Fairfield UNIVERSITY

CPSC 1101 Introduction to Computing

School of Engineering & Computing
Dept. of Computer Science & Engineering
Dr. Sidike Paheding

Agenda

- Operators and operands
- Operators: **pages 39 - 41** (arithmetic and assignment) and **65 - 71** (relational and logic)

Operators and Operands

- **Operators** are special tokens that represent computations like
  - addition
  - multiplication
  - Division
  - …
- **Operands** are the values that operators work on

The Operator *Assignment*

“=” is used to **assign** (or **re-assign**) a value to a variable

```
var1=(5+9)*(15-7)
var2=var1
```

What is the value of var1?

Operators

higher precedence ↑

parens	`(   )`
(Exponentiation)power	`**`
negate	`-`
times, mod, divide, integer division	`*  %  /  //`
add, subtract	`+  -`
compare	`>  ==  <  !=  <=  >=`
assign	`=`

Examples with two operands

```
1.   print(5 + 4)  #9
2.   25 / 4    #6.25
3.   25 // 4   #6
4.   25 % 4    #1
5.   3 ** 2    #9
```

Examples that show the order of precedence and use of parentheses

```
1. print("3 + 4 * 5: ", 3 + 4 * 5)

2. print("(3 + 4) * 5: ", (3 + 4) * 5)

3. print("2 raised to the 3rd: ", 2 ** 3)

4. print("-2 * 4: ", -2 * 4)
```

The Operator *mod (i.e. %)*

- **The operator mod is represented by "%". Syntax examples:**
  - `7 % 3      #1`
  - `16 % 7     #2`
- **x%y return the <u>remainder</u> when x is divided by y**

For what values of **x** are these **True**?

$$\left\{\begin{array}{l}\texttt{x\%2 == 0}\\ \texttt{x\%2 == 1}\\ \texttt{x\%4 == 0}\\ \texttt{x\%4 == 3}\end{array}\right.$$

Exercise

Write a Python program that

1. calculates the remainder of dividing 29 by 6
2. prints the result of the comparison between the reminder and 0.

Built-in Math Functions

- **abs(x)**: Returns the absolute value of a number.
  - Example: abs(-5) returns 5.
- **max(iterable)**: Returns the largest item from an iterable.
  - Example: max(1, 3, 2) returns 3.
- **min(iterable)**: Returns the smallest item from an iterable.
  - Example: min(1, 3, 2) returns 1.
- **sum(iterable)**: Returns the sum of all items in an iterable.
  - Example: sum([1, 2, 3]) returns 6.

Built-in Math Functions

- **round(x, n)**: Rounds a floating-point number to a given precision n.
  - Example: round(3.14159, 2) returns 3.14.
- **pow(base, exp)**: Returns base raised to the power exp.
  - Example: pow(2, 3) returns 8.
- **divmod(x, y)**: Returns a tuple containing the quotient and remainder when x is divided by y.
  - Example: divmod(9, 2) returns (4, 1).

Functions from the math Module

import math

Function	Description	Example
math.sqrt(x)	Returns the square root of x.	math.sqrt(16) 3→ 4.0
math.ceil(x)	Rounds x up to the nearest integer.	math.ceil(2.3) → 3
math.floor(x)	Rounds x down to the nearest integer.	math.floor(2.9) → 2
math.factorial(x)	Returns the factorial of x.	math.factorial(5) → 120
math.exp(x)	Returns e raised to the power of x.	math.exp(1) → 2.718281828459045
math.log(x, base)	Returns the logarithm of x to the given base.	math.log(100, 10) → 2.0
math.sin(x)	Returns the sine of x (in radians).	math.sin(math.pi / 2) → 1.0
math.cos(x)	Returns the cosine of x (in radians).	math.cos(0) → 1.0
math.tan(x)	Returns the tangent of x (in radians).	math.tan(math.pi / 4) → 1.0

Shortcut

- What does the following instruction do?

```
n1 += 2
```

- Are there “similar” “shortcuts” available in Python?

Shortcut

Operator	Shorthand	Expression	Description
+=	x+=y	x = x + y	Adds 2 numbers and assigns the result to left operand.
-=	x-= y	x = x -y	Subtracts 2 numbers and assigns the result to left operand.
***=**	x*= y	x = x*y	Multiplies 2 numbers and assigns the result to left operand.
/=	x/= y	x = x/y	Divides 2 numbers and assigns the result to left operand.
%=	x%= y	x = x%y	Computes the modulus of 2 numbers and assigns the result to left operand.
****=**	x**=y	x = x**y	Performs exponential (power) calculation on operators and assign value to the equivalent to left operand.

Exercise

```
x=12
y=7
x  +=  y   # print("x+=y:, x=", x)
x  -=  y   # print("x-=y:, x=", x)
x  *=  y   # print("x*=y:, x=", x)
x  /=  y   # print("x/=y:, x=", x)
x  %=  y   # print("x%=y:, x=", x)
x  **=  y # print("x**=y:, x=", x)
```

Logical Operators

and (logical and)	**True if both operands are True**	**Example: a and b**
or (logical or)	**True if either of the operands is True**	**Example: a or b**
not (logical not)	**True if the operands is False**	**Example: not a**

Truth Table

a	**b**	**a and b**	**a or b**	**not a**
0 # false	0 # false	0 # false	0 # false	1 # true
0 # false	1 # true	0 # false	1 # true	1 # true
1 # true	0 # false	0 # false	1 # true	0 # false
1 # true	1 # true	1 # true	1 # true	0 # false

Relational operators

Operator	Name
`==`	Equal to
`!=`	Not equal to
`>`	Greater than
`<`	Less than
`>=`	Greater than or equal to
`<=`	Less than or equal to

Boolean expressions

```
age == 5                  # variable equal to numeric literal
first_name == "John"      # variable equal to string literal

quantity != 0             # variable not equal to numeric literal

distance > 5.6            # variable greater than numeric literal
fuel_req < fuel_cap       # variable less than variable

distance >= limit         # variable greater than or equal to variable
stock <= reorder_point    # variable less than or equal to variable

rate / 100 >= 0.1         # expression greater than or equal to literal
```

How to assign a Boolean value to a variable

```
active = True             # variable is set to Boolean True value
active = False            # variable is set to Boolean False value
```

Logical operators

Operator	Name
and	AND
or	OR
not	NOT

highest precedence

Order of precedence

NOT operator

AND operator

OR operator

Boolean expressions that use logical operators

```
# The AND operator
age >= 65 and city == "Chicago"

# The OR operator
city == "Greenville" or age >= 65

# The NOT operator
not age >= 65

# Two AND operators
age >= 65 and city == "Greenville" and state == "SC"

# Two OR operators
age >= 65 or age <= 18 or status == "retired"

# AND and OR operators with parens to clarify sequence of
operations
(age >= 65 and status == "retired") or age < 18

# AND and OR operators with parens to change sequence of operations
age >= 65 and (status == "retired" or state == "SC")
```

Some string comparisons

Condition	Boolean result
`"apple" < "Apple"`	`False`
`"App" < "Apple"`	`True`
`"1" < "5"`	`True`
`"10" < "5"`	`True`

Python compares strings lexicographically, character by character, based on their Unicode values.

The sort sequence of digits and letters

Digits from 0-9

Uppercase letters from A-Z

Lowercase letters from a-z

Logical Expressions

```
print('ab' and 'cd' or 'ef')
```

- In logical expressions, non-empty strings are considered True.
- The *and* operator returns the second operand if both are truthy.
  - 'ab' and 'cd' evaluates to 'cd’.
- The *or* operator returns the first truthy operand.
  - 'cd' or 'ef' evaluates to 'cd’.
  - Therefore, the output is 'cd'.

Precedence	Operators	Description	Associativity
1	**()**	Parentheses	Left to right
2	**x[index]**, **x[index:index]**	Subscription, slicing	Left to right
3	await x	Await expression	N/A
4	**	Exponentiation	Right to left
5	**+x**, **-x**, **~x**	Positive, negative, bitwise NOT	Right to left
6	*****, **@**, **/**, **//**, **%**	Multiplication, matrix, division, floor division, remainder	Left to right
7	**+**, **–**	Addition and subtraction	Left to right
8	**<<**, **>>**	Shifts	Left to right
9	**&**	Bitwise AND	Left to right
10	**^**	Bitwise XOR	Left to right
11	**|**	Bitwise OR	Left to right
12	**in**, **not in**, **is**, **is not**, **<**, **<=**, **>**, **>=**, **!=**, **==**	Comparisons, membership tests, identity tests	Left to Right
13	**not x**	Boolean NOT	Right to left
14	**and**	Boolean AND	Left to right
15	**or**	Boolean OR	Left to right
16	**if-else**	Conditional expression	Right to left

Class Assignment 1

Write a Python program that uses the functions from the module to perform the following tasks:

1) Calculate the square root of 64 and print the result.
2) Round the number 4.7 up to the nearest integer and print the result.
3) Round the number 7.3 down to the nearest integer and print the result.
4) Calculate the factorial of 6 and print the result.
5) Find the logarithm of 1000 with base 10 and print the result.
6) Compute the tangent of π/4 and print the result.

Class Assignment 2: 4 Fun

- Complete fourfours.py example

Fairfield UNIVERSITY

Thank you for participating in CPSC 1101 - Intro to Computing.

Are there any questions?

Sidike Paheding, Ph.D.

spaheding@fairfield.edu