CSC148-Section:L0301/L0401
Week#5-Monday

Instructed by
AbdulAziz Al-Helali
a.alhelali@mail.utoronto.ca
Office hours: Wednesday 11-1, BA2230.

Slides adapted from Professor Danny Heap course material
winter17
Announcements

• Midterm #1: Wednesday in Exam Center
Possible test topics:

• Class design (main noun/verbs/ attributes)
• Special methods ( __init__/__str__/__eq__ etc)
• Inheritance
  • Subclasses
    • declare class Son(Father)
    • Inherit methods -> use unchanged
    • Extend methods -> use, then add some bits
    • Override -> replace completely
Possible test topics:

• Testing, exceptions
  • We will not ask you to write unittest classes or methods
  • Know exception hierarchy
• ADTs, stacks, queues, sacks
  • Familiar with add/remove/is_empty
• Linked Lists
  • Modify LL
  • Walk LL
Possible test topics:

• Write Docstrings
• Read Docstrings
• Your code should fit the space provided. If not think about your approach.
valid sudoku

• what makes a sudoku square valid?
valid sudoku

• what makes a sudoku square valid?

• valid rows

• valid columns

• valid subsquares
code it!

• Follow: **Top – down** design
  • Divide the work into smaller problems
  • Write a method for each smaller problem
  • Make each method readable not too many lines
    • If a method implementation requires you to scroll down your screen to read it, it means it is long make it shorter by building helper methods.
      • This **reduces the errors** in your code and **make it readable**
Top – down design

• Create an example sudoku grid
• Create digit set

```python
# an example of a grid, for testing
_GRID = [[5, 3, 4, 6, 7, 8, 9, 1, 2],
          [6, 7, 2, 1, 9, 5, 3, 4, 8],
          [1, 9, 8, 3, 4, 2, 5, 6, 7],
          [8, 5, 9, 7, 6, 1, 4, 2, 3],
          [4, 2, 6, 8, 5, 3, 7, 9, 1],
          [7, 1, 3, 9, 2, 4, 8, 5, 6],
          [9, 6, 1, 5, 3, 7, 2, 8, 4],
          [2, 8, 7, 4, 1, 9, 6, 3, 5],
          [3, 4, 5, 2, 8, 6, 1, 7, 9]]

# an example of a digit_set for testing
_DIGIT_SET = {1, 2, 3, 4, 5, 6, 7, 8, 9}
```
Top – down design

• what makes a sudoku square valid?
  • valid rows
  • valid columns
  • valid subsquares
def valid_sudoku(grid: list[list], digit_set: set) -> bool:
    """Return whether grid represents a valid, complete sudoku.

    Assume grid is square (as many rows as columns) and has the same number of rows as elements of digit_set.
    >>> valid_sudoku(_GRID, _DIGIT_SET)
    True
    >>> g = [[x for x in row] for row in _GRID]
    >>> g[0][1] = 5
    >>> valid_sudoku(g, _DIGIT_SET)
    False
    """
Top – down design

• Create an example

```python
def valid_sudoku(grid: list[list], digit_set: set) -> bool:
    # insures we have square grid num cols == num rows
    assert all([len(r) == len(grid) for r in grid])
    # insures grid len == digit set
    assert len(grid) == len(digit_set)
    # call methods that will implemented later
    return _all_rows_valid(grid, digit_set) and
    _all_columns_valid(grid, digit_set) and
    _all_subquares_valid(grid, digit_set)
```
Top – down design

• what makes a sudoku square valid?
  • valid rows
  • valid columns
  • valid subsquares
def _all_rows_valid(grid, digit_set):
    return all([_list_valid(r, digit_set) for r in grid])
list_Valid

def _list_valid(r, digit_set):
    return set(r) == digit_set
Top – down design

• what makes a sudoku square valid?
  • valid rows
  • valid columns
  • valid subsquares
def _all_columns_valid(grid, digit_set):
    return all([_list_valid(c, digit_set) for c in _columns(grid)])
columns

def _columns(grid):
    return [_column(i, grid) for i in range(len(grid))]
def _column(i, grid):
    return [r[i] for r in grid]
Top – down design

• what makes a sudoku square valid?
  • valid rows
  • valid columns
  • valid subsquares
def _all_subquares_valid(grid, digit_set):
    return all([_list_valid(s, digit_set) for s in _subquares(grid)])