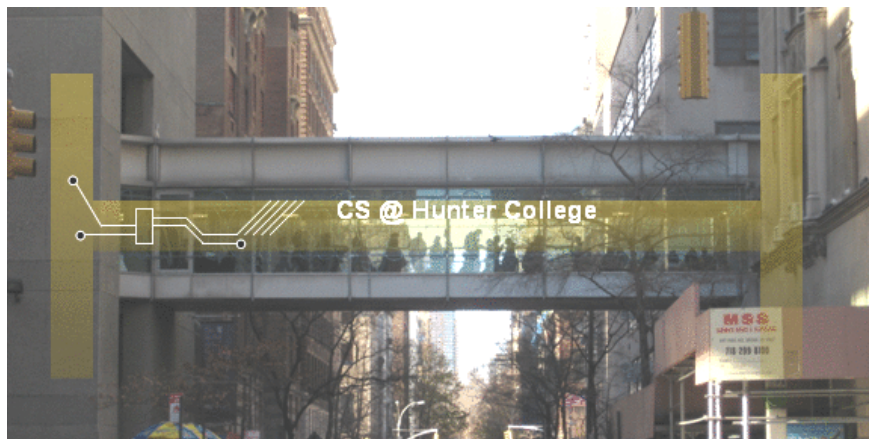


CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From email

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

- **Can I work ahead?**

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

- **Can I work ahead?**

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

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

- **Can I work ahead?**

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

- **When is the midterm?**

Frequently Asked Questions

From email

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

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

When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Lab Quiz (Gradescope) can be submitted only once.

Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

- **Can I work ahead?**

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

- **When is the midterm?**

There is no midterm. Instead there's required quizzes and programming assignments.

Today's Topics



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

In Pairs or Triples...

Some review and some novel challenges:

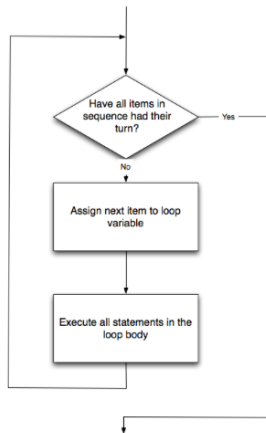
```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12   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

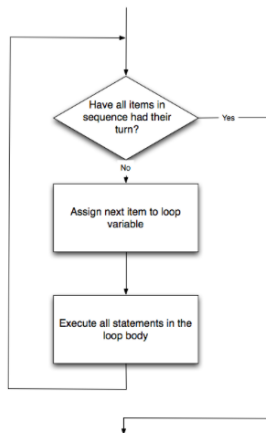
for-loop



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

How to Think Like CS, §4.5

for-loop



How to Think Like CS, §4.5

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

where `list` is a list of items:

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

Today's Topics



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

More on range():

```
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 the list `[0,1,2,3,...,100]`, you would write:

range()



Simplest version:

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

```
range(101)
```


`range()`

What if you wanted to start somewhere else:



range()

What if you wanted to start somewhere else:

- `range(start, stop)`



range()



What if you wanted to start somewhere else:

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

range()



What if you wanted to start somewhere else:

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

range()



What if you wanted to start somewhere else:

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

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

range()

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



range()

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

- `range(start, stop, step)`



range()

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

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



range()

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

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



range()

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

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

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



In summary: `range()`



The three versions:

In summary: `range()`



The three versions:

- `range(stop)`

In summary: `range()`



The three versions:

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

In summary: `range()`



The three versions:

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

Today's Topics



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

Variables

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



Variables

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



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

(wiki)

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	!	17	Q	33	A	49	1
2	"	18	R	34	B	50	2
3	#	19	S	35	C	51	3
4	\$	20	T	36	D	52	4
5	%	21	U	37	E	53	5
6	&	22	V	38	F	54	6
7	'	23	W	39	G	55	7
8	(24	X	40	H	56	8
9)	25	Y	41	I	57	9
10	*	26	Z	42	J	58	.
11	+	27	[43	K	59	,
12	,	28	\	44	L	60	:
13	-	29]	45	M	61	;
14	.	30	^	46	N	62	'
15	/	31	_	47	O	63	~
16		32	`	48	P	64	
17	A	33	a	49	Q	65	
18	B	34	b	50	R	66	
19	C	35	c	51	S	67	
20	D	36	d	52	T	68	
21	E	37	e	53	U	69	
22	F	38	f	54	V	70	
23	G	39	g	55	W	71	
24	H	40	h	56	X	72	
25	I	41	i	57	Y	73	
26	J	42	j	58	Z	74	
27	K	43	k	59	[75	
28	L	44	l	60	\	76	
29	M	45	m	61]	77	
30	N	46	n	62	^	78	
31	O	47	o	63	_	79	
32	P	48	p	64		80	
33	Q	49	q	65		81	
34	R	50	r	66		82	
35	S	51	s	67		83	
36	T	52	t	68		84	
37	U	53	u	69		85	
38	V	54	v	70		86	
39	W	55	w	71		87	
40	X	56	x	72		88	
41	Y	57	y	73		89	
42	Z	58	z	74		90	
43	[59	{	75		91	
44	\	60		76		92	
45]	61	}	77		93	
46	^	62	~	78		94	
47	_	63		79		95	
48		64		80		96	
49		65		81		97	
50		66		82		98	
51		67		83		99	
52		68		84		100	
53		69		85		101	
54		70		86		102	
55		71		87		103	
56		72		88		104	
57		73		89		105	
58		74		90		106	
59		75		91		107	
60		76		92		108	
61		77		93		109	
62		78		94		110	
63		79		95		111	
64		80		96		112	
65		81		97		113	
66		82		98		114	
67		83		99		115	
68		84		100		116	
69		85		101		117	
70		86		102		118	
71		87		103		119	
72		88		104		120	
73		89		105		121	
74		90		106		122	
75		91		107		123	
76		92		108		124	
77		93		109		125	
78		94		110		126	
79		95		111		127	
80		96		112		128	
81		97		113		129	
82		98		114		130	
83		99		115		131	
84		100		116		132	
85		101		117		133	
86		102		118		134	
87		103		119		135	
88		104		120		136	
89		105		121		137	
90		106		122		138	
91		107		123		139	
92		108		124		140	
93		109		125		141	
94		110		126		142	
95		111		127		143	
96		112		128		144	
97		113		129		145	
98		114		130		146	
99		115		131		147	
100		116		132		148	
101		117		133		149	
102		118		134		150	
103		119		135		151	
104		120		136		152	
105		121		137		153	
106		122		138		154	
107		123		139		155	
108		124		140		156	
109		125		141		157	
110		126		142		158	
111		127		143		159	
112		128		144		160	
113		129		145		161	
114		130		146		162	
115		131		147		163	
116		132		148		164	
117		133		149		165	
118		134		150		166	
119		135		151		167	
120		136		152		168	
121		137		153		169	
122		138		154		170	
123		139		155		171	
124		140		156		172	
125		141		157		173	
126		142		158		174	
127		143		159		175	
128		144		160		176	
129		145		161		177	
130		146		162		178	
131		147		163		179	
132		148		164		180	
133		149		165		181	
134		150		166		182	
135		151		167		183	
136		152		168		184	
137		153		169		185	
138		154		170		186	
139		155		171		187	
140		156		172		188	
141		157		173		189	
142		158		174		190	
143		159		175		191	
144		160		176		192	
145		161		177		193	
146		162		178		194	
147		163		179		195	
148		164		180		196	
149		165		181		197	
150		166		182		198	
151		167		183		199	
152		168		184		200	
153		169		185		201	
154		170		186		202	
155		171		187		203	
156		172		188		204	
157		173		189		205	
158		174		190		206	
159		175		191		207	
160		176		192		208	
161		177		193		209	
162		178		194		210	
163		179		195		211	
164		180		196		212	
165		181		197		213	
166		182		198		214	
167		183		199		215	
168		184		200		216	
169		185		201		217	
170		186		202		218	
171		187		203		219	
172		188		204		220	
173		189		205		221	
174		190		206		222	
175		191		207		223	
176		192		208		224	
177		193		209		225	
178		194		210		226	
179		195		211		227	
180		196		212		228	
181		197		213		229	
182		198		214		230	
183		199		215		231	
184		200		216		232	
185		201		217		233	
186		202		218		234	
187		203		219		235	
188		204		220		236	
189		205		221		237	
190		206		222		238	
191		207		223		239	
192		208		224		240	
193		209		225		241	
194		210		226		242	
195		211		227		243	
196		212		228		244	
197		213		229		245	
198		214		230		246	
199		215		231		247	
200		216		232		248	
201		217		233		249	
202		218		234		250	
203		219		235		251	
204		220		236		252	
205		221		237		253	
206		222		238		254	
207		223		239		255	
208		224		240		256	
209		225		241		257	

Converting from Character to Code:

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

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

ASCII TABLE

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	0x10	32	@	48	0x30
1		17	0x11	33	A	49	0x31
2		18	0x12	34	B	50	0x32
3		19	0x13	35	C	51	0x33
4		20	0x14	36	D	52	0x34
5		21	0x15	37	E	53	0x35
6		22	0x16	38	F	54	0x36
7		23	0x17	39	G	55	0x37
8		24	0x18	40	H	56	0x38
9		25	0x19	41	I	57	0x39
10		26	0x1A	42	J	58	0x3A
11		27	0x1B	43	K	59	0x3B
12		28	0x1C	44	L	60	0x3C
13		29	0x1D	45	M	61	0x3D
14		30	0x1E	46	N	62	0x3E
15		31	0x1F	47	O	63	0x3F
16	0x10	32	@	64	0x40	80	0x50
17	0x11	33	A	65	a	81	0x51
18	0x12	34	B	66	b	82	0x52
19	0x13	35	C	67	c	83	0x53
20	0x14	36	D	68	d	84	0x54
21	0x15	37	E	69	e	85	0x55
22	0x16	38	F	70	f	86	0x56
23	0x17	39	G	71	g	87	0x57
24	0x18	40	H	72	h	88	0x58
25	0x19	41	I	73	i	89	0x59
26	0x1A	42	J	74	j	90	0x5A
27	0x1B	43	K	75	k	91	0x5B
28	0x1C	44	L	76	l	92	0x5C
29	0x1D	45	M	77	m	93	0x5D
30	0x1E	46	N	78	n	94	0x5E
31	0x1F	47	O	79	o	95	0x5F
32	@	80	0x50	96	0x60	112	0x70
33	A	81	0x51	97	0x61	113	0x71
34	B	82	0x52	98	0x62	114	0x72
35	C	83	0x53	99	0x63	115	0x73
36	D	84	0x54	100	0x64	116	0x74
37	E	85	0x55	101	0x65	117	0x75
38	F	86	0x56	102	0x66	118	0x76
39	G	87	0x57	103	0x67	119	0x77
40	H	88	0x58	104	0x68	120	0x78
41	I	89	0x59	105	0x69	121	0x79
42	J	90	0x5A	106	0x6A	122	0x7A
43	K	91	0x5B	107	0x6B	123	0x7B
44	L	92	0x5C	108	0x6C	124	0x7C
45	M	93	0x5D	109	0x6D	125	0x7D
46	N	94	0x5E	110	0x6E	126	0x7E
47	O	95	0x5F	111	0x6F	127	0x7F
48	0x30	112	0x70	128	0x80	144	0x90
49	0x31	113	0x71	129	0x81	145	0x91
50	0x32	114	0x72	130	0x82	146	0x92
51	0x33	115	0x73	131	0x83	147	0x93
52	0x34	116	0x74	132	0x84	148	0x94
53	0x35	117	0x75	133	0x85	149	0x95
54	0x36	118	0x76	134	0x86	150	0x96
55	0x37	119	0x77	135	0x87	151	0x97
56	0x38	120	0x78	136	0x88	152	0x98
57	0x39	121	0x79	137	0x89	153	0x99
58	0x3A	122	0x7A	138	0x8A	154	0x9A
59	0x3B	123	0x7B	139	0x8B	155	0x9B
60	0x3C	124	0x7C	140	0x8C	156	0x9C
61	0x3D	125	0x7D	141	0x8D	157	0x9D
62	0x3E	126	0x7E	142	0x8E	158	0x9E
63	0x3F	127	0x7F	143	0x8F	159	0x9F
64	0x40	128	0x80	144	0x90	160	0xA0
65	a	129	0x81	145	0x91	161	0xA1
66	b	130	0x82	146	0x92	162	0xA2
67	c	131	0x83	147	0x93	163	0xA3
68	d	132	0x84	148	0x94	164	0xA4
69	e	133	0x85	149	0x95	165	0xA5
70	f	134	0x86	150	0x96	166	0xA6
71	g	135	0x87	151	0x97	167	0xA7
72	h	136	0x88	152	0x98	168	0xA8
73	i	137	0x89	153	0x99	169	0xA9
74	j	138	0x8A	154	0x9A	170	0xAA
75	k	139	0x8B	155	0x9B	171	0xAB
76	l	140	0x8C	156	0x9C	172	0xAC
77	m	141	0x8D	157	0x9D	173	0xAD
78	n	142	0x8E	158	0x9E	174	0xAE
79	o	143	0x8F	159	0x9F	175	0xAF
80	0x50	144	0x90	160	0xA0	176	0xB0
81	0x51	145	0x91	161	0xA1	177	0xB1
82	0x52	146	0x92	162	0xA2	178	0xB2
83	0x53	147	0x93	163	0xA3	179	0xB3
84	0x54	148	0x94	164	0xA4	180	0xB4
85	0x55	149	0x95	165	0xA5	181	0xB5
86	0x56	150	0x96	166	0xA6	182	0xB6
87	0x57	151	0x97	167	0xA7	183	0xB7
88	0x58	152	0x98	168	0xA8	184	0xB8
89	0x59	153	0x99	169	0xA9	185	0xB9
90	0x5A	154	0x9A	170	0xAA	186	0xBA
91	0x5B	155	0x9B	171	0xAB	187	0xBB
92	0x5C	156	0x9C	172	0xAC	188	0xBC
93	0x5D	157	0x9D	173	0xAD	189	0xBD
94	0x5E	158	0x9E	174	0xAE	190	0xBE
95	0x5F	159	0x9F	175	0xAF	191	0xBF
96	0x60	160	0xA0	176	0xB0	192	0xC0
97	0x61	161	0xA1	177	0xB1	193	0xC1
98	0x62	162	0xA2	178	0xB2	194	0xC2
99	0x63	163	0xA3	179	0xB3	195	0xC3
100	0x64	164	0xA4	180	0xB4	196	0xC4
101	0x65	165	0xA5	181	0xB5	197	0xC5
102	0x66	166	0xA6	182	0xB6	198	0xC6
103	0x67	167	0xA7	183	0xB7	199	0xC7
104	0x68	168	0xA8	184	0xB8	200	0xC8
105	0x69	169	0xA9	185	0xB9	201	0xC9
106	0x6A	170	0xAA	186	0xBA	202	0xCA
107	0x6B	171	0xAB	187	0xBB	203	0xCB
108	0x6C	172	0xAC	188	0xBC	204	0xCC
109	0x6D	173	0xAD	189	0xBD	205	0xCD
110	0x6E	174	0xAE	190	0xBE	206	0xCE
111	0x6F	175	0xAF	191	0xBF	207	0xCF
112	0x70	176	0xB0	192	0xC0	208	0xD0
113	0x71	177	0xB1	193	0xC1	209	0xD1
114	0x72	178	0xB2	194	0xC2	210	0xD2
115	0x73	179	0xB3	195	0xC3	211	0xD3
116	0x74	180	0xB4	196	0xC4	212	0xD4
117	0x75	181	0xB5	197	0xC5	213	0xD5
118	0x76	182	0xB6	198	0xC6	214	0xD6
119	0x77	183	0xB7	199	0xC7	215	0xD7
120	0x78	184	0xB8	200	0xC8	216	0xD8
121	0x79	185	0xB9	201	0xC9	217	0xD9
122	0x7A	186	0xBA	202	0xCA	218	0xDA
123	0x7B	187	0xBB	203	0xCB	219	0xDB
124	0x7C	188	0xBC	204	0xCC	220	0xDC
125	0x7D	189	0xBD	205	0xCD	221	0xDD
126	0x7E	190	0xBE	206	0xCE	222	0xDE
127	0x7F	191	0xBF	207	0xCF	223	0xDF
128	0x80	192	0xC0	208	0xD0	224	0xE0
129	0x81	193	0xC1	209	0xD1	225	0xE1
130	0x82	194	0xC2	210	0xD2	226	0xE2
131	0x83	195	0xC3	211	0xD3	227	0xE3
132	0x84	196	0xC4	212	0xD4	228	0xE4
133	0x85	197	0xC5	213	0xD5	229	0xE5
134	0x86	198	0xC6	214	0xD6	230	0xE6
135	0x87	199	0xC7	215	0xD7	231	0xE7
136	0x88	200	0xC8	216	0xD8	232	0xE8
137	0x89	201	0xC9	217	0xD9	233	0xE9
138	0x8A	202	0xCA	218	0xDA	234	0xEA
139	0x8B	203	0xCB	219	0xDB	235	0xEB
140	0x8C	204	0xCC	220	0xDC	236	0xEC
141	0x8D	205	0xCD	221	0xDD	237	0xED
142	0x8E	206	0xCE	222	0xDE	238	0xEE
143	0x8F	207	0xCF	223	0xDF	239	0xEF
144	0x90	208	0xD0	224	0xE0	240	0xF0
145	0x91	209	0xD1	225	0xE1	241	0xF1
146	0x92	210	0xD2	226	0xE2	242	0xF2
147	0x93	211	0xD3	227	0xE3	243	0xF3
148	0x94	212	0xD4	228	0xE4	244	0xF4
149	0x95	213	0xD5	229	0xE5	245	0xF5
150	0x96	214	0xD6	230	0xE6	246	0xF6
151	0x97	215	0xD7	231	0xE7	247	0xF7
152	0x98	216	0xD8	232	0xE8	248	0xF8
153	0x99	217	0xD9	233	0xE9	249	0xF9
154	0x9A	218	0xDA	234	0xEA	250	0xFA
155	0x9B	219	0xDB	235	0xEB	251	0xFB
156	0x9C	220	0xDC	236	0xEC	252	0xFC
157	0x9D	221	0xDD	237	0xED	253	0xFD
158	0x9E	222	0xDE	238	0xEE	254	0xFE
159	0x9F	223	0xDF	239	0xEF	255	0xFF

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	SOH	17	Q	33	A	49	1
2	STX	18	R	34	B	50	2
3	ETX	19	S	35	C	51	3
4	END	20	T	36	D	52	4
5	SO	21	U	37	E	53	5
6	ST	22	V	38	F	54	6
7	ACK	23	W	39	G	55	7
8	BS	24	X	40	H	56	8
9	HT	25	Y	41	I	57	9
10	LF	26	Z	42	J	58	:
11	VT	27	[43	K	59	;
12	FF	28	\	44	L	60	<
13	CR	29]	45	M	61	=
14	SO	30	^	46	N	62	>
15	ST	31	_	47	O	63	?
16	P	32	@	48	0	64	SP
17	Q	33	A	49	1	65	a
18	R	34	B	50	2	66	b
19	S	35	C	51	3	67	c
20	T	36	D	52	4	68	d
21	U	37	E	53	5	69	e
22	V	38	F	54	6	70	f
23	W	39	G	55	7	71	g
24	X	40	H	56	8	72	h
25	Y	41	I	57	9	73	i
26	Z	42	J	58	:	74	j
27	[43	K	59	;	75	k
28	\	44	L	60	<	76	l
29]	45	M	61	=	77	m
30	^	46	N	62	>	78	n
31	_	47	O	63	?	79	o
32	@	48	0	64	SP	80	p
33	A	49	1	65	a	81	q
34	B	50	2	66	b	82	r
35	C	51	3	67	c	83	s
36	D	52	4	68	d	84	t
37	E	53	5	69	e	85	u
38	F	54	6	70	f	86	v
39	G	55	7	71	g	87	w
40	H	56	8	72	h	88	x
41	I	57	9	73	i	89	y
42	J	58	:	74	j	90	z
43	K	59	;	75	k	91	{
44	L	60	<	76	l	92	
45	M	61	=	77	m	93	}
46	N	62	>	78	n	94	~
47	O	63	?	79	o	95	DEL

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

Converting from Character to Code:

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

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	SOH	17	Q	33	A	49	1
2	STX	18	R	34	B	50	2
3	ETX	19	S	35	C	51	3
4	END	20	T	36	D	52	4
5	SO	21	U	37	E	53	5
6	SI	22	V	38	F	54	6
7	BS	23	W	39	G	55	7
8	HT	24	X	40	H	56	8
9	LF	25	Y	41	I	57	9
10	VT	26	Z	42	J	58	:
11	FF	27	[43	K	59	;
12		28	\	44	L	60	<
13	CR	29]	45	M	61	=
14	SO	30	^	46	N	62	>
15	SI	31	_	47	O	63	?

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

Converting from Character to Code:

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

ASCII TABLE

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

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

In Pairs or Triples...

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 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)
```

(Demo with pythonTutor)

Wrap

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

offset: how many letters past 'a'?

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

User Input

Covered in detail in Lab 2:

```
➔ 1 mess = input('Please enter a message: ')\n   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~~s~~Saturday~~s~~Sunday"

More on Strings: Splits

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

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

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

More on Strings: Splits

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

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

```
"FridaysSaturdaysSunday"  
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.

```
"FridaysSaturdaysSunday"  
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.

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

- Different delimiters give different lists:

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

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.

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

- Different delimiters give different lists:

```
days = s[:-1].split("day")  
"FrixxxsSaturxxxsSunxxx"  
days = ['Fri', 'sSatur', '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(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- In Python, we introduced:

- ▶ For-loops

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


Recap

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`

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

Recap

- In Python, we introduced:

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

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

Recap

- In Python, we introduced:

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

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

Recap

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

● In Python, we introduced:

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

Recap

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

● In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

Recap

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

- In Python, we introduced:

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

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~~s~~Saturday~~s~~Sunday"

More on Strings: Splits

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

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

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

More on Strings: Splits

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

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

```
"FridaysSaturdaysSunday"  
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.

```
"FridaysSaturdaysSunday"  
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.

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

- Different delimiters give different lists:

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

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.

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

- Different delimiters give different lists:

```
days = s[:-1].split("day")  
"FrixxxsSaturxxxsSunxxx"  
days = ['Fri', 'sSatur', '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`
- Remainder or Modulus:

`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`
- Remainder or Modulus:
`days = totalDays % 7`

`15 // 7 = 2`

`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`
- Exponentiaion:

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`
- Exponentiaion:
`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("8last 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("8last 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 "

- Also works for lists:

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blas 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 "

- Also works for lists:

`names[1:3]`

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8last 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 "

- Also works for lists:

`names[1:3]`

gives: ["Anna", "Alice"]

```
1 for d in range(10, 0, -1):
2     print(d)
3 print("Blas 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 "

- Also works for lists:

`names[1:3]`

gives: ["Anna", "Alice"]

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






```
1 for d in range(10, 0, -1):
2     print(d)
3 print("8Last 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)
```

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>	

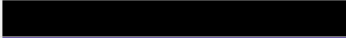




- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).

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>	

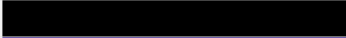




- Can specify by name.
- Can specify by numbers:
 - ▶ Amount of Red, Green, and Blue (RGB).
 - ▶ Adding light, not paint:

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.

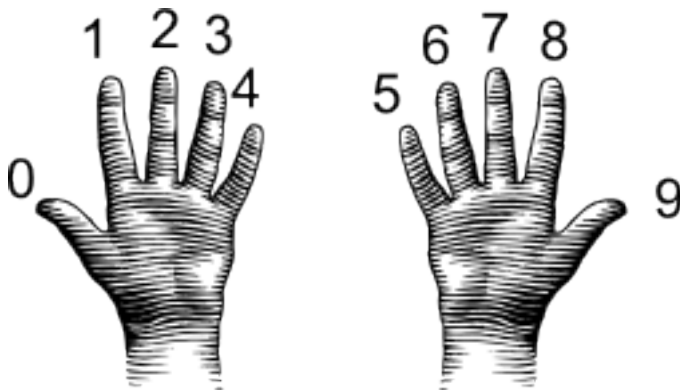
Colors

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)...

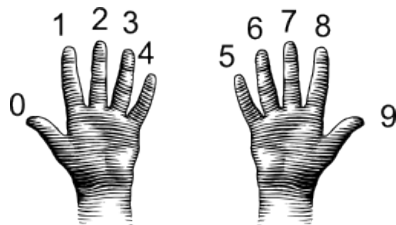
Decimal & Hexadecimal Numbers

Counting with 10 digits:



(from i-programmer.info)

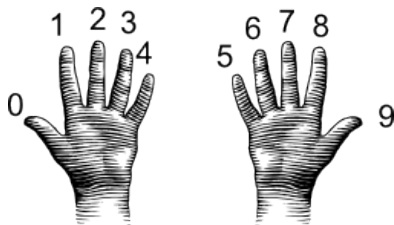
Decimal



(from i-programmer.info)

Decimal

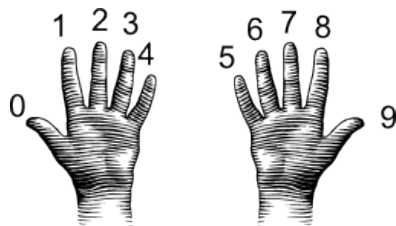
00 01 02 03 04 05 06 07 08 09



(from i-programmer.info)

Decimal

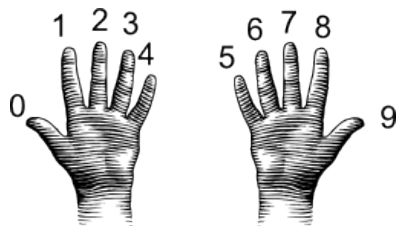
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19



(from i-programmer.info)

Decimal

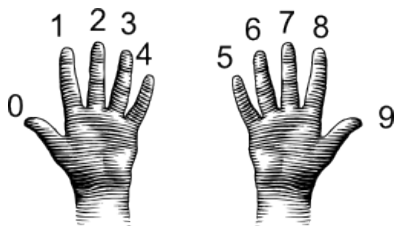
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29



(from i-programmer.info)

Decimal

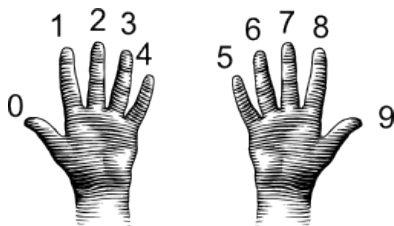
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39



(from i-programmer.info)

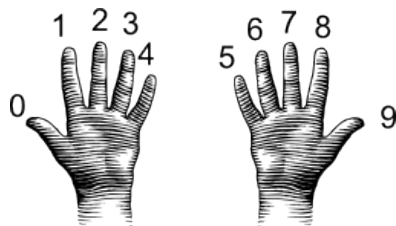
Decimal

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
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49



(from i-programmer.info)

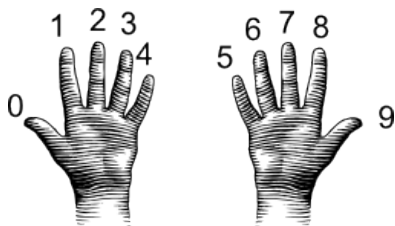
Decimal



(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
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

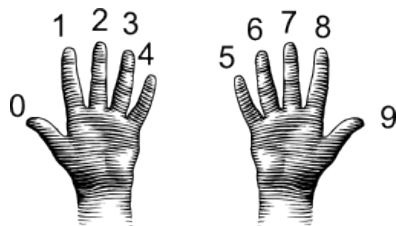
Decimal



(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

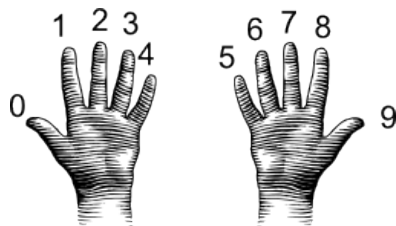
Decimal



(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

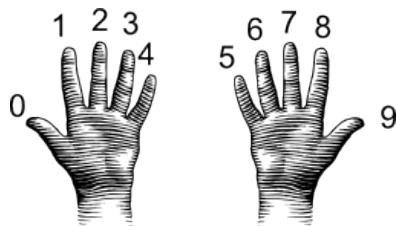
Decimal



(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

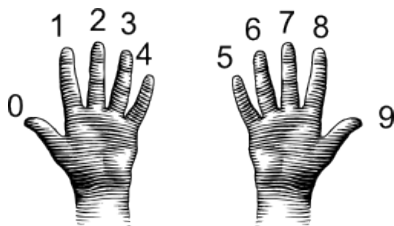
Decimal



(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



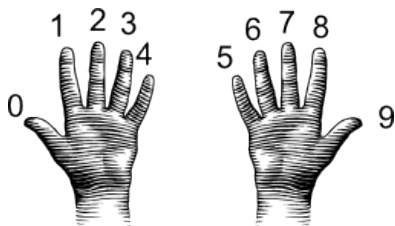
(from i-programmer.info)

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)

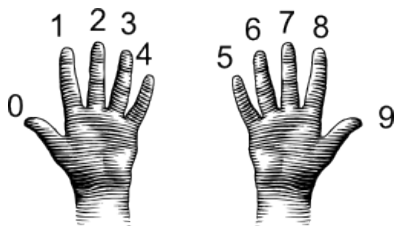
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

$$10^1 + 10^0$$

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

$$10^1 + 10^0$$

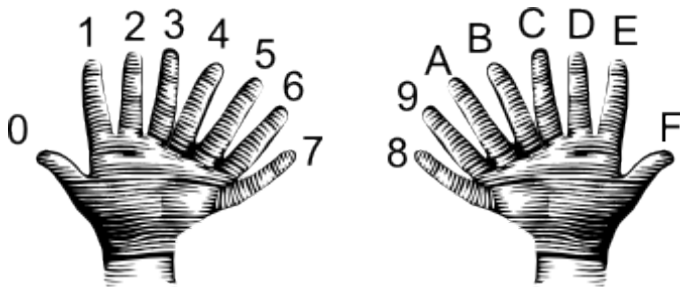
Max Number = 99

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

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

Decimal & Hexadecimal Numbers

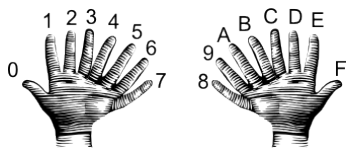
Counting with 16 digits:



(from i-programmer.info)

Hexadecimal

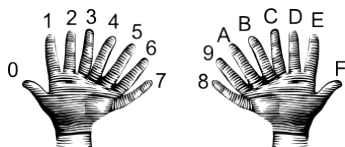
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F



(from i-programmer.info)

Hexadecimal

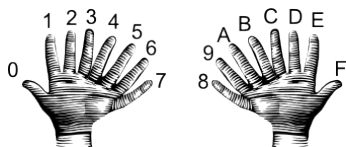
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F



(from i-programmer.info)

Hexadecimal

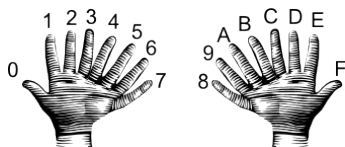
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F



(from i-programmer.info)

Hexadecimal

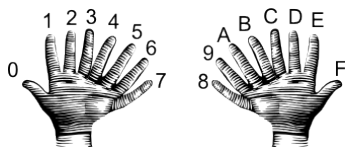
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F



(from i-programmer.info)

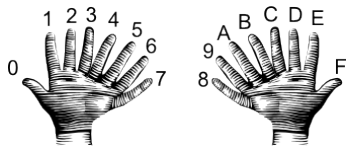
Hexadecimal

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)

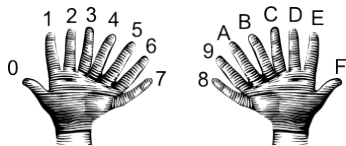
Hexadecimal



(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

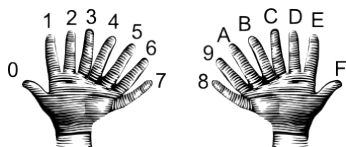
Hexadecimal



(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

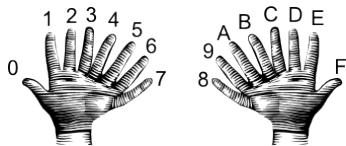
Hexadecimal



(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

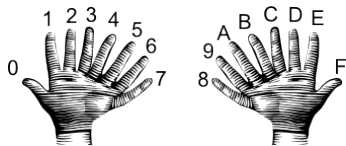
Hexadecimal



(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

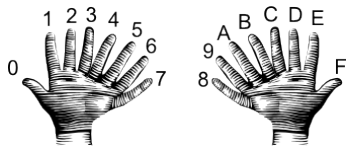
Hexadecimal



(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

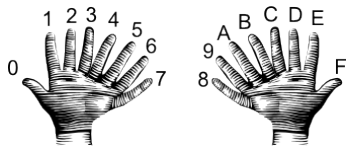
Hexadecimal



(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

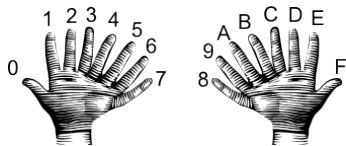
Hexadecimal



(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

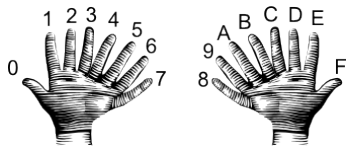
Hexadecimal



(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

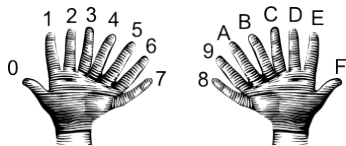
Hexadecimal



(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

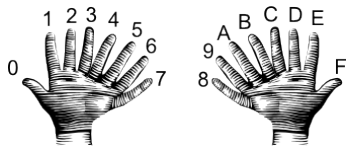
Hexadecimal



(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

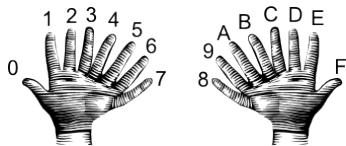
Hexadecimal



(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

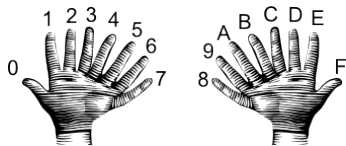


(from i-programmer.info)

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



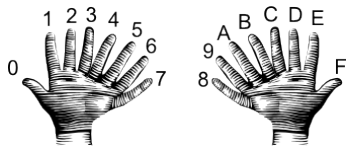
(from i-programmer.info)

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

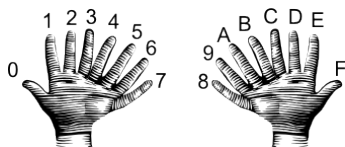
$$16^1 + 16^0$$

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 on Gradescope by 6pm on today

Class Reminders!



Before next class, don't forget to:

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