CS 112 – Introduction to Computing II

Wayne Snyder
Computer Science Department
Boston University

Today:
Arrays (1D and 2D)
Methods
Program structure
Fields vs local variables

Next time: Program structure continued: Classes and objects; public vs private; static vs non-static objects;
Reading assignments will be posted on the web site!

Java Data Types: Array

The fundamental data type in Python is a list, which stores a list of values:

Python:

```python
In [1]: A = [2, 3, 4, 6, 7]
In [2]: S = ["hi", "there", "folks!"]
In [3]: X = [3.14, 3.1415, 3.141592]
In [4]: A[0]
Out[4]: 2
In [5]: S[3]
Traceback (most recent call last):
  File ".\example-5-2cfe081d48e3\", line 1, in <module>
  S[3]
IndexError: list index out of range
In [6]: for i in range(len(X)):
   ...:    print(X[i])
  3.14
  3.1415
  3.141592
In [7]:
```
Java Data Types: Array

The fundamental data type in Python is a list, which stores a list of values; in Java, a sequence of values is stored in an array:

Python:
```python
In [1]: A = [2, 3, 4, 6, 7]
In [2]: S = ['hi', 'there', 'folks!']
In [3]: X = [3.14, 3.1415, 3.141592]
In [4]: A[0]
Out[4]: 2
In [5]: S[3]
Traceback (most recent call last):
  File "<ipython-input-5-2cf1e81d48e3>", line 3
IndexError: list index out of range
```

Java:
```java
int[] A = { 2, 3, 4, 6, 7 };
String[] S = { "hi", "there", "folks!" };
double[] X = { 3.14, 3.1415, 3.141592 };
System.out.println( A[0] );
System.out.println( S[3] );
for ( int i = 0; i < X.length; ++i )
  System.out.println( X[i] );
```

Same: Access elements using [...]

Indices range from 0 ... length – 1
For loop used to run through entire array
Java Data Types: Array

The fundamental data type in Python is a list, which stores a list of values; in Java, a sequence of values is stored in an array:

<table>
<thead>
<tr>
<th>Python</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>In [1]: A = [2, 3, 4, 6, 7]</td>
<td>int[] A = {2, 3, 4, 6, 7};</td>
</tr>
<tr>
<td>In [2]: S = ['hi', 'there', 'folks!']</td>
<td>String S = {&quot;hi&quot;, &quot;there&quot;, &quot;folks&quot;};</td>
</tr>
<tr>
<td>In [4]: A[0]</td>
<td>System.out.println(A[0]);</td>
</tr>
<tr>
<td>Out[4]: 2</td>
<td>System.out.println(S[1]);</td>
</tr>
<tr>
<td>In [5]: S[3]</td>
<td></td>
</tr>
<tr>
<td>Traceback (most recent call last):</td>
<td></td>
</tr>
<tr>
<td>File &quot;&lt;python-input-5-2cf1e81d4e8e3=&quot;&quot;,</td>
<td>java.lang.ArrayIndexOutOfBoundsException: 3</td>
</tr>
<tr>
<td>S[3]</td>
<td>at SampleProgram.main(SampleProgram.java:19)</td>
</tr>
<tr>
<td>IndexError: list index out of range</td>
<td></td>
</tr>
</tbody>
</table>

In [6]: for i in range(len(X)):           |
|   ...:                                       |
|   print(X[i])                                |
| 3.14 3.1415 3.141592                         |

In [7]:

Same: Access elements using [...]  
Indices range from 0 ... length – 1  
For loop used to run through entire array  
Error if index is not in range.

Not same: Initialize array using { ... } instead of [ ... ]
Java Data Types: Array

The fundamental data type in Python is a list, which stores a list of values; in Java, a sequence of values is stored in an array:

Python:
In [1]: A = [2, 3, 4, 6, 7]
In [2]: S = ['hi', 'there', 'folks!']
In [3]: X = [3.14, 3.1415, 3.141592]
In [4]: A[0]
Out[4]: 2
In [5]: S[3]
Traceback (most recent call last):
  File "<ipython-input-5-2cf3e81d4be3>", line 1
    S[3]
IndexError: list index out of range

Java:
int[] A = { 2, 3, 4, 6, 7 };
String[] S = { "hi", "there", "folks!" };
double[] X = { 3.14, 3.1415, 3.141592 };
System.out.println( A[0] );
System.out.println( S[3] );
for( int i = 0; i < X.length; ++i )
    System.out.println( X[i] );

Not same: Initialize array using ( ... ) instead of [ ... ]
Must declare with array type with [ ] and all elements must be of same type.
Length of array is name-of-array.length

Java Data Types: Array

The fundamental data type in Python is a list, which stores a list of values; in Java, a sequence of values is stored in an array:

Python:
In [1]: A = [2, 3, 4, 6, 7]
In [2]: S = ['hi', 'there', 'folks!']
In [3]: X = [3.14, 3.1415, 3.141592]
In [4]: A[0]
Out[4]: 2
In [5]: S[3]
Traceback (most recent call last):
  File "<ipython-input-5-2cf3e81d4be3>", line 1
    S[3]
IndexError: list index out of range

Java:
int[] A = { 2, 3, 4, 6, 7 };
String[] S = { "hi", "there", "folks!" };
double[] X = { 3.14, 3.1415, 3.141592 };
System.out.println( A[0] );
System.out.println( S[3] );
for( int i = 0; i < X.length; ++i )
    System.out.println( X[i] );

Not same: Initialize array using ( ... ) instead of [ ... ]
Must declare with array type with [ ] and all elements must be of same type.
Length of array is name-of-array.length
Java Data Types: Array

**Same:** We can change the value in a particular location in the sequence:

**Python:**

```python
In [7]: A = [2, 3, 4, 6, 7]
In [8]: A
Out[8]: [2, 3, 4, 6, 7]
In [10]: A
Out[10]: [2, 3, 12, 6, 7]
```

**Java:**

```java
int[] A = {2, 3, 4, 6, 7};
System.out.println("n[ ");
for (int i = 0; i < A.length-1; ++i )
    System.out.println( A[i] + ", ");
System.out.println( A[A.length - 1] + "]");
System.out.println("n [ ");
for (int i = 0; i < A.length-1; ++i )
    System.out.println( A[i] + ", ");
System.out.println( A[A.length - 1] + "]");
```

> run sampleProgram
`[ 2, 3, 12, 6, 7]`

**Not same:** We can NOT change the size of the sequence at run time, as we can in Python:

**Python:**

```python
In [10]: A
Out[10]: [2, 3, 12, 6, 7]
In [11]: A.append(23)
```

**Java:**

```java
SIMPLY NOT POSSIBLE!
```

**Java:**

```java
In [12]: A
Out[12]: [2, 3, 12, 6, 7, 23]
```
The reason has to do with strong typing: in Java, we have to allocate memory for the array when we create it. We can’t change the size once this is done. To add to the array, we would have to redo the whole process.

The computer’s main memory is essentially a very big array holding 32- or 64-bit numbers which are interpreted as the various Java data types; each slot in the array has an index, called its “address.”

```java
int[] A = { 1, 1, 2, 3, 5, 8 }; // Some Fibonacci numbers
```

The array indices are translated into memory addresses: 0 -> 103
1 -> 104, etc.

Programming with Arrays in Java

We can create an array using an array literal when the array is declared:

```java
int[] A = { 1, 1, 2, 3, 5, 8 }
```

```java
for(int i = 0; i < A.length; ++i)
    System.out.print(A[i] + " ");
System.out.println();
```

Welcome to DrJava. Work: run SampleProgram
```
1 1 2 3 5 8
```

The array A:

0: 1
1: 1
2: 2
3: 3
4: 5
5: 8
We can create an array using an array literal when the array is declared:

```
int[] A = { 1, 2, 3, 5, 8 };
```

Or we can declare a “default array” which is initialized with the default types (int: 0, double: 0.0, boolean: false):

```
int[] A = new int[6];
```

We can then put values into this “default array” using assignments:

```
A[2] = 2;
```

```java
// Some Fibonacci numbers
int[] A = { 1, 1, 2, 3, 5, 8 };
for(int i = 0; i < A.length; ++i)
    System.out.print(A[i] + " ");
System.out.println();
```

The array A:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

```java
// Some Fibonacci numbers
int[] A = { 1, 2, 3, 5, 8 };
for(int i = 0; i < A.length; ++i)
    System.out.print(A[i] + " ");
System.out.println();
```

The array A:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>
Java Data Types: Array

Warning: You can NOT use an array literal to assign to an array, EXCEPT when you first declare it; this is why an array literal is called an "array initializer":

```java
// Some Fibonacci numbers
int[] A = new int[6];
A = { 1, 1, 2, 3, 5, 8 };```

Two dimensional arrays are implemented as "array of arrays" and can be visualized as a matrix of values; they are declared using an array initializer:

```
// Part of Pascal's Triangle
int[][] A = { { 1, 1, 1 }, { 1, 2, 3 }, { 1, 3, 6 }, { 1, 4, 10 } };

// print out the matrix by rows and columns
for (int row = 0; row < A.length; ++row) {
    for (int col = 0; col < A[row].length; ++col) {
        System.out.print(A[row][col] + " ");
    }
    System.out.println();
}
```
Java Data Types: Array

Or they can be created using explicit assignments:

```java
int[][] A = new int[4][3];
```

When the rows are given first, this is called row-major order.

---

Java Data Types: Array

Or they can be created using explicit assignments:

```java
int[][] A = new int[4][3];
```

// Part of Pascal's Triangle

```java
for(int col = 0; col < A[0].length; ++col) 
    A[0][col] = 1;

for(int row = 1; row < A.length; ++row) {
    A[row][0] = 1;
    for(int col = 1; col < A[0].length; ++col) 
}
```
One way to write a program is to put ALL code into the main method:

```java
public class PascalsTriangle {
    public static void main(String[] args) {
        // initialize empty array
        int[] A = new int[100][100];

        // fill in first row with all 1's
        for(int col = 0; col < A[0].length; ++col)
            A[0][col] = 1;

        // fill in rest of rows; first column 0, rest add from left and above
        for(int row = 1; row < A.length; ++row) {
            A[row][0] = 1;
            for(int col = 1; col < A[0].length; ++col)
        }

        // print out the matrix by rows and columns
        for(int row = 0; row < A.length; ++row) {
            for(int col = 0; col < A[0].length; ++col)
                System.out.print(A[row][col] + " ");
            System.out.println();
        }
    }
}
```
A better way is to use “Procedural Decomposition” and break the code into methods which are called by main (and possibly each other):

```java
public class PascalTriangle {
    static int[][] getTriangle(int n) {
        // Initialize empty array
        int[][] A = new int[n][n];
        // Fill in first row with all 1's
        A[0][0] = 1;
        // Fill in rest of rows; first column 1, rest odd from left and above
        for (int row = 1; row < n; ++row) {
            for (int col = 1; col < A[row].length; ++col) {
            }
        }
        return A;
    }
    static void printTriangle(int[][] A) {
        // Print out the matrix by rows and columns
        for (int row = 0; row < A.length; ++row) {
            for (int col = 0; col < A[row].length; ++col) {
                System.out.print(A[row][col] + " ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args) {
        for (int n = 1; n <= 3; ++n) {
            printTriangle(getTriangle(n));
            System.out.println();
        }
    }
}
```

**Advantages:**

1. Avoid duplicating code by cutting and pasting – and avoid the “multiple update problem” (when you fix one instance, you have to find all the other instances).

2. Better organization: you have labelled various parts of your code, so you can understand the whole thing better – you can just focus on coding one method at a time!

Anatomy of a method definition:

If a method does not return a value, then the return type is void, and the return statement (without an argument) is optional.
Methods, like operators, can be overloaded:

```java
public class OverloadTest {
    
    static int sum(int n, int m) {
        System.out.println("Calling sum(int n, int m)...");
        return n+m;
    }

    static double sum(double x, double y) {
        System.out.println("Calling sum(double x, double y)...");
        return x+y;
    }

    public static void main(String[] args) {
        System.out.println("\nTry sum(2,3)...");
        System.out.println("Calling sum(int n, int m)...");
        System.out.println("Returns 5");
        System.out.println("Try sum(2.3, 3.1)..." );
        System.out.println("Calling sum(double x, double y)... Returns 5.4");
        System.out.println("Try sum(2, 3.1)...");
        System.out.println("Calling sum(double x, double y)... Returns 5.1" );
    }
}
```

Widening Conversion

![Conversion Example](image-url)