UMBC

AN HONORS UNIVERSITY IN MARYLAND

CMSC201 Computer Science I for Majors

Lecture 04 – Expressions

Prof. Katherine Gibson

Last Class We Covered

- Variables
  - Rules for naming
  - Different types
  - How to use them
- Printing output to the screen
- Getting input from the user
  - Mad Libs

Any Questions from Last Time?

Today’s Objectives

- To learn more about expressions
- To learn Python’s operators
  - Including mod and integer division
- To understand the order of operations
- To learn more about types
  - How to cast to a type
- To understand the use of constants

Expressions

- Expressions are code that produces or calculates new data and data values
- Allow us to program interesting things
- Always on the **right hand side** of the assignment operator

Pop Quiz!

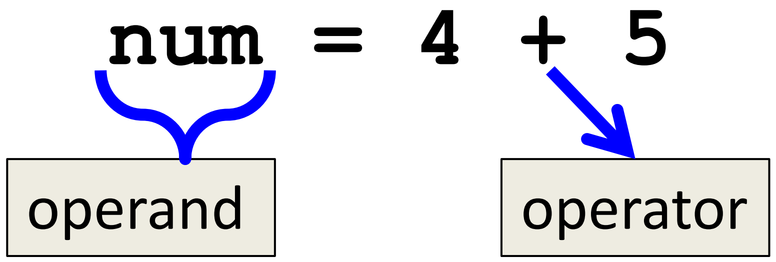
- Which of the following examples are correct?

1. `500 = numStudents`
2. `numStudents = 500`
3. `numCookies * cookiePrice = total`
4. `mpg = miles_driven / gallons_used`
5. `"Hello World!" = message`
6. `_CMSC201_doge_ = "Very learning"`
7. `60 * hours = days * 24 * 60`

Python’s Operators

Python Basic Operators

- Operators are the constructs which can manipulate the value of operands
- Consider the expression:



- Here, **`num`** is the operand and **+** is the operator

Types of Operators in Python

- Arithmetic Operators

focus of today's lecture

- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Operators in Python

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Integer division
%	Modulo (remainder)
**	Exponentiation

Operators – Addition & Subtraction

- "Lowest" priority in the order of operations
  - Can only change this with parentheses
- Function as they normally do

- Examples:
  1. `cash = cash - bills`
  2. `(5 + 7) / 2`
  3. `( ((2 + 4) * 5) / (9 - 6) )`

Operators – Multiplication & Division

- Higher priority in the order of operations than addition and subtraction
- Function as they normally do

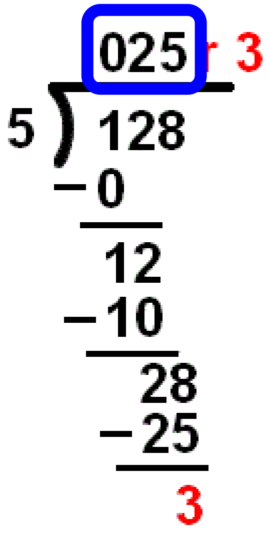
- Examples:
  1. `tax = subtotal * 0.06`
  2. `area = PI * (radius * radius)`
  3. `tsp = tbsp * 3`

Operators – Integer Division

- Reminder: integers (or ints) are **whole numbers**
  - What do you think integer division is?

- Remember division in grade school?

- Integer division is division without decimals, and in which we discard the remainder from our answer



Examples: Integer Division

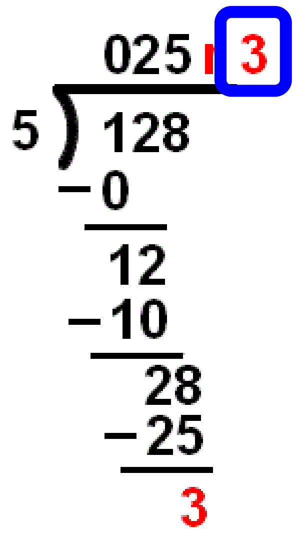
- Integer division uses double slashes (`//`)

- Examples:
  1. `7 /  5        = 1.4`
  2. `7 // 5        = 1`
  3. `2 /  8        = 0.25`
  4. `2 // 8        = 0`
  5. `4 // 17 // 5 =  0`

$\xrightarrow{\hspace{6em}}$ evaluate from left to right

Operators – Modulo

- Also called “modulo,” “modulus,” or “mod”
- Example: `17 % 5 = 2`
  - What do you think mod does?
- Remember division in grade school?
- Mod gives you the remainder from integer division



Examples: Mod

- Mod uses the percent sign (`%`)

- Examples:
  1. `7  % 5     = 2`
  2. `5  % 9     = 5`
  3. `17 % 6     = 5`
  4. `22 % 4     = 2`
  5. `48692451673 % 2 = 1`

Operators – Exponentiation

- “Exponentiation” is just another word for raising one number to the power of another

- Examples:
  1. `binary8 = 2 ** 8`
  2. `squarea = squareLen ** 2`
  3. `cubeVolume = squareLen ** 3`

Order of Operations

- Expressions are evaluated in what direction?

Operator(s)	Priority
**	highest
/ * // %	
+ -	lowest

- What can change this ordering?
  - Parentheses

Types in Python

Variable Types

- There are many different kinds of variables!
  - Numbers
    - Whole numbers (Integers)
    - Decimals (Floats)
  - Booleans (`True` and `False`)
  - Strings (collections of characters)

Finding a Variable’s Type

- To find what type a variable is, use **`type()`**

- Example:

```
>>> a = 3.0
>>> type(a)
<class 'float'>
```

```
>>> b = "moo"
>>> type(b)
<class 'str'>
```

Division: Floats and Integers

- Floats (decimals) and integers (whole numbers) behave very differently in Python
  - And in many other programming languages
- Biggest difference is with how division works
  - In Python 2, all integers use integer division
  - In Python 3, we have to explicitly call integer division
    - Otherwise, we perform decimal division
  - Floats automatically perform decimal division

Division Examples

- What do the following expressions evaluate to?
  1. `4 /  3   =` 1.3333333333333333
  2. `4 // 3   =` 1
  3. `4 // 3.0 =` 1.0
  4. `8 /  3   =` $2.666666666666\boxed{6667}$
  5. `8 /  2   =` 4
  6. `5 /  7   =` $0.714285714285\boxed{7143}$
  7. `5 // 7   =` 0

Floating Point Errors

- In base 10, some numbers are approximated:
  - 0.666666666666666666666666667…
  - 3.14159265358979323846264338328…
- The same is true for base 2
  - 0.000110011001100110011001100… (0.1 in base 10)
- This leads to rounding errors with floats
  - **Don't compare floats after you've done division!**

Casting to a Type

- We can change a variable from one type to another using casting

- Example:

```
>>> e = 2.718
>>> int(e)
2
>>> str(e)
'2.718'
```

type you want to cast to, then the variable to cast *"change e to an integer"*

Constants

What are Constants?

- Constants are values that are **not** generated by the user or by the code
  - But are used a great deal in the program
- Constants should be ALL CAPS with a "**_**" (underscore) to separate the words
  - Coding standards

Using Constants

- Calculating the total for a shopping order

```
MD_TAX          = 0.06
subtotal = input("Enter subtotal:")
tax = subtotal * MD_TAX
total = tax + subtotal
print("Your total is:", total)
```

easy to change if the tax rate changes

we know exactly what this number is for

“Magic” Numbers

- “Magic” numbers are numbers used directly in the code – should be replaced with constants

- Examples:
  - Mathematical numbers (pi, e, etc.)
  - Program properties (window size, min and max)
  - Important values (tax rate, maximum number of students, credits required to graduate, etc.

“Magic” Numbers Example

- You’re looking at the code for a virtual casino
  - You see the number 21
  - What does it mean?

```
if (value < 21)
```

- Blackjack? Drinking age? VIP room numbers?

```
if (customerAge < DRINKING_AGE)
```

- Also helpful if the drinking age changes – why?
  - Don’t have to figure out which “21”s to change

Are Constants Really Constant?

- In some languages (like C, C++, and Java), you can create variables that CANNOT be changed
- This is <u>not possible</u> with Python variables
  - Part of why coding standards are so important
  - If you see code that changes the value of a variable called **`MAX_ENROLL`**, you know that's a constant, and shouldn't be changed

Quick Note: Version of Python

- Before you run any Python code, you need to tell GL you want to use Python 3 instead:

```
/usr/bin/scl enable python33 bash
```

- You can double-check which version with the command `python -v`
  - It will print out a bunch of text, but near the bottom you should see "`Python 3.3.2`"

Announcements

- Your Lab 2 is an online lab this week!
  – Due by this Friday (Sept 11th) at 8:59:59 PM

- Homework 2 is out
  – Due by Tuesday (Sept 15th) at 8:59:59 PM

- Both of these assignments are on Blackboard
  – Weekly Agendas are also on Blackboard