CSC148 winter 2018

functional programming, top-down

week 5

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.teach.cs.toronto.edu/~csc148h/winter/
416-978-5899

February 5, 2018
Outline

idiomatic python
going with the (pep) tide

Python is more flexible than the community you are coding in. Try to figure out what the \textcolor{red}{python way} is

\begin{itemize}
\item don’t re-invent the wheel (except for academic exercises), e.g. sum, set \textcolor{red}{\rightarrow data type} \textcolor{red}{\rightarrow built-in functions}
\item use comprehensions when you mean to produce a new list (tuple, dictionary, set, …) \textcolor{red}{\rightarrow clearer than alternative}
\item any \textcolor{red}{\approx} \exists \quad all \textcolor{red}{\approx} \forall
\item use ternary if when you want an expression that evaluates in different ways, depending on a condition
\end{itemize}
example: add (cubes of) first 10 natural numbers

- You’ll be generating a new list from \texttt{range(1, 11)}, so use a comprehension

- You want to add all the numbers in the resulting list, so use \texttt{sum}

\texttt{try sum of cubes of 1 \ldots 10}
euclidean distance in 3 dimensions... or more

works for dimensions > 3!

Suppose $L = [x, y, z]$, using $L$, compute:

$$\sqrt{x^2 + y^2 + z^2}.$$
average string length

Try a really large list of words...

Suppose \( L = ["my", "dog", "has", "fancy", "fleas"] \), compute the average string length using \( L \)
try big list with any/all

with open("/usr/share/dict/words", "r") as words_file:
    word_list = words_file.read().split("\n")

- What's average length?
- Do any contain "yx"?
list differences, lists without duplicates

- python lists allow duplicates, python sets don’t
  \[
  \text{list (set (my_list))}
  \]

- python sets have a set-difference operator
  \[
  \{1, 2, 3\} - \{2, 3, 4\}
  \]

- python built-in functions list() and set() convert types
possible test topics
include...

- class design
- special methods
- subclasses
- inheritance
- testing, exceptions
- ADTs, stacks, queues, sacks
- linked lists
valid sudoku

what makes a sudoku square valid?

- valid rows
- valid columns
- valid subsquares

write a simple top-level function with these 3 ideas
def valid_sudoku(grid, digit_set: set) -> bool:
    
    """
    Return whether grid represents a valid, complete sudoku.
    """

    assert all([len(r) == len(grid) for r in grid])
    assert len(grid) == len(digit_set)
    return (_all_rows_valid(grid, digit_set) and 
            _all_columns_valid(grid, digit_set) and 
            _all_subquares_valid(grid, digit_set))

  technical detail: these don't exist (yet)
code those non-existent helpers!

def _all_rows_valid(grid, digit_set: set) -> bool:
    
    Return whether all rows in grid are valid and complete.

    Assume grid has same number of rows as elements of digit_set
    and grid has same number of columns as rows.
    
    assert all([len(r) == len(grid) for r in grid])
    assert len(grid) == len(digit_set)
    return all([_list_valid(r, digit_set) for r in grid])

% doesn't exist yet
def _list_valid(r, digit_set: set) -> bool:
    """
    Return whether r contains each element of digit_set
    exactly once.
    """
    Assume r has same number of elements as digit_set.
    """
    assert len(r) == len(digit_set)
    return set(r) == digit_set

    we've bottomed out...