

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Review of Lecture 1: turtle graphics

- Imagine a turtle has a pen; when it moves forward some distance, a line is drawn on the screen.
- The turtle can also turn left some amount of degrees.

```
1 import turtle
2
3 t = turtle.Turtle()
4
5 #draw side one
6 t.forward(100)
7 t.left(120)
8
9 #draw side two
10 t.forward(100)
11 t.left(120)
12
13 #draw side three
14 t.forward(100)
15 t.left(120)
```

Review of Lecture 1: for-loops

- The previous program used the turtle module to draw a triangle
- Rewrite the program using a for-loop

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

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

Draw a polygon with $n \geq 3$ sides

Pseudocode describes the general algorithm our program will follow; it is language-agnostic and can be translated into any programming language.

Import the turtle library

Instantiate a turtle object called `t`

Initialize `n` to be an integer ≥ 3

Repeat the following `n` times:

- (1) `t` moves forward a fixed distance
- (2) `t` turns left $360 / n$ degrees

Today's Topics



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

Today's Topics

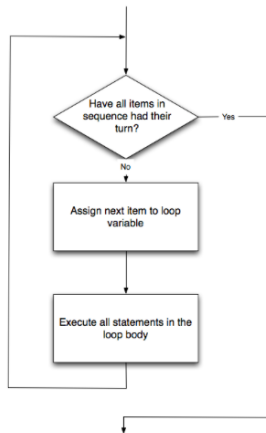


- **For-loops**
- `range()`
- Variables
- Strings
- ASCII

Group Work: predict what will be printed

```
1
2 for j in [0,1,2,3,4,5]:
3     print(j)
4 for count in range(6):
5     print(count)
6 for color in ["red", "green", "blue"]:
7     print(color)
8 for i in range(2):
9     for j in range(2):
10        print("hello from inner loop")
11        print("hello from outer loop")
```

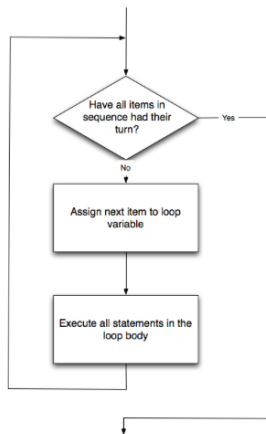
for-loop



```
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
- Strings
- ASCII

More on range(): predict what will be printed

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

[link to range demo](#)

range()

Simplest version:

- `range(stop)`



range()



Simplest version:

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

range()



Simplest version:

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

range()



Simplest version:

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

```
range(101)
```

range()

What if you wanted to start somewhere else:



range()

What if you wanted to start somewhere else:

- `range(start, stop)`



range()



What if you wanted to start somewhere else:

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

range()



What if you wanted to start somewhere else:

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

range()



What if you wanted to start somewhere else:

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

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

`range()`

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



range()

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

- `range(start, stop, step)`



range()

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

- `range(start, stop, step)`
- Produces a list: `[start, start+1*step, start+2*step, start+3*step, ..., last]`
(where last is the largest `start+k*step` less than stop)



range()



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

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

range()



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

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

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

In summary: `range()`



The three versions:

In summary: `range()`



The three versions:

- `range(stop)`

In summary: `range()`



The three versions:

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

In summary: `range()`



The three versions:

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

Today's Topics



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

Variables

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



Variables



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

Variables



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

Variable Names



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

Today's Topics



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

String Methods

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

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

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).

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`.

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.

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.

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?

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?

Lecture Slip

- How can we use the `count()` method to solve our lecture slip problem?

Lecture Slip

- How can we use the `count()` method to solve our lecture slip problem?

```
1 words = "oranges banana apple grapes kiwis "  
2  
3 num = words.count("s ")  
4  
5 print(num)
```

More on Strings: Indexing & Substrings

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

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

More on Strings: Indexing & Substrings

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

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

More on Strings: Indexing & Substrings

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

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

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

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

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

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

- `s[0]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

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

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

- `s[1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

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

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

- `s[-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[-1]` is "s".

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[3:6]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[:3]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

- `s[:-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

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

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

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

Today's Topics



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

Standardized Code for Characters

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

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 between Character and Code:

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

ASCII TABLE

Decimal	Hex Char	Octal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		00		128	80	192	C0
1		01		129	81	193	C1
2		02		130	82	194	C2
3		03		131	83	195	C3
4		04		132	84	196	C4
5		05		133	85	197	C5
6		06		134	86	198	C6
7		07		135	87	199	C7
8		10		136	88	200	C8
9		11		137	89	201	C9
10		12		138	8A	202	CA
11		13		139	8B	203	CB
12		14		140	8C	204	CC
13		15		141	8D	205	CD
14		16		142	8E	206	CE
15		17		143	8F	207	CF
16		20		144	90	208	D0
17		21		145	91	209	D1
18		22		146	92	210	D2
19		23		147	93	211	D3
20		24		148	94	212	D4
21		25		149	95	213	D5
22		26		150	96	214	D6
23		27		151	97	215	D7
24		30		152	98	216	D8
25		31		153	99	217	D9
26		32		154	9A	218	DA
27		33		155	9B	219	DB
28		34		156	9C	220	DC
29		35		157	9D	221	DD
30		36		158	9E	222	DE
31		37		159	9F	223	DF
32	space	40		160	A0	224	E0
33	!	21		161	A1	225	E1
34	"	22		162	A2	226	E2
35	#	23		163	A3	227	E3
36	\$	24		164	A4	228	E4
37	%	25		165	A5	229	E5
38	&	26		166	A6	230	E6
39	'	27		167	A7	231	E7
40	(28		168	A8	232	E8
41)	29		169	A9	233	E9
42	*	30		170	AA	234	EA
43	+	31		171	AB	235	EB
44	,	32		172	AC	236	EC
45	-	33		173	AD	237	ED
46	.	34		174	AE	238	EE
47	/	35		175	AF	239	EF
48	0	30		176	B0	240	F0
49	1	31		177	B1	241	F1
50	2	32		178	B2	242	F2
51	3	33		179	B3	243	F3
52	4	34		180	B4	244	F4
53	5	35		181	B5	245	F5
54	6	36		182	B6	246	F6
55	7	37		183	B7	247	F7
56	8	30		184	B8	248	F8
57	9	31		185	B9	249	F9
58	:	32		186	BA	250	FA
59	;	33		187	BB	251	FB
60	<	34		188	BC	252	FC
61	=	35		189	BD	253	FD
62	>	36		190	BE	254	FE
63	?	37		191	BF	255	FF

Converting between Character and 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	00		1	01		2	02		3	03	
4	04		5	05		6	06		7	07	
8	08		9	09		10	0A		11	0B	
12	0C		13	0D		14	0E		15	0F	
16	10		17	11		18	12		19	13	
20	14		21	15		22	16		23	17	
24	18		25	19		26	1A		27	1B	
28	1C		29	1D		30	1E		31	1F	
32	20	Space	33	21	!	34	22	"	35	23	#
36	24	\$	37	25	%	38	26	&	39	27	'
40	28	(41	29)	42	2A	*	43	2B	+
44	2C	,	45	2D	-	46	2E	.	47	2F	/
48	30	0	49	31	1	50	32	2	51	33	3
52	34	4	53	35	5	54	36	6	55	37	7
56	38	8	57	39	9	58	3A	:	59	3B	;
60	3C	<	61	3D	=	62	3E	>	63	3F	?
64	40	@	65	41	A	66	42	B	67	43	C
68	44	D	69	45	E	70	46	F	71	47	
72	48		73	49		74	4A		75	4B	
76	4C		77	4D		78	4E		79	4F	
80	50		81	51		82	52		83	53	
84	54		85	55		86	56		87	57	
88	58		89	59		90	5A		91	5B	
92	5C		93	5D		94	5E		95	5F	
96	60		97	61	a	98	62	b	99	63	c
100	64	d	101	65	e	102	66	f	103	67	
104	68		105	69		106	6A		107	6B	
108	6C		109	6D		110	6E		111	6F	
112	70		113	71		114	72		115	73	
116	74		117	75		118	76		119	77	
120	78		121	79		122	7A		123	7B	
124	7C		125	7D		126	7E		127	7F	
128	80		129	81		130	82		131	83	
132	84		133	85		134	86		135	87	
136	88		137	89		138	8A		139	8B	
140	8C		141	8D		142	8E		143	8F	
144	90		145	91		146	92		147	93	
148	94		149	95		150	96		151	97	
152	98		153	99		154	9A		155	9B	
156	9C		157	9D		158	9E		159	9F	
160	A0		161	A1		162	A2		163	A3	
164	A4		165	A5		166	A6		167	A7	
168	A8		169	A9		170	AA		171	AB	
172	AC		173	AD		174	AE		175	AF	
176	B0		177	B1		178	B2		179	B3	
180	B4		181	B5		182	B6		183	B7	
184	B8		185	B9		186	BA		187	BB	
188	BC		189	BD		190	BE		191	BF	
192	C0		193	C1		194	C2		195	C3	
196	C4		197	C5		198	C6		199	C7	
200	C8		201	C9		202	CA		203	CB	
204	CC		205	CD		206	CE		207	CF	
208	D0		209	D1		210	D2		211	D3	
212	D4		213	D5		214	D6		215	D7	
216	D8		217	D9		218	DA		219	DB	
220	DC		221	DD		222	DE		223	DF	
224	E0		225	E1		226	E2		227	E3	
228	E4		229	E5		230	E6		231	E7	
232	E8		233	E9		234	EA		235	EB	
236	EC		237	ED		238	EE		239	EF	
240	F0		241	F1		242	F2		243	F3	
244	F4		245	F5		246	F6		247	F7	
248	F8		249	F9		250	FA		251	FB	
252	FC		253	FD		254	FE		255	FF	

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

Converting between Character and 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	00		1	01		2	02		3	03	
4	04		5	05		6	06		7	07	
8	08		9	09		10	0A		11	0B	
12	0C		13	0D		14	0E		15	0F	
16	10		17	11		18	12		19	13	
20	14		21	15		22	16		23	17	
24	18		25	19		26	1A		27	1B	
28	1C		29	1D		30	1E		31	1F	
32	20	Space	33	21	!	34	22	"	35	23	#
36	24	\$	37	25	%	38	26	&	39	27	'
40	28	(41	29)	42	2A	*	43	2B	+
44	2C	,	45	2D	-	46	2E	.	47	2F	/
48	30	0	49	31	1	50	32	2	51	33	3
52	34	4	53	35	5	54	36	6	55	37	7
56	38	8	57	39	9	58	3A	:	59	3B	;
60	3C	<	61	3D	=	62	3E	>	63	3F	?
64	40	@	65	41	A	66	42	B	67	43	C
68	44	D	69	45	E	70	46	F	71	47	
72	48		73	49		74	4A		75	4B	
76	4C		77	4D		78	4E		79	4F	
80	50		81	51		82	52		83	53	
84	54		85	55		86	56		87	57	
88	58		89	59		90	5A		91	5B	
92	5C		93	5D		94	5E		95	5F	
96	60		97	61	a	98	62	b	99	63	c
100	64	d	101	65	e	102	66	f	103	67	
104	68		105	69		106	6A		107	6B	
108	6C		109	6D		110	6E		111	6F	
112	70		113	71		114	72		115	73	
116	74		117	75		118	76		119	77	
120	78		121	79		122	7A		123	7B	
124	7C		125	7D		126	7E		127	7F	
128	80		129	81		130	82		131	83	
132	84		133	85		134	86		135	87	
136	88		137	89		138	8A		139	8B	
140	8C		141	8D		142	8E		143	8F	
144	90		145	91		146	92		147	93	
148	94		149	95		150	96		151	97	
152	98		153	99		154	9A		155	9B	
156	9C		157	9D		158	9E		159	9F	
160	A0		161	A1		162	A2		163	A3	
164	A4		165	A5		166	A6		167	A7	
168	A8		169	A9		170	AA		171	AB	
172	AC		173	AD		174	AE		175	AF	
176	B0		177	B1		178	B2		179	B3	
180	B4		181	B5		182	B6		183	B7	
184	B8		185	B9		186	BA		187	BB	
188	BC		189	BD		190	BE		191	BF	
192	C0		193	C1		194	C2		195	C3	
196	C4		197	C5		198	C6		199	C7	
200	C8		201	C9		202	CA		203	CB	
204	CC		205	CD		206	CE		207	CF	
208	D0		209	D1		210	D2		211	D3	
212	D4		213	D5		214	D6		215	D7	
216	D8		217	D9		218	DA		219	DB	
220	DC		221	DD		222	DE		223	DF	
224	E0		225	E1		226	E2		227	E3	
228	E4		229	E5		230	E6		231	E7	
232	E8		233	E9		234	EA		235	EB	
236	EC		237	ED		238	EE		239	EF	
240	F0		241	F1		242	F2		243	F3	
244	F4		245	F5		246	F6		247	F7	
248	F8		249	F9		250	FA		251	FB	
252	FC		253	FD		254	FE		255	FF	

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

Converting between Character and Code:

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

ASCII TABLE



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

ord() and chr()

(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	00		1	01		2	02		3	03	
4	04		5	05		6	06		7	07	
8	08		9	09		10	0A		11	0B	
12	0C		13	0D		14	0E		15	0F	
16	10		17	11		18	12		19	13	
20	14		21	15		22	16		23	17	
24	18		25	19		26	1A	A	27	1B	B
28	1C	C	29	1D	D	30	1E	E	31	1F	F
32	20	Space	33	21	!	34	22	"	35	23	#
36	24	\$	37	25	%	38	26	&	39	27	'
40	28	(41	29)	42	2A	*	43	2B	+
44	2C	,	45	2D	-	46	2E	.	47	2F	/
48	30	0	49	31	1	50	32	2	51	33	3
52	34	4	53	35	5	54	36	6	55	37	7
56	38	8	57	39	9	58	3A	:	59	3B	;
60	3C	<	61	3D	=	62	3E	>	63	3F	?
64	40	@	65	41	A	66	42	B	67	43	C
68	44	D	69	45	E	70	46	F	71	47	
72	48	a	73	49	b	74	4A	c	75	4B	d
76	4C	e	77	4D	f	78	4E	g	79	4F	h
80	50	i	81	51	j	82	52	k	83	53	l
84	54	m	85	55	n	86	56	o	87	57	p
88	58	q	89	59	r	90	5A	s	91	5B	t
92	5C	u	93	5D	v	94	5E	w	95	5F	x
96	60	y	97	61	z	98	62	[99	63	\
100	64]	101	65	^	102	66	_	103	67	`
104	68	{	105	69	}	106	6A	~	107	6B	
108	6C		109	6D		110	6E		111	6F	
112	70		113	71		114	72		115	73	
116	74		117	75		118	76		119	77	
120	78		121	79		122	7A		123	7B	
124	7C		125	7D		126	7E		127	7F	
128	80		129	81		130	82		131	83	
132	84		133	85		134	86		135	87	
136	88		137	89		138	8A		139	8B	
140	8C		141	8D		142	8E		143	8F	
144	90		145	91		146	92		147	93	
148	94		149	95		150	96		151	97	
152	98		153	99		154	9A		155	9B	
156	9C		157	9D		158	9E		159	9F	
160	A0		161	A1		162	A2		163	A3	
164	A4		165	A5		166	A6		167	A7	
168	A8		169	A9		170	AA		171	AB	
172	AC		173	AD		174	AE		175	AF	
176	B0		177	B1		178	B2		179	B3	
180	B4		181	B5		182	B6		183	B7	
184	B8		185	B9		186	BA		187	BB	
188	BC		189	BD		190	BE		191	BF	
192	C0		193	C1		194	C2		195	C3	
196	C4		197	C5		198	C6		199	C7	
200	C8		201	C9		202	CA		203	CB	
204	CC		205	CD		206	CE		207	CF	
208	D0		209	D1		210	D2		211	D3	
212	D4		213	D5		214	D6		215	D7	
216	D8		217	D9		218	DA		219	DB	
220	DC		221	DD		222	DE		223	DF	
224	E0		225	E1		226	E2		227	E3	
228	E4		229	E5		230	E6		231	E7	
232	E8		233	E9		234	EA		235	EB	
236	EC		237	ED		238	EE		239	EF	
240	F0		241	F1		242	F2		243	F3	
244	F4		245	F5		246	F6		247	F7	
248	F8		249	F9		250	FA		251	FB	
252	FC		253	FD		254	FE		255	FF	

ord() and chr()

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

- ord():
input type: character
output type: integer

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		1	01		2	02		3	03	
4	04		5	05		6	06		7	07	
8	08		9	09		10	0A		11	0B	
12	0C		13	0D		14	0E		15	0F	
16	10		17	11		18	12		19	13	
20	14		21	15		22	16		23	17	
24	18		25	19		26	1A		27	1B	
28	1C		29	1D		30	1E		31	1F	
32	20	Space	33	21	!	34	22	"	35	23	#
36	24	\$	37	25	%	38	26	&	39	27	'
40	28	(41	29)	42	2A	*	43	2B	+
44	2C	,	45	2D	-	46	2E	.	47	2F	/
48	30	0	49	31	1	50	32	2	51	33	3
52	34	4	53	35	5	54	36	6	55	37	7
56	38	8	57	39	9	58	3A	:	59	3B	;
60	3C	<	61	3D	=	62	3E	>	63	3F	?
64	40	@	65	41	A	66	42	B	67	43	C
68	44	D	69	45	E	70	46	F	71	47	
72	48		73	49		74	4A		75	4B	
76	4C		77	4D		78	4E		79	4F	
80	50		81	51		82	52		83	53	
84	54		85	55		86	56		87	57	
88	58		89	59		90	5A		91	5B	
92	5C		93	5D		94	5E		95	5F	
96	60		97	61		98	62		99	63	
100	64		101	65		102	66		103	67	
104	68		105	69		106	6A		107	6B	
108	6C		109	6D		110	6E		111	6F	
112	70		113	71		114	72		115	73	
116	74		117	75		118	76		119	77	
120	78		121	79		122	7A		123	7B	
124	7C		125	7D		126	7E		127	7F	
128	80		129	81		130	82		131	83	
132	84		133	85		134	86		135	87	
136	88		137	89		138	8A		139	8B	
140	8C		141	8D		142	8E		143	8F	
144	90		145	91		146	92		147	93	
148	94		149	95		150	96		151	97	
152	98		153	99		154	9A		155	9B	
156	9C		157	9D		158	9E		159	9F	
160	A0		161	A1		162	A2		163	A3	
164	A4		165	A5		166	A6		167	A7	
168	A8		169	A9		170	AA		171	AB	
172	AC		173	AD		174	AE		175	AF	
176	B0		177	B1		178	B2		179	B3	
180	B4		181	B5		182	B6		183	B7	
184	B8		185	B9		186	BA		187	BB	
188	BC		189	BD		190	BE		191	BF	
192	C0		193	C1		194	C2		195	C3	
196	C4		197	C5		198	C6		199	C7	
200	C8		201	C9		202	CA		203	CB	
204	CC		205	CD		206	CE		207	CF	
208	D0		209	D1		210	D2		211	D3	
212	D4		213	D5		214	D6		215	D7	
216	D8		217	D9		218	DA		219	DB	
220	DC		221	DD		222	DE		223	DF	
224	E0		225	E1		226	E2		227	E3	
228	E4		229	E5		230	E6		231	E7	
232	E8		233	E9		234	EA		235	EB	
236	EC		237	ED		238	EE		239	EF	
240	F0		241	F1		242	F2		243	F3	
244	F4		245	F5		246	F6		247	F7	
248	F8		249	F9		250	FA		251	FB	
252	FC		253	FD		254	FE		255	FF	

ord() and chr()

(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	00		9	09		18	12		27	1B	
1	01		10	0A		19	13		28	1C	
2	02		11	0B		20	14	space	29	1D	
3	03		12	0C		21	15	!	30	1E	
4	04		13	0D		22	16	"	31	1F	
5	05		14	0E		23	17	#	32	20	space
6	06		15	0F		24	18	\$	33	21	!
7	07		16	10		25	19	%	34	22	"
8	08		17	11		26	1A	&	35	23	#
9	09		18	12		27	1B	'	36	24	\$
10	0A		19	13		28	1C	(37	25	%
11	0B		20	14	space	29	1D)	38	26	&
12	0C		21	15	!	30	1E	*	39	27	'
13	0D		22	16	"	31	1F	+	40	28	(
14	0E		23	17	#	32	20	,	41	29)
15	0F		24	18	\$	33	21	-	42	2A	*
16	10		25	19	%	34	22	.	43	2B	+
17	11		26	1A	&	35	23	:	44	2C	,
18	12		27	1B	'	36	24	;	45	2D	;
19	13		28	1C	(37	25	<	46	2E	<
20	14	space	29	1D)	38	26	=	47	2F	=
21	15	!	30	1E	*	39	27	>	48	30	0
22	16	"	31	1F	+	40	28	?	49	31	1
23	17	#	32	20	,	41	29	@	50	32	2
24	18	\$	33	21	!	42	2A	A	51	33	3
25	19	%	34	22	"	43	2B	a	52	34	4
26	1A	&	35	23	#	44	2C	A	53	35	5
27	1B	'	36	24	\$	45	2D	a	54	36	6
28	1C	(37	25	%	46	2E	A	55	37	7
29	1D)	38	26	&	47	2F	a	56	38	8
30	1E	*	39	27	'	48	30	A	57	39	9
31	1F	+	40	28	(49	31	a	58	3A	0
32	20	space	41	29)	50	32	A	59	3B	1
33	21	!	42	2A	*	51	33	a	60	3C	2
34	22	"	43	2B	+	52	34	A	61	3D	3
35	23	#	44	2C	,	53	35	a	62	3E	4
36	24	\$	45	2D	;	54	36	A	63	3F	5
37	25	%	46	2E	<	55	37	a	64	40	6
38	26	&	47	2F	=	56	38	A	65	41	7
39	27	'	48	30	>	57	39	a	66	42	8
40	28	(49	31	?	58	3A	A	67	43	9
41	29)	50	32	@	59	3B	a	68	44	0
42	2A	*	51	33	A	60	3C	A	69	45	1
43	2B	+	52	34	a	61	3D	a	70	46	2
44	2C	,	53	35	A	62	3E	a	71	47	3
45	2D	;	54	36	A	63	3F	a	72	48	4
46	2E	<	55	37	a	64	40	A	73	49	5
47	2F	=	56	38	a	65	41	A	74	4A	6
48	30	0	57	39	A	66	42	a	75	4B	7
49	31	1	58	3A	a	67	43	A	76	4C	8
50	32	2	59	3B	a	68	44	A	77	4D	9
51	33	3	60	3C	A	69	45	a	78	4E	0
52	34	4	61	3D	a	70	46	A	79	4F	1
53	35	5	62	3E	a	71	47	A	80	50	2
54	36	6	63	3F	a	72	48	A	81	51	3
55	37	7	64	40	A	73	49	a	82	52	4
56	38	8	65	41	A	74	4A	a	83	53	5
57	39	9	66	42	a	75	4B	a	84	54	6
58	3A	0	67	43	a	76	4C	A	85	55	7
59	3B	1	68	44	a	77	4D	A	86	56	8
60	3C	2	69	45	a	78	4E	A	87	57	9
61	3D	3	70	46	A	79	4F	a	88	58	0
62	3E	4	71	47	A	80	50	a	89	59	1
63	3F	5	72	48	A	81	51	a	90	5A	2
64	40	6	73	49	a	82	52	A	91	5B	3
65	41	7	74	4A	a	83	53	A	92	5C	4
66	42	8	75	4B	a	84	54	A	93	5D	5
67	43	9	76	4C	A	85	55	a	94	5E	6
68	44	0	77	4D	A	86	56	a	95	5F	7
69	45	1	78	4E	a	87	57	A	96	60	8
70	46	2	79	4F	a	88	58	A	97	61	9
71	47	3	80	50	a	89	59	A	98	62	0
72	48	4	81	51	a	90	5A	A	99	63	1
73	49	5	82	52	a	91	5B	A	100	64	2
74	4A	6	83	53	a	92	5C	A			
75	4B	7	84	54	a	93	5D	A			
76	4C	8	85	55	A	94	5E	a			
77	4D	9	86	56	A	95	5F	a			
78	4E	0	87	57	a						
79	4F	1	88	58	a						
80	50	2	89	59	a						
81	51	3	90	5A	A						
82	52	4	91	5B	A						
83	53	5	92	5C	A						
84	54	6	93	5D	A						
85	55	7	94	5E	a						
86	56	8	95	5F	a						
87	57	9	96	60	A						
88	58	0	97	61	a						
89	59	1	98	62	A						
90	5A	2	99	63	A						
91	5B	3	100	64	A						

- ord():
input type: character
output type: integer
- chr():
input type: integer
output type: character

ord() and chr()

(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	00		9	09		18	12		27	1B	
1	01		10	0A		19	13		28	1C	
2	02		11	0B		20	14	space	29	1D	
3	03		12	0C		21	15	!	30	1E	0
4	04		13	0D		22	16	"	31	1F	1
5	05		14	0E		23	17	#	32	20	2
6	06		15	0F		24	18	\$	33	21	3
7	07		16	10		25	19	%	34	22	"
8	08		17	11		26	1A	&	35	23	#
9	09		18	12		27	1B	'	36	24	\$
10	0A		19	13		28	1C	(37	25	%
11	0B		20	14	space	29	1D)	38	26	&
12	0C		21	15	!	30	1E	*	39	27	'
13	0D		22	16	"	31	1F	+	40	28	(
14	0E		23	17	#	32	20	,	41	29)
15	0F		24	18	\$	33	21	-	42	2A	*
16	10		25	19	%	34	22	.	43	2B	+
17	11		26	1A	&	35	23	:	44	2C	,
18	12		27	1B	'	36	24	;	45	2D	;
19	13		28	1C	(37	25	<	46	2E	<
20	14	space	29	1D)	38	26	=	47	2F	=
21	15	!	30	1E	*	39	27	>	48	30	>
22	16	"	31	1F	+	40	28	?	49	31	?
23	17	#	32	20	,	41	29	@	50	32	@
24	18	\$	33	21	-	42	2A	A	51	33	A
25	19	%	34	22	.	43	2B	B	52	34	B
26	1A	&	35	23	:	44	2C	C	53	35	C
27	1B	'	36	24	;	45	2D	D	54	36	D
28	1C	(37	25	<	46	2E	E	55	37	E
29	1D)	38	26	=	47	2F	F	56	38	F
30	1E	*	39	27	'	48	30		57	39	
31	1F	+	40	28	(49	31		58	3A	
32	20	,	41	29)	50	32		59	3B	
33	21	-	42	2A	*	51	33		60	3C	
34	22	.	43	2B	+	52	34		61	3D	
35	23	:	44	2C	,	53	35		62	3E	
36	24	;	45	2D	;	54	36		63	3F	
37	25	<	46	2E	<	55	37		64	40	
38	26	=	47	2F	=	56	38		65	41	a
39	27	'	48	30	'	57	39		66	42	b
40	28	(49	31	(58	3A		67	43	c
41	29)	50	32)	59	3B		68	44	d
42	2A	*	51	33	*	60	3C		69	45	e
43	2B	+	52	34	+	61	3D		70	46	f
44	2C	,	53	35	,	62	3E		71	47	g
45	2D	;	54	36	;	63	3F		72	48	h
46	2E	<	55	37	<	64	40		73	49	i
47	2F	=	56	38	=	65	41		74	4A	j
48	30	>	57	39	>	66	42		75	4B	k
49	31	?	58	3A	?	67	43		76	4C	l
50	32	@	59	3B	@	68	44		77	4D	m
51	33	A	60	3C	A	69	45		78	4E	n
52	34	B	61	3D	B	70	46		79	4F	o
53	35	C	62	3E	C	71	47		80	50	p
54	36	D	63	3F	D	72	48		81	51	q
55	37	E	64	40	E	73	49		82	52	r
56	38	F	65	41	F	74	4A		83	53	s
57	39		66	42		75	4B		84	54	t
58	3A		67	43		76	4C		85	55	u
59	3B		68	44		77	4D		86	56	v
60	3C		69	45		78	4E		87	57	w
61	3D		70	46		79	4F		88	58	x
62	3E		71	47		80	50		89	59	y
63	3F		72	48		81	51		90	5A	z
64	40		73	49		82	52		91	5B	[
65	41	a	74	4A		83	53		92	5C	\
66	42	b	75	4B		84	54		93	5D]
67	43	c	76	4C		85	55		94	5E	^
68	44	d	77	4D		86	56		95	5F	_
69	45	e	78	4E		87	57		96	60	`
70	46	f	79	4F		88	58		97	61	
71	47	g	80	50		89	59		98	62	
72	48	h	81	51		90	5A		99	63	
73	49	i	82	52		91	5B		100	64	
74	4A	j	83	53		92	5C				
75	4B	k	84	54		93	5D				
76	4C	l	85	55		94	5E				
77	4D	m	86	56		95	5F				
78	4E	n	87	57							
79	4F	o	88	58							
80	50	p	89	59							
81	51	q	90	5A							
82	52	r	91	5B							
83	53	s	92	5C							
84	54	t	93	5D							
85	55	u	94	5E							
86	56	v	95	5F							
87	57	w									
88	58	x									
89	59	y									
90	5A	z									
91	5B	[
92	5C	\									
93	5D]									
94	5E	^									
95	5F	_									
96	60	`									
97	61										
98	62										
99	63										
100	64										

- `ord()`:
input type: character
output type: integer
- `chr()`:
input type: integer
output type: character
- What is `chr(33)`?

ord() and chr()

(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	00		9	09		18	12		27	1B	
1	01		10	0A		19	13		28	1C	
2	02		11	0B		20	14		29	1D	
3	03		12	0C		21	15	!	30	1E	
4	04		13	0D		22	16	"	31	1F	
5	05		14	0E		23	17	#	32	20	
6	06		15	0F		24	18	\$	33	21	
7	07		16	10		25	19	%	34	22	
8	08		17	11		26	1A	&	35	23	
9	09		18	12		27	1B	'	36	24	
10	0A		19	13		28	1C	(37	25	
11	0B		20	14		29	1D)	38	26	
12	0C		21	15	!	30	1E	*	39	27	
13	0D		22	16	"	31	1F	+	40	28	
14	0E		23	17	#	32	20	,	41	29	
15	0F		24	18	\$	33	21	;	42	2A	
16	10		25	19	%	34	22	:	43	2B	
17	11		26	1A	&	35	23	<	44	2C	
18	12		27	1B	'	36	24	=	45	2D	
19	13		28	1C	(37	25	>	46	2E	
20	14		29	1D)	38	26	?	47	2F	
21	15	!	30	1E	*	39	27	@	48	30	
22	16	"	31	1F	+	40	28	A	49	31	
23	17	#	32	20	,	41	29	B	50	32	
24	18	\$	33	21	;	42	2A	C	51	33	
25	19	%	34	22	:	43	2B	D	52	34	
26	1A	&	35	23	<	44	2C	E	53	35	
27	1B	'	36	24	=	45	2D	F	54	36	
28	1C	(37	25	>	46	2E		55	37	
29	1D)	38	26	?	47	2F		56	38	
30	1E	*	39	27	@	48	30		57	39	
31	1F	+	40	28	A	49	31		58	3A	
32	20	,	41	29	B	50	32		59	3B	
33	21	;	42	2A	C	51	33		60	3C	
34	22	:	43	2B	D	52	34		61	3D	
35	23	<	44	2C	E	53	35		62	3E	
36	24	=	45	2D	F	54	36		63	3F	
37	25	>	46	2E		55	37		64	40	
38	26	?	47	2F		56	38		65	41	a
39	27	@	48	30		57	39		66	42	b
40	28	A	49	31		58	3A		67	43	c
41	29	B	50	32		59	3B		68	44	d
42	2A	C	51	33		60	3C		69	45	e
43	2B	D	52	34		61	3D		70	46	f
44	2C	E	53	35		62	3E		71	47	g
45	2D	F	54	36		63	3F		72	48	h
46	2E		55	37		64	40		73	49	i
47	2F		56	38		65	41		74	4A	j
48	30		57	39		66	42		75	4B	k
49	31		58	3A		67	43		76	4C	l
50	32		59	3B		68	44		77	4D	m
51	33		60	3C		69	45		78	4E	n
52	34		61	3D		70	46		79	4F	o
53	35		62	3E		71	47		80	50	
54	36		63	3F		72	48		81	51	
55	37		64	40		73	49		82	52	
56	38		65	41		74	4A		83	53	
57	39		66	42		75	4B		84	54	
58	3A		67	43		76	4C		85	55	
59	3B		68	44		77	4D		86	56	
60	3C		69	45		78	4E		87	57	
61	3D		70	46		79	4F		88	58	
62	3E		71	47		80	50		89	59	
63	3F		72	48		81	51		90	5A	
64	40		73	49		82	52		91	5B	
65	41	a	74	4A		83	53		92	5C	
66	42	b	75	4B		84	54		93	5D	
67	43	c	76	4C		85	55		94	5E	
68	44	d	77	4D		86	56		95	5F	
69	45	e	78	4E		87	57		96	60	
70	46	f	79	4F		88	58		97	61	
71	47	g	80	50		89	59		98	62	
72	48	h	81	51		90	5A		99	63	
73	49	i	82	52		91	5B		100	64	
74	4A	j	83	53		92	5C				
75	4B	k	84	54		93	5D				
76	4C	l	85	55		94	5E				
77	4D	m	86	56		95	5F				
78	4E	n	87	57							
79	4F	o	88	58							
80	50		89	59							
81	51		90	5A							
82	52		91	5B							
83	53		92	5C							
84	54		93	5D							
85	55		94	5E							
86	56		95	5F							
87	57										
88	58										
89	59										
90	5A										
91	5B										
92	5C										
93	5D										
94	5E										
95	5F										
96	60										
97	61										
98	62										
99	63										
100	64										

- `ord()`:
input type: character
output type: integer
- `chr()`:
input type: integer
output type: character
- What is `chr(33)`?
- What is `ord("$")`?

The plus (+) operator for numbers and strings



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

The plus (+) operator for numbers and strings



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

The plus (+) operator 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`.

The plus (+) operator for numbers and strings



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

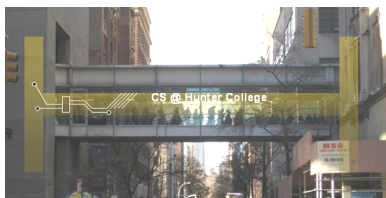
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

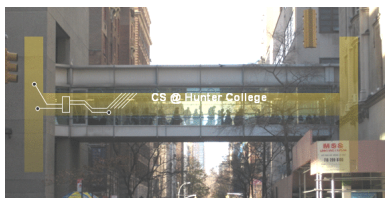
Weekly Reminders!



Before next lecture, don't forget to:

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

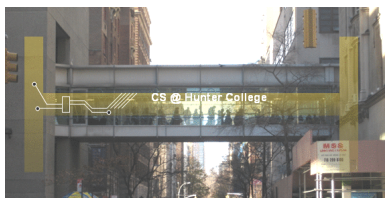
Weekly Reminders!



Before next lecture, don't forget to:

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

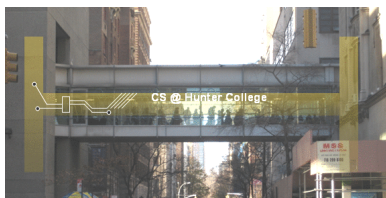
Weekly Reminders!



Before next lecture, don't forget to:

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

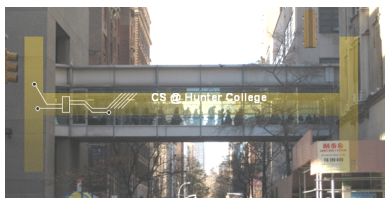
Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Lecture Slips & Writing Boards



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