CS 105: Introduction to Computer Science

Prof. Thao Nguyen

Spring 2025

Materials adapted from Dave Wonnacott

Variables and Objects so far

- Variables are names given to "objects" (drawn with an arrow in our diagrams)
  - e.g., number_of_cookies: int = 3
  - includes "parameters", e.g. "n_cookies", if we have
    def eat_cookies(n_cookies: int) -> None:
    and
    eat_cookies(3)
- Types of objects we've seen:
  - **int** (integers) and **float** (floating-point approximations of real numbers)
    - manipulate with +, -, *, /, **, %, // and get more numbers (see tutorial on python.org)
    - manipulate with <=, <, ==, >=, >, != and get True or False
  - **bool** (Boolean values, i.e., True and False)
    - manipulate/combine with and, or, not
    - often, though not always, used with "if" (but, could be put in a variable, returned ...)

Digression: Arithmetic Operations

When working with numbers,

- The / operation tries to find the exact quotient, report it as a float
  - 12/3 # produces 4.0 (the .0 is a visual indicator that Python is "thinking of" this as a float)
  - 22/7 # produces something like 3.142857142857143 (exact details are approximate)
  - *also*, when working with float values, beware of == comparisons! (try rounding first)
    - prefer round(22/7, 3) == 3.143 to an attempt at exact comparison
- The // operation finds the whole-number quotient (think back to 1st grade)
  - 12//3 # produces 4
  - 22//7 # produces 3
- The % operation finds the remainder
  - 12%3 # produces 0 ... nothing remains when we divide 12 into three parts of size 4
  - 22%7 # produces 1 ... 22 == 3*7**+1**
  - 12.75%0.5 # produces 0.25, since 12.75 == 25*0.5**+0.25**

Group exercise: sum the digits of a two-digit number

Use either command-line Python or pythontutor or the Python Console in pycharm

Write a function that takes a two-digit number, returns the sum of its digits, e.g., sum_digits(52) gives 7; sum_digits(89) gives 17

(We could do this with a 90-way if/then else, but that would be the hard way...)

String objects and operations

- String objects represent text
  - Indicated in a program with either *single quotes* (just apostrophes) or *quotes* or *triple-quotes*
    - "this is an example string"
    - 'this is a 2nd string; note they can contain punctuation and numerals, etc.'
    - """this is a triple-quoted string; it's o.k. if they span multiple lines, but this one doesn't"""
  - Save as a variable in the usual way, indicate the type with *str*
    - first_name: str = 'Thao'
    - last_name: str = 'Nguyen'
- Operations on string objects
  - first_name + " " + last_name # “+” puts strings together ... why do we use two "+" here?
  - len(first_name) == 4 # len finds the length ... note the quotes aren't part of the string!
  - ==, <=, etc., compare strings in *dictionary order* (expanded to allow non-letters **aNd CaSe!**)
  - [start:end:step] # extracts parts of string

Group Exercise: Blend Names

String slicing examples, discusses J.D.'s "paper towel" metaphor

Exercise: write a function that, given two names, returns the first half of the first name plus the second half of the second

For example,

blend_names("Dave", "Steven") gives "Daven"
blend_names("Steven", "Dave") gives "Steve"
blend_names("Thao", "Suzanne") gives either "Thanne" or "Thnne"

Function/Algorithm Design so far

Which of these should be use, to square or cube something?

What about to find the alphabetically-earliest letter in a three-letter word?

- Relate to a solved problem/library function
- Design by cases
- Top-down design

Group exercise: start writing "power" and "earliest_letter" functions