ECOR 1041
Computation and Programming

Review: Representation of Numbers

Introduction to Python:

Variables, Visualizing Code Execution, Coding Conventions

Copyright © 2007 - 2024, Department of Systems and Computer Engineering

Carleton University

References

- *Practical Programming*, 3rd ed., Chapter 2, pp. 15 – 27
  - This is the last six sections of Chapter 2:
    - Variables and Computer Memory: Remembering Values
    - How Python Tells You Something Went Wrong
    - A Single Statement That Spans Multiple Lines
    - Describing Code
    - Making Code Readable
    - The Object of This Chapter

General Notes

- Textbook information is in the course outline (page 3)
  - The course outline is in General Course Information in Brightspace
- No, you do **not** need to memorize the conversion chart provided for Lab 1
- Calculators will **<u>not</u>** be permitted on the exams
  - This is a good thing! Calculations will be easier.

Modulus (remainder, %) again ** new

- How to calculate % (in Python):

1. By inspection (i.e. you just look at it and you know)
2. Using a number line (as in lecture 2)
3. If you find calculating // easier then from the formula:

   a = a//b * b + a%b, we get:

   a % b = a - a // b * b

e.g. Calculate: 7 % -5. 7 / -5 is -1.4, so 7 // -5 is -2 (next smallest integer).

Thus 7 % -5 = 7 - (-2 * -5) = 7 - 10 = -3

Lecture Objectives

- Review representing numbers and discuss fractions
- Use the Python shell to learn about computation with variables
- Start to develop a “mental model” of computation
  - Learn how to reason about and visualize the execution of Python code
- Understand why we have coding conventions and learn some of the basic coding conventions

Review Representing Numbers and Discuss How to Deal with Fractions

Carleton University

Learning Outcomes

- Practice binary addition
- Practice binary to decimal and decimal to binary conversion with integers
- Learn how to convert fractions

Number Conversion: Review

- Either way of converting decimal to binary is fine
  - You only need to know one way
- If you are undecided which you prefer, go with the division by 2 method
- What about fractions?
  - We will do an example
  - Instead of division by 2, we multiply by 2

Review: Binary Addition

- Counting in decimal: 0 1 2 3 4 5 6 7
- Counting in binary: 0 1 10 11 100 101 110 111
  - i.e. $2_{10} = 10_2$, $3_{10} = 11_2$, $4_{10} = 100_2$, etc.
- So, in binary: 1 + 1 = 10, or 0, carry 1
  - And 1 + 1 + 1 = 11, or 1, carry 1, etc.
- Let us add two binary numbers: 01010 + 11111:

$$\begin{array}{r} 01010 \\ \underline{11111} \end{array}$$

Review: Binary Addition

Step by step, from left to right (top line is the carries):

```
          1      11     111      111
01010   01010  01010  01010    01010
11111   11111  11111  11111    11111
    1      01    001   1001   101001
```

Double check:

$01010 = 8 + 2 = 10_{10}$

$11111 = 16 + 8 + 4 + 2 + 1 = 31_{10}$,

$101001 = 32 + 8 + 1 = 41_{10}$

Decimal to Binary Conversion Example

Convert 23_{10} to binary using division by 2, using // and %:

23 / 2 = 11 with a remainder of **1**, i.e.

23 = 11 * 2 + 1 = 23 // 2 * 2 + 23 % 2

11 / 2 = 5 with a remainder of **1**, i.e.

11 = 5 * 2 + 1 = 11 // 2 * 2 + 11 % 2

5 / 2 = 2 with a remainder of **1**, i.e.

5 = 2 * 2 + 1 = 5 // 2 * 2 + 5 % 2

2 / 2 = 1 with a remainder of **0**, i.e.

2 = 1 * 2 + 0 = 2 // 2 * 2 + 2 % 2

1 / 2 = 0 with a remainder of **1**, i.e.

1 = 0 * 2 + 1 = 1 // 2 * 2 + 1 % 2

Thus 23 decimal is binary:
(write from bottom to top)
10111
Double check:
10111 = 16 + 4 + 2 + 1 = 23

Another Decimal to Binary Conversion Example

Convert 37_{10} to binary using division by 2:

37 / 2 = 18 with remainder 1

18 / 2 = 9 with remainder 0

9 / 2 = 4 with remainder 1

4 / 2 = 2 with remainder 0

2 / 2 = 1 with remainder 0

1 / 2 = 0 with remainder 1

Thus 37 decimal is binary:
(write from bottom to top)
100101
Double check:
100101 = 32 + 4 + 1 = 37

Decimal to Binary Conversion Example for a Fraction

Convert 0.375_{10} to binary using **multiplication** by 2:

0.375 * 2 = **<u>0</u>**.75: number before the "." is the first binary digit, i.e. 0

0.75 * 2 = **<u>1</u>**.5: so 1 is the next binary digit, and use 0.5 in next step

0.5 * 2 = **<u>1</u>**.0: so 1 is the next binary digit, and we are done (as 0 is left)

Thus 0.375_{10} is 0.011_2 (write digits from top to bottom)

Double check:

$0.011 = 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 0.25 + 0.125 = 0.375_{10}$

Decimal to Binary Conversion Example for Number with Fraction

How would we convert 37.375_{10} to binary?

1. Convert 37_{10} to binary.
2. Convert 0.375_{10} to binary.
3. Add the two answers together (i.e. 100101.011).
4. Double check your answer (convert back to decimal).

Variables

Carleton University

Section Objective

- Use the Python shell to learn about computation with variables

Learning Outcomes (Vocabulary)

- Know the meaning of these words, in the context of computation
  - Variable
  - Assignment (binding)
  - Dynamic typing

Learning Outcomes

- Evaluate arithmetic expressions consisting of literal values, **variables**, and operators, using the same evaluation rules as Python; in other words, predict the values Python will calculate when it evaluates these expressions
  - You will be asked to do this on exams
- Use the Python interpreter to verify your predictions

What is a Variable (Mathematics)?

- In mathematics, a variable is a symbol that represents a mathematical object (e.g., a number, vector, matrix, etc.)

- Example: $y = f(x) = 3x^2 + 42$
  - Variable x is the *argument* of function f
  - Variable y is the *value* of function f

What is a Variable (Programming)?

- In a programming language, a variable is a storage location in a computer's memory that contains a value, and a symbolic name that is used to refer to the location

- We will now use the shell to help us learn about computation with variables in Python
  - An annotated transcript is posted on Brightspace

Variable Names: Syntax Rules

- Are made up of letters, numbers and underscores
- Must begin with a letter or an underscore
- 👍 `x`, `colour`, `total_score`, `UPPER_LIMIT`, `__version__`
- 👎 `67s` (begins with a number)
- 👎 `cost$` (`$` is not a letter, number or underscore)
- 👎 `total-score` (hyphen interpreted as “minus”)

Variable Names: Syntax Rules

- Case is important!

- `degrees_C`, `Degrees_C` and `DEGREES_C` are different names

Summary: Assignment Statements

- General form:

  variable = *expression*

- *expression* is evaluated, then the resulting value is assigned (bound) to *variable*
  - *variable* is created if it does not exist
  - If *variable* exists, a new variable is <u>not</u> created

Summary: Assignment Statements

- When variables are used in an expression and the expression is evaluated, Python substitutes the values to which the variables are bound, then applies the operators

Summary: Dynamic Typing

- Python uses *dynamic typing*
  - a variable can refer to values of different types at different points in a program's execution

Visualizing Code Execution

Carleton University

Section Objectives

- Start to develop a “mental model” of computation
  - Learn how to reason about and visualize the execution of Python code

Learning Outcomes (Vocabulary)

- Know the meaning of these words, in the context of computation
  - Global frame
  - Program state

Learning Outcomes

- Know how to use a web-based tool that visualizes code execution
- Trace short Python programs “by hand” and:
  - explain what happens, step-by-step, as the computer executes each statement
  - draw diagrams that depict the variables in the program’s *global frame* and the objects that are bound to the variables

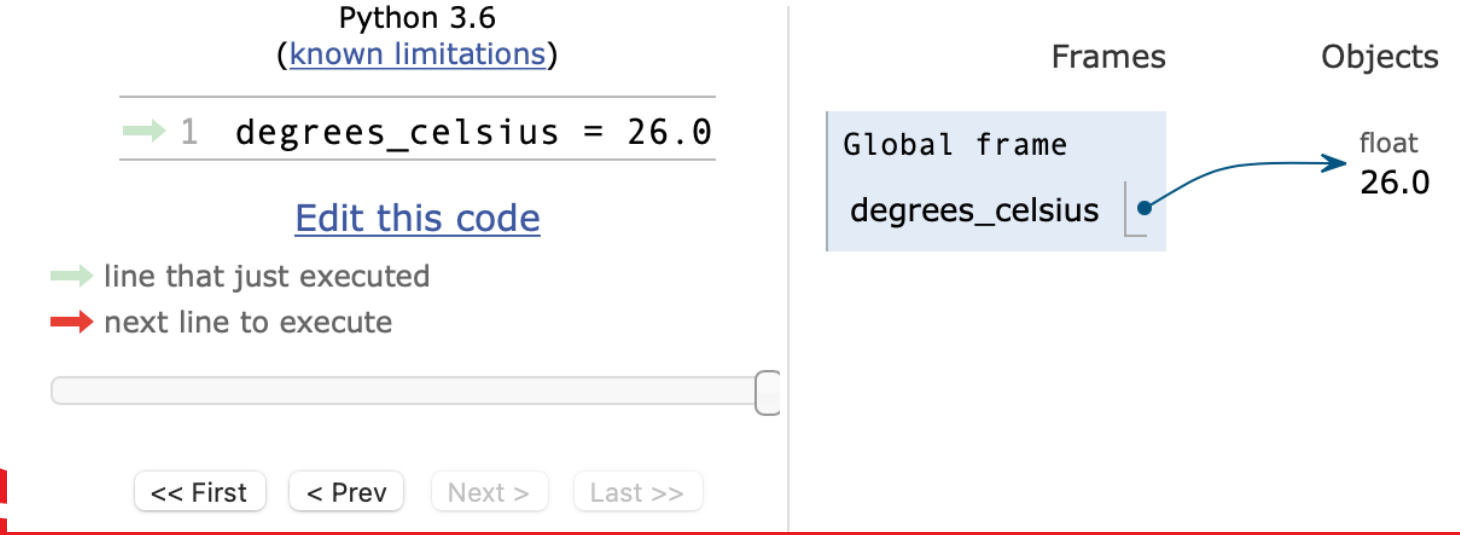
Python Tutor

- A visualization tool that depicts a Python program's *state* as its code is executed, step-by-step
- Python Tutor runs in a Web browser:

  http://pythontutor.com/visualize.html
- We will now use Python Tutor to visualize the execution of code that uses variables
- Ensure that you select "render all objects on the heap (Python/Java)" in the middle drop-down menu

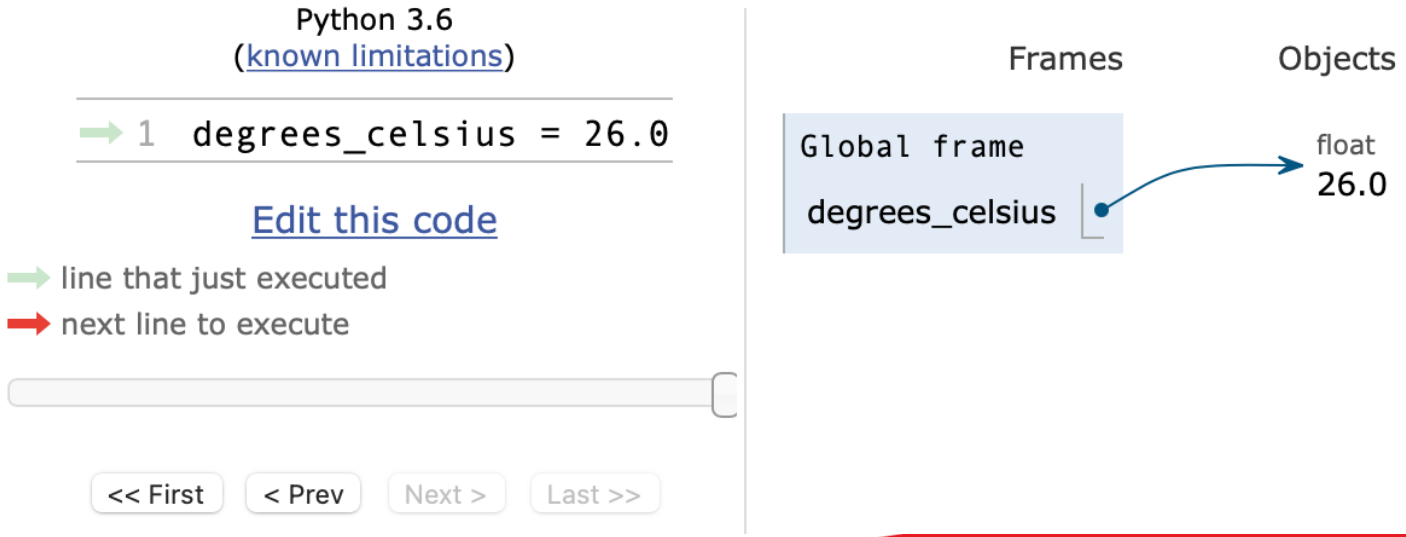
Summary: Python Tutor

- Variables are drawn in the *Global frame* in the Frames column as they are created
  - We will learn about *function activation frames* later…



Summary: Python Tutor

- In Python, every value is stored in an *object*
- Objects (type and value) are drawn in the Objects column

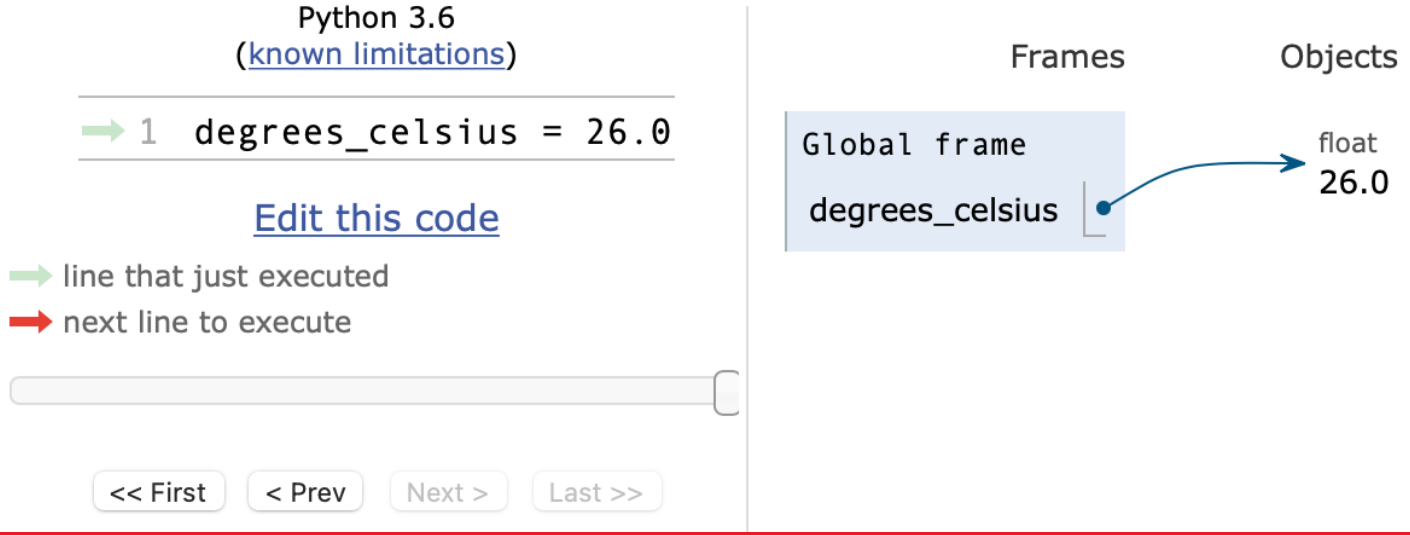


Summary: Python Tutor

- When a value is assigned to a variable, the memory address of the object containing the value is stored in the variable
  - We say that the variable *refers* to the object or is *bound* to the object
- PyTutor provides different ways to display this information

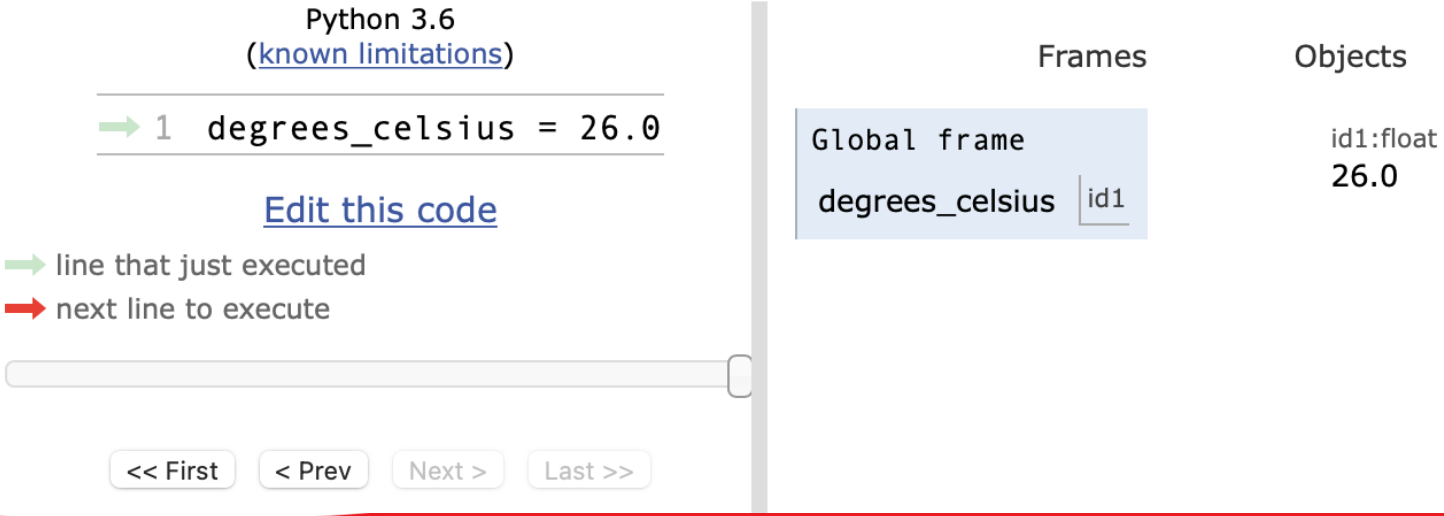
Summary: Python Tutor

- In "draw pointers as arrows" configuration, the memory address of an object is depicted as an arrow pointing from the variable to the object



Summary: Python Tutor

- In "use text labels for pointers" configuration, the memory address of an object is rendered as a symbol (e.g., `id1`) in the variable and the object



Coding Conventions

Carleton University

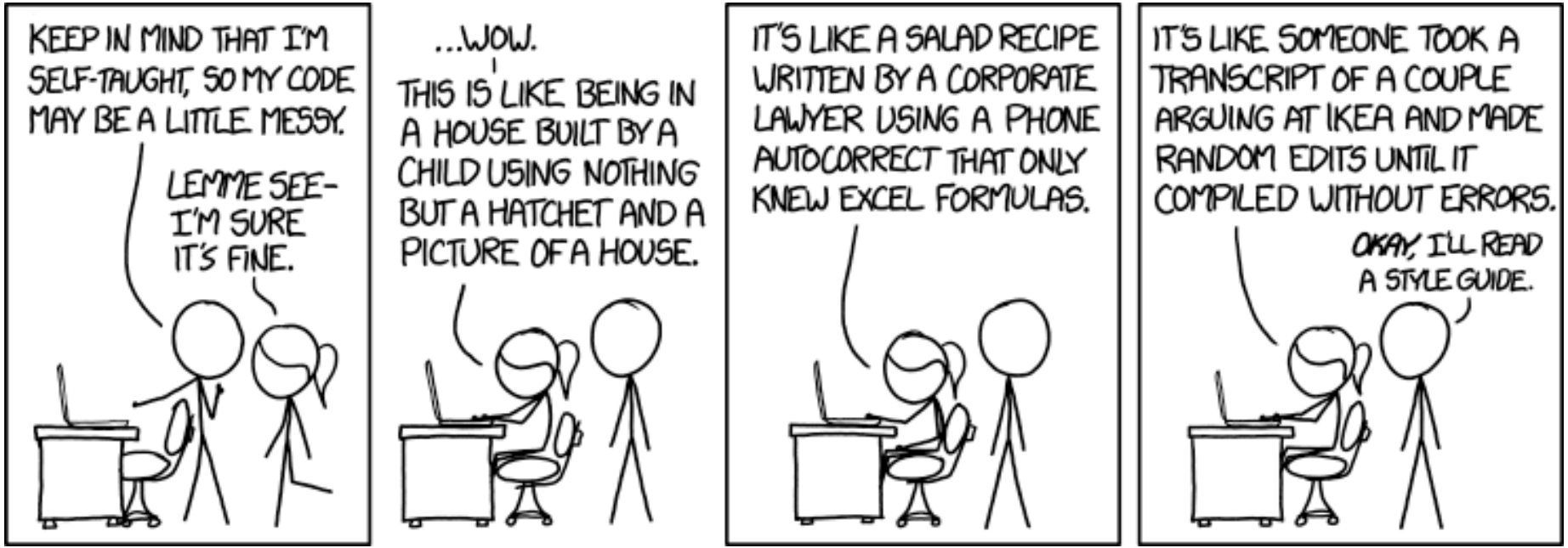
Section Objective

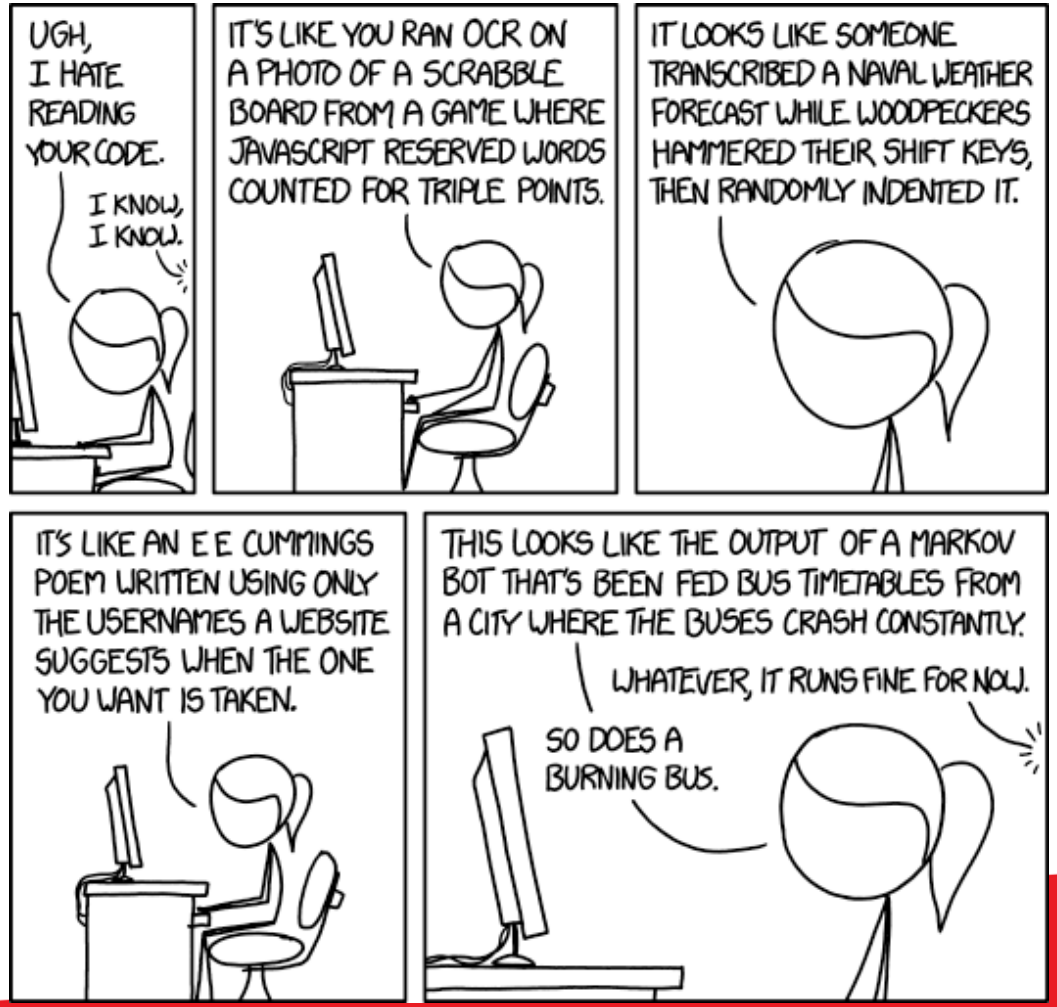
- Understand why we have coding conventions and learn some of the basic coding conventions

What are Coding Conventions?

- “Code is read much more than it is written” (PEP 8 - *Style Guide for Python Code*)
  - https://peps.python.org/pep-0008
- Coding conventions are guidelines/recommendations that help us write programs that are easier to read, understand and maintain

Code Quality @ xkcd.com




UGH, I HATE READING YOUR CODE.
I KNOW, I KNOW.
IT'S LIKE YOU RAN OCR ON A PHOTO OF A SCRABBLE BOARD FROM A GAME WHERE JAVASCRIPT RESERVED WORDS COUNTED FOR TRIPLE POINTS.
IT LOOKS LIKE SOMEONE TRANSCRIBED A NAVAL WEATHER FORECAST WHILE WOODPECKERS HAMMERED THEIR SHIFT KEYS, THEN RANDOMLY INDENTED IT.
IT'S LIKE AN E E CUMMINGS POEM WRITTEN USING ONLY THE USERNAMES A WEBSITE SUGGESTS WHEN THE ONE YOU WANT IS TAKEN.
THIS LOOKS LIKE THE OUTPUT OF A MARKOV BOT THAT'S BEEN FED BUS TIMETABLES FROM A CITY WHERE THE BUSES CRASH CONSTANTLY.
WHATEVER, IT RUNS FINE FOR NOW.
SO DOES A BURNING BUS.

Conventions for Names

- thEre iS a GoOD rEasON wHy WorDs haVE A StaNDaRd caPITaLizAtIon sCHemE

Conventions: Variable Names

- Variable names are (usually) written in lowercase
- If a name consists of multiple words, the words are separated by single underscores
  - This style is often called `pothole_case` or `snake_case`

Conventions: Constants

- Variables whose values should not be changed are known as *constants*
- Names of constants are written in upper case (so that we can distinguish them from variables)
  - If the name has multiple words, the words are separated by underscores (often called `SCREAMING_SNAKE_CASE`)
- 👍 `LBS_PER_KG = 2.20462`

Conventions: Use Descriptive Names

- Is it obvious that this code converts a temperature from Fahrenheit to Celsius?

```
>>> a = 80
>>> b = 5 / 9 * (a - 32)
>>> b
26.666666666666668
```

Conventions: Use Descriptive Names

- Descriptive names help us understand the code

```
>>> degrees_F = 80
>>> degrees_C = 5 / 9 * (degrees_F - 32)

>>> fahrenheit = 80
>>> celsius = 5 / 9 * (fahrenheit - 32)
```

Conventions: Use Descriptive Names

- Descriptive names reduce the need to write comments that explain your code
- Do not do this:

```
# a is the temperature in degrees F
# b is the temperature in degrees C
>>> a = 80
>>> b = 5 / 9 * (a - 32)
```

Conventions: Whitespace

- Put one space on either side of the binary operators (`+`, `-`, `*`, `/`, `%`, `//`, etc.) and `=`

- 👎 `degrees_C=5/9*(degrees_F-32)`
- 👍 `degrees_C = 5 / 9 * (degrees_F - 32)`

Conventions: Whitespace

- Do not put spaces immediately inside parentheses; that is, after "`(`" and before "`)`"

- 👎 `degrees_C = 5 / 9 * (  degrees_F - 32  )`

Conventions: Whitespace

- Do not use unnecessary parentheses
- 👎 `degrees_C = ((5 / 9) * (degrees_F - 32))`
- Learn the precedence of Python's operators
- The only parenthesis that are required here are the ones enclosing the subtraction
- 👍 `degrees_C = 5 / 9 * (degrees_F - 32)`

Conventions: Comments

- Assume the person reading your code knows Python
- Do not write comments that just restate what the code does in English

- 👎 `x = x + 1  # Increment x`
- 👍 `x = x + 1  # Compensate for border`

Recap of Learning Outcomes

Carleton University

Learning Outcomes

- Practice binary addition
- Practice binary to decimal and decimal to binary conversion with integers
- Learn how to convert fractions

Learning Outcomes (Vocabulary)

- Know the meaning of these words, in the context of computation
  - Variable
  - Assignment (binding)
  - Dynamic typing

Learning Outcomes

- Evaluate arithmetic expressions consisting of literal values, **variables**, and operators, using the same evaluation rules as Python; in other words, predict the values Python will calculate when it evaluates these expressions
  - You will be asked to do this on exams
- Use the Python interpreter to verify your predictions

Learning Outcomes (Vocabulary)

- Know the meaning of these words, in the context of computation
  - Global frame
    - Rectangle labeled with “global frame” in the Frames column that contains the names of all the variables
  - Program state
    - Entire memory diagram: both the Frames and Objects columns

Learning Outcomes

- Know how to use a web-based tool that visualizes code execution
- Trace short Python programs “by hand” and:
  - explain what happens, step-by-step, as the computer executes each statement
  - draw diagrams that depict the variables in the program’s *global frame* and the objects that are bound to the variables