CMPT 120

Topic: Lists – Part 2
Last Lecture

• Design and implement a recursive function
• Box trace (hand trace with boxes) a recursive function to figure out what it produces
Learning outcomes

At the end of this course, a student is expected to:

• Create (design), analyze, and explain the behaviour of simple algorithms:
  • **Solve problems by designing** simple algorithms, e.g., basic calculations, searching in strings and **lists**, etc…
  • Create (design) small to medium size programs using Python:
    • **Create programs** that search or construct **lists** and strings
    • Create programs that modify **lists** in-place
  • Use Python's data types appropriately:
    • Compare and contrast mutable and immutable data types (**lists** vs. strings, numeric values)
Today’s Menu

• Another look at lists
Lists – so far

• We saw that a list can contain elements of various data types

```python
prices = [1.20, 0.75, 4.50]
names = ["Mike", "Xinghua", "Lise"]
somePrimes = [1, 3, 5, 7, 11, 13]
underTheBed = [3, "old socks"]
```
Creating lists – so far

• Here are ways of creating lists we have used so far:
  
  • aList = list() or bList = list("132")
  • cList = [] or dList = ['1', '3', '2']
  • eList = ['4'] + dList
  • fList = eList[2:]
  • gList = sorted(eList)
  • equationList =
Another way of creating a list

- **List comprehension**

  **Example 1:**
  
  \[
  \text{max} = 5 \\
  \text{list1} = ["*" \text{ for number in range}(\text{max})] \\
  \]

  **Example 2:**
  
  \[
  \text{length} = 4 \\
  \text{list2} = [\text{number for number in range}(\text{length})] \\
  \]

  **Example 3:**
  
  \[
  \text{operandList} = ["4", "5"] \\
  \text{operandList} = [\text{int}(\text{operandList}[i]) \text{ for i in range}(<\text{len(operandList)'}\text{)])] \\
  \]
Another look at lists

• We saw that a list can contain elements of various data types
• What if a list contains lists?

\[
\text{stdInfo} = ["Mike", [112, "B Street"], "YVR"]
\]
\[
\text{stdGrades} = [ [3,4.5,4], [3.5,5,"-" ] ]
\]

What would we use a list of lists for in our programs?
2 dimensional array or matrix

• **Question**: What if a problem statement (Step 1) ask us to manipulate a matrix?

• **Answer**: In Step 2, we could use a list of lists to represent a matrix

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\]

• **Answer**: In Step 2, we could use a list of lists to represent a matrix

• And in Step 4, the Python code representing the above data would look like:

```python
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```
Access elements in a list of lists

myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

stdInfo = ["Mike", [112, "B Street"], "YVR"]
Slicing a list of lists

myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Other use for a list of lists

- What else could a list of lists represent in our programs?
How to create such a list of lists –> **a grid** – Take 1

```python
# Set variables
row = 5
column = 3
symbol = "-"
grid = list()
# Create a grid
for aRow in range(row):
    listRow = list()
    for aColumn in range(column):
        listRow.insert(aColumn, symbol)
    grid.insert(aRow, listRow)
```
How to create such a list of lists -> a grid – Take 2

```python
# Set variables
row = 5
column = 3
cellContent = " - "

# Create a grid
grid = [[cellContent for aColumn in range(column)]
         for aRow in range(row)]
```
How to print such a list of lists

Take 1:
print(grid)

Take 2:
for aRow in range(row):
    print(grid[aRow])

Take 3:
# pint the list using join() method
for aRow in range(len(grid)):
    print(' '.join(grid[aRow]))
Summary

- Another look at lists
  - List of Lists
- What they could represent in our programs
  - 2D matrix
  - Grid
- How to create a list of lists
- How to print a list of lists
Next Lecture

- File I/O