Arrays

- Array Structure
- Array ADT
- 1-D Arrays
- Python List
- 2-D Arrays
The Array Structure

An array is the most basic type of container.

- Arrangement of some data/objects
- Most languages provide arrays as a primitive type.
  - allow for creating multiple dimension arrays
- Can be used with a wide range of problems
1-D array (Linear array)

A sequence structure:
• Composed of multiple elements.
• Elements are stored in contiguous bytes of memory.
• Entire contents is known by a single name.
• Individual elements can be accessed by subscript.
Array -vs- Python list

• Similar:
  • Both are sequences
  • composed of multiple sequential elements
  • can be accessed by position

• Two major differences:
  • Arrays only have 3 operations:
    • array creation
    • reading a specific element
    • writing a specific element
  • The size of an array is fixed
Why Study Arrays?

• Python provides the list structure as its mutable sequence type
• Do we really need arrays?
  • Many languages only provide the array structure
  • Both structures have their uses
When to use Arrays?

- Arrays are best suited to problems where:
  - maximum number of elements is known up front.
    - array size is fixed
    - the list has extra space that can be wasteful.
  - only a limited number of operations are needed.
    - arrays have 3 operations.
    - the list can manage the items in the container.
1-D Array ADT

- A 1-D array is a collection of contiguous elements with each element identified by integer subscript.
  - Subscripts start at 0.
  - Once created, array size can not be changed.

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array( size )</td>
</tr>
<tr>
<td>length()</td>
</tr>
<tr>
<td>getitem( index)</td>
</tr>
<tr>
<td>setitem( index, value )</td>
</tr>
<tr>
<td>clear( value )</td>
</tr>
<tr>
<td>iterator()</td>
</tr>
</tbody>
</table>
# Fill a 1-D array with random values, then print the values.
from arrays import Array
import random

# The constructor is called to create the array.
valueList = Array( 100 )

# Fill the array with random floating-point values.
for i in range( len(valueList) ):
    valueList[i] = random.random()

# Print the values, one per line.
for i in range( len(valueList) ) :
    print( valueList[i] )
Array- Example 2

```python
# Count the number of occurrences of each letter in a text file.
from arrays import Array

# Create an array for the counters.
theCounters = Array(127)
theCounters.clear(0)

# Open the text file for reading and extract each letter.
theFile = open('atextfile.txt', 'r')
for line in theFile:
    for letter in line:
        code = ord(letter)
        theCounters[code] += 1

# Close the file
theFile.close()

# Print the results.
for i in range(26):
    print("%c - %4d   %c - %4d" %
           (chr(65+i), theCounters[65+i], chr(97+i), theCounters[97+i]))
```
### ASCII Table

<table>
<thead>
<tr>
<th>Dec</th>
<th>HxOct</th>
<th>Char</th>
<th>Dec</th>
<th>HxOct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>HxOct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>HxOct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>HxOct</th>
<th>Html</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td><strong>(null)</strong></td>
<td>32</td>
<td>040</td>
<td>#32;</td>
<td>Space</td>
<td>64</td>
<td>100</td>
<td>#64;</td>
<td>B</td>
<td>96</td>
<td>140</td>
<td>#96;</td>
<td>`</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td><strong>(start of heading)</strong></td>
<td>33</td>
<td>041</td>
<td>#33;</td>
<td>65</td>
<td>101</td>
<td>#65;</td>
<td>A</td>
<td>97</td>
<td>141</td>
<td>#97;</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td><strong>(start of text)</strong></td>
<td>34</td>
<td>042</td>
<td>#34;</td>
<td>66</td>
<td>102</td>
<td>#66;</td>
<td>B</td>
<td>98</td>
<td>142</td>
<td>#98;</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>003</td>
<td><strong>(end of text)</strong></td>
<td>35</td>
<td>043</td>
<td>#35;</td>
<td>67</td>
<td>103</td>
<td>#67;</td>
<td>C</td>
<td>99</td>
<td>143</td>
<td>#99;</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>004</td>
<td><strong>(end of transmission)</strong></td>
<td>36</td>
<td>044</td>
<td>#36;</td>
<td>68</td>
<td>104</td>
<td>#68;</td>
<td>D</td>
<td>100</td>
<td>144</td>
<td>#100;</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>005</td>
<td><strong>(enquiry)</strong></td>
<td>37</td>
<td>045</td>
<td>#37;</td>
<td>69</td>
<td>105</td>
<td>#69;</td>
<td>E</td>
<td>101</td>
<td>145</td>
<td>#101;</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>006</td>
<td><strong>(acknowledge)</strong></td>
<td>38</td>
<td>046</td>
<td>#38;</td>
<td>70</td>
<td>106</td>
<td>#70;</td>
<td>F</td>
<td>102</td>
<td>146</td>
<td>#102;</td>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>007</td>
<td><strong>(bell)</strong></td>
<td>39</td>
<td>047</td>
<td>#39;</td>
<td>71</td>
<td>107</td>
<td>#71;</td>
<td>G</td>
<td>103</td>
<td>147</td>
<td>#103;</td>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>010</td>
<td><strong>(backspace)</strong></td>
<td>40</td>
<td>050</td>
<td>#40;</td>
<td>72</td>
<td>110</td>
<td>#72;</td>
<td>H</td>
<td>104</td>
<td>150</td>
<td>#104;</td>
<td>h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>011</td>
<td><strong>(horizontal tab)</strong></td>
<td>41</td>
<td>051</td>
<td>#41;</td>
<td>73</td>
<td>111</td>
<td>#73;</td>
<td>I</td>
<td>105</td>
<td>151</td>
<td>#105;</td>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>012</td>
<td><strong>(NL line feed, new line)</strong></td>
<td>42</td>
<td>052</td>
<td>#42;</td>
<td>74</td>
<td>112</td>
<td>#74;</td>
<td>J</td>
<td>106</td>
<td>152</td>
<td>#106;</td>
<td>j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>013</td>
<td><strong>(vertical tab)</strong></td>
<td>43</td>
<td>053</td>
<td>#43;</td>
<td>75</td>
<td>113</td>
<td>#75;</td>
<td>K</td>
<td>107</td>
<td>153</td>
<td>#107;</td>
<td>k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>014</td>
<td><strong>(MP form feed, new page)</strong></td>
<td>44</td>
<td>054</td>
<td>#44;</td>
<td>76</td>
<td>114</td>
<td>#76;</td>
<td>L</td>
<td>108</td>
<td>154</td>
<td>#108;</td>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>015</td>
<td><strong>(carriage return)</strong></td>
<td>45</td>
<td>055</td>
<td>#45;</td>
<td>77</td>
<td>115</td>
<td>#77;</td>
<td>M</td>
<td>109</td>
<td>155</td>
<td>#109;</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>016</td>
<td><strong>(shift out)</strong></td>
<td>46</td>
<td>056</td>
<td>#46;</td>
<td>78</td>
<td>116</td>
<td>#78;</td>
<td>N</td>
<td>110</td>
<td>156</td>
<td>#110;</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>017</td>
<td><strong>(shift in)</strong></td>
<td>47</td>
<td>057</td>
<td>#47;</td>
<td>79</td>
<td>117</td>
<td>#79;</td>
<td>O</td>
<td>111</td>
<td>157</td>
<td>#111;</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>020</td>
<td><strong>(data link escape)</strong></td>
<td>48</td>
<td>060</td>
<td>#48;</td>
<td>80</td>
<td>120</td>
<td>#80;</td>
<td>P</td>
<td>112</td>
<td>160</td>
<td>#112;</td>
<td>p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>021</td>
<td><strong>(device control 1)</strong></td>
<td>49</td>
<td>061</td>
<td>#49;</td>
<td>81</td>
<td>121</td>
<td>#81;</td>
<td>Q</td>
<td>113</td>
<td>161</td>
<td>#113;</td>
<td>q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>022</td>
<td><strong>(device control 2)</strong></td>
<td>50</td>
<td>062</td>
<td>#50;</td>
<td>82</td>
<td>122</td>
<td>#82;</td>
<td>R</td>
<td>114</td>
<td>162</td>
<td>#114;</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>023</td>
<td><strong>(device control 3)</strong></td>
<td>51</td>
<td>063</td>
<td>#51;</td>
<td>83</td>
<td>123</td>
<td>#83;</td>
<td>S</td>
<td>115</td>
<td>163</td>
<td>#115;</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>024</td>
<td><strong>(device control 4)</strong></td>
<td>52</td>
<td>064</td>
<td>#52;</td>
<td>84</td>
<td>124</td>
<td>#84;</td>
<td>T</td>
<td>116</td>
<td>164</td>
<td>#116;</td>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>025</td>
<td><strong>(negative acknowledge)</strong></td>
<td>53</td>
<td>065</td>
<td>#53;</td>
<td>85</td>
<td>125</td>
<td>#85;</td>
<td>U</td>
<td>117</td>
<td>165</td>
<td>#117;</td>
<td>u</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>026</td>
<td><strong>(synchronous idle)</strong></td>
<td>54</td>
<td>066</td>
<td>#54;</td>
<td>86</td>
<td>126</td>
<td>#86;</td>
<td>V</td>
<td>118</td>
<td>166</td>
<td>#118;</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>027</td>
<td><strong>(end of trans. block)</strong></td>
<td>55</td>
<td>067</td>
<td>#55;</td>
<td>87</td>
<td>127</td>
<td>#87;</td>
<td>W</td>
<td>119</td>
<td>167</td>
<td>#119;</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>030</td>
<td><strong>(cancel)</strong></td>
<td>56</td>
<td>070</td>
<td>#56;</td>
<td>88</td>
<td>130</td>
<td>#88;</td>
<td>X</td>
<td>120</td>
<td>170</td>
<td>#120;</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>031</td>
<td><strong>(end of medium)</strong></td>
<td>57</td>
<td>071</td>
<td>#57;</td>
<td>89</td>
<td>131</td>
<td>#89;</td>
<td>Y</td>
<td>121</td>
<td>171</td>
<td>#121;</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>032</td>
<td><strong>(substitute)</strong></td>
<td>58</td>
<td>072</td>
<td>#58;</td>
<td>90</td>
<td>132</td>
<td>#90;</td>
<td>Z</td>
<td>122</td>
<td>172</td>
<td>#122;</td>
<td>z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>033</td>
<td><strong>(escape)</strong></td>
<td>59</td>
<td>073</td>
<td>#59;</td>
<td>91</td>
<td>133</td>
<td>#91;</td>
<td>\</td>
<td>123</td>
<td>173</td>
<td>#123;</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>034</td>
<td><strong>(file separator)</strong></td>
<td>60</td>
<td>074</td>
<td>#60;</td>
<td>92</td>
<td>134</td>
<td>#92;</td>
<td>]</td>
<td>124</td>
<td>174</td>
<td>#124;</td>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>035</td>
<td><strong>(group separator)</strong></td>
<td>61</td>
<td>075</td>
<td>#61;</td>
<td>93</td>
<td>135</td>
<td>#93;</td>
<td>^</td>
<td>125</td>
<td>175</td>
<td>#125;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>036</td>
<td><strong>(record separator)</strong></td>
<td>62</td>
<td>076</td>
<td>#62;</td>
<td>94</td>
<td>136</td>
<td>#94;</td>
<td>_</td>
<td>126</td>
<td>176</td>
<td>#126;</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>037</td>
<td><strong>(unit separator)</strong></td>
<td>63</td>
<td>077</td>
<td>#63;</td>
<td>95</td>
<td>137</td>
<td>#95;</td>
<td>`</td>
<td>127</td>
<td>177</td>
<td>#127;</td>
<td>\</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Array Implementation

- Python is built using the C language.
  - High-level compiled language.
  - Provides syntax for working with the hardware.
- Python provides the `ctypes` module
  - Access to C data types and functionality.
  - Provides for hardware-supported arrays.
  - Requires knowledge of C language.
  - Not meant for direct use in programs.
Hardware Array: Creation

- Create a hardware array

```
import ctypes

ArrayType = ctypes.py_object * 5
slots = ArrayType()
```

- fixed size
- each element stores a reference to an object.
Hardware Array: Initialize

- A hardware array has to be initialized before it can be used.

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

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

- References to any type of Python can be stored in any array element.

slots[1] = 12
slots[3] = 54
slots[4] = 37
Hardware Array: Remove

- Items can be removed from the array.

```
slots[3] = None
```

```
slots
    □→  ▪  12  ▪  54  ▪  39
  0  1  2  3  4

slots
    □→  ▪  12  ▪  ▪  39
  0  1  2  3  4
```
import ctypes

class Array:
    def __init__(self, size):
        assert size > 0, "Array size must be > 0"
        self._size = size

        PyArrayType = ctypes.py_object * size
        self._elements = PyArrayType()

        self.clear(None)

    def __len__(self):
        return self._size

    def clear(self, value):
        for i in range(len(self)):
            self._elements[i] = value

    def __iter__(self):
        return ArrayIterator(_self.elements)
class Array:
    # ...
    def __getitem__(self, index):
        assert index >= 0 and index < len(self),
        "Array subscript out of range"
        return self._elements[index]

    def __setitem__(self, index, value):
        assert index >= 0 and index < len(self),
        "Array subscript out of range"
        self._elements[index] = value
Array ADT Implementation

```python
# An iterator for the Array ADT.
class _ArrayIterator :
    def __init__( self, theArray ) :
        self._arrayRef = theArray
        self._curNdx = 0

    def __iter__( self ) :
        return self

    def __next__( self ) :
        if self._curNdx < len( self._arrayRef ) :
            entry = self._arrayRef[ self._curNdx ]
            self._curNdx += 1
            return entry
        else :
            raise StopIteration
```
The Python List

- A mutable sequence type container.
  - Provides operations for managing the collection.
  - Can grow and/or shrink as needed.
  - Implemented using an array.
List: Construction

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

```python
pyList = [ 4, 12, 2, 34, 17 ]
```
List: Implementation

- An array is used to store the items of the list.
  - Created bigger than needed.
  - Has capacity for future items.
- **subarray** – the items are stored in a contiguous subset of the array.
List: Appending an Item

- New items can be added at the end of the list.

```python
pyList.append(50)
```

- When space is available, the item is stored in the next slot.
List: Appending an Item

- What happens when the array becomes full?
  
  pyList.append(18)
  pyList.append(64)
  pyList.append(6)

- There is no space for value 6.
Lists: Expanding The List

**Step 1:** create a new array, double the size.

```
tempArray
```

**Step 2:** copy the items from original array to the new array.
Lists: Expanding The List

**Step 3:** replace the original array with the new array.

**Original array**

<table>
<thead>
<tr>
<th>pyList</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: 4</td>
</tr>
<tr>
<td>1: 12</td>
</tr>
<tr>
<td>2: 2</td>
</tr>
<tr>
<td>3: 34</td>
</tr>
<tr>
<td>4: 17</td>
</tr>
<tr>
<td>5: 50</td>
</tr>
<tr>
<td>6: 18</td>
</tr>
<tr>
<td>7: 64</td>
</tr>
</tbody>
</table>

**New array**

<table>
<thead>
<tr>
<th>pyList</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: 4</td>
</tr>
<tr>
<td>1: 12</td>
</tr>
<tr>
<td>2: 2</td>
</tr>
<tr>
<td>3: 34</td>
</tr>
<tr>
<td>4: 17</td>
</tr>
<tr>
<td>5: 50</td>
</tr>
<tr>
<td>6: 18</td>
</tr>
<tr>
<td>7: 64</td>
</tr>
<tr>
<td>8: 6</td>
</tr>
</tbody>
</table>

**Step 4:** store value 6 in the next slot of the new array.

<table>
<thead>
<tr>
<th>pyList</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: 4</td>
</tr>
<tr>
<td>1: 12</td>
</tr>
<tr>
<td>2: 2</td>
</tr>
<tr>
<td>3: 34</td>
</tr>
<tr>
<td>4: 17</td>
</tr>
<tr>
<td>5: 50</td>
</tr>
<tr>
<td>6: 18</td>
</tr>
<tr>
<td>7: 64</td>
</tr>
<tr>
<td>8: 6</td>
</tr>
<tr>
<td>9: 6</td>
</tr>
<tr>
<td>10: 6</td>
</tr>
<tr>
<td>11: 6</td>
</tr>
<tr>
<td>12: 6</td>
</tr>
<tr>
<td>13: 6</td>
</tr>
<tr>
<td>14: 6</td>
</tr>
<tr>
<td>15: 6</td>
</tr>
</tbody>
</table>
List: Extending

- The entire contents of a list can be appended to a second list.

```python
pyListA = [34, 12]
pyListB = [4, 6, 31, 9]
pyListA.extend(pyListB)
```
List: Inserting Items

- An item can be inserted anywhere within the list.
  
  `pyList.insert( 3, 79 )`
List: Removing Items

- An item can be removed from position of the list.

```python
pyList.pop(0)
```
List: Removing Items

- Removing the last item in the list.

```
pyList.pop()
```
List: Slices

- Slicing a list creates a new list from a contiguous subset of elements.

\[ \text{aSlice} = \text{pyList}[2:5] \]
2-D Arrays

- Arrays can be defined with multiple dimensions.
- Two-dimensional arrays:
  - organize the data in rows and columns
  - element access: \([i, j]\)
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 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 subscript (start at 0).
  - Once created, array size can not be changed.

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array2D( nrows, ncols )</td>
</tr>
<tr>
<td>numRows()</td>
</tr>
<tr>
<td>numCols()</td>
</tr>
<tr>
<td>clear( value )</td>
</tr>
<tr>
<td>getItem( i₁, i₂ )</td>
</tr>
<tr>
<td>setItem( i₁, i₂, value )</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>96</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>91</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>73</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>82</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>88</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>64</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>85</td>
<td>89</td>
<td></td>
</tr>
</tbody>
</table>
from array import Array2D

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

    # Extract the first two values; indicate the size of the array.
umExams = int( gradeFile.readline() )
numStudents = int( gradeFile.readline() )

    # Create the 2-D array to store the grades.
examGrades = Array2D( numStudents, numExams )

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

    # Close the text file.
gradeFile.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
```
2-D Array Example

```python
# Compute each student's average exam grade.
for i in range( numStudents ):
    total = 0
    for j in range( numExams ):
        total += examGrades[i,j]
    examAvg = total / numExams
    print( "%2d: %6.2f" % (i+1, examAvg) )
```
Implementing the 2-D Array

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

```python
class Array2D:
    def __init__(self, numRows, numCols):
        self._theRows = Array(numRows)
        for i in range(numRows):
            self._theRows[i] = Array(numCols)

    def numRows(self):
        return len(self._theRows)

    def numCols(self):
        return len(self._theRows[0])

    def clear(self, value):
        for row in self._theRows:
            row.clear(value)
```
2-D Array Implementation

- Subscript notation:
  \[ y = x[r, c] \quad x[r, c] = z \]

- Subscripts are passed to the methods as a tuple.
- Must verify the size of the tuple.
2-D Array Implementation

class Array2D:
    # ...

    def __getitem__(self, ndxTuple):
        assert len(ndxTuple) == 2, "Invalid number of array subscripts."
        row = ndxTuple[0]
        col = ndxTuple[1]
        assert row >= 0 and row < self.numRows() and col >= 0 and col < self.numCols(), "Array subscript out of range."
        the1dArray = self._theRows[row]
        return the1dArray[col]
class Array2D:
    # ...

    def __setitem__(self, ndxTuple, value):
        assert len(ndxTuple) == 2, "Invalid number of array subscripts."
        row = ndxTuple[0]
        col = ndxTuple[1]
        assert row >= 0 and row < self.numRows() and col >= 0 and col < self.numCols(), "Array subscript out of range."
        the1dArray = self._theRows[row]
        the1dArray[col] = value
Single Array Implementation

We can also use a single 1-D array.

• Entire contents is stored in one array.
• Common approach of many languages.
  • Row-major order
  • Column-major order
Row-Major Order

- The rows are stored sequentially, one at a time.

Physical storage of a 2-D array using row-major order.
Column-Major Order

- The columns are stored sequentially, one at a time.
Index Computation

- When accessing a 2-D array element, how do we find the actual element within the 1-D array?
  - Must calculate the offset within the 1-D array.
  - Ex: access element (2, 3) using row-major order.
Index Computation

- Given an $m \times n$ array, derive an index formula:
  - Element $(0, 0)$ maps to index $0$.
  - Element $(1, 0)$ maps to index $1n$.
  - Element $(2, 0)$ maps to index $2n$. 
Index Computation

- First element of the $i^{th}$ row maps to index $i \times n$.
- Row subscript indicates how many rows to skip.
Index Computation

- After skipping $i$ complete rows, then skip $j$ columns.
- Column subscript indicates how many rows to skip on the $i^{th}$ row.

- The formula for a 2D array is:
  \[
  \text{index}_2(i, j) = i \times n + j
  \]