Course	
Term	
Week	
Date	
Chapter. Topic	3. Decision Structures and Boolean Logic

Boolean Logic

Siva R Jasthi

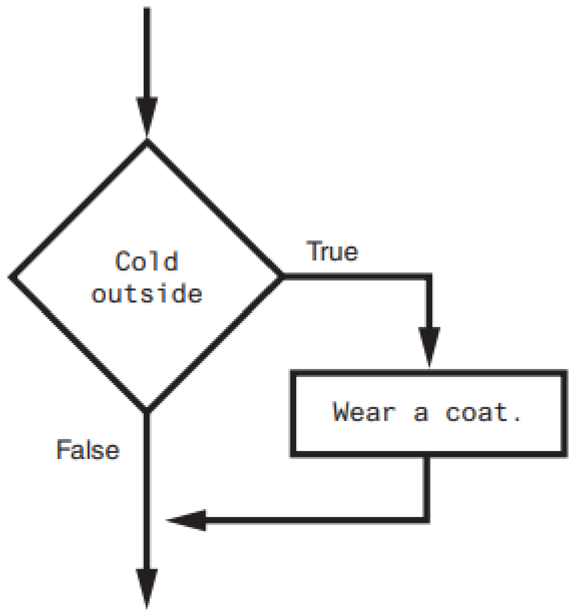
Computer Science and Cybersecurity

Metropolitan State University

Outline

1. Boolean Logic
   - and
   - or
   - not
2. Short-circuit evaluation
   - OR stops at first True
   - AND stops at first False

3. Conditions (aka Branching)
   - if
   - if .. else
   - if – elif
   - if – elif – elif – else
   - Nested if conditions



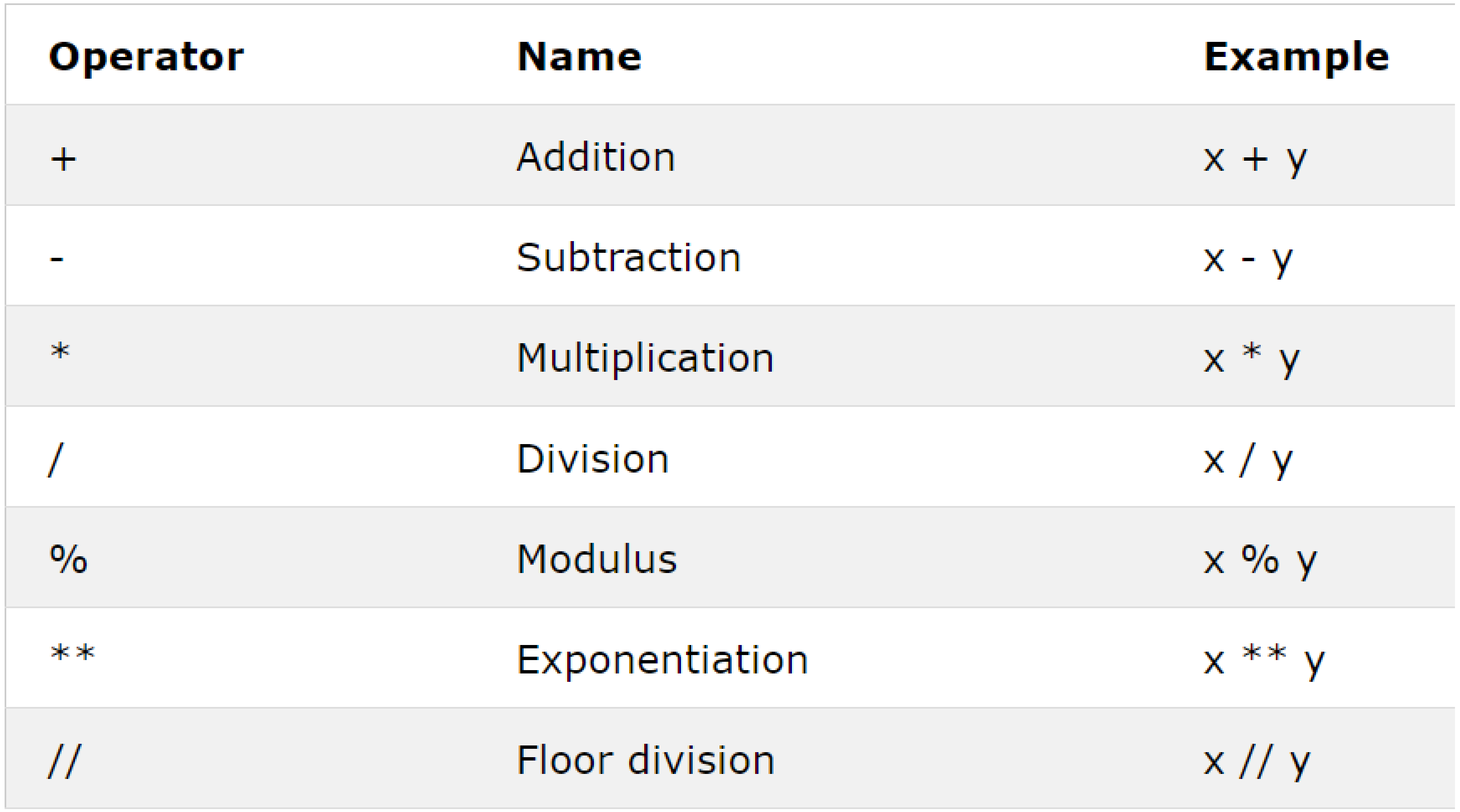
Operators

Python divides the operators in the following groups:

Type	Notes
Arithmetic Operators	+ - * % / // **
Assignment operators	=
Comparison operators	==, !=, >, <, >=, <=
Logical operators	and, or, not
Identity operators	
Membership operators	
Bitwise operators	

https://www.w3schools.com/python/python_operators.asp

Arithmetic Operators



Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

Only assignment operator you need to know is = (equals to sign).

```
>>> age = 17
>>> print(age)
17
>>> age = age + 1
>>> print (age)
18
```

Assignment Operators (Short-hand notation)

Python also supports short-hand notation for the assignments.

```
>>> age = 17
>>> print(age)
17
>>> age = age + 1
>>> print (age)
18
>>> age += 1
>>> print(age)
19
>>>
```

Regular notation

Short-hand notation

I recommend using the regular notation in this course. Even though it involves a couple of extra key strokes, it is easy to read.

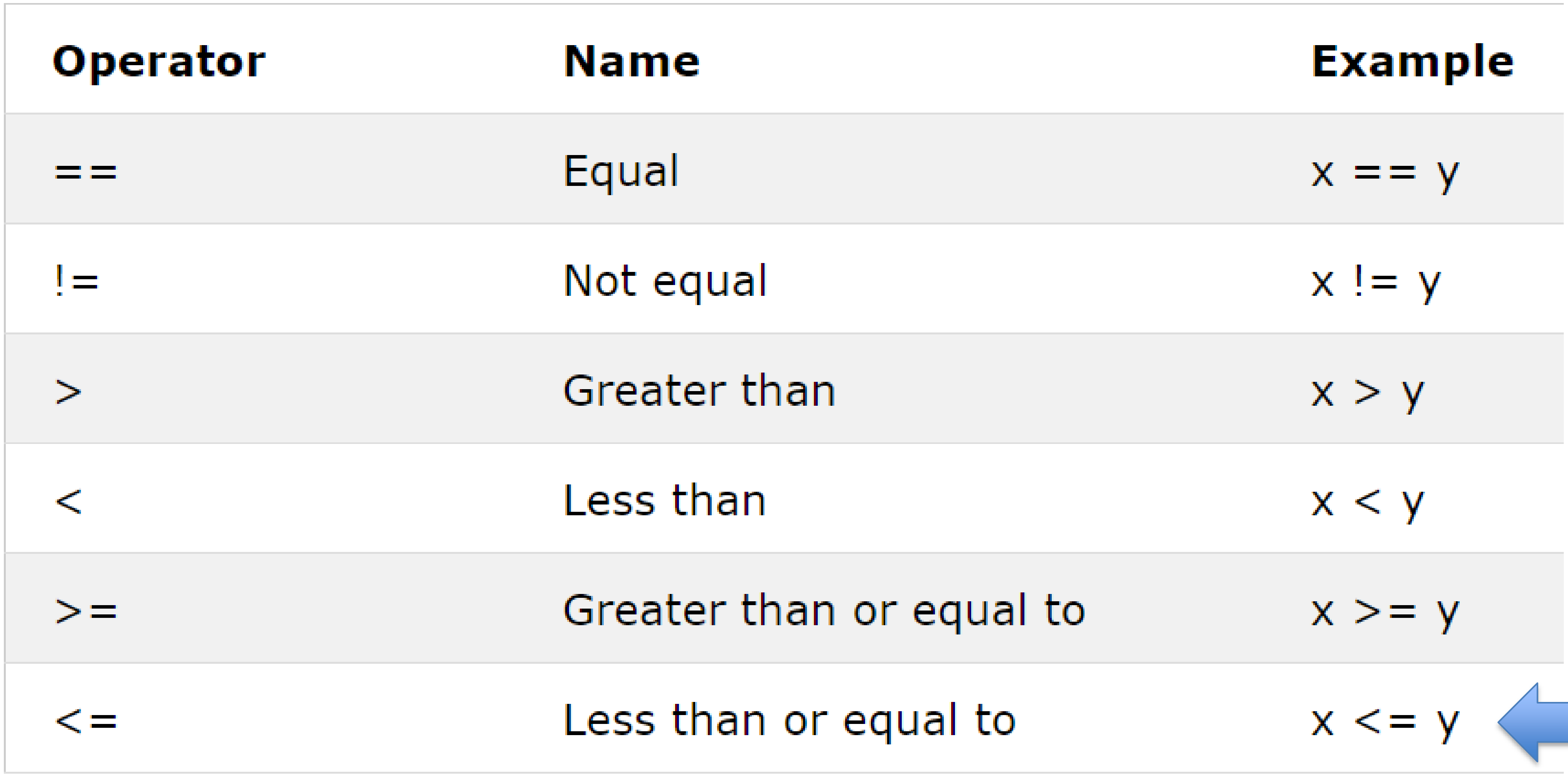
Assignment Operators (Short-hand notation)

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

I recommend using the regular notation in this course. Even though it involves a couple of extra key strokes, it is easy to read.

Comparison Operators

Comparison Operators are used to compare two values.
And the result is always a Boolean value (True or False).



Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Comparison Operators (Examples)

Comparison Operators are used to compare two values.
And the result is always a Boolean value (True or False).

```
>>> a=100
>>> b=10
>>> c=100

>>> a==b
False
>>> a!=b
True
>>> b!=a
True
>>> b==a
False
>>> a>b
True
>>> b>a
False
```

```
>>> a<b
False
>>> b>a
False
>>> a>=b
True
>>> a<=b
False
>>> a>=c
True
>>> a<=c
True
```

```
>>> a > 87
True
>>> a < 999
True
>>> 67 < 90
True
>>> 67 > 90
False
>>> 789 * 764 < 568 * 987
False
>>> a + b < b - c
False
>>> a * b ==  b * c
True
```

Comparison Operators (Comparing Strings)

You can also use comparison operators to compare strings.

Strings are compared based on the **lexicographical (dictionary)** order.

```
>>> "hello" == "hola!"
False
>>> "hello" < "hola"
True
>>> "hello" < "holaaaaaaaa"
True
>>> "a" < "z"
True
>>> "a" < "A"
False
>>> “mumbai" != “madras"
True
```

https://careerkarma.com/blog/python-compare-strings/

Comparison Operators: One caution

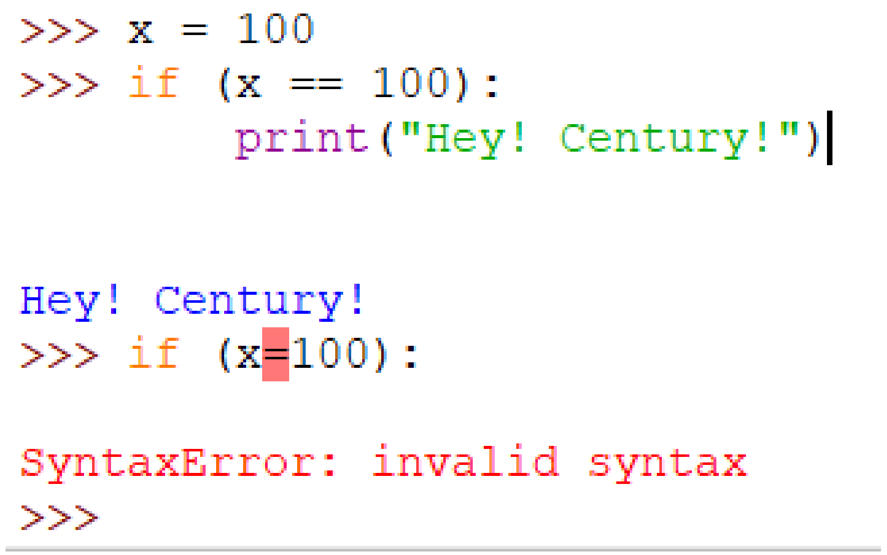
Equals To vs Double Equals To

=

Assigns a value; Assignment Operator;
Example: a = 10

==

Equality (Comparison) Operator; Compares two values
Example: a = = 10



```
>>> x = 100
>>> if (x == 100):
        print("Hey! Century!")


Hey! Century!
>>> if (x=100):

SyntaxError: invalid syntax
>>>
```

Logical Operators

Logical operators (and or not) are used to combine conditional statements:

x = 4

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

AND

When both conditions are true, the combined condition is true.
When one of the conditions is false, then the combined condition is false.

A = 10
B = 5

(A < B) and (B < A) ➔ False
False and True

(A == 10) → True
(B == 5) → True

(A == 10) and (B == 5) → True

A < 15 → True
A < 5 → False
(A < 15) and (A < 5) → False

(A==B) → False
(B >= A) → False
(A == B) and (B >= A) → False

OR

If one of the conditions is TRUE, then the whole condition is TRUE
If both the conditions are FALSE, then the whole condition is False.

A = 10
B = 5

A < B False
B < A True
(A < B) or (B < A) $\rightarrow$ False or True $\rightarrow$ True

(A == 10) $\rightarrow$ True
(B == 5) $\rightarrow$ True

(A == 10) or (B == 5) $\rightarrow$ True or True $\rightarrow$ True

A < 15 $\rightarrow$ True
A < 5 $\rightarrow$ False
(A < 15) or (A < 5) $\rightarrow$ True or False $\rightarrow$ True

(A==B) $\rightarrow$ False
(B >= A) $\rightarrow$ False
(A == B) or (B >= A) $\rightarrow$ False or False $\rightarrow$ False

NOT

Opposite of True is False
not (True) = False

Opposite of False is True
not (False) = True

A = 10
B = 5

(A == 10) → True
not (A == 10)
not (True) → False

B == 5 → True
not (B == 5) → False

B > 6 → False
not (B > 6) → True

A == 20 → False
B == 5 → True

(A == 20) or (B==5) →False or True → True

not ((A == 20) or (B==5)) → False

not (not ((A == 20) or (B==5))) → True

Logical Operators (Example 1)

Logical operators (and or not) are used to combine conditional statements:

```
>>> number_1 = 100
>>> number_2 = 10
>>> (number_1 == 100) and (number_2 == 10)
True
>>> (number_1 == 100) and (number_2 == 100)
False

>>> (number_1 == 100) or (number_2 == 100)
True
>>> (number_1 == 90) or (number_2 == 10)
True


>>> not((number_1 == 100) and (number_2 == 100))
True
>>> not((number_1 == 100) and (number_2 == 10))
False
```

Logical Operators (Example 2)

Logical operators (and or not) are used to combine conditional statements:

```
>>> day = "Saturday‟

>>> day == "Saturday"
True
>>> (day == "Saturday")  or (day == "sat")  or (day == "SAT")  or (day == "SATURDAY")
True
>>> day = "SAT"
>>> (day == "Saturday")  or (day == "sat")  or (day == "SAT")  or (day == "SATURDAY")
True

>>> day= "asjdflas"
>>> (day == "Saturday")  or (day == "sat")  or (day == "SAT")  or (day == "SATURDAY")
False
```

Logical Operators (Coding Convention)

Use parenthesis ( ) to group mini-expressions for readability and clarity.

```
>>> day = "SAT"


>>> (day == "Saturday")  or (day == "sat")
or (day == "SAT")  or (day == "SATURDAY")
True


>>> day == "Saturday"  or day == "sat"  or
day == "SAT"  or day == "SATURDAY"
True
```

Logical Operators and Truth Table

You can combine Booleans using the logical operators

A	B	A AND B	A OR B	NOT A
False	False			
False	True			
True	False			
True	True			

Summary: What did we learn today?

1. True or False
2. Boolean Logic