	\041
49	31	41	SO	\031	81	51	71	SIGPOLL	\01F	113	71	149	SIGPOLL	\042
50	32	42	SI	\032	82	52	72	SIGPOLL	\01F	114	72	150	SIGPOLL	\043
51	33	43	DC1	\033	83	53	73	SIGPOLL	\01F	115	73	151	SIGPOLL	\044
52	34	44	DC2	\034	84	54	74	SIGPOLL	\01F	116	74	152	SIGPOLL	\045
53	35	45	DC3	\035	85	55	75	SIGPOLL	\01F	117	75	153	SIGPOLL	\046
54	36	46	DC4	\036	86	56	76	SIGPOLL	\01F	118	76	154	SIGPOLL	\047
55	37	47	NAK	\037	87	57	77	SIGPOLL	\01F	119	77	155	SIGPOLL	\048
56	38	48	SYN	\038	88	58	78	SIGPOLL	\01F	120	78	156	SIGPOLL	\049
57	39	49	ETX	\039	89	59	79	SIGPOLL	\01F	121	79	157	SIGPOLL	\04A
58	3A	4A	ENQ	\03A	90	5A	7A	SIGPOLL	\01F	122	7A	158	SIGPOLL	\04B
59	3B	4B	ACK	\03B	91	5B	7B	SIGPOLL	\01F	123	7B	159	SIGPOLL	\04C
60	3C	4C	BT	\03C	92	5C	7C	SIGPOLL	\01F	124	7C	160	SIGPOLL	\04D
61	3D	4D	HT	\03D	93	5D	7D	SIGPOLL	\01F	125	7D	161	SIGPOLL	\04E
62	3E	4E	LF	\03E	94	5E	7E	SIGPOLL	\01F	126	7E	162	SIGPOLL	\04F
63	3F	4F	VT	\03F	95	5F	7F	SIGPOLL	\01F	127	7F	163	SIGPOLL	\050
64	40	50	FF	\040	96	60	80	SIGPOLL	\01F	128	80	164	SIGPOLL	\051
65	41	51	CR	\041	97	61	81	SIGPOLL	\01F	129	81	165	SIGPOLL	\052
66	42	52	SO	\042	98	62	82	SIGPOLL	\01F	130	82	166	SIGPOLL	\053
67	43	53	SI	\043	99	63	83	SIGPOLL	\01F	131	83	167	SIGPOLL	\054
68	44	54	DC1	\044	100	64	84	SIGPOLL	\01F	132	84	168	SIGPOLL	\055
69	45	55	DC2	\045	101	65	85	SIGPOLL	\01F	133	85	169	SIGPOLL	\056
70	46	56	DC3	\046	102	66	86	SIGPOLL	\01F	134	86	170	SIGPOLL	\057
71	47	57	DC4	\047	103	67	87	SIGPOLL	\01F	135	87	171	SIGPOLL	\058
72	48	58	NAK	\048	104	68	88	SIGPOLL	\01F	136	88	172	SIGPOLL	\059
73	49	59	SYN	\049	105	69	89	SIGPOLL	\01F	137	89	173	SIGPOLL	\05A
74	4A	5A	ETX	\04A	106	6A	8A	SIGPOLL	\01F	138	8A	174	SIGPOLL	\05B
75	4B	5B	ENQ	\04B	107	6B	8B	SIGPOLL	\01F	139	8B	175	SIGPOLL	\05C
76	4C	5C	ACK	\04C	108	6C	8C	SIGPOLL	\01F	140	8C	176	SIGPOLL	\05D
77	4D	5D	BT	\04D	109	6D	8D	SIGPOLL	\01F	141	8D	177	SIGPOLL	\05E
78	4E	5E	HT	\04E	110	6E	8E	SIGPOLL	\01F	142	8E	178	SIGPOLL	\05F
79	4F	5F	LF	\04F	111	6F	8F	SIGPOLL	\01F	143	8F	179	SIGPOLL	\060
80	50	60	VT	\050	112	70	90	SIGPOLL	\01F	144	90	180	SIGPOLL	\061
81	51	61	FF	\051	113	71	91	SIGPOLL	\01F	145	91	181	SIGPOLL	\062
82	52	62	CR	\052	114	72	92	SIGPOLL	\01F	146	92	182	SIGPOLL	\063
83	53	63	SO	\053	115	73	93	SIGPOLL	\01F	147	93	183	SIGPOLL	\064
84	54	64	SI	\054	116	74	94	SIGPOLL	\01F	148	94	184	SIGPOLL	\065
85	55	65	DC1	\055	117	75	95	SIGPOLL	\01F	149	95	185	SIGPOLL	\066
86	56	66	DC2	\056	118	76	96	SIGPOLL	\01F	150	96	186	SIGPOLL	\067
87	57	67	DC3	\057	119	77	97	SIGPOLL	\01F	151	97	187	SIGPOLL	\068
88	58	68	DC4	\058	120	78	98	SIGPOLL	\01F	152	98	188	SIGPOLL	\069
89	59	69	NAK	\059	121	79	99	SIGPOLL	\01F	153	99	189	SIGPOLL	\06A
90	5A	6A	SYN	\05A	122	7A	9A	SIGPOLL	\01F	154	9A	190	SIGPOLL	\06B
91	5B	6B	ETX	\05B	123	7B	9B	SIGPOLL	\01F	155	9B	191	SIGPOLL	\06C
92	5C	6C	ENQ	\05C	124	7C	9C	SIGPOLL	\01F	156	9C	192	SIGPOLL	\06D
93	5D	6D	ACK	\05D	125	7D	9D	SIGPOLL	\01F	157	9D	193	SIGPOLL	\06E
94	5E	6E	BT	\05E	126	7E	9E	SIGPOLL	\01F	158	9E	194	SIGPOLL	\06F
95	5F	6F	HT	\05F	127	7F	9F	SIGPOLL	\01F	159	9F	195	SIGPOLL	\070
96	60	70	LF	\060	128	80	A0	SIGPOLL	\01F	160	A0	196	SIGPOLL	\071
97	61	71	VT	\061	129	81	A1	SIGPOLL	\01F	161	A1	197	SIGPOLL	\072
98	62	72	FF	\062	130	82	A2	SIGPOLL	\01F	162	A2	198	SIGPOLL	\073
99	63	73	CR	\063	131	83	A3	SIGPOLL	\01F	163	A3	199	SIGPOLL	\074
100	64	74	SO	\064	132	84	A4	SIGPOLL	\01F	164	A4	200	SIGPOLL	\075
101	65	75	SI	\065	133	85	A5	SIGPOLL	\01F	165	A5	201	SIGPOLL	\076
102	66	76	DC1	\066	134	86	A6	SIGPOLL	\01F	166	A6	202	SIGPOLL	\077
103	67	77	DC2	\067	135	87	A7	SIGPOLL						

- `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	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	'\f'
35	23	\25	5B	43	\r	99	63	'\r'
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.
- Example: `chr(97)` returns 'a'.

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'
'\0F9A0'	'0F9A0'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A2'	'0F9A2'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A4'	'0F9A4'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A6'	'0F9A6'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A8'	'0F9A8'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A9'	'0F9A9'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A00'	'0F9A00'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A02'	'0F9A02'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A04'	'0F9A04'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A06'	'0F9A06'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A08'	'0F9A08'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A09'	'0F9A09'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A0A'	'0F9A0A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A0B'	'0F9A0B'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A0C'	'0F9A0C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A0D'	'0F9A0D'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A0E'	'0F9A0E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A0F'	'0F9A0F'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A10'	'0F9A10'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A11'	'0F9A11'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A12'	'0F9A12'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A13'	'0F9A13'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A14'	'0F9A14'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A15'	'0F9A15'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A16'	'0F9A16'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A17'	'0F9A17'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A18'	'0F9A18'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A19'	'0F9A19'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A1A'	'0F9A1A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A1B'	'0F9A1B'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A1C'	'0F9A1C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A1D'	'0F9A1D'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A1E'	'0F9A1E'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A1F'	'0F9A1F'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A20'	'0F9A20'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A21'	'0F9A21'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A22'	'0F9A22'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A23'	'0F9A23'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A24'	'0F9A24'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A25'	'0F9A25'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A26'	'0F9A26'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A27'	'0F9A27'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0F9A28'	'0F9A28'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0F9A29'	'0F9A29'	'0A5'	'\0FD'	'0FD'	'0A6'
'\0F9A2A'	'0F9A2A'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0F9A2B'	'0F9A2B'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0F9A2C'	'0F9A2C'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0F9A2D'	'0F9A2D'				

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)
```



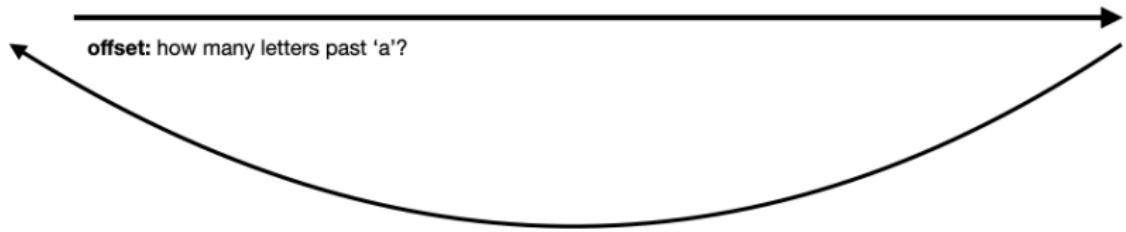
Python Tutor

```
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 the 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



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`.

Lecture Quiz

- Log-in to Gradescope
- Find LECTURE 2 Quiz
- Take the quiz
- **You have 3 minutes**

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- **Strings**
- Guests: Internships, Advising & 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[:-1].split("s")
```

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- `s[0]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- `s[1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- `s[-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- $s[-1]$ is 's'.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- `s[3:6]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- `s[3:6]` is ‘day’.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- `s[:3]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

- `s[:-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

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

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

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

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

"Friday~~X~~Saturday~~X~~Sunday"

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXSaturdayXSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXSaturdayXSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXSaturdayXSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXXXXSaturdayXXXXSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")
```

```
"FridayXXXXsaturdayXXXXsunday"
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXSaturdayXSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")
```

```
"FriXXXySaturXXXySunXXX"  
days = ['Fri', 'sSatur', 'sSun']
```

Today's Topics



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

Guest Speakers

- Announcement on Blackboard:
 - ▶ Advising
 - ▶ Programs and Clubs Handout
 - ▶ Internships Handout
 - ▶ Hunter CS Handbook
 - ▶ PreTech Center (formerly CUNY2X) Newsletter

Recap

- In Python, we introduced:

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

Recap

```
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)