INDIAN INSTITUTE OF TECHNOLOGY MADRAS · सिद्धिर्भवति कर्मजा ·

IIT Madras

ONLINE DEGREE

Pseudocode: Introducing lists

Collections

- Variables keep track of intermediate values

Collections

- Variables keep track of intermediate values
- Often we need to keep track of a collection of values
    - Students with highest marks in Physics
    - Customers who have bought food items from SV Stores
    - Nouns that follow an adjective

Collections

- Variables keep track of intermediate values
- Often we need to keep track of a collection of values
    - Students with highest marks in Physics
    - Customers who have bought food items from SV Stores
    - Nouns that follow an adjective
- Simplest collection is a list
    - Sequence of values
    - Single variable refers to the entire sequence

Collections

- Variables keep track of intermediate values
- Often we need to keep track of a collection of values
    - Students with highest marks in Physics
    - Customers who have bought food items from SV Stores
    - Nouns that follow an adjective
- Simplest collection is a list
    - Sequence of values
    - Single variable refers to the entire sequence
    - Notation for lists
    - Primitive operations to manipulate lists

Pseudocode for lists

- Sequence within square brackets
    - `[1,13,2]`
    - `["Vedanayagam","cane", "Monday","school"]`
    - `[]` — empty list

Pseudocode for lists

- Sequence within square brackets
    - `[1,13,2]`
    - `["Vedanayagam","cane", "Monday","school"]`
    - `[]` — empty list
- Append two lists, `l1 ++ l2`
    - `l1` is `[1,13]`, `l2` is `[2,17,1]`
    - `l1++l2` is `[1,13,2,17,1]`

Pseudocode for lists

- Sequence within square brackets
  - `[1,13,2]`
  - `["Vedanayagam","cane", "Monday","school"]`
  - `[]` — empty list
- Append two lists, `l1 ++ l2`
  - `l1` is `[1,13]`, `l2` is `[2,17,1]`
  - `l1++l2` is `[1,13,2,17,1]`
- Extend `l` with item `x`
  - `l = l ++ [x]`

Pseudocode for lists

- Sequence within square brackets
  - `[1,13,2]`
  - `["Vedanayagam","cane", "Monday","school"]`
  - `[]` — empty list
- Append two lists, `l1 ++ l2`
  - `l1` is `[1,13]`, `l2` is `[2,17,1]`
  - `l1++l2` is `[1,13,2,17,1]`
- Extend `l` with item `x`
  - `l = l ++ [x]`
- Examples
  - List of students born in May

```
mayList = []
while (Table 1 has more rows) {
   Read the first row X in Table 1
   if (X.MonthOfBirth == "May") {
     mayList = mayList ++
                         [X.Seqno]
   }
   Move X to Table 2
}
```

Pseudocode for lists

- Sequence within square brackets
  - `[1,13,2]`
  - `["Vedanayagam","cane", "Monday","school"]`
  - `[]` — empty list
- Append two lists, `l1 ++ l2`
  - `l1` is `[1,13]`, `l2` is `[2,17,1]`
  - `l1++l2` is `[1,13,2,17,1]`
- Extend `l` with item `x`
  - `l = l ++ [x]`
- Examples
  - List of students born in May
  - List of students from Chennai

```
chennaiList = []
while (Table 1 has more rows) {
   Read the first row X in Table 1
   if (X.TownCity == "Chennai") {
     chennaiList = chennaiList ++
                            [X.Seqno]
   }
   Move X to Table 2
}
```

Processing lists

- Typically, we need to iterate over a list
    - Examine each item
    - Process it appropriately

Processing lists

- Typically, we need to iterate over a list
    - Examine each item
    - Process it appropriately
-
```
foreach x in l {
    Do something with x
}
```
    - `x` iterates through values in `l`

Processing lists

- Typically, we need to iterate over a list
  - Examine each item
  - Process it appropriately
- `foreach x in l {`

  ```
      Do something with x
  }
  ```
  - `x` iterates through values in `l`
- Example
  - All students born in May who are from Chennai
  - Nested `foreach`

```
mayChennaiList = []
foreach x in mayList {
    foreach y in chennaiList {
      if (x == y) {
        mayChennaiList =
               mayChennaiList ++ [x]
      }
    }
}
```

Summary

- A list is a sequence of values
- Write a list as `[x1,x2,...,xn]`
- Combine lists using `++`
  - `[x1,x2] ++ [y1,y2,y3]` $\mapsto$ `[x1,x2,y1,y2,y3]`
- Extending list `l` by an item `x`
  - `l = l ++ [x]`
- `foreach` iterates through values in a list

```
foreach x in l {
    Do something with x
}
```