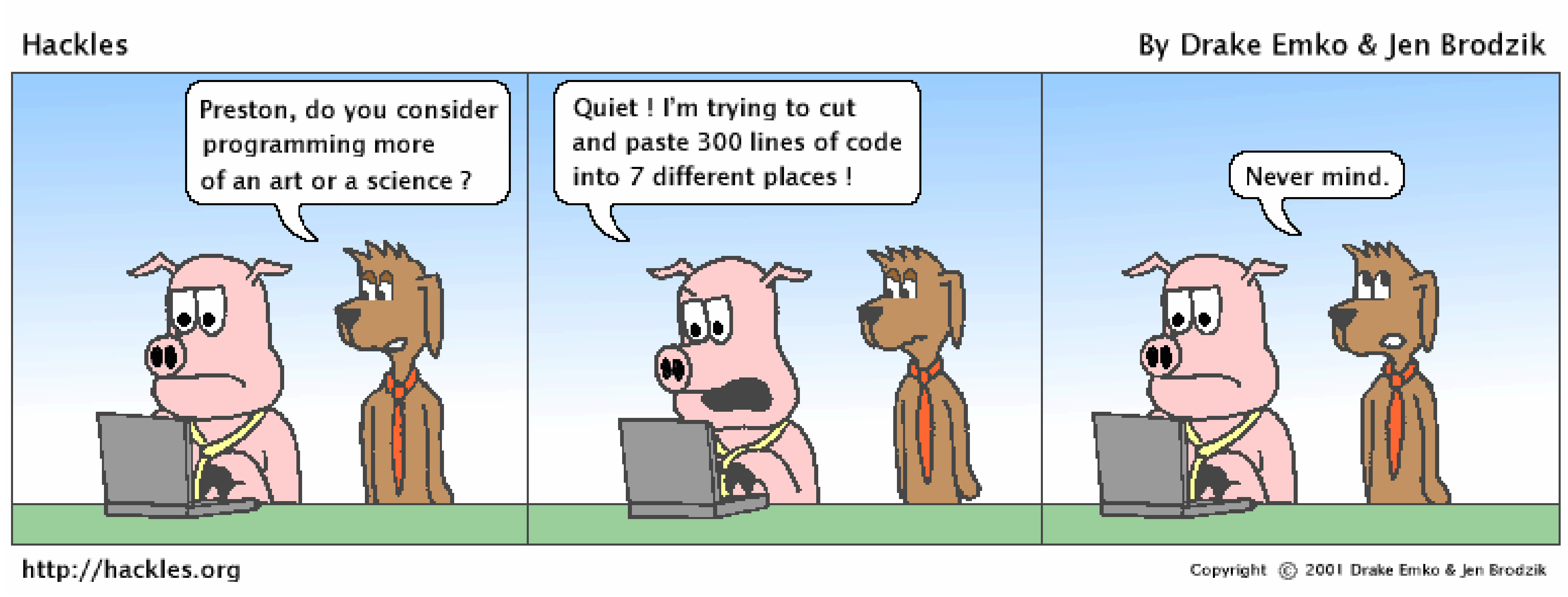
CSc 110, Autumn 2016

Lecture 3: Expressions and Variables


Hackles
By Drake Emko & Jen Brodzik
Preston, do you consider programming more of an art or a science ?
Quiet ! I'm trying to cut and paste 300 lines of code into 7 different places !
Never mind.
http://hackles.org
Copyright © 2001 Drake Emko & Jen Brodzik

Data and expressions

Data types

- Internally, computers store everything as 1s and 0s

```
104   → 01101000
'hi'  → 0110100001101001
'h'   → 01101000
```

- How are `h` and `104` differentiated?

- **type**: A category or set of data values.
  - Constrains the operations that can be performed on data
  - Many languages ask the programmer to specify types
  - Examples: integer, real number, string

Python's number types

Name	Description		Examples
`int`	integers	(up to $2^{31} - 1$)	`42, -3, 0, 926394`
`float`	real numbers		`3.1, -0.25`
`complex`			

Expressions

- **expression**: A value or operation that computes a value.
  - Examples:

    ```
    1 + 4 * 5
    (7 + 2) * 6 / 3
    42
    ```

  - The simplest expression is a *literal value*.
  - A complex expression can use operators and parentheses.

Arithmetic operators

- **operator**: Combines multiple values or expressions.

`+`	addition
`-`	subtraction (or negation)
`*`	multiplication
`/`	division
`//`	integer division (a.k.a. leave off any remainder)
`%`	modulus (a.k.a. remainder)
`**`	exponent

- As a program runs, its expressions are *evaluated*.
  - `1 + 1` evaluates to `2`

Integer division with `//`

- When we divide integers with //, the quotient is also an integer.
  - `14 // 4` is `3`, not `3.5`

```
       3                4                 52
   _____            _____           _______
4 ) 14         10 ) 45         27 ) 1425
    12                40               135
    --                --               ---
     2                 5                75
                                        54
                                        --
                                        21
```

- More examples:
  - `32 // 5` is `6`
  - `84 // 10` is `8`
  - `156 // 100` is `1`

  - Dividing by 0 causes an error when your program runs.

Integer remainder with `%`

- The `%` operator computes the remainder from integer division.
  - `14 % 4` is `2`
  - `218 % 5` is `3`

```
        3                  43
4 ) 14               5 ) 218
    12                   20
     2                    18
                          15
                           3
```

What is the result?

```
45 % 6
2 % 2
8 % 20
11 % 0
```

- Applications of `%` operator:
  - Obtain last digit of a number: `230857 % 10` is `7`
  - Obtain last 4 digits: `658236489 % 10000` is `6489`
  - See whether a number is odd: `7 % 2` is `1`, `42 % 2` is `0`

Precedence

- **precedence**: Order in which operators are evaluated.
  - Generally operators evaluate left-to-right.
    `1 - 2 - 3` is `(1 - 2) - 3` which is `-4`

  - But `*` `/` `//` `%` have a higher level of precedence than `+` `-`

```
1 + 3 * 4                   is 13

6 + 8 // 2 * 3
6 +    4   * 3
6 +      12                 is 18
```

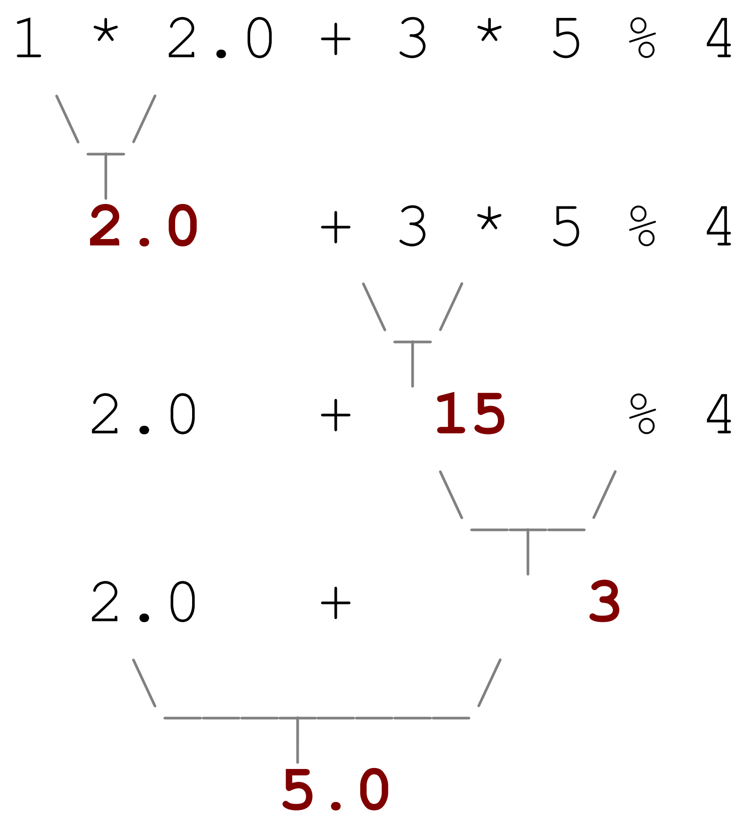
  - Parentheses can force a certain order of evaluation:

```
(1 + 3) * 4                 is 16
```

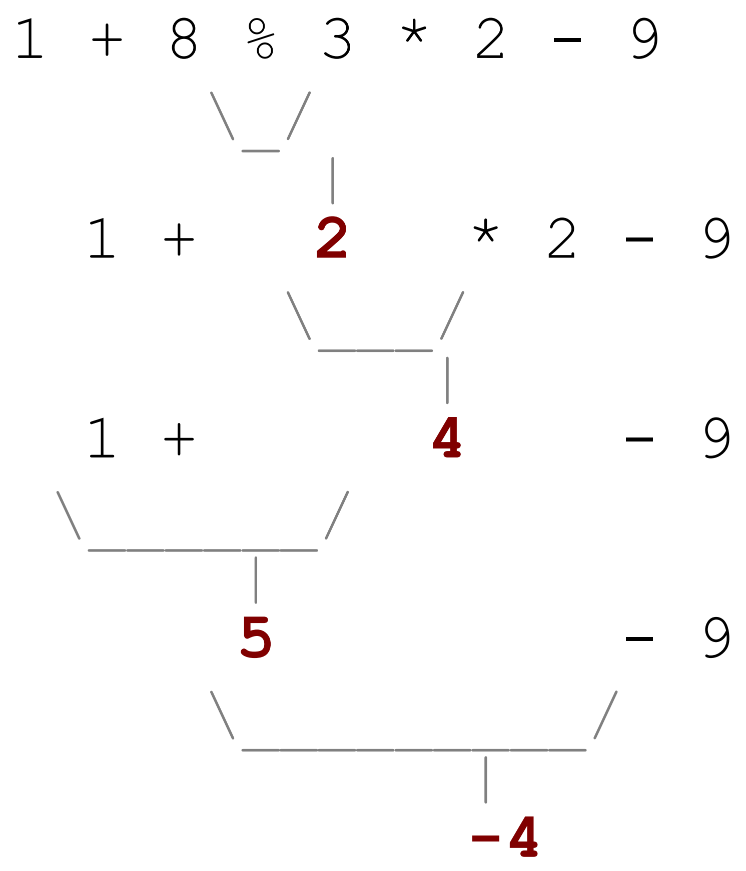
  - Spacing does not affect order of evaluation

```
1+3 * 4-2                   is 11
```

Precedence examples



```
1 * 2.0 + 3 * 5 % 4
  2.0   + 3 * 5 % 4
  2.0   +  15   % 4
  2.0   +       3
       5.0
```



```
1 + 8 % 3 * 2 - 9
  1 +   2   * 2 - 9
  1 +       4   - 9
      5         - 9
               -4
```

Precedence questions

- What values result from the following expressions?
  - `9 // 5`
  - `695 % 20`
  - `7 + 6 * 5`
  - `7 * 6 + 5`
  - `248 % 100 / 5`
  - `6 * 3 - 9 // 4`
  - `(5 - 7) * 2 ** 2`
  - `6 + (18 % (17 - 12))`

Receipt example

What's bad about the following code?

```
# Calculate total owed, assuming 8% tax / 15% tip
print("Subtotal:")
print(38 + 40 + 30)

print("Tax:")
print((38 + 40 + 30) * .08)

print("Tip:")
print((38 + 40 + 30) * .15)

print("Total:")
print(38 + 40 + 30 + (38 + 40 + 30) * .15 + (38 + 40 + 30) * .08)
```

- The subtotal expression `(38 + 40 + 30)` is repeated
- So many `print` statements

Variables

- **variable**: A piece of the computer's memory that is given a name and type, and can store a value.
  - Like preset stations on a car stereo, or cell phone speed dial:

  - Steps for using a variable:
    - *Declare/initialize* it - state its name and type and store a value into it
    - *Use* it - print it or use it as part of an expression

Declaration and assignment

- **variable declaration and assignment**:

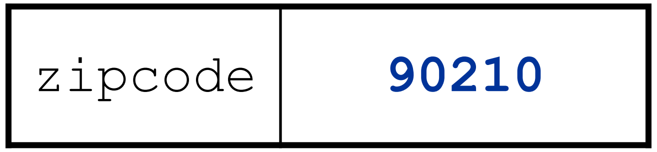
  Sets aside memory for storing a value and stores a value into a variable.
  - Variables must be declared before they can be used.
  - The value can be an expression; the variable stores its result.

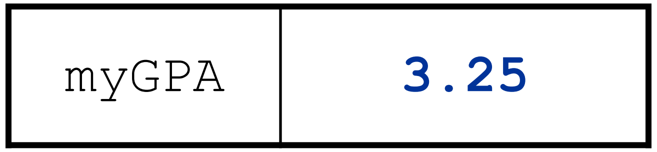
- Syntax:

  name = expression

  - **`zipcode = 90210`**

  - **`myGPA = 1.0 + 2.25`**



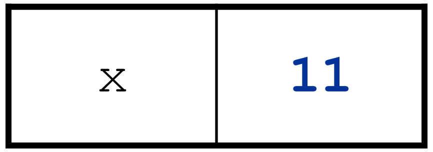


Using variables

- Once given a value, a variable can be used in expressions:

```
x = 3               # x is 3
y = 5 * x - 1       # now y is 14
```

- You can assign a value more than once:



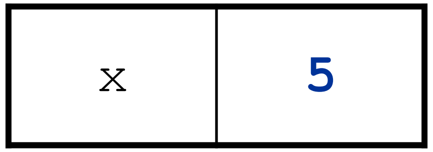
```
x = 3               # 3 here

x = 4 + 7           # now x is 11
```

Assignment and algebra

- Assignment uses = , but it is not an algebraic equation.
    - = means, *"store the value at right in variable at left"*
        - The right side expression is evaluated first,
          and then its result is stored in the variable at left.
- What happens here?

```
x = 3
x = x + 2    # ???
```



Receipt question

Improve the receipt program using variables.

```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    print("Subtotal:")
    print(38 + 40 + 30)

    print("Tax:")
    print((38 + 40 + 30) * .08)

    print("Tip:")
    print((38 + 40 + 30) * .15)

    print("Total:")
    print(38 + 40 + 30 + (38 + 40 + 30) * .15 + (38 + 40 + 30) * .08)
```

Printing a variable's value

- Use `+ str(`**`value`**`)` to print a string and a variable's value on one line.

  -
    ```
    grade = (95.1 + 71.9 + 82.6) / 3.0
    print("Your grade was " + str(grade))

    students = 11 + 17 + 4 + 19 + 14
    print("There are " + str(students) +
          " students in the course.")
    ```
  - Output:

    ```
    Your grade was 83.2
    There are 65 students in the course.
    ```

Receipt answer

```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    subtotal = 38 + 40 + 30         # int
    tax = subtotal * .08            # float
    tip = subtotal * .15            # float
    total = subtotal + tax + tip    # float

    print("Subtotal: " + str(subtotal))
    print("Tax: " + str(tax))
    print("Tip: " + str(tip))
    print("Total: " + str(total))
```