



CSCI 141

Lecture 7:
Conditionals:
`if, else, elif`

Announcements

Announcements

- A2 is out, due next **Tuesday** night. Get started early!

Announcements

- A2 is out, due next **Tuesday** night. Get started early!
- QOTD: late submissions are **not** accepted.

Announcements

- A2 is out, due next **Tuesday** night. Get started early!
- QOTD: late submissions are **not** accepted.
 - Not even at 1:07pm, after I've gone over the answers.

Announcements

- A2 is out, due next **Tuesday** night. Get started early!
- QOTD: late submissions are **not** accepted.
 - Not even at 1:07pm, after I've gone over the answers.
- The decimal-to-binary conversion question had an error in the solution. I regraded them all.

Announcements

- A2 is out, due next **Tuesday** night. Get started early!
- QOTD: late submissions are **not** accepted.
 - Not even at 1:07pm, after I've gone over the answers.
- The decimal-to-binary conversion question had an error in the solution. I regraded them all.
 - Points should be correct, but answers marked "incorrect" may not be incorrect, and vice versa.

Goals

- Understand the behavior of the equality comparison operators (`==`, `!=`) on non-numeric types.
- Know how to use an `if statement` to conditionally execute a block of code.
- Know how to use an `if/else statement` to choose which of two code blocks to execute.
- Understand how conditional statements can be `nested` to make decisions among more than two possibilities.
- Know how to use `if/elif/else` statements.

Last Time

- New type: `bool`
- New operators:
 - comparison `<`, `>`, `<=`, `>=`, `==`, `!=`
 - logical not, and, or
- Operator precedence

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Write the truth table:

		x xor y	
		y	
		T	F
x	T		
	F		

Note: xor is not a python operator.

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

`not (a and b) and not(not a and not b)`

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

`a or b`

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

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

not (a **and** b) **and** **not** (**not** a **and** **not** b)

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

not (**a and b**) **and not** (**not a and not b**)

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not (T and T) and not(not T and not T)`

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not (T and T) and not (not T and not T)`

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

`not T and not (not T and not T)`

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not T and not (not T and not T)`

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

not T and not (F and F)

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not T and not (F and F)`

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

not T and not F

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

`not T` and `not F`

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

F and not F

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

F and not F

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

F and T

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

F and T

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

F

		b	
		T	F
a	T		
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

F

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not (T and F) and not(not T and not F)`

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

`not (T and F) and not (not T and not F)`

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

not F and not (F and T)

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not F and not (F and T)`

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

not F and not F

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not F` and `not F`

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

T **and** T

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

T and T

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

T

		b	
		T	F
a	T	F	
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

T

		b	
		T	F
a	T	F	T
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not (F and T) and not(not F and not T)`

		b	
		T	F
a	T	F	T
	F		

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

`not (F and T) and not(not F and not T)`

		b	
		T	F
a	T	F	T
	F		

... F/T comes out the same as T/F

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

not (F **and** T) **and** **not**(**not** F **and** **not** T)

		b	
		T	F
a	T	F	T
	F	T	

... F/T comes out the same as T/F

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

not (F **and** F) **and** **not** (**not** F **and** **not** F)

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

not (F **and** F) **and** **not** (**not** F **and** **not** F)

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not (F and F) and not (not F and not F)`

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

not F and not (T and T)

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

`not F and not (T and T)`

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

not F and not T

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

`not F` and `not T`

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

T **and** F

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to a xor b?

T and F

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

F

		b	
		T	F
a	T	F	T
	F	T	

QOTD

Exclusive or ("xor"): True if exactly one of the operands is true. Which of the following evaluates to $a \text{ xor } b$?

F

		b	
		T	F
a	T	F	T
	F	T	F

QOTD

For each expression, give the type and value.

`True and False or True`

`2**3.0`

`not 1 + 5 // 2 == 3 and 4 < 5 or 4 != 5`

Boolean Expressions: Another worked example

What does this print?

```
print((3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5)
```

Boolean Expressions: Another worked example

What does this print?

```
print((3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5)
```

Worked solution is on the following slides for your reference.

Last Time:

Boolean Expressions

Another example: what does this print?

```
print((3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5)
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print((3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5)
```


Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True ) and 3 < 5 )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True) and 3 < 5 )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True) and 3 < 5 )
```

```
print( ( False or True) and 3 < 5 )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True) and 3 < 5 )
```

```
print( ( False or True ) and 3 < 5 )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True) and 3 < 5 )
```

```
print( ( False or True ) and 3 < 5 )
```

```
print( True and 3 < 5 )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True) and 3 < 5 )
```

```
print( ( False or True ) and 3 < 5 )
```

```
print( True and 3 < 5 )
```


Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True) and 3 < 5 )
```

```
print( ( False or True ) and 3 < 5 )
```

```
print( True and 3 < 5 )
```

```
print( True and True )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True ) and 3 < 5 )
```

```
print( ( False or True ) and 3 < 5 )
```

```
print( True and 3 < 5 )
```

```
print( True and True )
```

Last Time:

Boolean Expressions

Another example: what does this print?

```
print( (3 == 5 or (3 != 5 and 5 != 7)) and 3 < 5 )
```

```
print( (3 == 5 or ( True and True )) and 3 < 5 )
```

```
print( (3 == 5 or True ) and 3 < 5 )
```

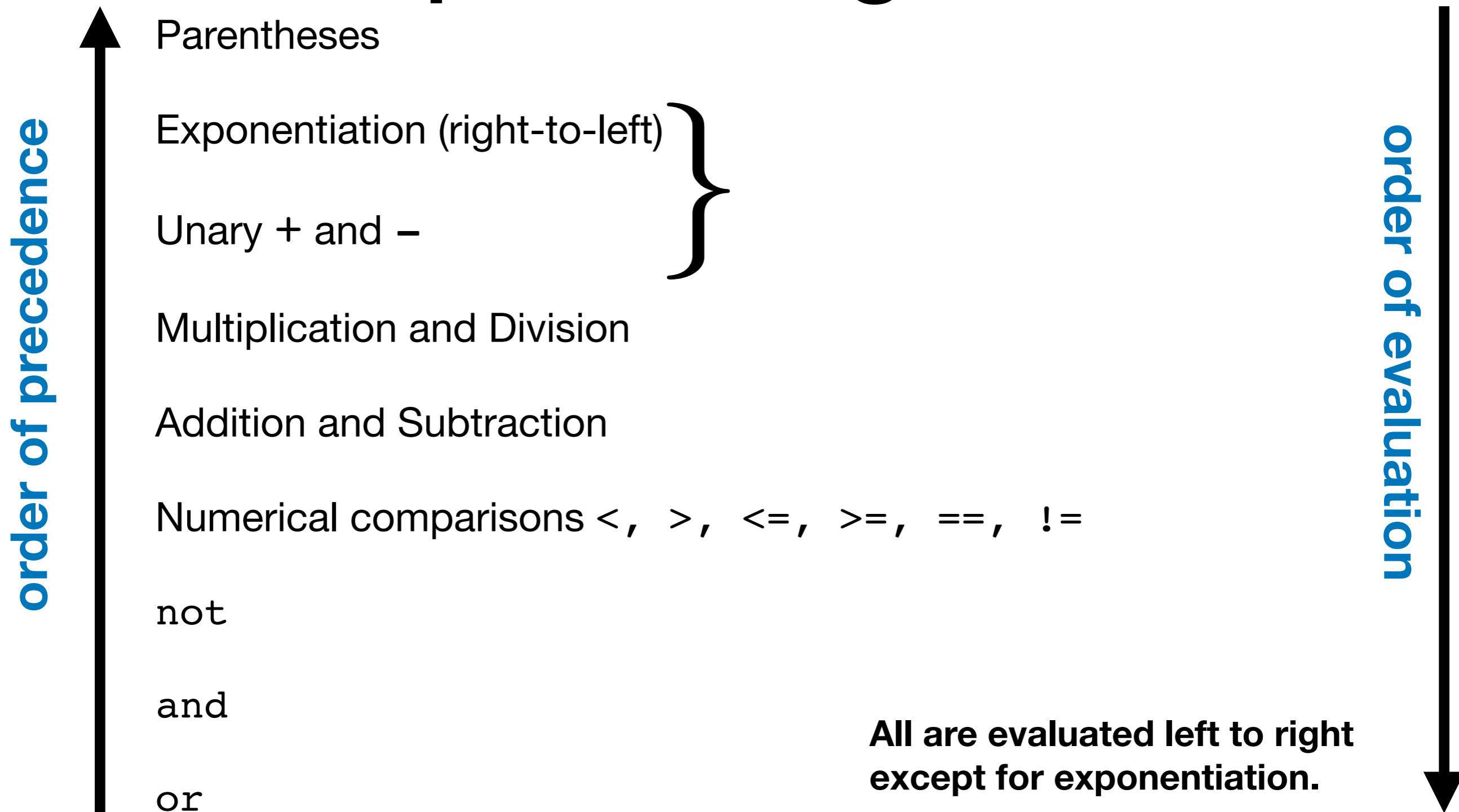
```
print( ( False or True ) and 3 < 5 )
```

```
print( True and 3 < 5 )
```

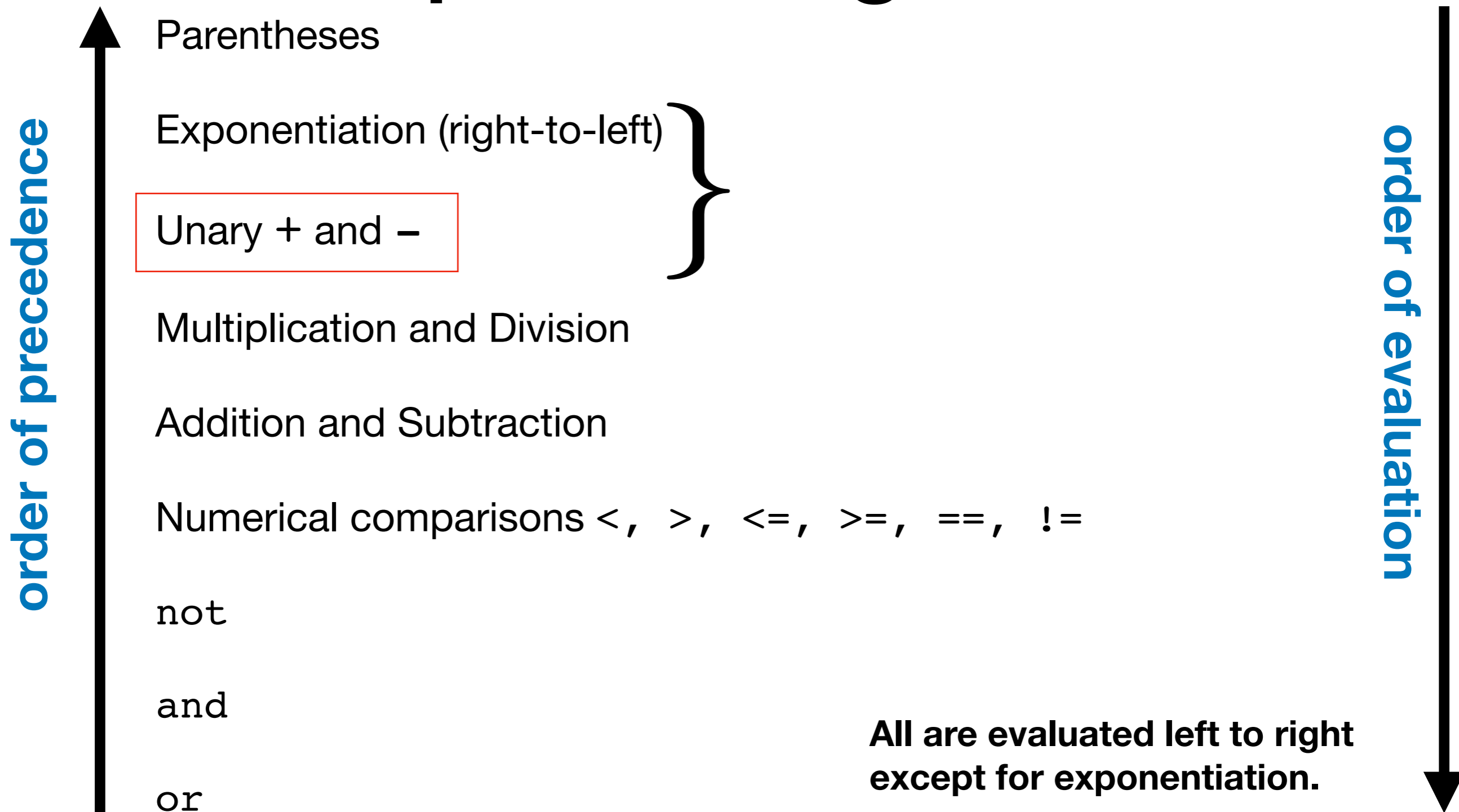
```
print( True and True )
```

```
print( True )
```

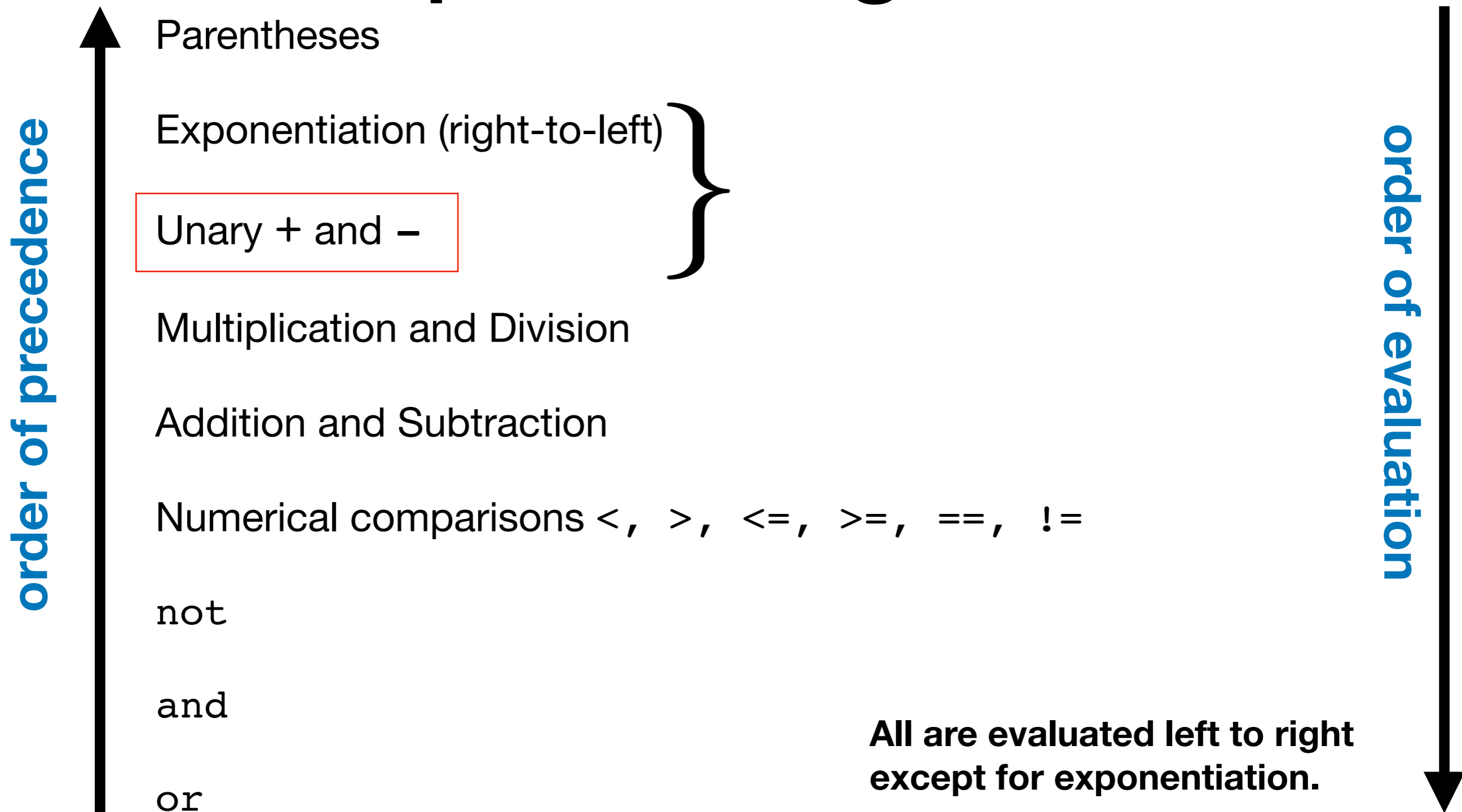
Operator Precedence, Updated Again



Operator Precedence, Updated Again

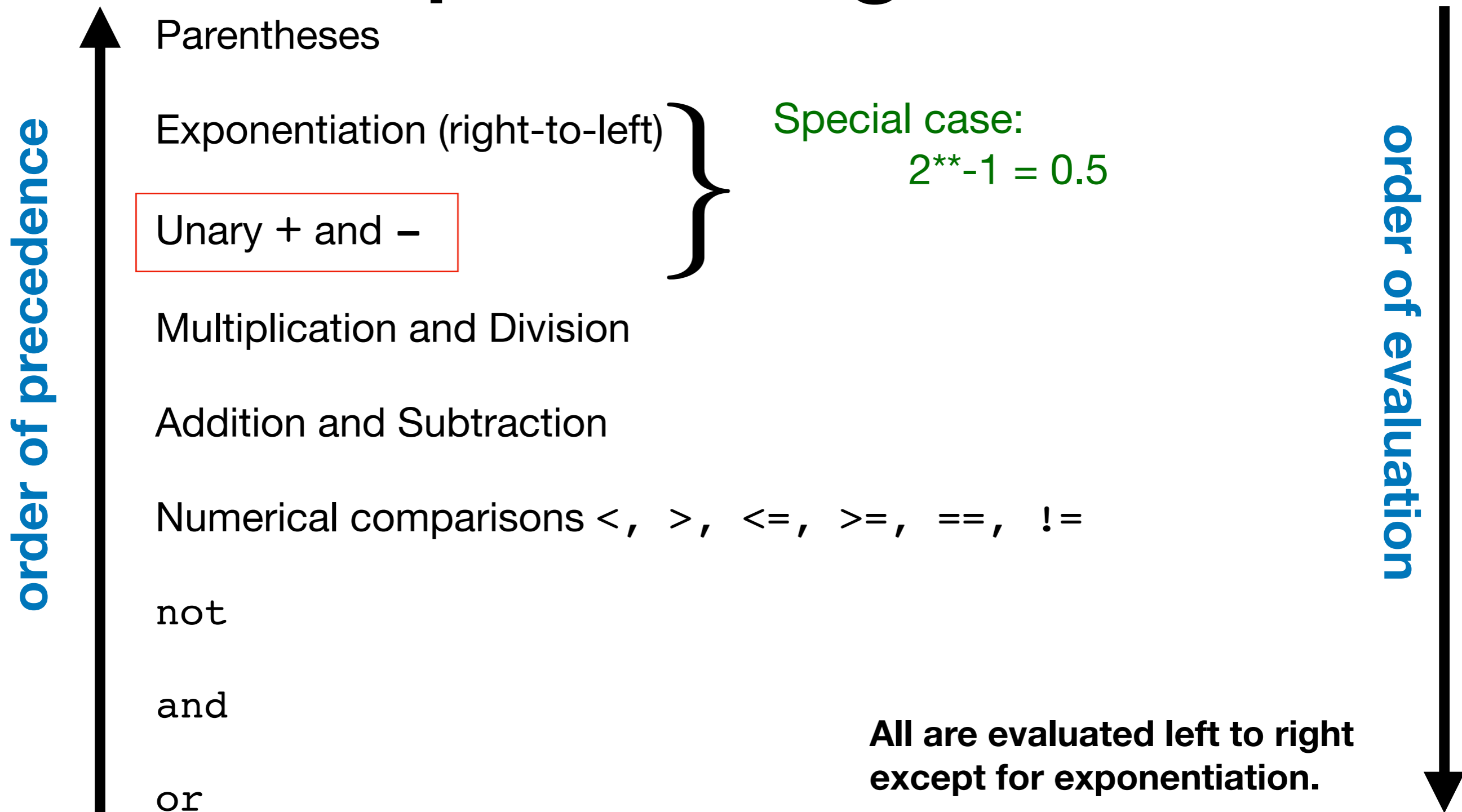


Operator Precedence, Updated Again

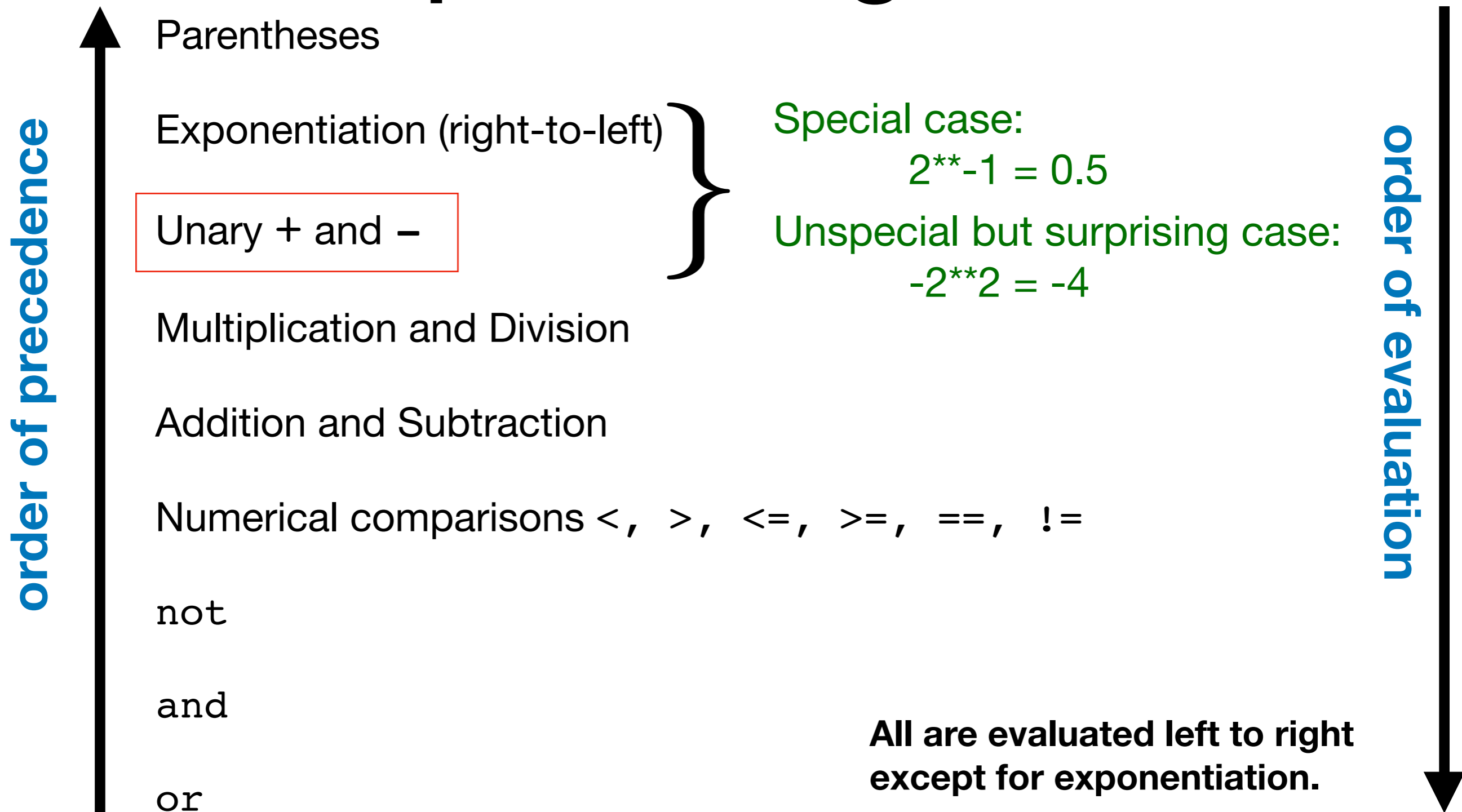


You can look up all the details: <https://docs.python.org/3/reference/expressions.html#operator-precedence>

Operator Precedence, Updated Again



Operator Precedence, Updated Again



You can look up all the details: <https://docs.python.org/3/reference/expressions.html#operator-precedence>

Equality Comparisons

- The operators `==` and `!=` check whether two values are equal or not.
- Unlike some operators (e.g., `//`), the concept of equality has meaning for some non-numeric types:

```
4 == 5  
"abc" == "bcd"  
"abc" == "abc"  
type(4) == type(5)  
5.0 == 5
```

Equality Comparisons

- The operators `==` and `!=` check whether two values are equal or not.
- Unlike some operators (e.g., `//`), the concept of equality has meaning for some non-numeric types:

```
4 == 5           => False
```

```
"abc" == "bcd"
```

```
"abc" == "abc"
```

```
type(4) == type(5)
```

```
5.0 == 5
```

Equality Comparisons

- The operators `==` and `!=` check whether two values are equal or not.
- Unlike some operators (e.g., `//`), the concept of equality has meaning for some non-numeric types:

```
4 == 5           => False
```

```
"abc" == "bcd"  => False
```

```
"abc" == "abc"
```

```
type(4) == type(5)
```

```
5.0 == 5
```

Equality Comparisons

- The operators `==` and `!=` check whether two values are equal or not.
- Unlike some operators (e.g., `//`), the concept of equality has meaning for some non-numeric types:

```
4 == 5 => False
```

```
"abc" == "bcd" => False
```

```
"abc" == "abc" => True
```

```
type(4) == type(5)
```

```
5.0 == 5
```

Equality Comparisons

- The operators `==` and `!=` check whether two values are equal or not.
- Unlike some operators (e.g., `//`), the concept of equality has meaning for some non-numeric types:

```
4 == 5 => False
```

```
"abc" == "bcd" => False
```

```
"abc" == "abc" => True
```

```
type(4) == type(5) => True
```

```
5.0 == 5
```

Equality Comparisons

- The operators `==` and `!=` check whether two values are equal or not.
- Unlike some operators (e.g., `//`), the concept of equality has meaning for some non-numeric types:

```
4 == 5           => False
"abc" == "bcd"  => False
"abc" == "abc"  => True
type(4) == type(5) => True
5.0 == 5        => True
```

Equality Comparisons

Lightning round!



True or False?

Equality Comparisons

Lightning round!

$$10 == 4 + 6$$



True or False?

Equality Comparisons

Lightning round!

$$10 == 4 + 6$$



True or False?

Equality Comparisons

Lightning round!

$$10 == 4 + 6$$

$$"abc" == "ab" + "c"$$



True or False?

Equality Comparisons

Lightning round!

10 == 4 + 6

"abc" == "ab" + "c"



True or False?

Equality Comparisons

Lightning round!

10 == 4 + 6

"abc" == "ab" + "c"

'abc' == "abc"



True or False?

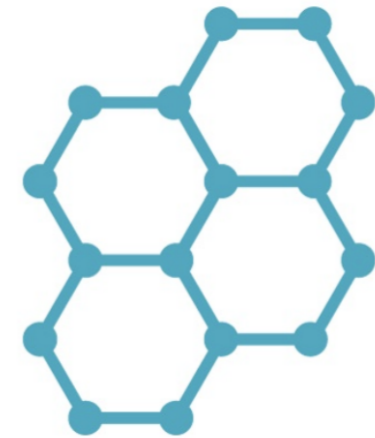
Equality Comparisons

Lightning round!

10 == 4 + 6

"abc" == "ab" + "c"

'abc' == "abc"



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6`

`"abc" == "ab" + "c"`

`'abc' == "abc"`

`"Scott" == "scott"`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6`

`"abc" == "ab" + "c"`

`'abc' == "abc"`

`"Scott" == "scott"`

`(4+3 > 5) == (1.0 > 4)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6`

`"abc" == "ab" + "c"`

`'abc' == "abc"`

`"Scott" == "scott"`

`(4+3 > 5) == (1.0 > 4)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6`

`"abc" == "ab" + "c"`

`'abc' == "abc"`

`"Scott" == "scott"`

`(4+3 > 5) == (1.0 > 4)`

`int(5.6) != int(5.1)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6 => True`

`"abc" == "ab" + "c"`

`'abc' == "abc"`

`"Scott" == "scott"`

`(4+3 > 5) == (1.0 > 4)`

`int(5.6) != int(5.1)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6 => True`

`"abc" == "ab" + "c" => True`

`'abc' == "abc"`

`"Scott" == "scott"`

`(4+3 > 5) == (1.0 > 4)`

`int(5.6) != int(5.1)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6 => True`

`"abc" == "ab" + "c" => True`

`'abc' == "abc" => True`

`"Scott" == "scott"`

`(4+3 > 5) == (1.0 > 4)`

`int(5.6) != int(5.1)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6 => True`

`"abc" == "ab" + "c" => True`

`'abc' == "abc" => True`

`"Scott" == "scott" => False`

`(4+3 > 5) == (1.0 > 4)`

`int(5.6) != int(5.1)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6 => True`

`"abc" == "ab" + "c" => True`

`'abc' == "abc" => True`

`"Scott" == "scott" => False`

`(4+3 > 5) == (1.0 > 4) => False`

`int(5.6) != int(5.1)`



True or False?

Equality Comparisons

Lightning round!

`10 == 4 + 6 => True`

`"abc" == "ab" + "c" => True`

`'abc' == "abc" => True`

`"Scott" == "scott" => False`

`(4+3 > 5) == (1.0 > 4) => False`

`int(5.6) != int(5.1) => False`



True or False?

Today

Today

- Last week: everything you already knew how to do using a calculator.

Today

- Last week: everything you already knew how to do using a calculator.



Today

- Last week: everything you already knew how to do using a calculator.
- Last lecture: representing and manipulating boolean (true/false) expressions and values.



Today

Today

- Last week: everything you already knew how to do using a calculator.
- Last lecture: representing and manipulating boolean (true/false) expressions and values.

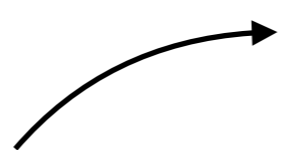
Today

- Last week: everything you already knew how to do using a calculator.
- Last lecture: representing and manipulating boolean (true/false) expressions and values.
- Today: Making **decisions** based on the value of a boolean expression.


Today

- Last week: everything you already knew how to do using a calculator.
- Last lecture: representing and manipulating boolean (true/false) expressions and values.
- Today: Making **decisions** based on the value of a boolean expression.

about **what** code to execute



Today

- Last week: everything you already knew how to do using a calculator.
- Last lecture: representing and manipulating boolean (true/false) expressions and values.
- Today: Making **decisions** based on the value of a boolean expression.  about **what** code to execute
- Also: a new kind of **statement!**

Let's talk about the weather

Let's talk about the weather

- You wish to write a software system that recommends what to wear/bring based on the current weather conditions.

Let's talk about the weather

- You wish to write a software system that recommends what to wear/bring based on the current weather conditions.
- In a later version, you will hook your software up to automated weather sensors that read temperature, wind, and precipitation data in real time.

Let's talk about the weather

- You wish to write a software system that recommends what to wear/bring based on the current weather conditions.
- In a later version, you will hook your software up to automated weather sensors that read temperature, wind, and precipitation data in real time.
- For now, we'll just ask the user.

Let's talk about the weather

Suppose we have `bool` variable `is_raining`

Here's the logic (pseudocode):

- if it is raining, tell the user to bring a raincoat

Here's the Python code:

Let's talk about the weather

Suppose we have `bool` variable `is_raining`

Here's the logic (pseudocode):

- if it is raining, tell the user to bring a raincoat

Here's the Python code:

```
if is_raining:  
    print("You should wear a raincoat!")
```

Let's talk about the weather

Suppose we have `bool` variables `is_raining` and `is_windy`

Here's the logic (pseudocode):

- if it is raining and windy, tell the user to bring a raincoat
- if is raining and not windy, tell the user to bring an umbrella

Here's the Python code:

Let's talk about the weather

Suppose we have bool variables `is_raining` and `is_windy`

Here's the logic (pseudocode):

- if it is raining and windy, tell the user to bring a raincoat
- if is raining and not windy, tell the user to bring an umbrella

Here's the Python code:


```
if is_raining and is_windy:
    print("You should wear a raincoat!")
if is_raining and not is_windy:
    print("You should bring an umbrella")
```


The `if` statement

```
if is_raining:  
    print("You should wear a raincoat!")
```

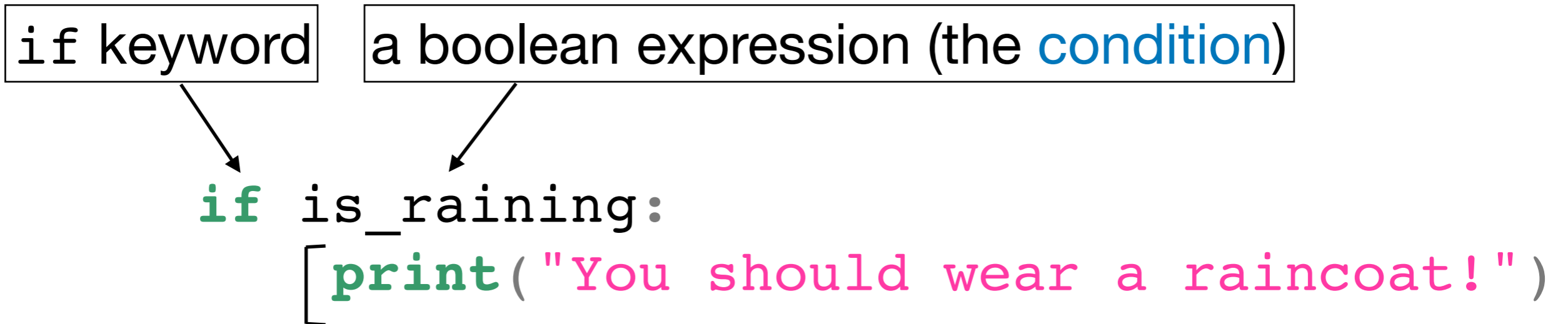
The `if` statement

`if` keyword

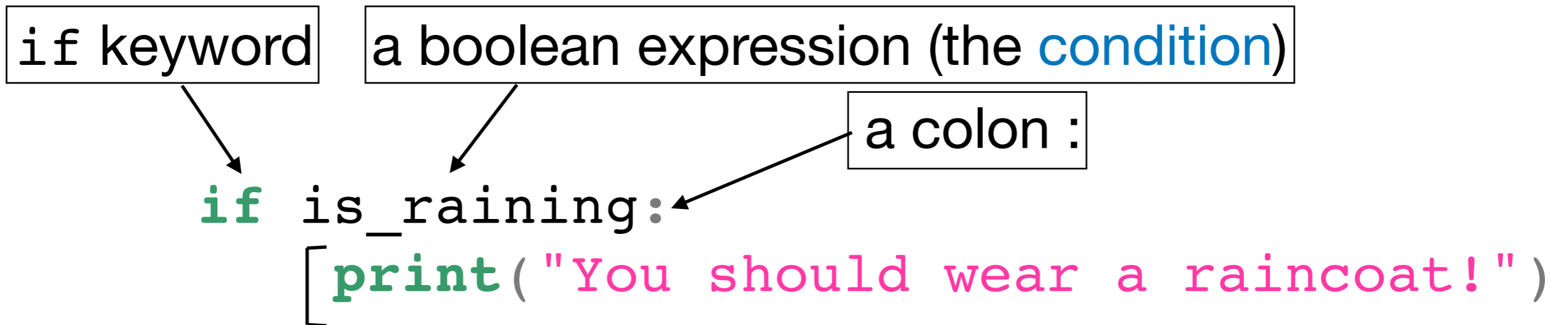


```
if is_raining:  
    print("You should wear a raincoat!")
```

The `if` statement



The `if` statement



The `if` statement

`if` keyword

a boolean expression (the **condition**)

a colon `:`

```
if is_raining:  
    print("You should wear a raincoat!")
```

an indented **code block**: one or more statements to be executed if the boolean expression evaluates to **True**

The `if` statement

`if` keyword

a boolean expression (the `condition`)

a colon `:`

```
if is_raining:  
    print("You should wear a raincoat!")
```

an indented `code block`: one or more statements to be executed if the boolean expression evaluates to **True**

Notes:

The `if` statement

`if` keyword

a boolean expression (the `condition`)

a colon `:`

```
if is_raining:  
    print("You should wear a raincoat!")
```

an indented `code block`: one or more statements to be executed if the boolean expression evaluates to **True**

Notes:

- In Python, the indentation is **required**.

The `if` statement

`if` keyword

a boolean expression (the **condition**)

a colon `:`

```
if is_raining:  
    print("You should wear a raincoat!")
```

an indented **code block**: one or more statements to be executed if the boolean expression evaluates to **True**

Notes:

- In Python, the indentation is **required**.
- Indenting with tabs or spaces is acceptable.

The `if` statement

`if` keyword

a boolean expression (the **condition**)

a colon `:`

```
if is_raining:  
    print("You should wear a raincoat!")
```

an indented **code block**: one or more statements to be executed if the boolean expression evaluates to **True**

Notes:

- In Python, the indentation is **required**.
- Indenting with tabs or spaces is acceptable.
- We'll use the most common convention: indent 4 spaces beyond the line with the `if`

The `if` statement

`if` keyword

a boolean expression (the `condition`)

a colon `:`

```
if is_raining:  
    print("You should wear a raincoat!")
```

an indented `code block`: one or more statements to be executed if the boolean expression evaluates to **True**

Notes:

- In Python, the indentation is **required**.
- Indenting with tabs or spaces is acceptable.
- We'll use the most common convention: indent 4 spaces beyond the line with the `if`
- Thonny follows this convention for you

Demo

Demo

- using the `is_raining` example
- if statement with a condition that evaluates to `True` vs `False`
- statements after the indented code block
- multiple lines in the code block

What if it's not raining?

What if we want to also print something in case it's **not raining**?

```
if is_raining:  
    print("Wear a raincoat!")
```

What if it's not raining?

What if we want to also print something in case it's **not raining**?

```
if is_raining:  
    print("Wear a raincoat!")  
if not is_raining:  
    print("Don't wear a raincoat!")
```

What if it's not raining?

What if we want to also print something in case it's **not raining**?

```
if is_raining:  
    print("Wear a raincoat!")  
if not is_raining:  
    print("Don't wear a raincoat!")
```

How many times did we check the value of `is_raining`?

What if it's not raining?

What if we want to also print something in case it's **not raining**?

```
if is_raining:  
    print("Wear a raincoat!")  
if not is_raining:  
    print("Don't wear a raincoat!")
```

How many times did we check the value of `is_raining`?

Could we do any better?

What if it's not raining?

What if we want to also print something in case it's **not raining**?

```
if is_raining:  
    print("Wear a raincoat!")  
if not is_raining:  
    print("Don't wear a raincoat!")
```

How many times did we check the value of `is_raining`?

Could we do any better?

Yes: it's a common use case to want to choose between two paths of execution (two code blocks).

The `if/else` Statement

```
if isRaining:  
    print("Wear a raincoat!")  
else:  
    print("Don't wear a raincoat!")
```

The `if/else` Statement

`if` keyword a boolean expression (the **condition**)

a colon :

an indented **code block** to be executed if the condition evaluates to **True**

```
if isRaining:  
    print("Wear a raincoat!")  
else:  
    print("Don't wear a raincoat!")
```

The `if/else` Statement

`if` keyword

a boolean expression (the **condition**)

a colon :

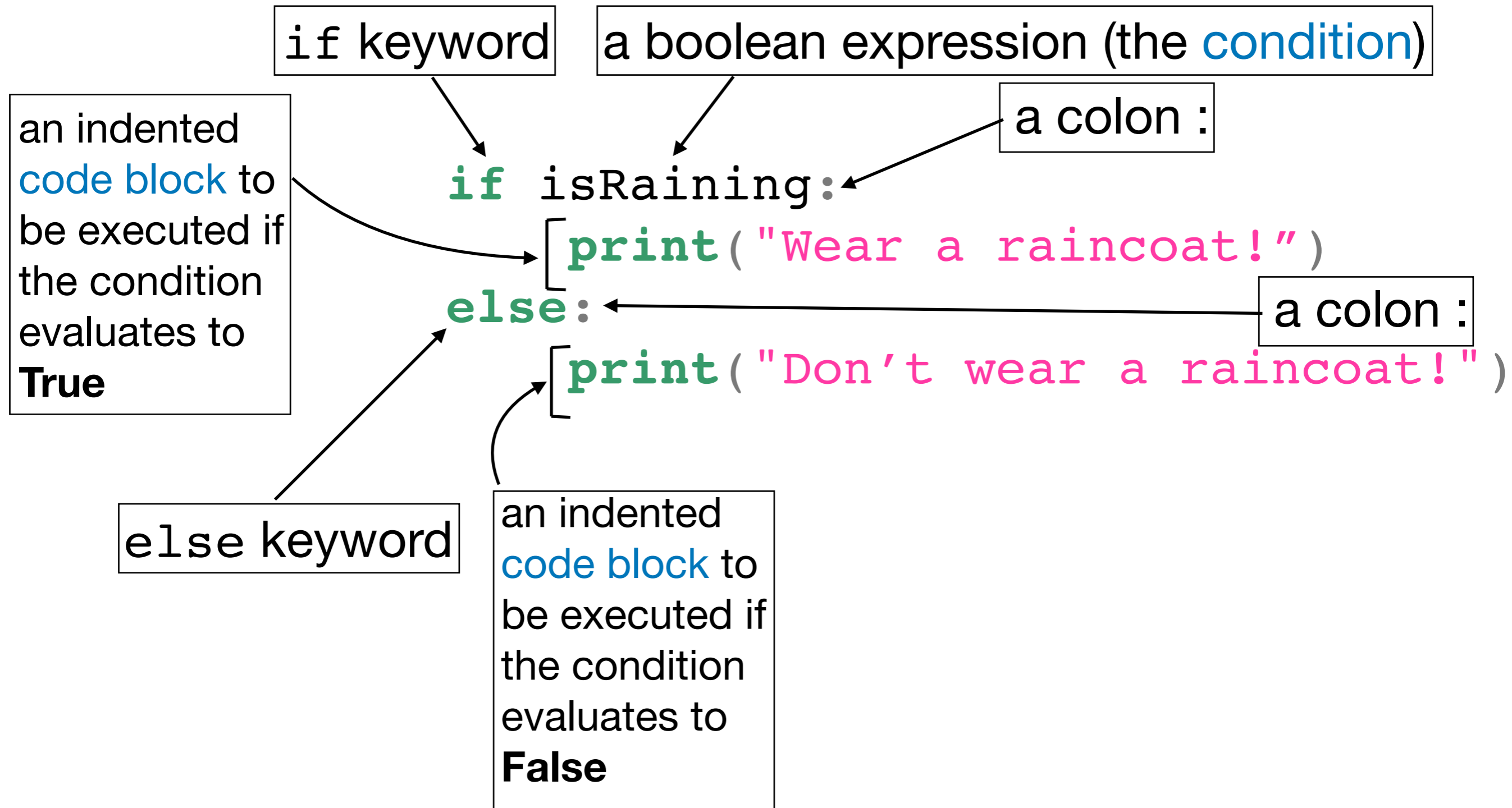
an indented **code block** to be executed if the condition evaluates to **True**

```
if isRaining:  
    print("Wear a raincoat!")  
else:  
    print("Don't wear a raincoat!")
```

a colon :

`else` keyword

The `if/else` Statement



if/else: the basics

What does the following program print?

```
if 2 + 5 == 5:  
    print(2 + 5)  
else:  
    print("not equal")
```



- A. $2 + 5$
- B. 7
- C. $2 + 5 == 5$
- D. not equal

if/else: the basics

What does the following program print?

```
a = 5
if a >= 5 and a <= 5:
    print(a)
else:
    print("nope")
```



- A. 5
- B. $a \geq 5$
- C. $a \leq 5$
- D. nope

if/else: the basics

What does the following program print?

```
a = 5
if a >= 5 and a <= 5:
    print(a)
else:
    print("nope")
```

Is there a better way to write the condition?



- A. 5
- B. $a \geq 5$
- C. $a \leq 5$
- D. nope

Aim for Simplicity

```
a = 5
if a >= 5 and a <= 5:
    print(a)
else:
    print("nope")
```

```
a = 5
if a == 5:
    print(a)
else:
    print("nope")
```

Aim for Simplicity

```
a = 5
if a >= 5 and a <= 5:
    print(a)
else:
    print("nope")
```

```
a = 5
if a == 5:
    print(a)
else:
    print("nope")
```

The program on the right does **exactly** the same thing, but is easier to read, and therefore is preferable.

Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```
# assume x and y are numbers
if x < y:
    print("x is less than y")
```

Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
```

Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

the **inner** if/else statement is the indented code block for the **else clause** of the **outer** if/else statement.

Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

Note: the conditions still have to be boolean expressions (i.e., they evaluate to True or False)

the **inner** if/else statement is the indented code block for the **else clause** of the **outer** if/else statement.

Nested Conditionals

Suppose $x = 4$ and $y = 5$. How many comparison operators ($<$, $>$) are evaluated by the following code?



- A. 0
- B. 1
- C. 2
- D. 3

```
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

Chained Conditionals: Demo

Task: Write a program to ask the user for their 141 section number and print out when their lab section happens.

```
>>> %Run section_times.py
Enter your CSCI 141 section number: 40372
Your lab is on Monday from 10 - 12.
>>>
```

Chained Conditionals: Demo

Chained Conditionals: Demo

- `sections.py`: with chained if/else statements
- `sections_elif.py`: with if/elif/else
- `sections_refactored.py`: refactored to set variables then call print once
- `sections_refactored.py`: with feature to check for conflicts with lab

Chained Conditionals: Syntax

elif keyword

an indented code block to be executed if none of the prior conditions was true and this **elif** condition is True

```
if isRaining and not isWindy:  
    print("Bring an umbrella!")  
elif isRaining and isWindy:  
    print("Wear a raincoat!")  
else:  
    print("No rain gear needed!")
```

an indented code block to be executed if the **none** of the above conditions was true

(an **else clause** is optional)