Intro to Computer Science II

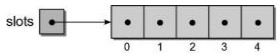
2D Arrays, Lists, and
User Modules

Arrays And Lists

- An array has to be created and initialized before it can be used.

```
slots = [None for i in range(5)]
for i in range(len(slots)):
  print(slots[i])
```

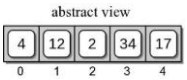
  - elements are like any other variable.
  - we must keep track of the size of the array.



List: Construction

- The Python list interface provides an abstraction to the actual underlying implementation.

```
py_list = [ 4, 12, 2, 34, 17 ]
```



2-D Arrays

- Arrays of 2 or more dimensions are not supported at the hardware level.
  - Most languages provide some mechanism for creating and managing multi-dimensional arrays.
  - 2-D arrays are very common data structure in computer science.

2-D Array ADT

- A 2-D *array* consists of a collection of elements organized into rows and columns.
  - Elements are referenced by row and column index (start at 0).
  - Once created, array size can not be changed.

- Array2D(nrows, ncols)
- num_rows()
- num_cols()
- clear(value)
- *getitem*(i_1, i_2)
- *setitem*(i_1, i_2, value)

2-D Array Example

- Suppose we have a text file containing exam grades for multiple students.
  - Extract the grades from the file.
  - Store them in a 2-D array.
  - Compute the average exam grades.
  - Example: *n* (7) students with *m* (3) grades each

```
7
3
90   96   92
85   91   89
82   73   84
69   82   86
95   88   91
78   64   84
92   85   89
```

2-D Array Example

avggrades.py

```
from array import Array2D

 # Open the text file for reading.
grade_file = open( filename, "r" )

 # Extract the first two values; indicate the size of the array.
num_exams = int( grade_file.readline() )
num_students = int( gradeFile.readline() )

 # Create the 2-D array to store the grades.
exam_grades = Array2D( num_students, num_exams )

 # Extract the grades from the remaining lines.
i = 0
for student in grade_file :
  grades = student.split()
  for j in range( num_exams ):
    exam_grades[i,j] = int( grades[j] )
  i += 1

 # Close the text file.
grade_file.close()
```


2-D Array Example

- The contents of the 2-D array produced by the previous code segment.

```
7
3
90   96   92
85   91   89
82   73   84
69   82   86
95   88   91
78   64   84
92   85   89
```

	0	1	2
0	90	96	92
1	85	91	89
2	82	73	84
3	69	82	86
4	95	88	91
5	78	64	84
6	92	85	89

2-D Array Example

avggrades.py

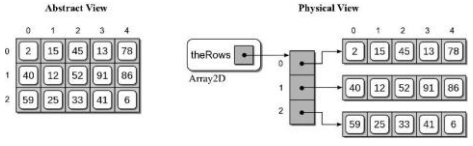
```
 # Compute each student's average exam grade.
for i in range( num_students ) :

  total = 0
  for j in range( num_exams ) :
    total += exam_grades[i,j]

  exam_avg = total / num_exams
  print( "%2d:  %6.2f" % (i+1, exam_avg) )
```


Implementing the 2-D Array

- There are various approaches that can be used to implement a 2-D array.
  - Use a single 1-D array with the elements arranged by row or column.
  - Use a 1-D array of 1-D arrays.
  - Use lists

Array of Arrays Implementation

- Each row is stored within its own 1-D array.
- A 1-D array is used to store references to each row array.



2-D Array Implementation

array.py

```
class Array2D :
  def __init__( self, n_rows, n_cols ):
    self._the_rows = Array( numRows )
    for i in range( n_rows ) :
      self._the_rows[i] = Array( n_cols )

  def num_rows( self ):
    return len( self._the_rows )

  def num_cols( self ):
    return len( self._the_rows[0] )

  def clear( self, value = 0):
    for row in range( self.num_rows() ):
      row.clear( value )
```

2-D Array Implementation

- Subscript notation:
  `y = x[r, c]  x[r, c] = z`
  - Subscripts are passed to the methods as a tuple.
  - Must verify the size of the tuple.

13

2-D Array Implementation

array.py

```
class Array2D :
# ...

  def __getitem__( self, ndx_tuple ):
    assert len(ndx_tuple) == 2, "Invalid number of array subscripts."
    row = ndx_tuple[0]
    col = ndx_tuple[1]
    assert row >= 0 and row < self.num_rows() \
       and col >= 0 and col < self.num_cols(), \
           "Array subscript out of range."
    the_row_array = self._the_rows[row]
    return the_row_array[col]
```

14

2-D Array Implementation

array.py

```
class Array2D :
# ...

  def __setitem__( self, ndx_tuple, value ):
    assert len(ndx_tuple) == 2, "Invalid number of array subscripts."
    row = ndx_tuple[0]
    col = ndx_tuple[1]
    assert row >= 0 and row < self.num_rows() \
       and col >= 0 and col < self.num_cols(), \
           "Array subscript out of range."
    the_row_array = self._the_rows[row]
    the_row_array[col] = value
```

15