Comparison Operators

What is a Comparison Operator?

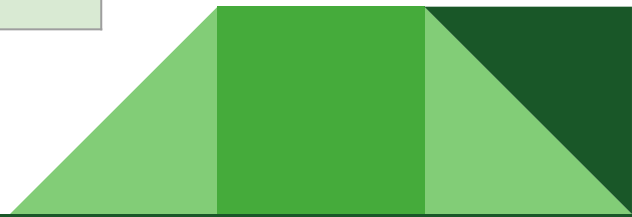
We've learned about **mathematical**, **string**, and **logical** operators so far - today we're going to learn yet another set of operators!

Comparison Operators do what you might expect - they *compare* two values.

They'll always result in a `Boolean` value, even if the values being compared aren't `Booleans`!

Comparison Operators (numbers)

Symbol	What it check	Example	Result of Example
==	Equals	`7 == 5`	`False`
!=	Not Equals	`10 != 3`	`True`
>	Greater Than	`8 > 2`	`True`
>=	Greater Than or Equal To	`12 >= 15`	`False`
<	Less Than	`1 < 0`	`False`
<=	Less Than or Equal To	`4 <= 9`	`True`



Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = int(input("How old are you? "))
old_enough = age >= 18
print("You can vote: " + str(old_enough))
```

```
How old are you?
```

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

Let's say the user enters **15**.

```
age = int(input("How old are you? "))
old_enough = age >= 18
print("You can vote: " + str(old_enough))
```

How old are you? **15**

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

Let's say the user enters **15**.

```
age = 15
old_enough = age >= 18
print("You can vote: " + str(old_enough))
```

How old are you? **15**

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = 15
old_enough = 15 >= 18
print("You can vote: " + str(old_enough))
```

How old are you? **15**

Let's say the user enters **15**. 15 is **not** greater than or equal to 18, therefore `old_enough` will be `False`.

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = 15
old_enough = False
print("You can vote: " + str(old_enough))
```

How old are you? **15**

Let's say the user enters **15**. 15 is **not** greater than or equal to 18, therefore `old_enough` will be `False`.

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = 15
old_enough = False
print("You can vote: " + str(old_enough))
```

Let's say the user enters **15**. 15 is **not** greater than or equal to 18, therefore `old_enough` will be `False`.

How old are you? **15**
You can vote: False

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = int(input("How old are you? "))
old_enough = age >= 18
print("You can vote: " + str(old_enough))
```

How old are you? **22**

Let's say the user enters **22** instead.

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = 22
old_enough = age >= 18
print("You can vote: " + str(old_enough))
```

How old are you? **22**

Let's say the user enters **22** instead.

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = 22
old_enough = 22 >= 18
print("You can vote: " + str(old_enough))
```

How old are you? **22**

Let's say the user enters **22** instead.
22 **is** greater than or equal to 18, therefore `old_enough` will be `True`.

Using Comparison Operators

We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = 22
old_enough = True
print("You can vote: " + str(old_enough))
```

How old are you? **22**

Let's say the user enters **22** instead.
22 **is** greater than or equal to 18, therefore `old_enough` will be `True`.

Using Comparison Operators

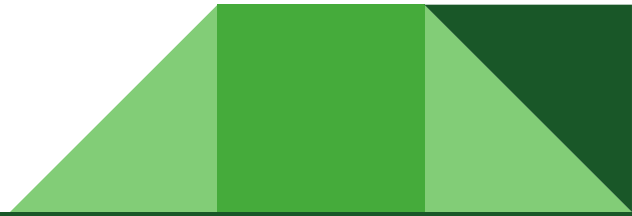
We are allowed to use the values stored within variables when we do a comparison - we don't only need to use literal values.

For example:

```
age = 22
old_enough = True
print("You can vote: " + str(old_enough))
```

How old are you? **22**
You can vote: True

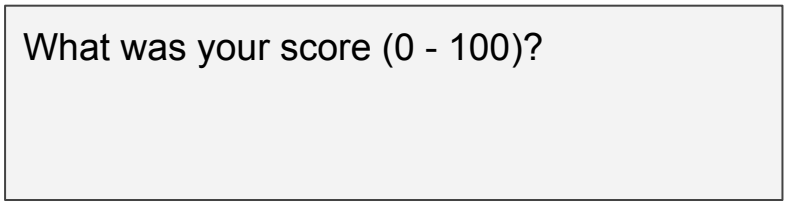
Let's say the user enters **22** instead.
22 **is** greater than or equal to 18, therefore `old_enough` will be `True`.



Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = int(input("What was your score (0 - 100)? "))
got_b = grade >= 80 and grade < 90
print("You got a B: " + str(got_b))
```



Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = int(input("What was your score (0 - 100)? "))
got_b = grade >= 80 and grade < 90
print("You got a B: " + str(got_b))
```

```
What was your score (0 - 100)? 92
```

Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = 92
got_b = grade >= 80 and grade < 90
print("You got a B: " + str(got_b))
```

What was your score (0 - 100)? **92**

Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = 92
got_b = 92 >= 80 and grade < 90
print("You got a B: " + str(got_b))
```

What was your score (0 - 100)? **92**

Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = 92
got_b = True and grade < 90
print("You got a B: " + str(got_b))
```

What was your score (0 - 100)? **92**

Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = 92
got_b = True and 92 < 90
print("You got a B: " + str(got_b))
```

What was your score (0 - 100)? **92**

Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = 92
got_b = True and False
print("You got a B: " + str(got_b))
```

What was your score (0 - 100)? **92**

Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = 92
got_b = False
print("You got a B: " + str(got_b))
```

What was your score (0 - 100)? **92**

Combining Comparison & Logical Operators

We can combine the use of these two types of operators! If we use an `and` or an `or` in between two comparisons, we can use the results of those comparisons for our logical operation!

```
grade = 92
got_b = False
print("You got a B: " + str(got_b))
```

```
What was your score (0 - 100)? 92
You got a B: False
```

Comparisons Operators (strings)

Symbol	What it check	Example	Result of Example
==	Equals	`"hi" == "hey"`	`False`
!=	Not Equals	`"oy" != "ahoy"`	`True`
>	Later in dictionary	`"yo" > "oy"`	`True`
>=	Later (or equal) in dictionary	`"hello" >= "hi"`	`False`
<	Earlier in dictionary	`"sup" < "hey"`	`False`
<=	Earlier (or equal) in dictionary	`"howdy" <= "yo"`	`True`

The ASCII Table

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]

ASCII stands for American Standard Code for Information Interchange!

Since this is the **standard**, just about every computer knows this set of number-character pairings, so we can use these values pretty reliably.

The ASCII Table

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]

Notice that the capital letters have a **lower** ASCII value associated with them - they are "smaller" than lowercase letters.

`"A"` is **less than** `"a"`.

The ASCII Table

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]

Here's the list of just about every character we can type in on our keyboards - ASCII values 0 - 31 are reserved for special non-keyboard characters.

Also see that digits are lower than **any** letter, upper OR lowercase.

Practicing Comparing Strings

```
"apple" > "banana"      ->

"apple" == "Apple"      ->

"banana" > "Banana"     ->

"APPLE" != "apple"      ->

"apple" <= "Apple"      ->
```

Practicing Comparing Strings

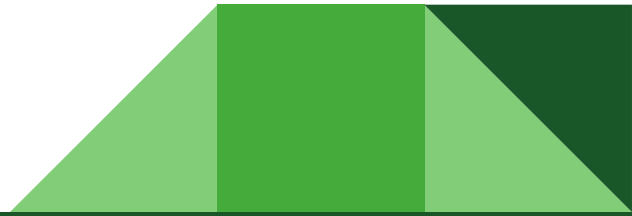
```
"apple" > "banana"      -> False

"apple" == "Apple"      ->

"banana" > "Banana"     ->

"APPLE" != "apple"      ->

"apple" <= "Apple"      ->
```



Practicing Comparing Strings

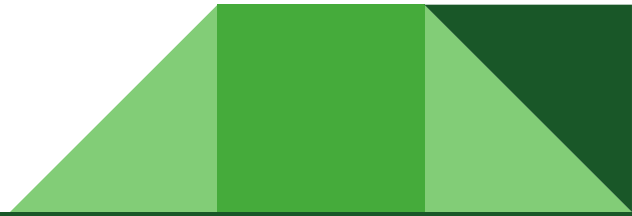
```
"apple" > "banana"      -> False

"apple" == "Apple"      -> False

"banana" > "Banana"     ->

"APPLE" != "apple"      ->

"apple" <= "Apple"      ->
```



Practicing Comparing Strings

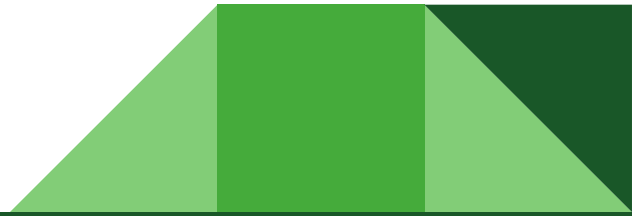
```
"apple" > "banana"      -> False

"apple" == "Apple"      -> False

"banana" > "Banana"     -> True

"APPLE" != "apple"      ->

"apple" <= "Apple"      ->
```



Practicing Comparing Strings

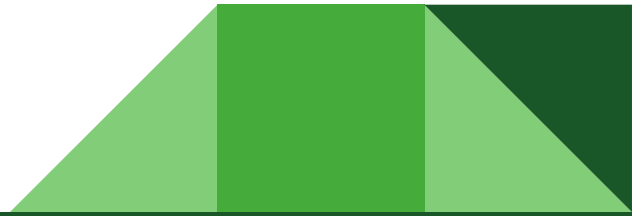
```
"apple" > "banana"      -> False

"apple" == "Apple"      -> False

"banana" > "Banana"     -> True

"APPLE" != "apple"      -> True

"apple" <= "Apple"      ->
```



Practicing Comparing Strings

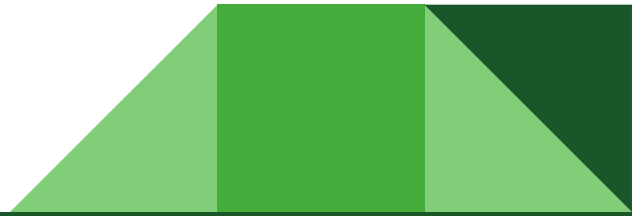
```
"apple" > "banana"      -> False

"apple" == "Apple"      -> False

"banana" > "Banana"     -> True

"APPLE" != "apple"      -> True

"apple" <= "Apple"      -> False
```



De Morgan's Laws

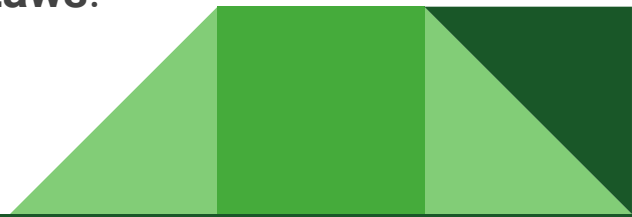
If I use a `not` on an `and` or an `or`, here's what will happen:

`not (a and b)` equals `not a or not b`

`not (a or b)` equals `not a and not b`

Essentially, we swap `and` to `or` (and vice-versa) and put a `not` in front of each of the original operands.

This property of logical operators is called **De Morgan's Laws**.
It's named after a 19th century mathematician.



Logical & Comparison Order of Operations

When we are combining both **Logical** and **Comparison** operators together, we will always evaluate our **Comparison** operators first, followed by out **Logical** operators.

We can use parentheses with these operators as well - parentheses are always first in the Order of Operations.