

# Boolean expressions Conditionals

Learning to test functions

# Relational operators

- ▶ Remember: `=` is the Python assignment operator
  - ▶ It is a command to evaluate the right-hand side and make the variable on the left refer to that result
  - ▶ In math (not Python!), `=` is a claim that two expressions are equal
- ▶ `==` is the Python operator that tests for equality
  - ▶ Other relational operators: `>` `>=` `<` `<=` `!=` (the last one means “not equal”)
  - ▶ They return `bool (Boolean)` values

→ True or False

# Concept Test

What is the output of the following code?

```
a = 3  
b = (a != 3)  
print(b)
```

*Boolean expression*

*b = (3 != 3)*

- ▶ A. True
- ▶ B. False
- ▶ C. 3
- ▶ D. Syntax error

*b = False*

# Functions returning Boolean values

Write a function that returns True if x is an integer otherwise returns False

Refer to code written in lecture



# Logical operators

- ▶ The logical operators take one (`not`) or two (`and`, `or`) bools and return a bool
- ▶ An expression involving `not` produces `True` if the original value is `False`, and `False` if the original value is `True`
- ▶ `And` produces `True` exactly when both of its operands are `True`
- ▶ `or` produces `True` exactly when at least one of its operands is `True`

not , and , or

not True → False

not False → True

In Python

0 → False, every other number is True

" " → Empty String is False  
every other string True

[ ] → Empty List is False

( ) → Empty Tuple False

True and True  $\rightarrow$  True

True and False  $\rightarrow$  False

False and True  $\rightarrow$  False

False and False  $\rightarrow$  False

$x < 10$  and  $\frac{x=5}{\top}$

$\xrightarrow{\text{Boolean expr.}}$        $\xrightarrow{\text{Boolean exp}}$

True Or True  $\rightarrow$  True

False Or True  $\rightarrow$  True

True Or False  $\rightarrow$  True

False Or False  $\rightarrow$  False

## Precedence

( ) Highest

\*\*

Unary negation, e.g. -x

-

\* / %

+ -

> < ==

not

and

or

= Lowest

## Caution Level

set equal to

=

divide

/

remainder

%

power

\*\*

is equal to

==

# Python Operators

(not a) and c

as usual

( )

/

%

\*\*

==

\*

+

>

<

-

( )

It's not worth remembering all these %+/\* things!

I'd go with parentheses over precedence



# Concept Test

What is the value of the expression at the bottom of the code?  
(Remember that not has the highest precedence, then and, then or.)

a = True

b = False

c = True

((not a) and b) or c

False and False

▶ A. True

▶ B. False

# More functions returning Boolean

For each of the following write a function that takes one parameter  $x$ , and returns true if the following condition is True, otherwise returns false

- ▶ A.  $x$  is an integer and its value is negative
- ▶ B.  $x$  is an odd integer (don't make assumptions about the value of  $x$ )

How would you modify the above code so that the function additionally prints a message when  $x$  is odd (instead of returning true)?

# If and If Else

```
if <condition>:  
    <sequence of statements>
```

If the condition evaluates to True, execute sequence of statements, otherwise jump to end of if block

```
if <condition>:  
    <sequence of statements-1>  
else:  
    <sequence-of-statements2>
```

If the condition evaluates to True, execute code inside if block, otherwise execute code in the else block

# Concept Test

What is the  
value of  $x$   
after this code  
executes?

```
x = 5  
if x > 2:  
    x = -3  
    x = 1  
else:  
    x = 3  
    x = 2
```

- ▶ A. -3
- ▶ B. 1
- ▶ C. 2
- ▶ D. 3
- ▶ E. 5

# Fizzbuzz

- Write a program for the game fizzbuzz
- Your program should take an input n
- If n is a multiple of 3, print Fizz
- If n is a multiple of 5, print Buzz
- If n is a multiple of both 3 and 5, print FizzBuzz
- If n if not a multiple of 3 or 5, print n

Please refer to code written in  
lecture