# 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

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))
```

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))
```

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))
```

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))
```

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

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.

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

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))
```

Let's say the user enters 22 instead.

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 22 instead.

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

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))
```

Let's say the user enters 22 instead

22 is greater than or equal to 18, therefore old\_enough will be True.

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))
```

Let's say the user enters 22 instead

22 **is** greater than or equal to 18, therefore old\_enough will be True.

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))
```

Let's say the user enters 22 instead

22 **is** greater than or equal to 18, therefore old\_enough will be True.

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

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))</pre>
```

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))</pre>
```

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))</pre>
```

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))</pre>
```

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))</pre>
```

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))</pre>
```

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))
```

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))
```

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	1	65	41	Α	97	61	a
2	2	[START OF TEXT]	34	22		66	42	В	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	1	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	1	105	69	i
10	Α	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	В	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	С	[FORM FEED]	44	2C	,	76	4C	L	108	6C	1
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	1	79	4F	0	111	6F	0
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	р
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	Υ	121	79	У
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	Ť
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	1	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]
		-	•			•		_	1		

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	Hav	Char	Decimal	—	Char	Decimal	Цох	Char	Decimal	Цох	Char
											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	В	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	С	99	63	С
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	е
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	Н	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49		105	69	i
10	Α	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	В	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C		76	4C	L	108	6C	1
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	1	79	4F	0	111	6F	0
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	Р	112	70	р
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	INEGATIVE ACKNOWLEDGE	53	35	5	85	55	U	117	75	u
22	16	ISYNCHRONOUS IDLE1	54	36	6	86	56	V	118	76	v
23	17	IENG OF TRANS. BLOCKI	55	37	7	87	57	W	119	77	w
24	18	ICANCEL1	56	38	8	88	58	Х	120	78	х
25	19	[END OF MEDIUM]	57	39	9	89	59	Υ	121	79	У
26	1A	ISUBSTITUTE1	58	3A		90	5A	Z	122	7A	Z
27	1B	[ESCAPE]	59	3B		91	5B	T.	123	7B	
28	1C	[FILE SEPARATOR]	60	3C	, <	92	5C	Ĭ.	124	7C	ř
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	1	125	7D	3
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3E	?	95	5F		127	7E	[DEL1
J1	11	[UNIT SELAKATUK]	05	31		95	51	-	12/	71	[DLL]

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	Hov	Char	Decimal		Char	Decimal	Цох	Char	ı Decimal	Цоу	Char
											Cilai
0	0 1	[NULL] [START OF HEADING]	32 33	20 21	[SPACE]	64 65	40 41	@ A	96 97	60 61	_
1	2	ISTART OF HEADING	34	22	:	66	41	В	98	62	a b
3	3	1-11-11	35	23	#	67	42	C	98	63	
4	4	[END OF TEXT] [END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENOUIRY]	37	25	%	69	45	E	100	65	-
5 6	6	[ENQUIRT] [ACKNOWLEDGE]	38	26	% &	70	45 46	Ē	101	66	e
7	7		39	27	OX I	71	47	G	102	67	T
, 8	8	[BELL] [BACKSPACE]	40	28	,	72	48	Н	103	68	g
9	9		41	28	(	73	48		104	69	h
_		[HORIZONTAL TAB]	41	29 2A	*	74			105	69 6A	
10	A B	[LINE FEED]	42	2A 2B	+	75	4A 4B	J K	106	6B	j
11	_	[VERTICAL TAB]	43		+	75 76	4B 4C		107		k
12 13	С	[FORM FEED]	45	2C	*	77		L M		6C	1
14	D E	[CARRIAGE RETURN] [SHIFT OUT]	46	2D 2E	-	78	4D 4E	N	109 110	6D 6E	m
15	F	ISHIFT IN1	47	2F	,	79	4E 4F	0	111	6F	n
16	10	[SHIFT IN] IDATA LINK ESCAPET	48	2F 30	0	80	4F 50	P	111	ог 70	0
16		[DATA LINK ESCAPE]  IDEVICE CONTROL 11	48	31	1	81	50	-	112	70	р
18	11		50	32	2	82	52	Q		72	q
	12 13	[DEVICE CONTROL 2]	51	33	3	83	53	R	114 115	73	r
19		[DEVICE CONTROL 3]	52	34	4	84		T			S
20	14	[DEVICE CONTROL 4]	53	35	5	85	54 55	-	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	54	35 36	6	86		U	117	75 76	u
22 23	16	[SYNCHRONOUS IDLE]	55	36 37	7	87	56 57	V	118	76 77	V
	17	[ENG OF TRANS. BLOCK]	56	38	8	88		W	119		W
24	18	[CANCEL]	57		_		58	X	120	78	X
25	19	[END OF MEDIUM]		39	9	89	59	Y	121	79	У
26	1A	[SUBSTITUTE]	58	3A	1	90	5A	Z	122	7A	Z
27	1B	[ESCAPE]	59	3B	1	91	5B	L	123	7B	1
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	1	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	Ž	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.

```
"apple" > "banana" ->
"apple" == "Apple" ->
"banana" > "Banana" ->
"APPLE" != "apple" ->
"apple" <= "Apple" ->
```

```
"apple" > "banana" -> False
"apple" == "Apple" ->
"banana" > "Banana" ->
"APPLE" != "apple" ->
"apple" <= "Apple" ->
```

```
"apple" > "banana" -> False
"apple" == "Apple" -> False
"banana" > "Banana" ->
"APPLE" != "apple" ->
"apple" <= "Apple" ->
```

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

"apple" == "Apple" -> False

"banana" > "Banana" -> True

"APPLE" != "apple" ->

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

```
"apple" > "banana" -> False
"apple" == "Apple" -> False
"banana" > "Banana" -> True
"APPLE" != "apple" -> True
"apple" <= "Apple" ->
```

```
"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 19<sup>th</sup> 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.