CS 112 – Introduction to Computing II

Wayne Snyder
Computer Science Department
Boston University

Today: Java basics:

- Compilation vs Interpretation
- Values
- Variables
- Types
- Operators and Expressions

Next Time: Java Statements, conditionals, and loops

Reading assignments are posted on the web site!

Computer Science

Compilation vs Interpretation

Computer Science

Python is an example of an interpreted language; the primary workflow is to interact with the interpreter as a fancy calculator with lots of features:

```
#    Xe.append(nextExponential(50))
#
#Xn = []
#
#for i in range(N):
#    Xn.append(norm.rvs(50,10))


#drawNormalPlot(Xu)
#
#drawNormalPlot(Xp)
#
#drawNormalPlot(Xe)
#
#drawNormalPlot(Xn)

# Normality testing by binning
# Separate data into bins of width S * sigma on each side of mea
# and graph percentages of data that are within

def drawNormal(mu, var):
    plt.figure(figsize=(15,7))

    sigma = var**0.5

    Xn = [x for x in np.arange(int(mu-4*sigma),int(mu+4*sigma)+1
    Yn = [phi(mu,var,x) for x in Xn]
    plt.plot(Xn,Yn, 'r-')
    plt.xlabel('Range')
    plt.ylabel('Probability')
    plt.xlim(int(mu-4*sigma),int(mu+4*sigma))
#    plt.title(r'N())
```

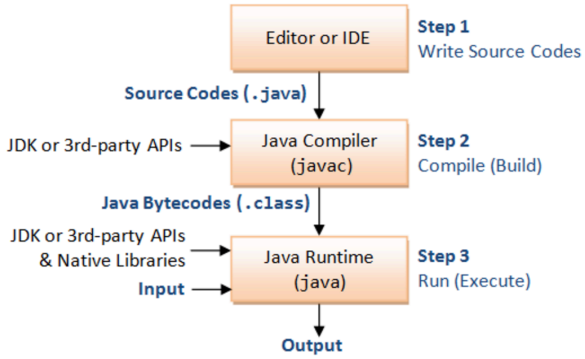
```
24994    67.21126
24995    69.50215
24996    64.54826
24997    64.69855
24998    67.52918
24999    68.87761
Name: Height, dtype: float64

In [106]: H.var()**0.5
Out[106]: 1.9016787712056042

In [107]: 3_4
  File "<ipython-input-107-c8029b172030>", line 1
    3_4
     ^
SyntaxError: invalid syntax


In [108]: 3 + 5
Out[108]: 8

In [109]: 2 + 9
Out[109]: 11

In [110]: H.var() ** 0.5
Out[110]: 1.9016787712056042

In [111]:
```

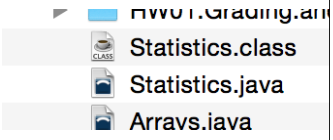
2

Compilation vs Interpretation

Computer Science

Java is an example of a language which is compiled; before running any code, your program (ending in .java) must be transformed into a lower-level form (an "executable file" ending in .class), and which is then passed to the interpreter, which runs the program and produces output:



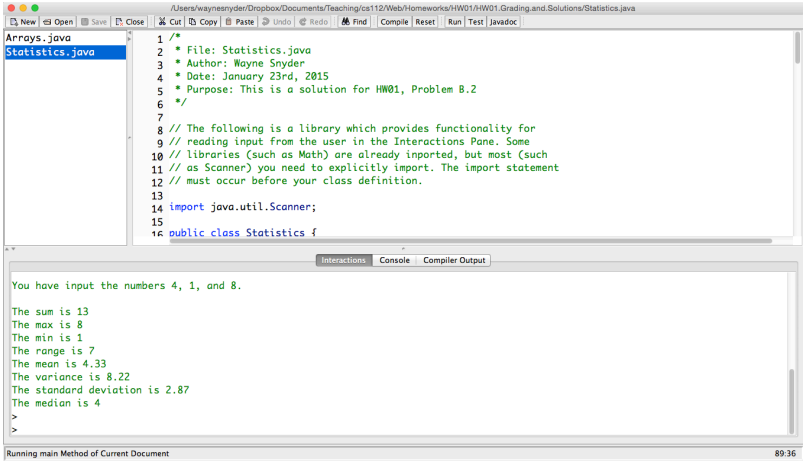


3

Compilation vs Interpretation

Computer Science

Dr. Java combines all the files necessary for your Java program with an editor, file manager, compiler, and an interpreter.



4

Java Basic Program Structure

Computer Science

Java programs are organized as classes (more on these next week!) stored in files with the suffix ".java", and with code written inside methods delimited by curly braces; each program must have a method called main, which contains the code that will be executed when you run your program:

Class name is same as file name

SampleProgram.java

Curly braces enclose classes and methods

```
public class SampleProgram {

   public static void main(String[] args) {

        //  Here is code that is executed when
        //  you run your program.

   }

}
```

Class

Method

5

Java Comments

Computer Science

Java has comments, exactly like Python, but with a different syntax:

Python:

""" and #

```
"""
File: Lab05.py
Author: Wayne Snyder
Purpose:  This collects together
discrete distributions.

"""

#  Import statements

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import numpy as np
import math
```

Java:

/* .... */ and //

```
/*
   File: Statistics.java
   Author: Wayne Snyder
   Date: January 23rd, 2015
   Purpose: This is a solution for
 */

// The following is a library whic
// reading input from the user in
// libraries (such as Math) are al
// as Scanner) you need to explici
// must occur before your class de
```

Java Values and Types

Computer Science

Java is a strongly-typed language supporting the following types of values:

Java Primitive Data Types

Type	Values	Default	Size	Range
byte	signed integers	0	8 bits	-128 to 127
short	signed integers	0	16 bits	-32768 to 32767
int	signed integers	0	32 bits	-2147483648 to 2147483647
long	signed integers	0	64 bits	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	0.0	32 bits	+/-1.4E-45 to +/-3.4028235E+38, +/-infinity, +/-0, NAN
double	IEEE 754 floating point	0.0	64 bits	+/-4.9E-324 to +/-1.7976931348623157E+308, +/-infinity, +/-0, NaN
char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF
boolean	true, false	false	1 bit used in 32 bit integer	NA

String “hi there” “”

Java Values and Types

Computer Science

Literal values are similar to Python:

int 4 -5

double 3.4 -2.34e10

char ‘a’ ‘\n’ ‘\t’ // single quotes for chars

boolean true false // note lower case

String “hi there” // must use double quotes

Note that String is capitalized

Java Values and Types

Computer Science

Python is "weakly typed": values have types but variables do not; variables are just names for any value you want and can be reused for any values; the only errors occur when variables have not yet been assigned values:

```
In [123]: X = 5

In [124]: X
Out[124]: 5

In [125]: X = 4.5

In [126]: X = "hi"

In [127]: X
Out[127]: 'hi'

In [128]: Z
Traceback (most recent call last):

  File "<ipython-input-128-41ff0912a07f>", line 1, in <module>
    Z

NameError: name 'Z' is not defined
```

9

Java Values and Types

Computer Science

Java is strongly-typed in that

All variables must be declared with a type before being used and can only be used for that type of value:

```
int x;                  // declare x to be int
x = 4;                  // assign value to x
System.out.print(x);
double y = 3.4;         // combine declaration and assignment
double z = x + y;
System.out.print(z);
```

Note: Java statements end in semicolon

10

Java Values and Types

Computer Science

During compilation, the types are checked and errors will be reported with line numbers and terse explanations:

```
1  public class SampleProgram {
2
3
4    public static void main(String[] args) {
5
6      int x;                  // declare x to be int
7      x = 3.4;                  // assign value to x
8      System.out.println(x);
9      double y = 3.4;         // combine declaration and assignment
10     double z = x + y;
11     System.out.println(z);
12
13   }
14
15 }
```

Interactions | Console | Compiler Output

```
1 error found:
File: /Users/waynesnyder/Dropbox/Documents/Teaching/cs112/SampleProgram.java  [line: 7]
Error: /Users/waynesnyder/Dropbox/Documents/Teaching/cs112/SampleProgram.java:7:
possible loss of precision
found   : double
required: int
```


Java Values and Types

Computer Science

This might seem unduly rigid, but the philosophy of strongly-typed languages is that specifying types makes programmers more careful about variables, and bugs and errors can be found during compilation, not when the program is running.

Values can be converted from one type to another implicitly or explicitly:

Widening Conversions (implicit):

Narrow types (less information) ⟶ Wider types (more information)

Example: `int` ⟶ `double`

```
double  x;
x  = 4;   // 4 is widened to 4.0 and then assigned
```

No error!

Java Values and Types

This might seem unduly rigid, but the philosophy of strongly-typed languages is that specifying types makes programmers more careful about variables, and bugs and errors can be found during compilation, not when the program is running.

Values can be converted from one type to another implicitly or explicitly:

Widening Conversions (implicit):

Narrow types (less information) ⟶ Wider types (more information)

Example: `int` ⟶ `double`

```
double  x;
x  = 4;   // 4 is widened to 4.0 and then assigned
```

Example 2: `char` ⟶ `int`

```
int  x;
x  = 'A';   // 'a' is converted to its ASCII
            //  value 65 and assigned to x
```

13

Java Values and Types

This might seem unduly rigid, but the philosophy of strongly-typed languages is that specifying types makes programmers more careful about variables, and bugs and errors can be found during compilation, not when the program is running.

Values can be converted from one type to another implicitly or explicitly:

Narrowing Conversions (you must specify a cast or else get an error):

Wider types (more information) ⟶ Narrower types (less information)

Example: `double` ⟶ `int`

```
int  x;
x  = 4.5;
```

Error!

14

Java Values and Types

Computer Science

This might seem unduly rigid, but the philosophy of strongly-typed languages is that specifying types makes programmers more careful about variables, and bugs and errors can be found during compilation, not when the program is running.

Values can be converted from one type to another implicitly or explicitly:

Narrowing Conversions (you must specify or else get an error):

Wider types (more information) → Narrower types (less information)

Example: `double` → `int`

```
int  x;
x  = 4.5;
```

Error!

Must explicitly tell Java to truncate the double value to an integer:

```
int  x;
x  = (int) 4.5;     // x gets 4
```

Cast

15

Java Values and Types

Computer Science

Narrowing →

String

boolean

Widening ←

16

Java Operators

Computer Science

Arithmetic operators are almost exactly the same as in Python:

Same:

+	addition	+=
-	subtraction	-=
*	multiplication	*=
%	modulus	%=
==	equals	
!=	not equal	
<	less	
<=	less or equal	
>	greater	
>=	greater or equal	

17

Java Operators

Computer Science

Arithmetic operators are almost exactly the same as in Python:

BUT some are different:

Python:	Java:
not	!
and	&&
or	||

In both languages, and and or are lazy:

(false and X) => false (without evaluating X)
(true or X) => true (without evaluating X)

18

Java Operators

Computer Science

Division is significantly different:

Python: two different division operators:

/ floating-point division /=

// integer division

Java: division operator is "overloaded":

/ returns an int if both operands are ints, otherwise returns double:

5 / 2 => 2 5.0 / 2 => 2.5 5.0 / 2.0 => 2.5

5 / (double) 2 => 2.5

19

Java Operators

Computer Science

There is no exponentiation operator in Java:

Python:

x ** 2 x squared

Java: have to use explicit math functions:

Math.pow(x,2) => returns double

Math.sqrt(2)

20

Java Operators

Computer Science

Finally, Java has several useful increment and decrement operators which Python lacks:

Java:

```
++x;   x++ ;      // same as x = x + 1    or   x += 1
--x;   x-- ;      //  same as x = x – 1    or   x -=  1
```

BUT these can be used as arithmetic expressions:

++x	has the value AFTER adding 1
x++	has the value BEFORE adding 1

```
int x = 4;          x = 4
int y = ++x;        y = 4   x = 5
int z = x++ ;       y = 4   z = 5    x = 6
```

21