CSCI 141

Lecture 4:
More Variables
Operators and Operands
Code Execution: Statements and Expressions

Announcements

- See the Canvas announcement about labs and A1
- CS Support wiki has useful info: https://support.cs.wwu.edu/index.php/Main_Page
- Labs are open to CS students 24/7 unless there’s a class.
- CF building is locked after 11pm, but you can stay later if you’re already inside.

Today's Quiz

- 3 minutes

Today’s Quiz

- 3 minutes
- Working with a neighbor: do your answers agree? (2 minutes)

Goals

- Understand how to use variables in assignment statements and elsewhere in place of values
- Know the rules for naming variables, and the conventions for deciding on good variable names.
- Know the definition and usage of operators and operands
    - Know how to use the following operators:
      `=, +, -, *, **, /, //, %`
- Understand the distinction between a statement and an expression.
- Understand function calls as expressions that evaluate to their return values.

Last time…

- A variable is a name in a program that refers to a piece of data (or a value).
- How do you use them?
  1. Decide what value you want to store in the variable
  2. Decide on a sensible name
  3. In your program, use the **assignment operator** to store that value in the variable:

```
my_age = 32
```

The assignment operator.

Why are variables useful?

- Remember those numbers from Monday?

5, 8, 12, 44, 89, 65, 43, -67, 43.4, 32

Using Variables

- Assigning a value is **not** stating an equality, like in math: it's storing a value.

```
my_age = 32
my_age = 33
```

A variable's value can be **updated** (overwritten) by a new value using the assignment operator.

✗ "my_age equals 32"

✓ "my_age becomes 32"

✓ "my_age gets 32"

✓ "the variable my_age takes on the value 32"

Variables and Assignment

What is the value of the variables a and b at the end of this program?

```
a = 5
b = 5
a = 6
b = 7
```

```
A. a: 5, b: 6

B. a: 6, b: 5

C. a: 6, b: 7

D. a: 7, b: 7
```

What can you do with variables?

- Use them anywhere you’d use a value!

```
print(5)
```

```
a = 5
print(a)
```

- These two programs both print 5.

Using Variables

Which of the following programs does not print the same thing as the others?

A:
```
a = 14
b = 3
print(a, b)
```

B:
```
a = 3
b = 14
print(14, 3)
```

C:
```
a = 14
b = a
print(a, b)
```

D:
```
a = 3
b = 14
print(14, a)
```

Using Variables

Which of the following programs does not print the same thing as the others?

A:
```
a = 14
b = 3
print(a, b)
```
14, 3

B:
```
a = 3
b = 14
print(14, 3)
```
14, 3

C:
```
a = 14
b = a
print(a, b)
```
14, 14

D:
```
a = 3
b = 14
print(14, a)
```
14, 3

Variable Names

- How do you use variables?
    1. Decide what value you want to store in the variable
    2. **Decide on a sensible name**
    3. In your program, use the assignment operator to store that value in the variable
- Great power, great responsibility: variables names can be almost anything!

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- **Valid** variable names:
  - start with a letter or an underscore (_)
  - can contain any letters and digits
  - are case-sensitive (name is not the same as Name)
  - are not the same as any Python language **keywords** (words that already mean something else):

```
False, None, True, and, as, assert, async, await, break, class,
continue, def, del, elif, else, except, finally, for, from,
global, if, import, in, is, lambda, nonlocal, not, or, pass,
raise, return, try, while, with, yield
```

`True` ✗ `2plus2` ✗ ✓ `a_number` ✓ `firstOfThreeValues`

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- A **good** variable name:
  - is descriptive - tell a reader what data they refer to
  - is not too long

    these depend on context!
  - follows a standard naming convention, e.g.:
    - starts with lower case letter
    - words are separated by underscores

✓`current_time` ✗`a4` ✓`hair_color`

✗`midterm_exam_grade_as_a_percent`

Statements and Expressions

- A statement is a line (or multiple lines) of code that Python can execute.

  `my_name = “Scott”` is an assignment statement

  A statement in Python does not evaluate to a value!

- An expression is a combination of values, variables, operators, and function calls that can be evaluated to determine its value.

```
type(32)
2+2
int(a)
int(b) * 4
```

are all expressions

The notation => is often used to mean “evaluates to”:

```
2 + 2 => 4
```

“two plus two evaluates to four”

NB: => is **not** a Python operator

Operators

- Operators are special symbols that represent computations we can perform.

- Operands are the values that an operator performs its computations on.

- We've seen one already: the assignment operator.

Its first (left) operand

Its second (right) operand

```
my_age = 32
```

The assignment operator.

Operators

Some Python operators:

```
=
+
-
*
/
**
//
%
```

Some of these probably look familiar…

Operators

Some Python operators:

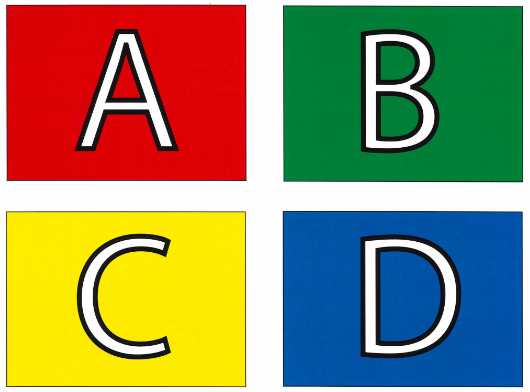
`=`	Assignment operator: stores a value in a variable
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`**`	
`//`	
`%`	

These ones do exactly what you think.

Operators

Some Python operators:

`=`	Assignment operator: stores a value in a variable
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`**`	
`//`	
`%`	



This one too, with one quirk:
In Python, division **always** returns a `float`.

```
3.0 / 2 => 1.5
7 / 2   => 3.5
4 / 2   => ??
```

ABCD:

A. 2
B. 4
C. 2.0
D. 4.0

Operators

Some Python operators:

`=`	Assignment operator: stores a value in a variable
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`**`	
`//`	
`%`	

A B

C D

This one too, with one quirk:
In Python, division **always** returns a `float`.

```
3.0 / 2 => 1.5
7 / 2   => 3.5
4 / 2   => 2.0
```

ABCD:
A. 2
B. 4
C. 2.0
D. 4.0

Operators

Some Python operators:

`=`	Assignment operator: stores a value in a variable
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`**`	Exponentiation
`//`	
`%`	

The exponentiation operator raises the left operand to the power of the right operand.

Math: $2^4 = 2 * 2 * 2 * 2 = 16$

Python: `2**4  =>  16`

Base Exponent

Operators

Some Python operators:

`=`	Assignment operator: stores a value in a variable
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`**`	Exponentiation
`//`	Integer division
`%`	Modulus (remainder)

Integer division does division and evaluates to the integer **quotient**

Math: 7 / 2 is 3 with remainder 1

Python: `7 // 2 => 3`

Operators

Some Python operators:

`=`	Assignment operator: stores a value in a variable
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`**`	Exponentiation
`//`	Integer division
`%`	Modulus (remainder)

The modulus operator does division and evaluates to the integer **remainder**

Math: 7 / 2 is 3 with remainder 1

Python: `7 % 2 => 1`

Examples

```
64 % 2

37 % 2

18 // 4

18 / 4
```

Examples

```
64 % 2 => 0

37 % 2 => 1

18 // 4 => 4

18 / 4 => 4.5
```

Function Calls, Revisited

- A function can take inputs called arguments
- A function can give back an output, called its return value.
- A function call is an expression that evaluates to the its return value.
  - `int(4.6)` evaluates to `4`
  - `print` does not return a value, so `print(4.6)` evaluates to `None`, a special keyword meaning no value

Demo

Demo

- Arithmetic operators and expressions
- printing from a program vs evaluating expressions in the shell
- function call with no return value
- expression on its own line

Putting it all together

- Consider this program:

```
a = 4
b = float(2 + a)
```

- What happens when we execute it?

Putting it all together

- Consider this program:

```
a = 4
b = float(2 + a)
```

- What happens when we execute it?
  - the value 4 gets stored in a

Putting it all together

- Consider this program:

```
a = 4
b = float(2 + a)
```

- What happens when we execute it?
  - the value 4 gets stored in a
  - the expression 2+a is evaluated, resulting in the value 6

Putting it all together

- Consider this program:

```
a = 4
b = float(6)
```

- What happens when we execute it?
  - the value 4 gets stored in a
  - the expression 2+a is evaluated, resulting in the value 6

Putting it all together

- Consider this program:

```
a = 4
b = float(6)
```

- What happens when we execute it?
  - the value 4 gets stored in a
  - the expression 2+a is evaluated, resulting in the value 6
  - 6 is passed into the float function

Putting it all together

- Consider this program:

```
a = 4
b = 6.0
```

- What happens when we execute it?
  - the value 4 gets stored in a
  - the expression 2+a is evaluated, resulting in the value 6
  - 6 is passed into the float function
  - the float function converts 6 to a float and returns 6.0

Putting it all together

- Consider this program:

```
 a = 4
 b = 6.0
```

- What happens when we execute it?
    - the value 4 gets stored in a
    - the expression 2+a is evaluated, resulting in the value 6
    - 6 is passed into the float function
    - the float function converts 6 to a float and returns 6.0
    - the value 6.0 gets stored in variable b