

# CSci 127: Introduction to Computer Science



[hunter.cuny.edu/csci](https://hunter.cuny.edu/csci)

# Review of Lecture 1: range function

- Simplest version of range function is `range(n)`, where `n` is a positive integer. Function `range(n)` returns a list of integers ranging (see how the name is coming from?) from 0 to `n-1`, a total of `n` integer. For example, `range(5)` returns a list `[0, 1, 2, 3, 4]`, while `range(-1)` returns an empty list.
- Usage: `range(n)` is part of `for` statement to repeat something for `n` times.
- Common mistake 1: use floating point number as parameters for range function. For example, `range(1.2)` is wrong.
- Common mistake 2: use square brackets `[` and `]` instead of parentheses `(` and `)` after range function. Note that `range` is a function name, it is followed by a pair of matched parentheses that enclose parameters for range function. For example, `range[2]` is wrong while `range(2)` is correct.

# Review of Lecture 1: turtle graphics

- Imagine a turtle has a pen, when it moves some distance, say, 100 pixels, a distance unit in digital world, a line is drawn on the screen.
- Then the turtle turn left 120 degrees.
- Repeat the above two steps for 3 times.

```
1 import turtle
2
3 t = turtle.Turtle()
4
5 t.fd(100) # same as t.forward(100), ie, t moves forward 100 pixels
6 t.left(120) # t turns left 90 degrees
7
8 t.fd(100)
9 t.left(120)
10
11 t.fd(100)
12 t.left(120)
```

## Review of Lecture 1: use turtle to draw a triangle

- After turning 120 degrees for three times, the turtle turns around in a circle and comes back to its start point.
- Use for-statement and range function to rewrite the above program.

```
1 import turtle
2
3 t = turtle.Turtle()
4
5 for i in range(3):
6     t.fd(100) #same as t.forward(100)
7     t.left(120)
```

For more commands, read [turtle documentation](#)



# Draw a polygon with $n \geq 3$ sides using turtle

Pseudocode (not actual code) work with any language with turtle library.  
Also called algorithm, consisting of step to step instructions.

```
import turtle library
```

```
instantiate a turtle object called t
```

```
initialize n to be an integer at least 3
```

```
Repeat the following for n times
```

```
(1) t moves forward t fixed number of distance
```

```
(2) t turns left  $360 / n$  degrees
```

Explanation: every time a turtle turns  $360 / n$ , after n rounds, move back to the start point.

# Programming Environment

- Best choice: use command line and vi / Emac editors. Avoid using IDE (Integrated development environment).
- Second choice: Install Python and IDLE, an IDE for Python.
- Last choice (not encouraged): use online editor.
  - ▶ [onlinegdb](#) for non-turtle-graphics programs
  - ▶ [trinket](#) for turtle graphics programs.
- In windows, install WSL (Windows Subsystem for Linux) for command line testing.
- Lab 1001 G has Linux laptops, borrow one from TAs. Need department's permission to borrow overnight.
- You can borrow a Lenovo laptop free for a semester from the 2nd floor of library (at Audio/Video department, need an additional photo ID besides school's ID).

# Today's Topics



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

# Today's Topics



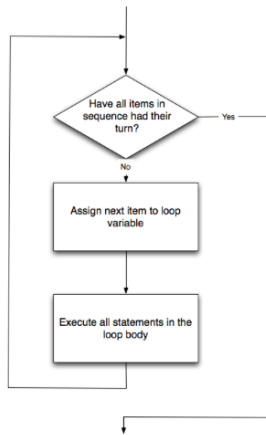
- **For-loops**
- `range()`
- Variables
- Characters
- Strings

## Group Work: predict what will be printed

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

[link to program](#)

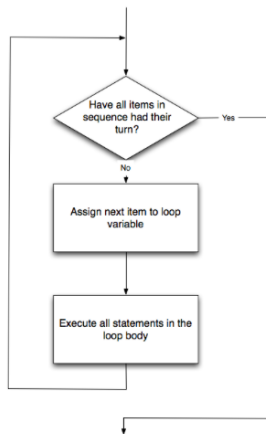
# for-loop



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

*How to Think Like CS, §4.5*

# for-loop



*How to Think Like CS, §4.5*

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

where `list` is a list of items:

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

# Today's Topics



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



More on range(): predict what will be printed

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

[link to range demo](#)

# range()

Simplest version:

- `range(stop)`



# range()



Simplest version:

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

# range()



Simplest version:

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

# range()



Simplest version:

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

```
range(101)
```

range()

What if you wanted to start somewhere else:



range()

What if you wanted to start somewhere else:

- `range(start, stop)`



# range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:  
`[start, start+1, ..., stop-1]`



# range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:  
`[start, start+1, ..., stop-1]`
- For example, if you want the the list  
`[10, 11, ..., 20]`  
you would write:

# range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:  
`[start, start+1, ..., stop-1]`
- For example, if you want the the list  
`[10, 11, ..., 20]`  
you would write:

```
range(10, 21)
```

`range()`

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



# range()

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

- `range(start, stop, step)`



# range()

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

- `range(start, stop, step)`
- Produces a list:  
`[start, start+step, start+2*step..., last]`  
(where last is the largest  $start+k*step$  less than stop)



# range()



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

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

# range()



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

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

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

In summary: `range()`



The three versions:



In summary: `range()`



The three versions:

- `range(stop)`

In summary: `range()`



The three versions:

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

In summary: `range()`



The three versions:

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

# Today's Topics



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

# 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
  - ▶ **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  
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.
- 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	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

(wiki)

# Converting from Character to Code:

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

## ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	0x10	32	0x20	48	0x30
1	SOH	17	0x11	33	0x21	49	0x31
2	STX	18	0x12	34	0x22	50	0x32
3	ETX	19	0x13	35	0x23	51	0x33
4	END	20	0x14	36	0x24	52	0x34
5	SO	21	0x15	37	0x25	53	0x35
6	SI	22	0x16	38	0x26	54	0x36
7	DEL	23	0x17	39	0x27	55	0x37
8		24	0x18	40	0x28	56	0x38
9		25	0x19	41	0x29	57	0x39
10	LF	26	0x1A	42	0x2A	58	0x3A
11	VT	27	0x1B	43	0x2B	59	0x3B
12	FF	28	0x1C	44	0x2C	60	0x3C
13		29	0x1D	45	0x2D	61	0x3D
14		30	0x1E	46	0x2E	62	0x3E
15		31	0x1F	47	0x2F	63	0x3F
16	0x20	48	0x30	64	0x40	80	0x50
17	0x21	49	0x31	65	0x41	81	0x51
18	0x22	50	0x32	66	0x42	82	0x52
19	0x23	51	0x33	67	0x43	83	0x53
20	0x24	52	0x34	68	0x44	84	0x54
21	0x25	53	0x35	69	0x45	85	0x55
22	0x26	54	0x36	70	0x46	86	0x56
23	0x27	55	0x37	71	0x47	87	0x57
24	0x28	56	0x38	72	0x48	88	0x58
25	0x29	57	0x39	73	0x49	89	0x59
26	0x2A	58	0x3A	74	0x4A	90	0x5A
27	0x2B	59	0x3B	75	0x4B	91	0x5B
28	0x2C	60	0x3C	76	0x4C	92	0x5C
29	0x2D	61	0x3D	77	0x4D	93	0x5D
30	0x2E	62	0x3E	78	0x4E	94	0x5E
31	0x2F	63	0x3F	79	0x4F	95	0x5F
32	0x30	64	0x40	80	0x50	96	0x60
33	0x31	65	0x41	81	0x51	97	0x61
34	0x32	66	0x42	82	0x52	98	0x62
35	0x33	67	0x43	83	0x53	99	0x63
36	0x34	68	0x44	84	0x54	100	0x64
37	0x35	69	0x45	85	0x55	101	0x65
38	0x36	70	0x46	86	0x56	102	0x66
39	0x37	71	0x47	87	0x57	103	0x67
40	0x38	72	0x48	88	0x58	104	0x68
41	0x39	73	0x49	89	0x59	105	0x69
42	0x3A	74	0x4A	90	0x5A	106	0x6A
43	0x3B	75	0x4B	91	0x5B	107	0x6B
44	0x3C	76	0x4C	92	0x5C	108	0x6C
45	0x3D	77	0x4D	93	0x5D	109	0x6D
46	0x3E	78	0x4E	94	0x5E	110	0x6E
47	0x3F	79	0x4F	95	0x5F	111	0x6F
48	0x40	80	0x50	96	0x60	112	0x70
49	0x41	81	0x51	97	0x61	113	0x71
50	0x42	82	0x52	98	0x62	114	0x72
51	0x43	83	0x53	99	0x63	115	0x73
52	0x44	84	0x54	100	0x64	116	0x74
53	0x45	85	0x55	101	0x65	117	0x75
54	0x46	86	0x56	102	0x66	118	0x76
55	0x47	87	0x57	103	0x67	119	0x77
56	0x48	88	0x58	104	0x68	120	0x78
57	0x49	89	0x59	105	0x69	121	0x79
58	0x4A	90	0x5A	106	0x6A	122	0x7A
59	0x4B	91	0x5B	107	0x6B	123	0x7B
60	0x4C	92	0x5C	108	0x6C	124	0x7C
61	0x4D	93	0x5D	109	0x6D	125	0x7D
62	0x4E	94	0x5E	110	0x6E	126	0x7E
63	0x4F	95	0x5F	111	0x6F	127	0x7F
64	0x50	96	0x60	112	0x70	128	0x80
65	0x51	97	0x61	113	0x71	129	0x81
66	0x52	98	0x62	114	0x72	130	0x82
67	0x53	99	0x63	115	0x73	131	0x83
68	0x54	100	0x64	116	0x74	132	0x84
69	0x55	101	0x65	117	0x75	133	0x85
70	0x56	102	0x66	118	0x76	134	0x86
71	0x57	103	0x67	119	0x77	135	0x87
72	0x58	104	0x68	120	0x78	136	0x88
73	0x59	105	0x69	121	0x79	137	0x89
74	0x5A	106	0x6A	122	0x7A	138	0x8A
75	0x5B	107	0x6B	123	0x7B	139	0x8B
76	0x5C	108	0x6C	124	0x7C	140	0x8C
77	0x5D	109	0x6D	125	0x7D	141	0x8D
78	0x5E	110	0x6E	126	0x7E	142	0x8E
79	0x5F	111	0x6F	127	0x7F	143	0x8F
80	0x60	112	0x70	128	0x80	144	0x90
81	0x61	113	0x71	129	0x81	145	0x91
82	0x62	114	0x72	130	0x82	146	0x92
83	0x63	115	0x73	131	0x83	147	0x93
84	0x64	116	0x74	132	0x84	148	0x94
85	0x65	117	0x75	133	0x85	149	0x95
86	0x66	118	0x76	134	0x86	150	0x96
87	0x67	119	0x77	135	0x87	151	0x97
88	0x68	120	0x78	136	0x88	152	0x98
89	0x69	121	0x79	137	0x89	153	0x99
90	0x6A	122	0x7A	138	0x8A	154	0x9A
91	0x6B	123	0x7B	139	0x8B	155	0x9B
92	0x6C	124	0x7C	140	0x8C	156	0x9C
93	0x6D	125	0x7D	141	0x8D	157	0x9D
94	0x6E	126	0x7E	142	0x8E	158	0x9E
95	0x6F	127	0x7F	143	0x8F	159	0x9F
96	0x70	128	0x80	144	0x90	160	0xA0
97	0x71	129	0x81	145	0x91	161	0xA1
98	0x72	130	0x82	146	0x92	162	0xA2
99	0x73	131	0x83	147	0x93	163	0xA3
100	0x74	132	0x84	148	0x94	164	0xA4
101	0x75	133	0x85	149	0x95	165	0xA5
102	0x76	134	0x86	150	0x96	166	0xA6
103	0x77	135	0x87	151	0x97	167	0xA7
104	0x78	136	0x88	152	0x98	168	0xA8
105	0x79	137	0x89	153	0x99	169	0xA9
106	0x7A	138	0x8A	154	0x9A	170	0xAA
107	0x7B	139	0x8B	155	0x9B	171	0xAB
108	0x7C	140	0x8C	156	0x9C	172	0xAC
109	0x7D	141	0x8D	157	0x9D	173	0xAD
110	0x7E	142	0x8E	158	0x9E	174	0xAE
111	0x7F	143	0x8F	159	0x9F	175	0xAF
112	0x80	144	0x90	160	0xA0	176	0xB0
113	0x81	145	0x91	161	0xA1	177	0xB1
114	0x82	146	0x92	162	0xA2	178	0xB2
115	0x83	147	0x93	163	0xA3	179	0xB3
116	0x84	148	0x94	164	0xA4	180	0xB4
117	0x85	149	0x95	165	0xA5	181	0xB5
118	0x86	150	0x96	166	0xA6	182	0xB6
119	0x87	151	0x97	167	0xA7	183	0xB7
120	0x88	152	0x98	168	0xA8	184	0xB8
121	0x89	153	0x99	169	0xA9	185	0xB9
122	0x8A	154	0x9A	170	0xAA	186	0xBA
123	0x8B	155	0x9B	171	0xAB	187	0xBB
124	0x8C	156	0x9C	172	0xAC	188	0xBC
125	0x8D	157	0x9D	173	0xAD	189	0xBD
126	0x8E	158	0x9E	174	0xAE	190	0xBE
127	0x8F	159	0x9F	175	0xAF	191	0xBF
128	0x90	160	0xA0	176	0xB0	192	0xC0
129	0x91	161	0xA1	177	0xB1	193	0xC1
130	0x92	162	0xA2	178	0xB2	194	0xC2
131	0x93	163	0xA3	179	0xB3	195	0xC3
132	0x94	164	0xA4	180	0xB4	196	0xC4
133	0x95	165	0xA5	181	0xB5	197	0xC5
134	0x96	166	0xA6	182	0xB6	198	0xC6
135	0x97	167	0xA7	183	0xB7	199	0xC7
136	0x98	168	0xA8	184	0xB8	200	0xC8
137	0x99	169	0xA9	185	0xB9	201	0xC9
138	0x9A	170	0xAA	186	0xBA	202	0xCA
139	0x9B	171	0xAB	187	0xBB	203	0xCB
140	0x9C	172	0xAC	188	0xBC	204	0xCC
141	0x9D	173	0xAD	189	0xBD	205	0xCD
142	0x9E	174	0xAE	190	0xBE	206	0xCE
143	0x9F	175	0xAF	191	0xBF	207	0xCF
144	0xA0	176	0xB0	192	0xC0	208	0xD0
145	0xA1	177	0xB1	193	0xC1	209	0xD1
146	0xA2	178	0xB2	194	0xC2	210	0xD2
147	0xA3	179	0xB3	195	0xC3	211	0xD3
148	0xA4	180	0xB4	196	0xC4	212	0xD4
149	0xA5	181	0xB5	197	0xC5	213	0xD5
150	0xA6	182	0xB6	198	0xC6	214	0xD6
151	0xA7	183	0xB7	199	0xC7	215	0xD7
152	0xA8	184	0xB8	200	0xC8	216	0xD8
153	0xA9	185	0xB9	201	0xC9	217	0xD9
154	0xAA	186	0xBA	202	0xCA	218	0xDA
155	0xAB	187	0xBB	203	0xCB	219	0xDB
156	0xAC	188	0xBC	204	0xCC	220	0xDC
157	0xAD	189	0xBD	205	0xCD	221	0xDD
158	0xAE	190	0xBE	206	0xCE	222	0xDE
159	0xAF	191	0xBF	207	0xCF	223	0xDF
160	0xB0	192	0xC0	208	0xD0	224	0xE0
161	0xB1	193	0xC1	209	0xD1	225	0xE1
162	0xB2	194	0xC2	210	0xD2	226	0xE2
163	0xB3	195	0xC3	211	0xD3	227	0xE3
164	0xB4	196	0xC4	212	0xD4	228	0xE4
165	0xB5	197	0xC5	213	0xD5	229	0xE5
166	0xB6	198	0xC6	214	0xD6	230	0xE6
167	0xB7	199	0xC7	215	0xD7	231	0xE7
168	0xB8	200	0xC8	216	0xD8	232	0xE8
169	0xB9	201	0xC9	217	0xD9	233	0xE9
170	0xBA	202	0xCA	218	0xDA	234	0xEA
171	0xBB	203	0xCB	219	0xDB	235	0xEB
172	0xBC	204	0xCC	220	0xDC	236	0xEC
173	0xBD	205	0xCD	221	0xDD	237	0xED
174	0xBE	206	0xCE	222	0xDE	238	0xEE
175	0xBF	207	0xCF	223	0xDF	239	0xEF
176	0xC0	208	0xD0	224	0xE0	240	0xF0
177	0xC1	209	0xD1	225	0		

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

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

- `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'.)*

## ASCII TABLE

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

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



# 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	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		1	SP	2	!"#\$%	3	&'()*+,-./:;<=>?@
4	A-Z	11	[ \ ] ^ _ `	22	{   } ~	33	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
44	0-9	55	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	66	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	77	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
88	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	99	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	110	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	121	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
132	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	143	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	154	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	165	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
176	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	187	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	198	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	209	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
220	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	231	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	242	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	253	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
264	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	275	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	286	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	297	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
308	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	319	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	330	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	341	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
352	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	363	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	374	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	385	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
396	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	407	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	418	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	429	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
440	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	451	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	462	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	473	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
484	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	495	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	506	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	517	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
528	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	539	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	550	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	561	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
572	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	583	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	594	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	605	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
616	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	627	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	638	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	649	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
660	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	671	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	682	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	693	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
704	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	715	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	726	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	737	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
748	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	759	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	770	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	781	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
792	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	803	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	814	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	825	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
836	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	847	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	858	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	869	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
880	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	891	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	902	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	913	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
924	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	935	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	946	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	957	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @
968	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	979	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	990	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @	1001	! " # \$ % & ' ( ) * + , - . / : ; < = > ? @

- `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'.)

## ASCII TABLE

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

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

## In Pairs or Triples...

```
1 for c in range(65,90):
2     print(chr(c))
3
4 message = "I love Python"
5 newMessage = ""
6 for c in message:
7     print(ord(c)) #Print the Unicode of c
8     print(chr(ord(c)+1)) #Print the next
9         character
10    newMessage = newMessage + chr(ord(c)+1) #
11        add to the new message
12 print("The coded message is", newMessage)
```

[link to python turtor demo](#)

# In Pairs or Triples...

Predict what will be printed.

```
1 for c in "World":  
2     print(c, c, '#')
```

## Cesar Cipher: hints for P9 of programming assignments

```
1 word = input("Enter a string: ")
2 codedWord = ""
3 shift = 2 #shift two letters
4 for ch in word:
5     offset = ord(ch) - ord('A') #relative
6         distance to 'A'
7     wrap = (offset + shift) % ? #what is ?
8     #TODO: compute the new letter
9     #TODO: add the newChar to the coded word
10 print("After shifting", shift, "letters,", \
11        word, "becomes", codedWord)
```

## User Input

Enter name and the year you are a freshman, print and calculate graduate year. *Covered in detail in Lab 2:*

```
1 name = input("Enter name: ")
2 year = int(input("Enter freshman year: "))
3
4 print("Hello, " + name) #same as print("
    Hello,", name)
5 graduateYear = year + 4
6 print("Will graduate in "+str(graduateYear))
7 #same:print("Will graduate in",graduateYear)
8 #str(graduateYear) converts int graduateYear
    to string. Suppose graduateYear is 2023,
    then str(graduateYear) is "2023".
```

## 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.
- $s = \text{"hi"} + \text{"Mom"}$  stores "hiMom" in memory locations  $s$ .
- $s = s + \text{"A"}$  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(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

# Recap

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

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

# Recap

- In Python, we introduced:

- ▶ For-loops
- ▶ range()

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

# Recap

- In Python, we introduced:

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

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

# Recap

- In Python, we introduced:

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

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



# Recap

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

- In Python, we introduced:

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

# Recap

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

● In Python, we introduced:

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

# Recap

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

- In Python, we introduced:

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

# Recap

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

- In Python, we introduced:

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

# Practice Quiz & Final Questions



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

# Practice Quiz & Final Questions



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

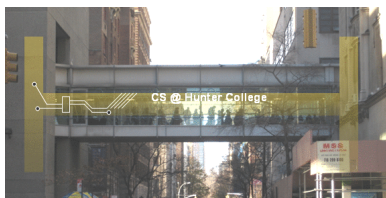
# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

# Weekly Reminders!

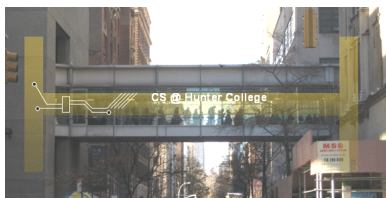


Before next lecture, don't forget to:

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



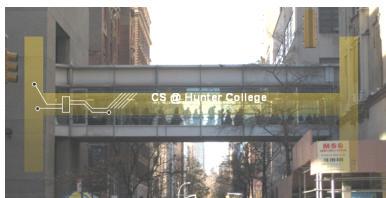
# Weekly Reminders!



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

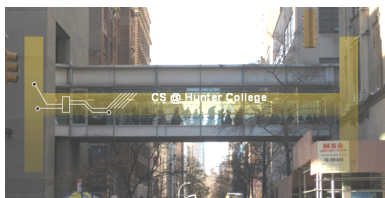
# Weekly Reminders!



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**)

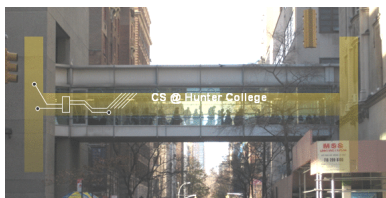
# Weekly Reminders!



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.