

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Today's Topics



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

In Pairs or Triples...

Some review and some novel challenges:

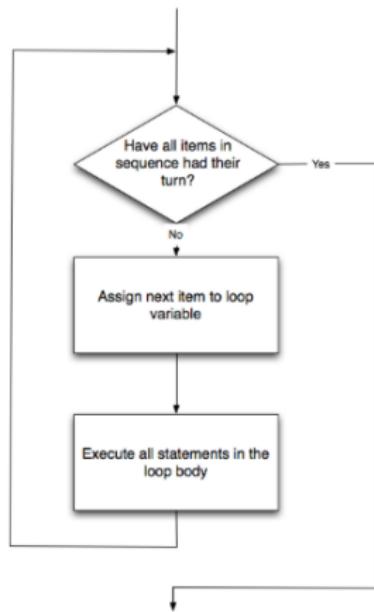
```
1 #Predict what will be printed:  
2 for i in range(4):  
3     print('The world turned upside down')  
4 for j in [0,1,2,3,4,5]:  
5     print(j)  
6 for count in range(6):  
7     print(count)  
8 for color in ['red', 'green', 'blue']:  
9     print(color)  
10    for i in range(2):  
11      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

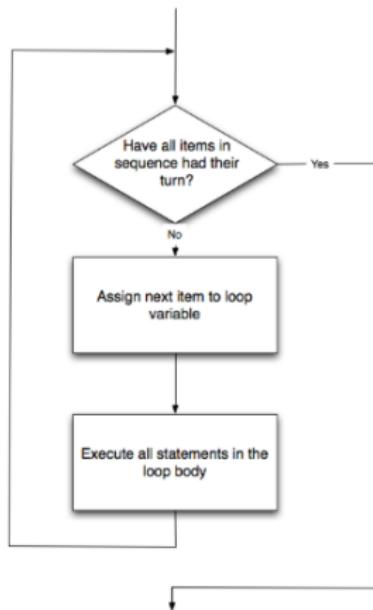
for-loop



```
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

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 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 list
`[5,10,...,50]`
you would write:



range()



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

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

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

In summary: range()



The three versions:

In summary: range()



The three versions:

- `range(stop)`

In summary: range()



The three versions:

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

In summary: range()



The three versions:

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

Today's Topics



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

Variables

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



Variables

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



Variables

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



Variables



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

Variables



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

Variables



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

Variables



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

Variable Names

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



Variable Names

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



Variable Names



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

Variable Names



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

Variable Names



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

Today's Topics



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

Standardized Code for Characters

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

Standardized Code for Characters

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

Standardized Code for Characters

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

ASCII TABLE

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

(wiki)

Converting from Character to Code:

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

ASCII TABLE

Binary Hex Char	Binary Hex Char	Decimal Hex Char	Decimal Non Char
00000000	00000000	00	0
00000001	00000001	01	1
00000002	00000002	02	2
00000003	00000003	03	3
00000004	00000004	04	4
00000005	00000005	05	5
00000006	00000006	06	6
00000007	00000007	07	7
00000008	00000008	08	8
00000009	00000009	09	9
0000000A	0000000A	0A	A
0000000B	0000000B	0B	B
0000000C	0000000C	0C	C
0000000D	0000000D	0D	D
0000000E	0000000E	0E	E
0000000F	0000000F	0F	F
00000010	00000010	10	16
00000011	00000011	11	17
00000012	00000012	12	18
00000013	00000013	13	19
00000014	00000014	14	20
00000015	00000015	15	21
00000016	00000016	16	22
00000017	00000017	17	23
00000018	00000018	18	24
00000019	00000019	19	25
0000001A	0000001A	1A	26
0000001B	0000001B	1B	27
0000001C	0000001C	1C	28
0000001D	0000001D	1D	29
0000001E	0000001E	1E	2A
0000001F	0000001F	1F	2B
00000020	00000020	20	32
00000021	00000021	21	33
00000022	00000022	22	34
00000023	00000023	23	35
00000024	00000024	24	36
00000025	00000025	25	37
00000026	00000026	26	38
00000027	00000027	27	39
00000028	00000028	28	40
00000029	00000029	29	41
0000002A	0000002A	2A	42
0000002B	0000002B	2B	43
0000002C	0000002C	2C	44
0000002D	0000002D	2D	45
0000002E	0000002E	2E	46
0000002F	0000002F	2F	47
00000030	00000030	30	48
00000031	00000031	31	49
00000032	00000032	32	50
00000033	00000033	33	51
00000034	00000034	34	52
00000035	00000035	35	53
00000036	00000036	36	54
00000037	00000037	37	55
00000038	00000038	38	56
00000039	00000039	39	57
0000003A	0000003A	3A	58
0000003B	0000003B	3B	59
0000003C	0000003C	3C	60
0000003D	0000003D	3D	61
0000003E	0000003E	3E	62
0000003F	0000003F	3F	63
00000040	00000040	40	64
00000041	00000041	41	65
00000042	00000042	42	66
00000043	00000043	43	67
00000044	00000044	44	68
00000045	00000045	45	69
00000046	00000046	46	70
00000047	00000047	47	71
00000048	00000048	48	72
00000049	00000049	49	73
0000004A	0000004A	4A	74
0000004B	0000004B	4B	75
0000004C	0000004C	4C	76
0000004D	0000004D	4D	77
0000004E	0000004E	4E	78
0000004F	0000004F	4F	79
00000050	00000050	50	80

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		32	20		64	40	
1	01	!>	33	21	!	65	41	A
2	02	!>	34	22	!	66	42	B
3	03	!>	35	23	!	67	43	C
4	04	!>	36	24	!	68	44	D
5	05	!>	37	25	!	69	45	E
6	06	!>	38	26	!	70	46	F
7	07	!>	39	27	!	71	47	G
8	08	!>	40	28	!	72	48	H
9	09	!>	41	29	!	73	49	I
10	0A	!>	42	2A	!	74	4A	J
11	0B	!>	43	2B	!	75	4B	K
12	0C	!>	44	2C	!	76	4C	L
13	0D	!>	45	2D	!	77	4D	M
14	0E	!>	46	2E	!	78	4E	N
15	0F	!>	47	2F	!	79	4F	O
16	10	!>	48	30	!	80	50	P
17	11	!>	49	31	!	81	51	Q
18	12	!>	4A	32	!	82	52	R
19	13	!>	4B	33	!	83	53	S
20	14	!>	4C	34	!	84	54	T
21	15	!>	4D	35	!	85	55	U
22	16	!>	4E	36	!	86	56	V
23	17	!>	4F	37	!	87	57	W
24	18	!>	50	38	!	88	58	X
25	19	!>	51	39	!	89	59	Y
26	1A	!>	52	3A	!	90	5A	Z
27	1B	!>	53	3B	!	91	5B	
28	1C	!>	54	3C	!	92	5C	
29	1D	!>	55	3D	!	93	5D	
30	1E	!>	56	3E	!	94	5E	
31	1F	!>	57	3F	!	95	5F	
32	20	!>	58	40	!	96	60	
33	21	!>	59	41	!	97	61	
34	22	!>	5A	42	!	98	62	
35	23	!>	5B	43	!	99	63	
36	24	!>	5C	44	!	100	64	
37	25	!>	5D	45	!	101	65	
38	26	!>	5E	46	!	102	66	
39	27	!>	5F	47	!	103	67	
40	28	!>	60	48	!	104	68	
41	29	!>	61	49	!	105	69	
42	2A	!>	62	4A	!	106	6A	
43	2B	!>	63	4B	!	107	6B	
44	2C	!>	64	4C	!	108	6C	
45	2D	!>	65	4D	!	109	6D	
46	2E	!>	66	4E	!	110	6E	
47	2F	!>	67	4F	!	111	6F	
48	30	!>	68	50	!	112	70	
49	31	!>	69	51	!	113	71	
50	32	!>	6A	52	!	114	72	
51	33	!>	6B	53	!	115	73	
52	34	!>	6C	54	!	116	74	
53	35	!>	6D	55	!	117	75	
54	36	!>	6E	56	!	118	76	
55	37	!>	6F	57	!	119	77	
56	38	!>	70	58	!	120	78	
57	39	!>	71	59	!	121	79	
58	3A	!>	72	5A	!	122	7A	
59	3B	!>	73	5B	!	123	7B	
60	3C	!>	74	5C	!	124	7C	
61	3D	!>	75	5D	!	125	7D	
62	3E	!>	76	5E	!	126	7E	
63	3F	!>	77	5F	!	127	7F	

- `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	Char	Decimal	Hex	Char	Decimal	Hex	Char
	0	00		128	80		255	F0	
	1	01		129	81		256	F1	
	2	02		130	82		257	F2	
	3	03		131	83		258	F3	
	4	04		132	84		259	F4	
	5	05		133	85		260	F5	
	6	06		134	86		261	F6	
	7	07		135	87		262	F7	
	8	08		136	88		263	F8	
	9	09		137	89		264	F9	
	10	0A		138	8A		265	FA	
	11	0B		139	8B		266	FB	
	12	0C		140	8C		267	FC	
	13	0D		141	8D		268	FD	
	14	0E		142	8E		269	FE	
	15	0F		143	8F		270	FF	
	16	10		144	90		271		
	17	11		145	91		272		
	18	12		146	92		273		
	19	13		147	93		274		
	20	14		148	94		275		
	21	15		149	95		276		
	22	16		150	96		277		
	23	17		151	97		278		
	24	18		152	98		279		
	25	19		153	99		280		
	26	1A		154	9A		281		
	27	1B		155	9B		282		
	28	1C		156	9C		283		
	29	1D		157	9D		284		
	30	1E		158	9E		285		
	31	1F		159	9F		286		
	32	20		160	A0		287		
	33	21		161	A1		288		
	34	22		162	A2		289		
	35	23		163	A3		290		
	36	24		164	A4		291		
	37	25		165	A5		292		
	38	26		166	A6		293		
	39	27		167	A7		294		
	40	28		168	A8		295		
	41	29		169	A9		296		
	42	2A		170	AA		297		
	43	2B		171	AB		298		
	44	2C		172	AC		299		
	45	2D		173	AD		300		
	46	2E		174	AE		301		
	47	2F		175	AF		302		
	48	30		176	B0		303		
	49	31		177	B1		304		
	50	32		178	B2		305		
	51	33		179	B3		306		
	52	34		180	B4		307		
	53	35		181	B5		308		
	54	36		182	B6		309		
	55	37		183	B7		310		
	56	38		184	B8		311		
	57	39		185	B9		312		
	58	3A		186	BA		313		
	59	3B		187	BB		314		
	60	3C		188	BC		315		
	61	3D		189	BD		316		
	62	3E		190	BE		317		
	63	3F		191	BF		318		
	64	40		192	C0		319		
	65	41		193	C1		320		
	66	42		194	C2		321		
	67	43		195	C3		322		
	68	44		196	C4		323		
	69	45		197	C5		324		
	70	46		198	C6		325		
	71	47		199	C7		326		
	72	48		200	C8		327		
	73	49		201	C9		328		
	74	4A		202	CA		329		
	75	4B		203	CB		330		
	76	4C		204	CC		331		
	77	4D		205	CD		332		
	78	4E		206	CE		333		
	79	4F		207	CF		334		
	80	50		208	D0		335		
	81	51		209	D1		336		
	82	52		210	D2		337		
	83	53		211	D3		338		
	84	54		212	D4		339		
	85	55		213	D5		340		
	86	56		214	D6		341		
	87	57		215	D7		342		
	88	58		216	D8		343		
	89	59		217	D9		344		
	90	5A		218	DA		345		
	91	5B		219	DB		346		
	92	5C		220	DC		347		
	93	5D		221	DD		348		
	94	5E		222	DE		349		
	95	5F		223	DF		350		
	96	60		224	E0		351		
	97	61		225	E1		352		
	98	62		226	E2		353		
	99	63		227	E3		354		
	100	64		228	E4		355		
	101	65		229	E5		356		
	102	66		230	E6		357		
	103	67		231	E7		358		
	104	68		232	E8		359		
	105	69		233	E9		360		
	106	6A		234	EA		361		
	107	6B		235	EB		362		
	108	6C		236	EC		363		
	109	6D		237	ED		364		
	110	6E		238	EE		365		
	111	6F		239	EF		366		
	112	70		240			367		
	113	71		241			368		
	114	72		242			369		
	115	73		243			370		
	116	74		244			371		
	117	75		245			372		
	118	76		246			373		
	119	77		247			374		
	120	78		248			375		
	121	79		249			376		
	122	7A		250			377		
	123	7B		251			378		
	124	7C		252			379		
	125	7D		253			380		
	126	7E		254			381		
	127	7F		255			382		
	128	80		256			383		
	129	81		257			384		
	130	82		258			385		
	131	83		259			386		
	132	84		260			387		
	133	85		261			388		
	134	86		262			389		
	135	87		263			390		
	136	88		264			391		
	137	89		265			392		
	138	8A		266			393		
	139	8B		267			394		
	140	8C		268			395		
	141	8D		269			396		
	142	8E		270			397		
	143	8F		271			398		
	144	90		272			399		
	145	91		273			400		
	146	92		274			401		
	147	93		275			402		
	148	94		276			403		
	149	95		277			404		
	150	96		278			405		
	151	97		279			406		
	152	98		280			407		
	153	99		281			408		
	154	9A		282			409		
	155	9B		283			410		
	156	9C		284			411		
	157	9D		285			412		
	158	9E		286			413		
	159	9F		287			414		
	160	A0		288			415		
	161	A1		289			416		
	162	A2		290			417		
	163	A3		291			418		
	164	A4		292			419		
	165	A5		293			420		
	166	A6		294			421		
	167	A7		295			422		
	168	A8		296			423		
	169	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'
'\0FA0'	'0FA0'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FA2'	'0FA2'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FA4'	'0FA4'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FA6'	'0FA6'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FA8'	'0FA8'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FAA'	'0FAA'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FAC'	'0FAC'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FAD'	'0FAD'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FAC0'	'0FAC0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FAC2'	'0FAC2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FAC4'	'0FAC4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FAC6'	'0FAC6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FAC8'	'0FAC8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA'	'0FACA'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'\0FACA6'	'0FACA6'	'0A8'	'\0FD'	'0FD'	'0A9'
'\0FACA8'	'0FACA8'	'0A0'	'\0FD'	'0FD'	'0A1'
'\0FACA0'	'0FACA0'	'0A2'	'\0FD'	'0FD'	'0A3'
'\0FACA2'	'0FACA2'	'0A4'	'\0FD'	'0FD'	'0A5'
'\0FACA4'	'0FACA4'	'0A6'	'\0FD'	'0FD'	'0A7'
'					

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	SIGTSTP	\005
3	03	3	ETX	\003	35	23	43	SIGCONT	\006	67	43	103	SIGCONT	\007
4	04	4	ENQ	\004	36	24	44	SIGQUIT	\007	68	44	104	SIGKILL	\008
5	05	5	ACK	\005	37	25	45	SIGPOLL	\009	69	45	105	SIGSTOP	\010
6	06	6	NAK	\006	38	26	46	SIGCONT	\011	70	46	106	SIGTSTP	\012
7	07	7	SYN	\007	39	27	47	SIGCONT	\013	71	47	107	SIGCONT	\014
8	08	10	CAN	\008	40	28	48	SIGPOLL	\015	72	48	108	SIGPOLL	\016
9	09	11	EOT	\009	41	29	49	SIGCONT	\017	73	49	109	SIGCONT	\018
10	0A	12	EM	\00A	42	2A	4A	SIGPOLL	\019	74	4A	110	SIGPOLL	\020
11	0B	13	END	\00B	43	2B	4B	SIGCONT	\021	75	4B	111	SIGCONT	\022
12	0C	14	KILL	\00C	44	2C	4C	SIGCONT	\023	76	4C	112	SIGCONT	\024
13	0D	15	STX	\00D	45	2D	4D	SIGPOLL	\025	77	4D	113	SIGPOLL	\026
14	0E	16	ETX	\00E	46	2E	4E	SIGCONT	\027	78	4E	114	SIGCONT	\028
15	0F	17	ENQ	\00F	47	2F	4F	SIGPOLL	\029	79	4F	115	SIGPOLL	\030
16	10	20	ACK	\010	48	30	50	SIGCONT	\031	80	50	116	SIGCONT	\032
17	11	21	NAK	\011	49	31	51	SIGPOLL	\033	81	51	117	SIGPOLL	\034
18	12	22	SYN	\012	50	32	52	SIGCONT	\035	82	52	118	SIGCONT	\036
19	13	23	CAN	\013	51	33	53	SIGPOLL	\037	83	53	119	SIGPOLL	\038
20	14	24	EOT	\014	52	34	54	SIGCONT	\039	84	54	120	SIGCONT	\040
21	15	25	EM	\015	53	35	55	SIGPOLL	\041	85	55	121	SIGPOLL	\042
22	16	26	END	\016	54	36	56	SIGCONT	\043	86	56	122	SIGCONT	\044
23	17	27	KILL	\017	55	37	57	SIGPOLL	\045	87	57	123	SIGPOLL	\046
24	18	28	STX	\018	56	38	58	SIGCONT	\047	88	58	124	SIGCONT	\048
25	19	29	ETX	\019	57	39	59	SIGPOLL	\049	89	59	125	SIGPOLL	\050
26	1A	2A	ENQ	\01A	58	3A	5A	SIGCONT	\051	90	5A	126	SIGCONT	\052
27	1B	2B	ACK	\01B	59	3B	5B	SIGPOLL	\053	91	5B	127	SIGPOLL	\054

- `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'	'00000000'	'00000000'	'\001'	'00000001'	'00000001'
'\002'	'00000002'	'00000002'	'\003'	'00000003'	'00000003'
'\004'	'00000004'	'00000004'	'\005'	'00000005'	'00000005'
'\006'	'00000006'	'00000006'	'\007'	'00000007'	'00000007'
'\010'	'00000010'	'0000000A'	'\011'	'00000011'	'0000000B'
'\012'	'00000012'	'0000000C'	'\013'	'00000013'	'0000000D'
'\014'	'00000014'	'0000000E'	'\015'	'00000015'	'0000000F'
'\016'	'00000020'	'00000010'	'\017'	'00000021'	'00000011'
'\020'	'00000040'	'00000014'	'\021'	'00000041'	'00000015'
'\022'	'00000042'	'00000016'	'\023'	'00000043'	'00000017'
'\024'	'00000044'	'00000018'	'\025'	'00000045'	'00000019'
'\026'	'00000046'	'0000001A'	'\027'	'00000047'	'0000001B'
'\030'	'00000060'	'0000001C'	'\031'	'00000061'	'0000001D'
'\032'	'00000062'	'0000001E'	'\033'	'00000063'	'0000001F'
'\034'	'00000064'	'0000001F'	'\035'	'00000065'	'00000020'
'\036'	'00000066'	'00000021'	'\037'	'00000067'	'00000022'
'\040'	'00000100'	'00000024'	'\041'	'00000101'	'00000025'
'\042'	'00000102'	'00000026'	'\043'	'00000103'	'00000027'
'\044'	'00000104'	'00000028'	'\045'	'00000105'	'00000029'
'\046'	'00000106'	'0000002A'	'\047'	'00000107'	'0000002B'
'\050'	'00000140'	'00000034'	'\051'	'00000141'	'00000035'
'\052'	'00000142'	'00000036'	'\053'	'00000143'	'00000037'
'\054'	'00000144'	'00000038'	'\055'	'00000145'	'00000039'
'\056'	'00000146'	'0000003A'	'\057'	'00000147'	'0000003B'
'\060'	'00000200'	'00000044'	'\061'	'00000201'	'00000045'
'\062'	'00000202'	'00000046'	'\063'	'00000203'	'00000047'
'\064'	'00000204'	'00000048'	'\065'	'00000205'	'00000049'
'\066'	'00000206'	'0000004A'	'\067'	'00000207'	'0000004B'
'\070'	'00000240'	'00000054'	'\071'	'00000241'	'00000055'
'\072'	'00000242'	'00000056'	'\073'	'00000243'	'00000057'
'\074'	'00000244'	'00000058'	'\075'	'00000245'	'00000059'
'\076'	'00000246'	'0000005A'	'\077'	'00000247'	'0000005B'
'\080'	'00000300'	'00000064'	'\081'	'00000301'	'00000065'
'\082'	'00000302'	'00000066'	'\083'	'00000303'	'00000067'
'\084'	'00000304'	'00000068'	'\085'	'00000305'	'00000069'
'\086'	'00000306'	'0000006A'	'\087'	'00000307'	'0000006B'
'\090'	'00000340'	'00000074'	'\091'	'00000341'	'00000075'
'\092'	'00000342'	'00000076'	'\093'	'00000343'	'00000077'
'\094'	'00000344'	'00000078'	'\095'	'00000345'	'00000079'
'\096'	'00000346'	'0000007A'	'\097'	'00000347'	'0000007B'
'\098'	'00000380'	'00000084'	'\099'	'00000381'	'00000085'
'\100'	'00000382'	'00000086'	'\101'	'00000383'	'00000087'
'\102'	'00000384'	'00000088'	'\103'	'00000385'	'00000089'
'\104'	'00000386'	'0000008A'	'\105'	'00000387'	'0000008B'
'\106'	'00000388'	'0000008C'	'\107'	'00000389'	'0000008D'
'\108'	'0000038A'	'0000008E'	'\109'	'0000038B'	'0000008F'
'\110'	'0000038C'	'0000008F'	'\111'	'0000038D'	'00000090'
'\112'	'0000038E'	'00000091'	'\113'	'0000038F'	'00000092'
'\114'	'00000390'	'00000093'	'\115'	'00000391'	'00000094'
'\116'	'00000392'	'00000095'	'\117'	'00000393'	'00000096'
'\118'	'00000394'	'00000097'	'\119'	'00000395'	'00000098'
'\120'	'00000396'	'00000099'	'\121'	'00000397'	'0000009A'
'\122'	'00000398'	'0000009B'	'\123'	'00000399'	'0000009C'
'\124'	'0000039A'	'0000009D'	'\125'	'0000039B'	'0000009E'
'\126'	'0000039C'	'0000009F'	'\127'	'0000039D'	'000000A0'
'\128'	'00000400'	'000000A4'	'\129'	'00000401'	'000000A5'
'\130'	'00000402'	'000000A6'	'\131'	'00000403'	'000000A7'
'\132'	'00000404'	'000000A8'	'\133'	'00000405'	'000000A9'
'\134'	'00000406'	'000000AA'	'\135'	'00000407'	'000000AB'
'\136'	'00000408'	'000000AC'	'\137'	'00000409'	'000000AD'
'\138'	'0000040A'	'000000AE'	'\139'	'0000040B'	'000000AF'
'\140'	'0000040C'	'000000B4'	'\141'	'0000040D'	'000000B5'
'\142'	'0000040E'	'000000B6'	'\143'	'0000040F'	'000000B7'
'\144'	'00000410'	'000000B8'	'\145'	'00000411'	'000000B9'
'\146'	'00000412'	'000000BA'	'\147'	'00000413'	'000000BB'
'\148'	'00000414'	'000000BC'	'\149'	'00000415'	'000000BD'
'\150'	'00000416'	'000000BE'	'\151'	'00000417'	'000000BF'
'\152'	'00000418'	'000000C4'	'\153'	'00000419'	'000000C5'
'\154'	'0000041A'	'000000C6'	'\155'	'0000041B'	'000000C7'
'\156'	'0000041C'	'000000C8'	'\157'	'0000041D'	'000000C9'
'\158'	'0000041E'	'000000CA'	'\159'	'0000041F'	'000000CB'
'\160'	'00000420'	'000000D4'	'\161'	'00000421'	'000000D5'
'\162'	'00000422'	'000000D6'	'\163'	'00000423'	'000000D7'
'\164'	'00000424'	'000000D8'	'\165'	'00000425'	'000000D9'
'\166'	'00000426'	'000000DA'	'\167'	'00000427'	'000000DB'
'\168'	'00000428'	'000000DC'	'\169'	'00000429'	'000000DD'
'\170'	'0000042A'	'000000E4'	'\171'	'0000042B'	'000000E5'
'\172'	'0000042C'	'000000E6'	'\173'	'0000042D'	'000000E7'
'\174'	'0000042E'	'000000E8'	'\175'	'0000042F'	'000000E9'
'\176'	'00000430'	'000000FA'	'\177'	'00000431'	'000000FB'
'\178'	'00000432'	'000000FC'	'\179'	'00000433'	'000000FD'
'\180'	'00000434'	'000000FE'	'\181'	'00000435'	'000000FF'
'\182'	'00000436'	'000000F0'	'\183'	'00000437'	'000000F1'
'\184'	'00000438'	'000000F2'	'\185'	'00000439'	'000000F3'
'\186'	'0000043A'	'000000F4'	'\187'	'0000043B'	'000000F5'
'\188'	'0000043C'	'000000F6'	'\189'	'0000043D'	'000000F7'
'\190'	'0000043E'	'000000F8'	'\191'	'0000043F'	'000000F9'
'\192'	'00000440'	'000000FA'	'\193'	'00000441'	'000000FB'
'\194'	'00000442'	'000000FC'	'\195'	'00000443'	'000000FD'
'\196'	'00000444'	'000000FE'	'\197'	'00000445'	'000000FF'
'\198'	'00000446'	'000000F0'	'\199'	'00000447'	'000000F1'
'\200'	'00000448'	'000000F2'	'\201'	'00000449'	'000000F3'
'\202'	'0000044A'	'000000F4'	'\203'	'0000044B'	'000000F5'
'\204'	'0000044C'	'000000F6'	'\205'	'0000044D'	'000000F7'
'\206'	'0000044E'	'000000F8'	'\207'	'0000044F'	'000000F9'
'\208'	'00000450'	'000000FA'	'\209'	'00000451'	'000000FB'
'\210'	'00000452'	'000000FC'	'\211'	'00000453'	'000000FD'
'\212'	'00000454'	'000000FE'	'\213'	'00000455'	'000000FF'
'\214'	'00000456'	'000000F0'	'\215'	'00000457'	'000000F1'
'\216'	'00000458'	'000000F2'	'\217'	'00000459'	'000000F3'
'\218'	'0000045A'	'000000F4'	'\219'	'0000045B'	'000000F5'
'\220'	'0000045C'	'000000F6'	'\221'	'0000045D'	'000000F7'
'\222'	'0000045E'	'000000F8'	'\223'	'0000045F'	'000000F9'
'\224'	'00000460'	'000000FA'	'\225'	'00000461'	'000000FB'
'\226'	'00000462'	'000000FC'	'\227'	'00000463'	'000000FD'
'\228'	'00000464'	'000000FE'	'\229'	'00000465'	'000000FF'
'\230'	'00000466'	'000000F0'	'\231'	'00000467'	'000000F1'
'\232'	'00000468'	'000000F2'	'\233'	'00000469'	'000000F3'
'\234'	'0000046A'	'000000F4'	'\235'	'0000046B'	'000000F5'
'\236'	'0000046C'	'000000F6'	'\237'	'0000046D'	'000000F7'
'\238'	'0000046E'	'000000F8'	'\239'	'0000046F'	'000000F9'
'\240'	'00000470'	'000000FA'	'\241'	'00000471'	'000000FB'
'\242'	'00000472'	'000000FC'	'\243'	'00000473'	'000000FD'
'\244'	'00000474'	'000000FE'	'\245'	'00000475'	'000000FF'
'\246'	'00000476'	'000000F0'	'\247'	'00000477'	'000000F1'
'\248'	'00000478'	'000000F2'	'\249'	'00000479'	'000000F3'
'\250'	'0000047A'	'000000F4'	'\251'	'0000047B'	'000000F5'
'\252'	'0000047C'	'000000F6'	'\253'	'0000047D'	'000000F7'
'\254'	'0000047E'	'000000F8'	'\255'	'0000047F'	'000000F9'
'\256'	'00000480'	'000000FA'	'\257'	'00000481'	'000000FB'
'\258'	'00000482'	'000000FC'	'\259'	'00000483'	'000000FD'
'\260'	'00000484'	'000000FE'	'\261'	'00000485'	'000000FF'
'\262'	'00000486'	'000000F0'	'\263'	'00000487'	'000000F1'
'\264'	'00000488'	'000000F2'	'\265'	'00000489'	'000000F3'
'\266'	'0000048A'	'000000F4'	'\267'	'0000048B'	'000000F5'
'\268'	'0000048C'	'000000F6'	'\269'	'0000048D'	'000000F7'
'\270'	'0000048E'	'000000F8'	'\271'	'0000048F'	'000000F9'
'\272'	'00000490'	'000000FA'	'\273'	'00000491'	'000000FB'
'\274'	'00000492'	'000000FC'	'\275'	'00000493'	'000000FD'
'\276'	'00000494'	'000000FE'	'\277'	'00000495'	'000000FF'
'\278'	'00000496'	'000000F0'	'\279'	'00000497'	'000000F1'
'\280'	'00000498'	'000000F2'	'\281'	'00000499'	'000000F3'
'\282'	'0000049A'	'000000F4'	'\283'	'0000049B'	'000000F5'
'\284'	'0000049C'	'000000F6'	'\285'	'0000049D'	'000000F7'
'\286'	'0000049E'	'000000F8'	'\287'	'0000049F'	'000000F9'
'\288'	'000004A0'	'000000FA'	'\289'	'000004A1'	'000000FB'
'\290'	'000004A2'	'000000FC'	'\291'	'000004A3'	'000000FD'
'\292'	'000004A4'	'000000FE'	'\293'	'000004A5'	'000000FF'
'\294'	'000004A6'	'000000F0'	'\295'	'000004A7'	'000000F1'
'\296'	'000004A8'	'000000F2'	'\297'	'000004A9'	'000000F3'
'\298'	'000004AA'	'000000F4'	'\299'	'000004AB'	'000000F5'
'\300'	'000004AC'	'000000F6'	'\301'	'000004AD'	'000000F7'
'\302'	'000004AE'	'000000F8'	'\303'	'000004AF'	'000000F9'
'\304'	'000004B0'	'000000FA'	'\305'	'000004B1'	'000000FB'
'\306'	'000004B2'	'000000FC'	'\307'	'000004B3'	'000000FD'
'\308'	'000004B4'	'000000FE'	'\309'	'000004B5'	'000000FF'
'\310'	'000004B6'	'000000F0'	'\311'	'000004B7'	'000000F1'
'\312'	'000004B8'	'000000F2'	'\313'	'000004B9'	'000000F3'
'\314'	'000004BA'	'000000F4'	'\315'	'000004BB'	'000000F5'
'\316'	'000004BC'	'000000F6'	'\317'	'000004BD'	'000000F7'
'\318'	'000004BE'	'000000F8'	'\319'	'000004BF'	'000000F9'
'\320'	'000004C0'	'000000FA'	'\321'	'000004C1'	'000000FB'
'\322'	'000004C2'	'000000FC'	'\323'	'000004C3'	'000000FD'
'\324'	'000004C4'	'000000FE'	'\325'	'000004C5'	'000000FF'
'\326'	'000004C6'	'000000F0'	'\327'	'000004C7'	'000000F1'
'\328'	'000004C8'	'000000F2'	'\329'	'000004C9'	'000000F3'
'\330'	'000004CA'	'000000F4'	'\331'	'000004CB'	'000000F5'
'\332'	'000004CC'	'000000F6'	'\333'	'000004CD'	'000000F7'
'\334'	'000004CE'	'000000F8'	'\335'	'000004CF'	'000000F9'
'\336'	'000004D0'	'000000FA'	'\337'	'000004D1'	'000000FB'
'\338'	'000004D2'	'000000FC'	'\339'	'000004D3'	'000000FD'
'\340'	'000004D4'	'000000FE'	'\341'	'000004D5'	'000000FF'
'\342'	'000004D6'	'000000F0'	'\343'	'000004D7'	'000000F1'</td

In Pairs or Triples...

Some review and some novel challenges:

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



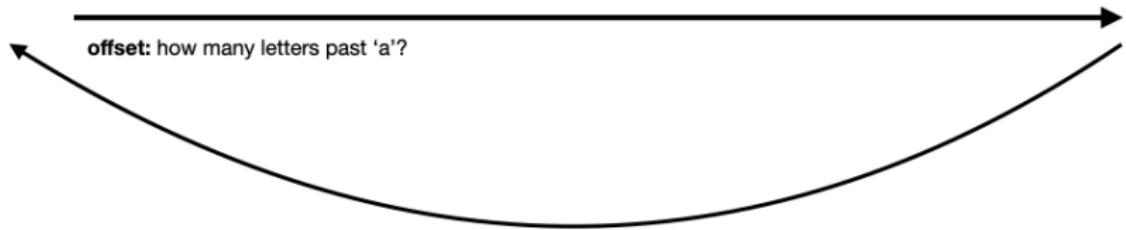
Python Tutor

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

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[:-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.

```
"Friday\xSaturday\xSunday"  
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")
```

```
"FridayXXXXsaturdayXXXXsundayXXXX"
```

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\xSaturday\xSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

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

```
"Fri\xySatur\xySun\xy"  
days = ['Fri', 'satur', 'sSun']
```

Today's Topics



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

Recap

- In Python, we introduced:

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

Recap

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

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

Recap

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

- In Python, we introduced:

- ▶ For-loops
- ▶ range()

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

5 Minute Break!



Today we have a second lecture portion! Take a quick break.

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
You must submit a file that contains python instructions and comments ONLY.

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
*You must submit a file that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.*

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
*You must submit a file that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.*

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
You must submit a file that contains python instructions and comments ONLY. Don't submit screenshots, those are images and the grading script cannot run and test your program that way. Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.
- **I missed the deadline for a programming assignment. What should I do?**

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
You must submit a file that contains python instructions and comments ONLY. Don't submit screenshots, those are images and the grading script cannot run and test your program that way. Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.
- **I missed the deadline for a programming assignment. What should I do?**
We do not accept late work but we drop the lowest 5 program grades.

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
*You must submit a file that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.*
- **I missed the deadline for a programming assignment. What should I do?**
*We do not accept late work but we drop the lowest 5 program grades.
Due dates are one week late to allow flexibility for emergencies.*

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
*You must submit a file that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.*
- **I missed the deadline for a programming assignment. What should I do?**
*We do not accept late work but we drop the lowest 5 program grades.
Due dates are one week late to allow flexibility for emergencies.
You must work on THIS WEEK'S PROGRAMS, that way you will never miss a deadline!!!*

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
*You must submit a file that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.*
- **I missed the deadline for a programming assignment. What should I do?**
*We do not accept late work but we drop the lowest 5 program grades.
Due dates are one week late to allow flexibility for emergencies.
You must work on THIS WEEK'S PROGRAMS, that way you will never miss a deadline!!!*
- **There is a typo in the slides, should I report it?**

Frequently Asked Questions

From emails.

- **Gradescope does not give me credit but my program runs on my computer.**
You must submit a file that contains python instructions and comments ONLY. Don't submit screenshots, those are images and the grading script cannot run and test your program that way. Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.
- **I missed the deadline for a programming assignment. What should I do?**
We do not accept late work but we drop the lowest 5 program grades. Due dates are one week late to allow flexibility for emergencies. You must work on THIS WEEK'S PROGRAMS, that way you will never miss a deadline!!!
- **There is a typo in the slides, should I report it?**
Yes, great catch! We really appreciate it when you tell us about any typos or errors, please send us email.

Today's Topics



- More on Strings
- Arithmetic
- Indexing and Slicing Lists
- Colors & Hexadecimal Notation

Today's Topics



- More on Strings
- Arithmetic
- Indexing and Slicing Lists
- Colors & Hexadecimal Notation

More on Strings...

From Final Exam, Fall 2017, Version 1, #1:

Name:

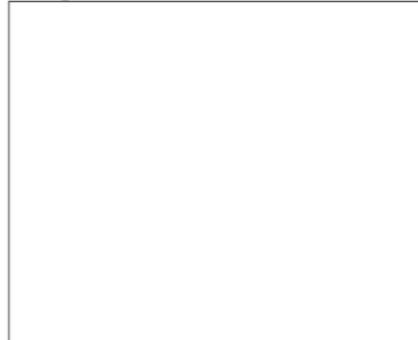
EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



More on Strings...

Name: _____

EmpID: _____

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:
`There are ??? fun days in a week`

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

There are ??? fun days in a week
Two of them are ???

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

There are ??? fun days in a week
Two of them are ???
My favorite ??? is Saturday.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

There are ??? fun days in a week
Two of them are ???
My favorite ??? is Saturday.
- Will get 1/3 to 1/2 points for writing down the basic structure.

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...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

There are 3 fun days in a week

Two of them are ???

My favorite ??? is Saturday.

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.

```
"Friday\xSaturday\xSunday"  
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")
```

```
"FridayXXXXsaturdayXXXXsundayXXXX"
```

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\xSaturday\xSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

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

```
"Fri\xySatur\xySun\xy"  
days = ['Fri', 'satur', 'sSun']
```

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Don't leave it blank— write what you know & puzzle out as much as possible:

There are 3 fun days in a week
Two of them are Friday Sunday
My favorite ??? is Saturday.

Today's Topics



- More on Strings
- **Arithmetic**
- Indexing and Slicing Lists
- Colors & Hexadecimal Notation

Arithmetic

Some arithmetic operators in Python:

- Addition:



Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`



Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction:



Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`



Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication:



Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`



Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division:

Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`

Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:

Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
`weeks = totalDays // 7`

$15 // 7 = 2$

Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
`weeks = totalDays // 7` 15 // 7 = 2
- Remainder or Modulus:

Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
`weeks = totalDays // 7` $15 // 7 = 2$
- Remainder or Modulus:
`days = totalDays % 7` $15 \% 7 = 1$

Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
`weeks = totalDays // 7` $15 // 7 = 2$
- Remainder or Modulus:
`days = totalDays % 7` $15 \% 7 = 1$
- Exponentiation:

Arithmetic



Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
`weeks = totalDays // 7` $15 // 7 = 2$
- Remainder or Modulus:
`days = totalDays % 7` $15 \% 7 = 1$
- Exponentiation:
`pop = 2**time`

Challenge:

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

Challenge:

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

Challenge:

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.

Challenge:

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.
- If the user enters, 8 and 20.

Challenge:

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.
- If the user enters, 8 and 20.
- If the user enters, 11 and 1.

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

```
| Enter starting time: 9
```

```
| Enter how long: 2
```

```
| Your event starts at 9 o'clock.
```

```
| Your event ends at 11 o'clock.
```

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 12 and 4.

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 12 and 4.

Enter starting time: 12

Enter how long: 4

Your event starts at 12 o'clock.

Your event ends at 4 o'clock.

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 8 and 20.

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 8 and 20.

Enter starting time: 8

Enter how long: 20

Your event starts at 8 o'clock.

Your event ends at 4 o'clock.

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 11 and 1.

Challenge:

What does this code do?

#Mystery code for lecture 3

```
startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 11 and 1.

Enter starting time: 11

Enter how long: 1

Your event starts at 11 o'clock.

Your event ends at 0 o'clock.

Today's Topics



- More on Strings
- Arithmetic
- **Indexing and Slicing Lists**
- Colors & Hexadecimal Notation

Challenge:

Mostly review:

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Python Tutor

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

(Demo with pythonTutor)

Review: range()



The three versions:

Review: range()



The three versions:

- range(stop)

Review: range()



The three versions:

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

Review: range()



The three versions:

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

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[5:8]`

gives: "Uni"

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[5:8]`

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

gives: "Uni"

- Also works for lists:

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[5:8]`

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

gives: "Uni"

- Also works for lists:

`names[1:3]`

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[5:8]`

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

gives: "Uni"

- Also works for lists:

`names[1:3]`

gives: ["Anna", "Alice"]

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

`s[5:8]`

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blast off!")
4
5 for num in range(5,8):
6     print(num, 2*num)
7
8 s = "City University of New York"
9 print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

gives: "Uni"

- Also works for lists:

`names[1:3]`

gives: ["Anna", "Alice"]

- Python also lets you “count backwards”: last element has index: `-1`.

Today's Topics



- More on Strings
- Arithmetic
- Indexing and Slicing Lists
- **Colors & Hexadecimal Notation**

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by name.

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

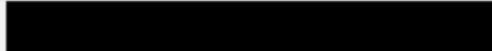
- Can specify by name.
- Can specify by numbers:

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).
 - ▶ Adding light, not paint:

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).
 - ▶ Adding light, not paint:
 - ★ Black: 0% red, 0% green, 0% blue

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).
 - ▶ Adding light, not paint:
 - ★ Black: 0% red, 0% green, 0% blue
 - ★ White: 100% red, 100% green, 100% blue

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):
 - ▶ Fractions of each:

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.

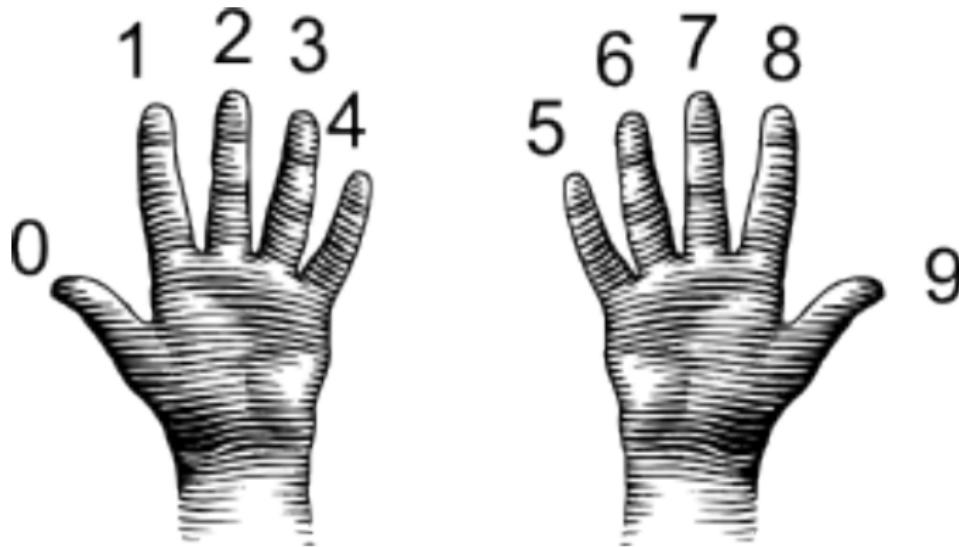
Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.
 - ▶ Hexcodes (base-16 numbers)...

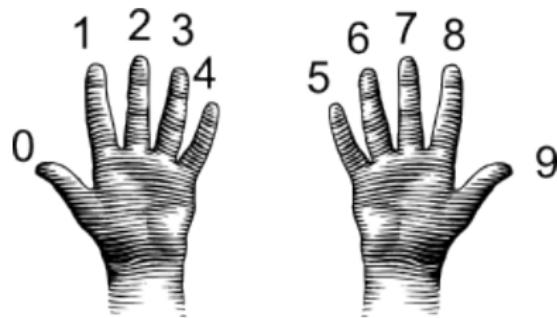
Decimal & Hexadecimal Numbers

Counting with 10 digits:



(from i-programmer.info)

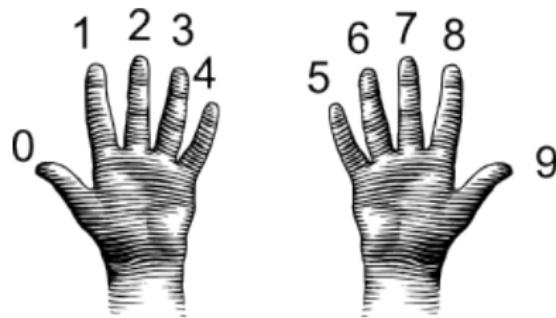
Decimal



(from i-programmer.info)

Decimal

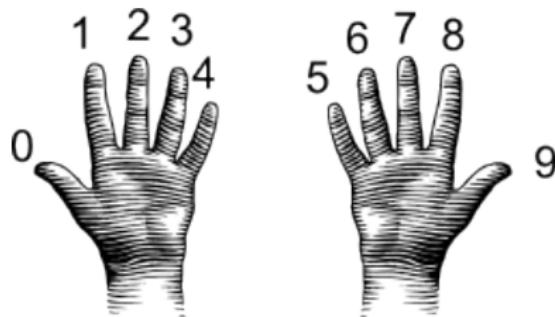
00 01 02 03 04 05 06 07 08 09



(from i-programmer.info)

Decimal

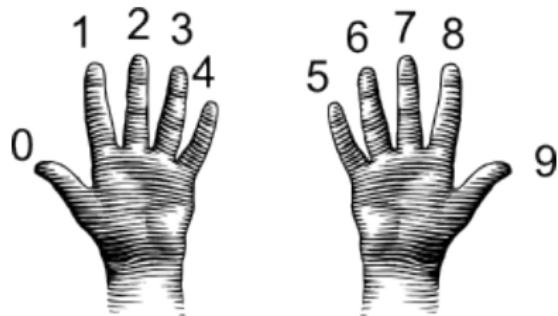
00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19



(from i-programmer.info)

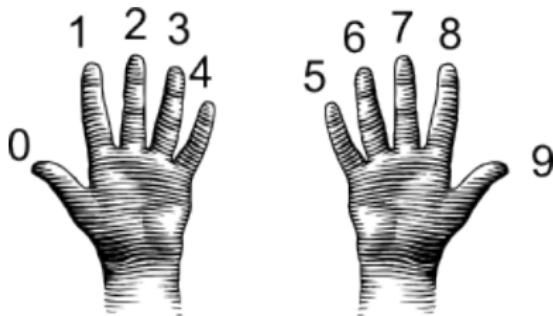
Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29



(from i-programmer.info)

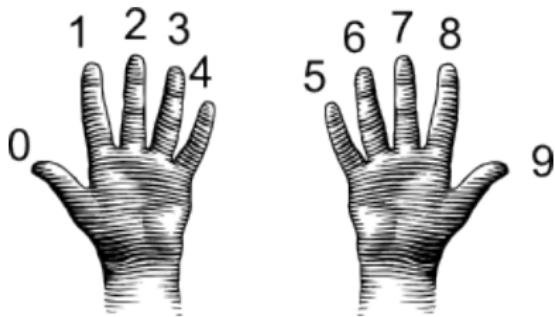
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

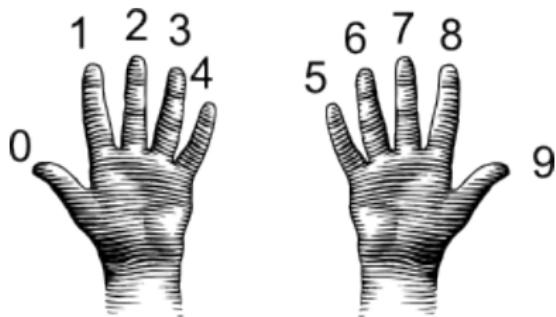
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49

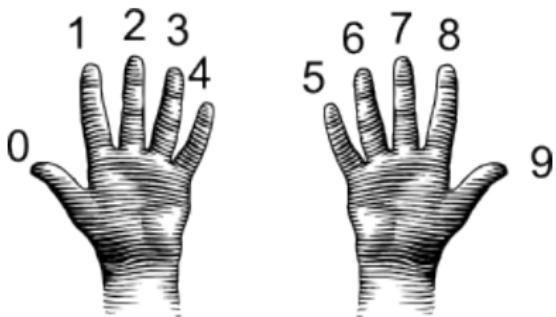
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

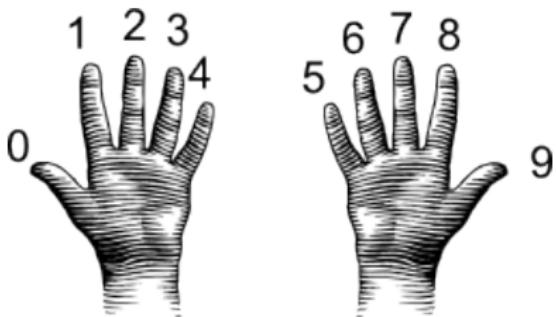
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69

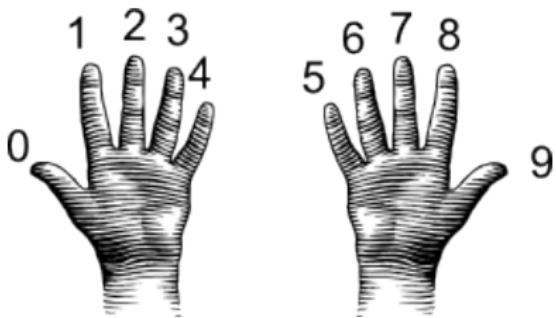
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79

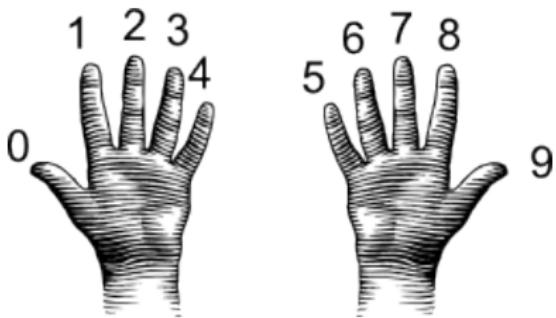
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
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

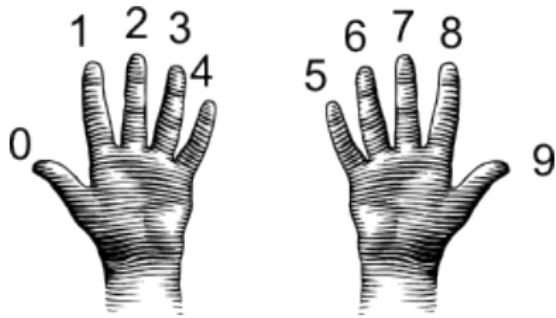
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
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	98	99

Decimal



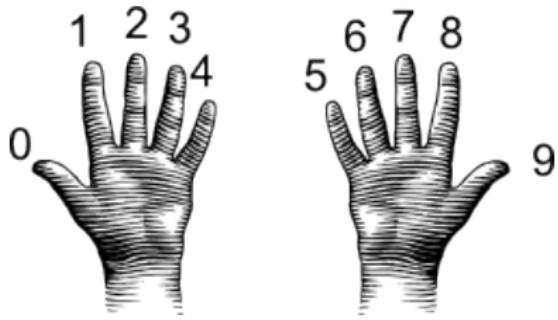
(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
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	98	99

$$10^1 + 10^0$$

Max Number = 99

Decimal



(from i-programmer.info)

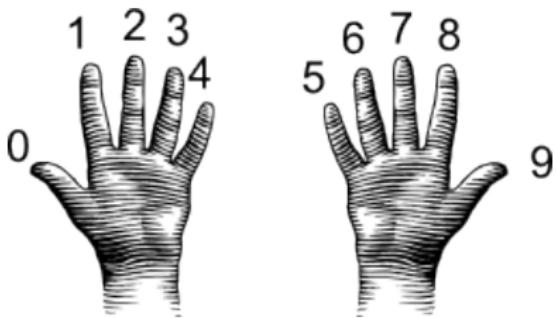
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

$$10^1 + 10^0$$

Max Number = 99

$$90 = (9 * 10^1) + (0 * 10^0)$$

Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
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	98	99

$$10^1 + 10^0$$

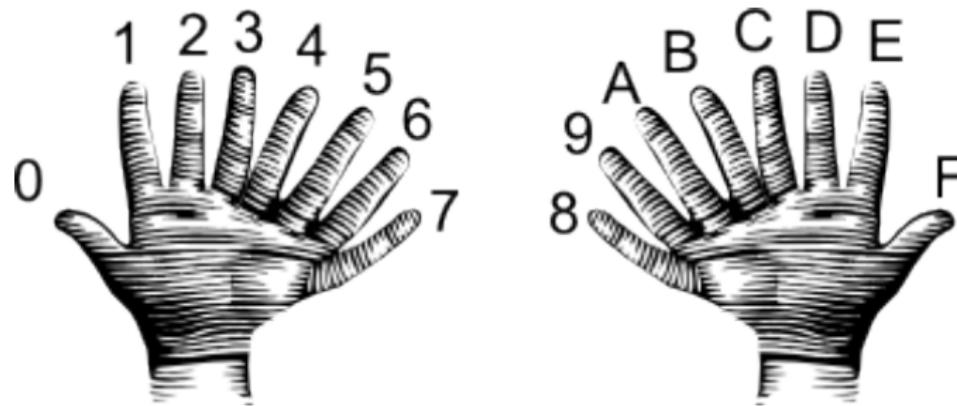
Max Number = 99

$$90 = (9 * 10^1) + (0 * 10^0)$$

$$99 = (9 * 10^1) + (9 * 10^0)$$

Decimal & Hexadecimal Numbers

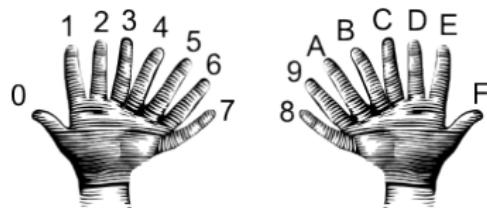
Counting with 16 digits:



(from i-programmer.info)

Hexadecimal

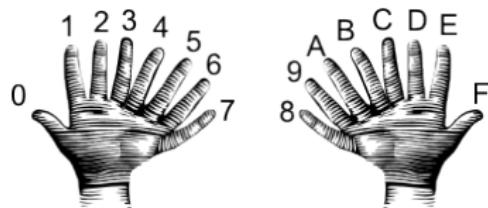
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F



(from i-programmer.info)

Hexadecimal

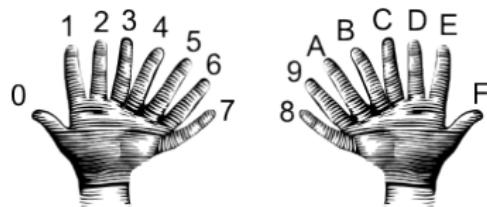
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F



(from i-programmer.info)

Hexadecimal

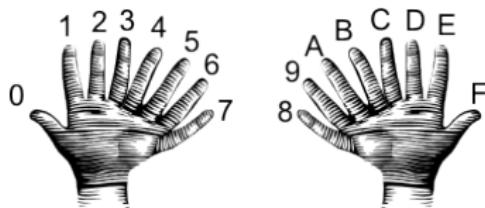
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F



(from i-programmer.info)

Hexadecimal

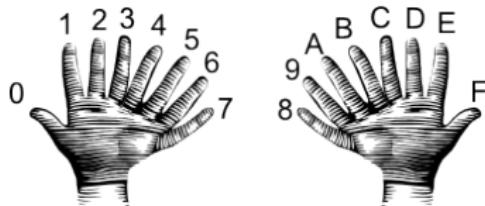
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F



(from i-programmer.info)

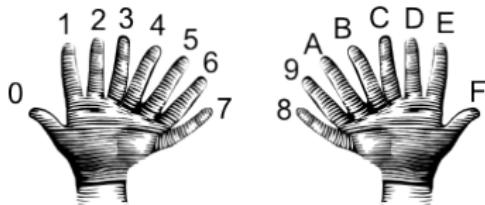
Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F



(from i-programmer.info)

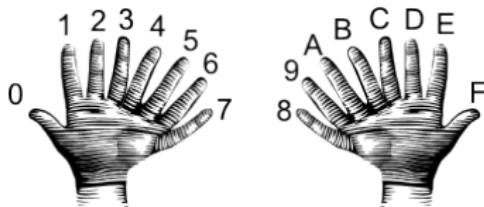
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

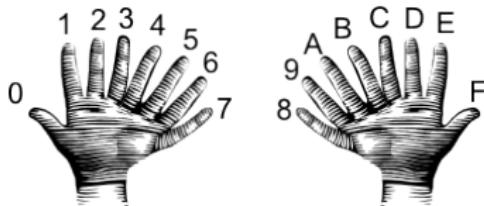
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F

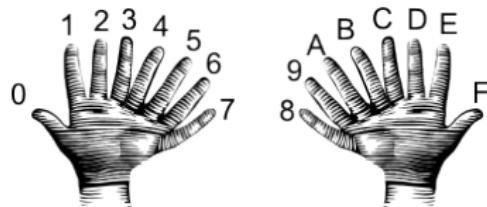
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F

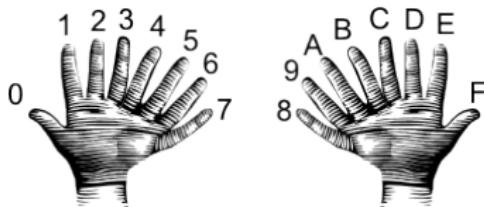
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

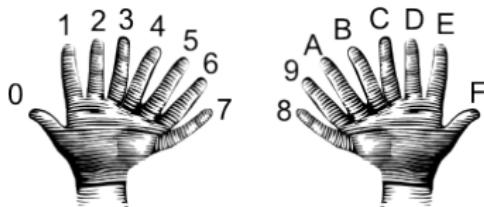
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F

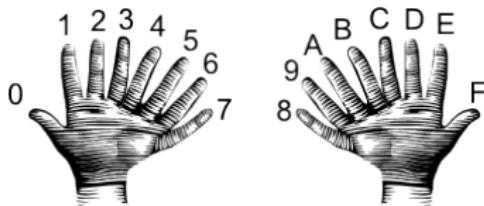
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF

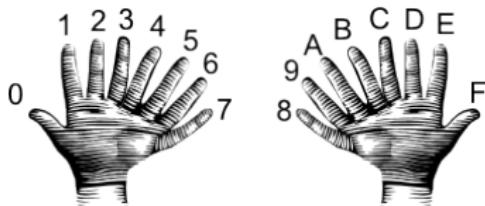
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF

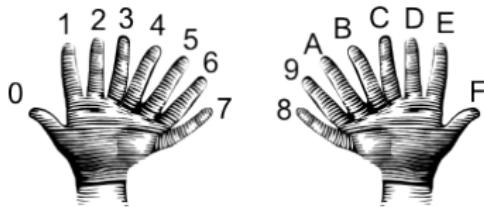
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

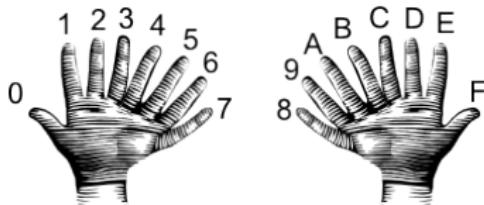
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

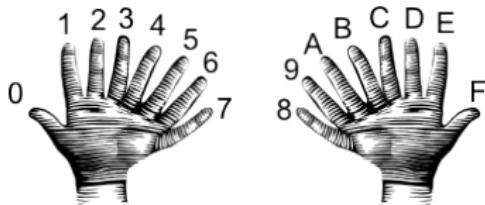
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF

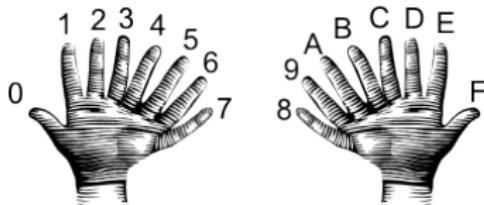
Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

Hexadecimal

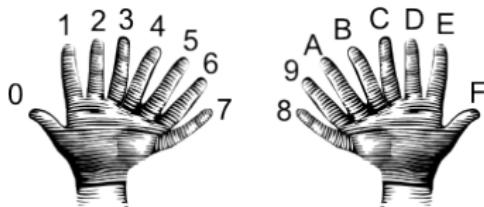


(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

$$16^1 + 16^0$$

Hexadecimal



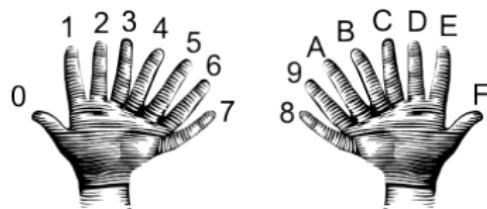
(from i-programmer.info)

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

$$16^1 + 16^0$$

Max Number = 255

Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

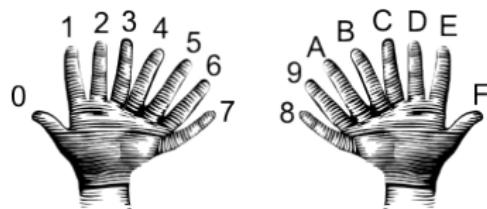
$$16^1 + 16^0$$

Max Number = 255

$$F0 = (F * 16^1) + (0 * 16^0)$$

$$F0 = (240) + (0) = 240$$

Hexadecimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

$$16^1 + 16^0$$

Max Number = 255

$$F0 = (F * 16^1) + (0 * 16^0)$$

$$F0 = (240) + (0) = 240$$

$$FF = (F * 16^1) + (F * 16^0)$$

$$FF = (240) + (15) = 255$$

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.
 - ▶ Hexcodes (base-16 numbers):

Colors

Color Name	HEX	Color
<u>Black</u>	<u>#000000</u>	
<u>Navy</u>	<u>#000080</u>	
<u>DarkBlue</u>	<u>#00008B</u>	
<u>MediumBlue</u>	<u>#0000CD</u>	
<u>Blue</u>	<u>#0000FF</u>	

- Can specify by numbers (RGB):
 - ▶ Fractions of each:
e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
 - ▶ 8-bit colors: numbers from 0 to 255:
e.g. (0, 255, 0) is no red, 100% green, and no blue.
 - ▶ Hexcodes (base-16 numbers):
e.g. #0000FF is no red, no green, and 100% blue.

Challenge:

Some review and some novel challenges:

```
1 import turtle
2 teddy = turtle.Turtle()
3
4 names = ["violet", "purple", "indigo", "lavender"]
5 for c in names:
6     teddy.color(c)
7     teddy.left(60)
8     teddy.forward(40)
9     teddy.dot(10)
10
11 teddy.penup()
12 teddy.forward(100)
13 teddy.pendown()
14
15 hexNames = ["#FF00FF", "#990099", "#550055", "#111111"]
16 for c in hexNames:
17     teddy.color(c)
18     teddy.left(60)
19     teddy.forward(40)
20     teddy.dot(10)
```



Trinkets

```
1 import turtle
2 teddy = turtle.Turtle()
3
4 names = ["violet", "purple", "indigo", "lavender"]
5 for c in names:
6     teddy.color(c)
7     teddy.left(60)
8     teddy.forward(40)
9     teddy.dot(10)
10
11 teddy.penup()
12 teddy.forward(100)
13 teddy.pendown()
14
15 hexNames = ["#FF00FF", "#990099", "#550055", "#111111"]
16 for c in hexNames:
17     teddy.color(c)
18     teddy.left(60)
19     teddy.forward(40)
20     teddy.dot(10)
```

(Demo with trinkets)

Recap



- In Python, we introduced:

Recap



- In Python, we introduced:
 - ▶ Indexing and Slicing Lists

Recap



- In Python, we introduced:
 - ▶ Indexing and Slicing Lists
 - ▶ Arithmetic

Recap



- In Python, we introduced:
 - ▶ Indexing and Slicing Lists
 - ▶ Arithmetic
 - ▶ Colors

Recap



- In Python, we introduced:
 - ▶ Indexing and Slicing Lists
 - ▶ Arithmetic
 - ▶ Colors
 - ▶ Hexadecimal Notation

Class Reminders!



Before next class, don't forget to:

- Review this week's Lab

Class Reminders!



Before next class, don't forget to:

- Review this week's Lab
- Take the Lab Quiz



Class Reminders!



Before next class, don't forget to:

- Review this week's Lab
- Take the Lab Quiz 
- Submit this class's 5 programming assignments (programs 6-15)