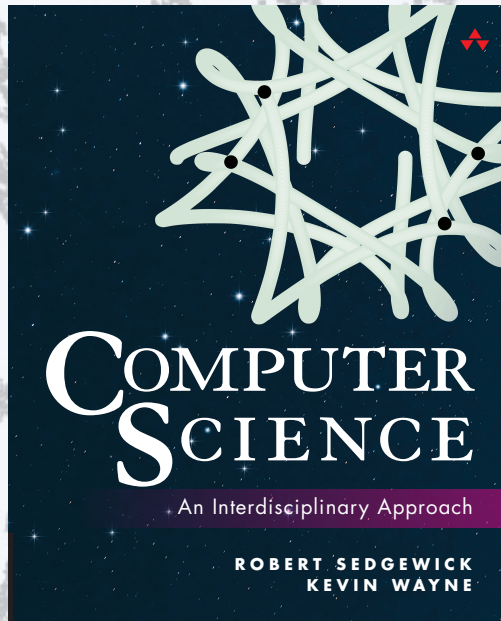


COMPUTER SCIENCE
SEDEGWICK / WAYNE
PART I: PROGRAMMING IN JAVA



<http://introcs.cs.princeton.edu>

10. Programming Languages

10. Programming Languages

- **Popular languages**
- Java in context
- Object-oriented programming
- Type checking
- Functional programming

The Tower of Babel

A story about the origins of multiple languages

- [After the flood]
“The whole earth was of one language and one speech.”
- They built a city and tower at Babel, believing that with a single language, people will be able to do anything they imagine.
- Yahweh disagrees and
“confounds the language of all the earth”
- Why?

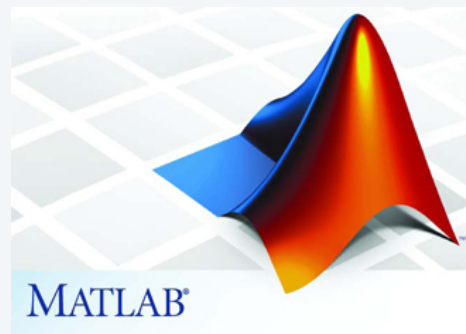
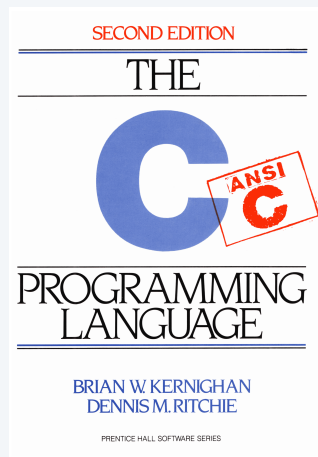


Proliferation of cultural differences (and multiple languages) is one basis of civilization.

Several ways to solve a transportation problem



Several ways to solve a programming problem



Java

You can write Java code.

3-sum

- Read int values from StdIn.
- Print triples that sum to 0.
- [See *Performance* lecture]

ThreeSum.java

```
public class ThreeSum
{
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);
        int[] a = new int[N];
        for (int i = 0; i < N; i++)
            a[i] = StdIn.readInt();
        for (int i = 0; i < N; i++)
            for (int j = i+1; j < N; j++)
                for (int k = j+1; k < N; k++)
                    if (a[i] + a[j] + a[k] == 0)
                        StdOut.println(a[i] + " " + a[j] + " " + a[k]);
    }
}
```



```
% more 8ints.txt
30 -30 -20 -10 40 0 10 5
% javac ThreeSum.java
% java ThreeSum 8 < 8ints.txt
30 -30 0
30 -20 -10
-30 -10 40
-10 0 10
```

C

You can *also* write C code.

Noticeable differences

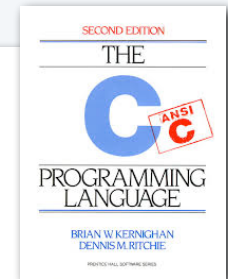
- library conventions
- array creation idiom
- standard input idiom
- pointer manipulation (stay tuned for Part 2)

ThreeSum.c

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    int N = atoi(argv[1]);
    int *a = malloc(N*sizeof(int));
    int i, j, k;
    for (i = 0; i < N; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < N; i++)
        for (j = i+1; j < N; j++)
            for (k = j+1; k < N; k++)
                if (a[i] + a[j] + a[k] == 0)
                    printf("%d %d %d\n", a[i], a[j], a[k]);
}
```

```
% more 8ints.txt
30 -30 -20 -10 40 0 10 5

% cc ThreeSum.c
% a.out 8 < 8ints.txt
30 -30 0
30 -20 -10
-30 -10 40
-10 0 10
```



A big difference between C and Java (there are many!)

NO DATA ABSTRACTION

- No objects in C.
- A C program is a set of static methods.

C++ (Stroustrup 1989)

- Adds data abstraction to C.
- "C with classes".
- Embodies many OOP innovations.



“There are only two kinds of programming languages: those people always [gripe] about and those nobody uses.”

– Bjarne Stroustrup



C++

You can *also* write C++ code.

Example 1. Use C++ like C.

Noticeable differences

- library conventions
- standard input idiom
- standard *output* idiom
- pointer manipulation
(stay tuned for Part 2)

ThreeSum.cxx

```
#include <iostream>
#include <cstdlib>
int main(int argc, char *argv[])
{
    int N = atoi(argv[1]);
    int *a = new int[N];
    int i, j, k;
    for (i = 0; i < N; i++)
        std::cin >> a[i];
    for (i = 0; i < N; i++)
        for (j = i+1; j < N; j++)
            for (k = j+1; k < N; k++)
                if (a[i] + a[j] + a[k] == 0)
                    std::cout << a[i] << " " << a[j] << " " << a[k] << std::endl;
}
```



```
% cpp ThreeSum.cxx
% a.out 8 < 8ints.txt
30 -30 0
30 -20 -10
-30 -10 40
-10 0 10
```

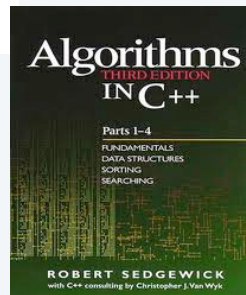
C++

You can *also* write C++ code.

Example 2. Use C++ like Java to implement the symbol table ADT (details in Part 2).

Challenges

- libraries/idioms
- **pointer manipulation**
- templates (generics)



1990

BST.cxx

```
template <class Item, class Key>
class ST
{
    private:
        struct node
        { Item item; node *left, *right;
          node(Item x)
          { item = x; left = 0; right = 0;}
        };
        typedef node *link;
        link head;

        Item searchR(link x, Key key)
        { if (x == 0) return 0;
          Key t = x->item.key();
          if (key == t) return x->item;
          if (key < t) return searchR(x->left, key);
          else return searchR(x->right, key);
        }
        ...
    }
}
```



A big difference between C/C++ and Java (there are many!)

C/C++: YOU are responsible for memory allocation

- Programs manipulate pointers.
- System provides memory allocation library.
- Programs explicitly call methods that “allocate” and “free” memory for objects.
- Pitfall: “memory leaks”.

Java: Automatic "garbage collection"

- System keeps track of references.
- System manages memory use.
- System reclaims memory that is no longer accessible from your program.

Fundamental challenge. C/C++ code that manipulates pointers is inherently *unsafe*.

C code that reuses an array name

```
double arr[] = calloc(5, sizeof(double));  
...  
free(arr);  
arr = calloc(10, sizeof(double));
```



Java code that reuses an array name

```
double[] arr = new double[5];  
...  
arr = new double[10];
```

Python

You can *also* use Python.

Example 1. Use Python like a calculator.



```
% python
Python 2.7.1 (r271:86832, Jun 16 2011, 16:59:05)
Type "help" for more information.
>>> 2+2
4
>>> (1 + sqrt(5))/2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
>>> import math
>>> (1 + math.sqrt(5))/2
1.618033988749895
```



Throw out
your calculator
(please)

Python

You can *also* write Python code.

Example 2. Use Python like Java.

Noticeable differences

- No braces (indents instead).
- No type declarations.
- Array creation idiom.
- I/O idioms.
- for (iterable) idiom.

threesum.py



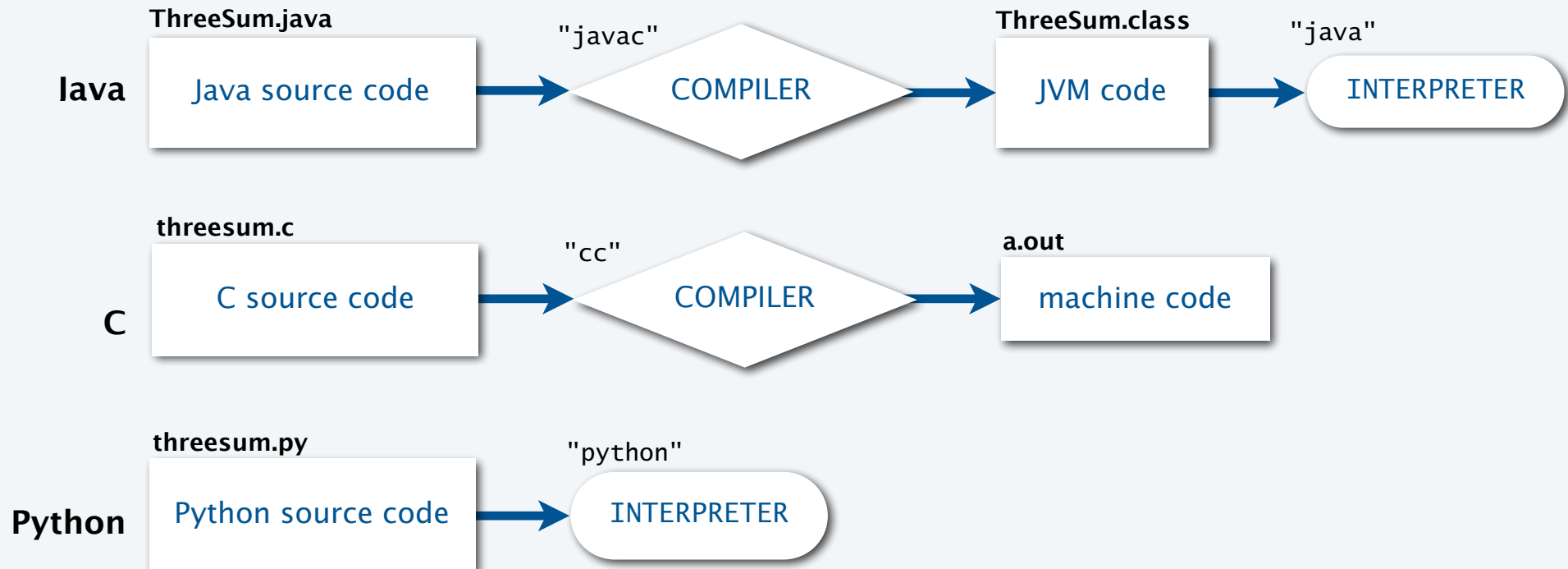
```
import sys
import stdio ← our stdio library (like the Java one)
N = int(sys.argv[1])
a = [0]*N
for i in range(N): ← range(8) is [0,1,2,3,4,5,6,7]
    a[i] = stdio.readInt() ← could use Python's sys.stdin
for i in range(N):
    for j in range(i+1, N):
        for k in range(j+1, N):
            if (a[i] + a[j] + a[k]) == 0: ← could use stdio
                print '%d %d %d\n', a[i], a[j], a[k]
```

```
% python threesum.py 8 < 8ints.txt
30 -30 0
30 -20 -10
-30 -10 40
-10 0 10
```

Compilation vs. Interpretation

Definition. A **compiler** translates your entire program to (virtual) machine code.

Definition. An **interpreter** simulates the operation of a (virtual) machine running your code.



A big difference between Python and C/C++/Java (there are many!)

NO COMPILE-TIME TYPE CHECKING

- No need to declare types of variables.
- System checks for type errors *only* at RUN time.

Implications

- Easier to write small programs.
- More difficult to debug large programs.



Using Python for large problems is playing with fire.

Typical (nightmare) scenario

- Scientist/programmer makes a small type error in a big program.
- Program runs for hours or days (because Python might be 10-100 times slower than Java).
- Program crashes without writing results.

Reasonable approaches

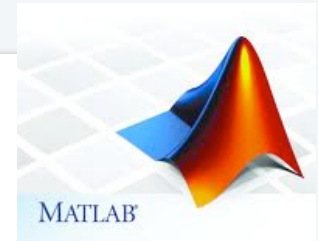
- Throw out your calculator; use Python.
- Prototype in Python, then convert to Java for "production" use.

Matlab

You can write Matlab code.

Example 1. Use Matlab like Java.

```
fileID = fopen('8ints.txt','r');
a = fscanf(fileID, '%d');
N = length(a);
for i = 1:N
    for j = i+1:N
        for k = j+1:N
            if (a(i) + a(j) + a(k)) == 0:
                fprintf('%4d %4d %4d\n', a(i), a(j), a(k))
            end
        end
    end
end
end
```



Example 2 (more typical). Use Matlab for matrix processing.

```
A = [1 3 5; 2 4 7]
B = [-5 8; 3 9; 4 0]
C = A*B
C =
    24    35
    30    52
```

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 7 \end{pmatrix} * \begin{pmatrix} -5 & 8 \\ 3 & 9 \\ 4 & 0 \end{pmatrix} = \begin{pmatrix} 24 & 35 \\ 30 & 52 \end{pmatrix}$$

Big differences between Matlab and C/C++/Java/Python (there are many!)

1. MATLAB IS NOT FREE.
2. Most Matlab programmers use only ONE data type (matrix).



Example. Matlab code " $i = 0$ " means

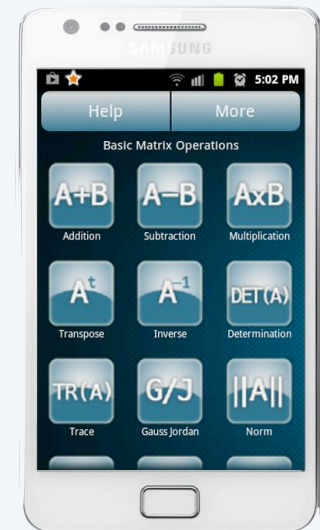
"redefine the value of the complex number i to be a 1-by-1 matrix whose entry is 0"

Notes

- Matlab is written in Java.
- The Java compiler and interpreters are written in C.
[Modern C compilers are written in C.]
- Good matrix libraries are available for C/C++/Java/Python.

Reasonable approaches

- Use Matlab as a "matrix calculator" and data analysis (if you own it).
- Convert to or use Java/C/C++ if you want to do anything else.



Matrix calculator
Android app

COMPUTER SCIENCE

SE D G E W I C K / W A Y N E

PART I: PROGRAMMING IN JAVA

Image sources

http://commons.wikimedia.org/wikipedia/commons/a/a8/Marten_van_Valckenborch_the_Elder_-_The_Tower_of_Babel_-_Google_Art_Project.jpg

http://en.wikipedia.org/wiki/Stealth_aircraft#/media/File:F-117_Nighthawk_Front.jpg

http://commons.wikimedia.org/wiki/File:Boeing_787_Dreamliner_N787BX.jpg

<http://commons.wikimedia.org/wiki/File:Bjarne-stroustrup.jpg>

10. Programming Languages

- Popular languages
- **Java in context**
- Object-oriented programming
- Type checking
- Functional programming

Why Java? (revisited)

Java features

- Widely used.
- Widely available.
- Continuously under development since early 1990s.
- Embraces full set of modern abstractions.
- Variety of automatic checks for mistakes in programs.

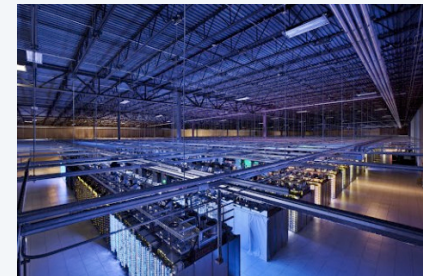


James Gosling






Java economy

- Mars rover.
- Cell phones.
- Blu-ray Disc.
- Web servers.
- Medical devices.
- Supercomputing.
- ...

← millions of developers
billions of devices



Why do we use Java in this course?

<i>language</i>	<i>widely used</i>	<i>widely available</i>	<i>full set of modern abstractions</i>	<i>modern libraries and systems</i>	<i>automatic checks for bugs</i>
	✓	✓	✗	✗	✓ ✗
	✓	✓	✓	maybe	✓ ✗
	✓	✓	✓	✓	✓
	✓	\$	maybe*	✓	✗
	✓	✓	maybe	✓	✗

not memory leaks

* OOP recently added but not embraced by most users

Why learn another programming language?

Good reasons to learn a programming language

- Offers something new.
- Need to interface with co-workers.
- Better than Java for the application at hand.
- Provides an intellectual challenge
- Opportunity to learn something about computation.
- Introduces a new programming style.



Something new: a few examples

1960s: Assembly language

- symbolic names
- relocatable code

IBM System/360 Reference Data

INSTRUCTION	OP	MODE	OPERANDS
ADD	11	1,2	R1,R2
ADDI	12	1,2	R1,R2,DISP
AND	13	1,2	R1,R2
ANDI	14	1,2	R1,R2,DISP
ASR	15	1,2	R1,R2
ASRI	16	1,2	R1,R2,DISP
BAL	17	1,2	R1,R2
BALR	18	1,2	R1,R2
BR	19	1,2	R1,R2
BRB	20	1,2	R1,R2
BRH	21	1,2	R1,R2
BRL	22	1,2	R1,R2
BRRL	23	1,2	R1,R2
BRBR	24	1,2	R1,R2
BRBL	25	1,2	R1,R2
BRBRH	26	1,2	R1,R2
BRBRHL	27	1,2	R1,R2
BRBLH	28	1,2	R1,R2
BRBLHL	29	1,2	R1,R2
BRHLH	30	1,2	R1,R2
BRHLHL	31	1,2	R1,R2
BRHLHLH	32	1,2	R1,R2
BRHLHLHL	33	1,2	R1,R2
BRHLHLHLH	34	1,2	R1,R2
BRHLHLHLHL	35	1,2	R1,R2
BRHLHLHLHLH	36	1,2	R1,R2
BRHLHLHLHLHL	37	1,2	R1,R2
BRHLHLHLHLHLH	38	1,2	R1,R2
BRHLHLHLHLHLHL	39	1,2	R1,R2
BRHLHLHLHLHLHLH	40	1,2	R1,R2
BRHLHLHLHLHLHLHL	41	1,2	R1,R2
BRHLHLHLHLHLHLHLH	42	1,2	R1,R2
BRHLHLHLHLHLHLHLHL	43	1,2	R1,R2
BRHLHLHLHLHLHLHLHLH	44	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHL	45	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLH	46	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHL	47	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLH	48	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHL	49	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLH	50	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHL	51	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLH	52	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHL	53	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLH	54	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHL	55	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLH	56	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHL	57	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLH	58	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHL	59	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLH	60	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHL	61	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLH	62	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHL	63	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLH	64	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHL	65	1,2	R1,R2
BRHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLHLH	66	1,2	R1,R2
BRHL	67	1,2	R1,R2
BRHLH	68	1,2	R1,R2
BRHL	69	1,2	R1,R2
BRHLH	70	1,2	R1,R2
BRHL	71	1,2	R1,R2
BRHLH	72	1,2	R1,R2
BRHL	73	1,2	R1,R2
BRHLH	74	1,2	R1,R2
BRHL	75	1,2	R1,R2
BRHLH	76	1,2	R1,R2
BRHL	77	1,2	R1,R2
BRHLH	78	1,2	R1,R2
BRHL	79	1,2	R1,R2
BRHLH	80	1,2	R1,R2
BRHL	81	1,2	R1,R2
BRHLH	82	1,2	R1,R2
BRHL	83	1,2	R1,R2
BRHLH	84	1,2	R1,R2
BRHL	85	1,2	R1,R2
BRHLH	86	1,2	R1,R2
BRHL	87	1,2	R1,R2
BRHLH	88	1,2	R1,R2
BRHL	89	1,2	R1,R2
BRHLH	90	1,2	R1,R2
BRHL	91	1,2	R1,R2
BRHLH	92	1,2	R1,R2
BRHL	93	1,2	R1,R2
BRHLH	94	1,2	R1,R2
BRHL	95	1,2	R1,R2
BRHLH	96	1,2	R1,R2
BRHL	97	1,2	R1,R2
BRHL	98	1,2	R1,R2
BRHL	99	1,2	R1,R2

Branch and Store (e)	BAS	4D	RX	R1,D2(X2,B2)
Branch on Condition	BCR	07	RR	M1,R2
Branch on Condition	BC	47	RX	M1,D2(X2,B2)
Branch on Count	BCTR	06	RR	R1,R2
Branch on Count	BCT	46	RX	R1,D2(X2,B2)
Branch on Index High	BXH	86	RS	R1,R3,D2(B2)
Branch on Index Low or Equal	BXLE	87	RS	R1,R3,D2(B2)
Compare (c)	CR	19	RR	R1,R2
Compare (c)	C	59	RX	R1,D2(X2,B2)
Compare Decimal (c,d)	CP	F9	SS	D1(L1,B1),D2(L2,B2)
Compare Halfword (c)	CH	49	RX	R1,D2(X2,B2)
Compare Logical (c)	CLR	15	RR	R1,R2
Compare Logical (c)	CL	55	RX	R1,D2(X2,B2)
Compare Logical (c)	CLC	D5	SS	D1(L,B1),D2(B2)
Compare Logical (c)	CLI	95	SI	D1(B1),I2
Convert to Binary	CVB	4F	RX	R1,D2(X2,B2)
Convert to Decimal	CVD	4E	RX	R1,D2(X2,B2)
Convert to Decimal	CV	83	SI	R1,R2
Convert to Decimal	CV	1D	RR	R1,R2

1970s: C

- “high-level” language
- statements, conditionals, loops
- machine-independent code
- functions and libraries



1990s: C++/Java

- data abstraction (OOP)
- extensive libraries





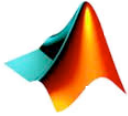






2000s: AJAX/PHP/Python/Ruby/Flash

- scripting
- libraries for web development



Programming styles

<i>style</i>	<i>execution model</i>	<i>examples</i>
procedural	step-by-step instruction execution usually compiled	
scripted	step-by-step command execution usually interpreted	 python 
special-purpose	optimized around certain data types	  MATLAB
object-oriented	focus on objects that do things	 Java 
functional	treats computation as the evaluation of functions, avoiding side effects and mutable types	 OCaml  HASKELL



COMPUTER SCIENCE
S E D G E W I C K / W A Y N E
PART I: PROGRAMMING IN JAVA

Image sources

http://commons.wikimedia.org/wiki/File:James_Gosling_2005.jpg

16. Programming Languages

- Popular languages
- Java in context
- **Object-oriented programming**
- Type checking
- Functional programming

Object-oriented programming

A different philosophy

- Software is a simulation of the real world.
- We know (approximately) how the real world works.
- Design software to (approximately) model the real world.



Procedural programming

- Tell the computer to do this. ← VERB-oriented
- Tell the computer to do that.

Object oriented programming (OOP)

- Programming paradigm based on data types.
- Identify things that are part of the problem domain or solution.
- Things in the world know something: instance variables.
- Things in the world do something: methods.



← NOUN-oriented

Why OOP?

Essential questions

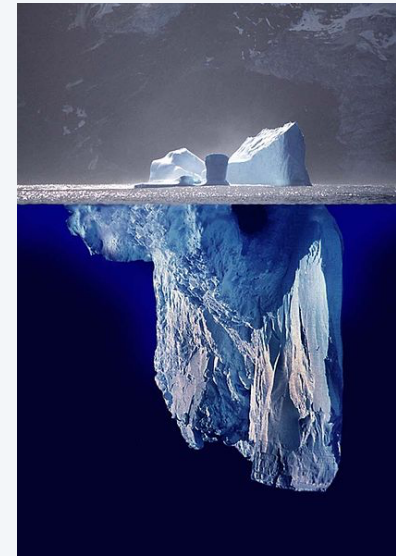
- Is my program easy to write?
- Is it easy to find errors and maintain my program?
- Is it correct and efficient?

Essential features of OOP

- **Encapsulation** to hide information to make programs robust.
- **Type checking** to avoid and find errors in programs.
- **Libraries** to reuse code.
- **Immutability** to guarantee stability of program data.

Does OOP make it easy to write and maintain correct and efficient programs?

- Difficult for you to know, because you haven't programmed in another style.
- Ongoing debate among experts intensifies as time goes on.
- Meanwhile, millions of people (including YOU) are reaping the benefits of OOP.



Warning: OOP involves deep, difficult, and controversial issues. Further study may be fruitful, but is likely to raise more questions than answers!

OOP pioneers

Kristen Nygaard and O.J. Dahl. (U. Oslo 1960s)

- Invented OOP for simulation.
- Developed Simula programming language.
- Studied formal reasoning about OO programs.



Kristen Nygaard and O.J. Dahl
2001 Turing Award

Alan Kay. (Xerox PARC 1970s)

- Developed Smalltalk programming language.
- Promoted OOP for widespread use.
- Computer science visionary.



Alan Kay
2003 Turing Award

Barbara Liskov. (MIT 1970s)

- Developed CLU programming language.
- Pioneered focus on data abstraction.
- Research provided basis for Java, C++, ...



Barbara Liskov
2008 Turing Award

Alan Kay: a computer science visionary

1970s



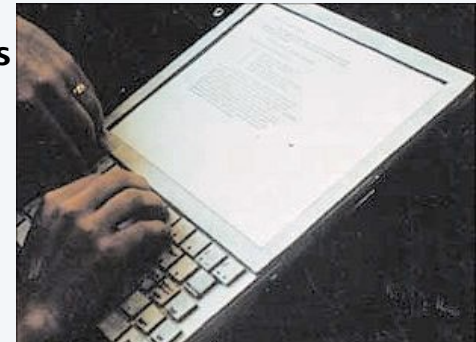
Typical "mainframe" computer: IBM 360/50

1970s



First PC: Xerox Alto

1970s

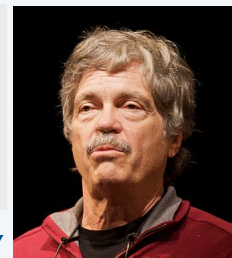


Alan Kay's vision for the future
Dynabook prototype
Key feature: **OOP software** (Smalltalk)

"The best way to predict the future is to invent it." (1971)
"The computer revolution hasn't happened yet." (1997)

↑
Still relevant today!

– Alan Kay



2010s



Modern personal computer
MacBook Air
Key feature: **OOP software** (Objective C)

COMPUTER SCIENCE
S E D G E W I C K / W A Y N E
PART I: PROGRAMMING IN JAVA

Image sources

http://commons.wikimedia.org/wiki/File:James_Gosling_2005.jpg

<http://commons.wikimedia.org/wiki/File:Iceberg.jpg>

[http://en.wikipedia.org/wiki/Alan_Kay#/media/File:Alan_Kay_\(3097597186\).jpg](http://en.wikipedia.org/wiki/Alan_Kay#/media/File:Alan_Kay_(3097597186).jpg)

<http://newsoffice.mit.edu/2009/turing-liskov-0310>

10. Programming Languages

- Popular languages
- Java in context
- Object-oriented programming
- **Type checking**
- Functional programming

Type checking

Static (compile-time) type checking (e.g. Java)

- All variables have declared types.
- System checks for type errors at *compile* time.

Dynamic (run-time) type checking (e.g. Python)

- Values, not variables, have defined types.
- System checks for type errors at *run* time.

← Java also does some run-time checking

Q. Which is best?

A. Religious wars ongoing!

- Static typing worth the trouble?
- Compiled code more efficient?
- Type-checked code more reliable?
- Advanced features (e.g. generics) too difficult to use with static typing?



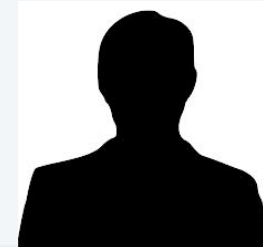
Example: Diametrically opposed points of view

Issue. Type checking or automated program testing?



“ Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence.”

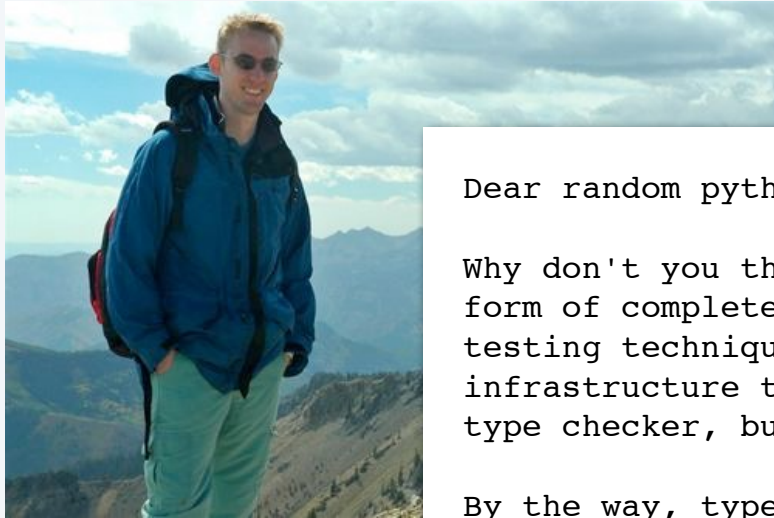
— Edsger Dijkstra (1969)



“ Since static type checking can't cover all possibilities, you will need automated testing. Once you have automated testing, static type checking is redundant.”

— Python blogger (2009)

A letter from Dave Walker



Dear random python blogger:

Why don't you think of static type checking as a complementary form of completely automated testing to augment your other testing techniques? I actually don't know of any other testing infrastructure that is as automated, fast and responsive as a type checker, but I'd be happy to learn.

By the way, type checking is a special kind of testing that scales perfectly to software of arbitrary size because it checks that the composition of 2 modules is ok based only on their interfaces, without re-examining their implementations.

Conventional testing does not scale the same way. Also, did you know that type checking is capable of guaranteeing the absence of certain classes of bugs? That is particularly important if you want your system to be secure. Python can't do that.

dpw (in mail to rs)

Programming folklore: Hungarian type system

Early programming languages had little support for types.

Hungarian type system (Charles Simonyi, 1970s)

- Encode type in first few characters of variable name.
- 8 character limit? Leave out the vowels, truncate.

Example. arru8Fbn

array of 8-bit integers (unsigned) variable name short for Fibonacci



Charles Simonyi
Introduced OOP to Microsoft

An advantage: Can “type check” while reading code.

A disadvantage: shrt vwl-lss vrbl nms.

Used in first version of Microsoft Word (and extensively before that time).

Lesson. Type-checking has *always* been important in large software systems.



COMPUTER SCIENCE
SE D G E W I C K / W A Y N E
PART I: PROGRAMMING IN JAVA

Image sources

http://en.wikipedia.org/wiki/Thirty_Years'_War#/media/File:Schlacht_am_Weißen_Berg_C-K_063.jpg

http://en.wikipedia.org/wiki/Edsger_W._Dijkstra#/media/File:Edsger_Wybe_Dijkstra.jpg

http://en.wikipedia.org/wiki/Charles_Simonyi

10. Programming Languages

- Popular languages
- Java in context
- Object-oriented programming
- Type checking
- **Functional programming**

Functional programming

Q. Why can't we use functions as arguments in Java programs?

A. We can, in Java 8! (Doing so is cumbersome in earlier versions.)


Functional programming treats computation as the evaluation of functions.

- Avoids side effects and mutable types.
- On-demand execution model.
- "What" rather than "how".

Advantages of functional programming

- Functions are first-class entities
(can be arguments and return values of other functions or stored as data).
- Often leads to more compact code than alternatives.
- More easily admits reasoning about correctness of code .
- More easily supports concurrency (programming on multiple processors).

Familiar examples:
differentiation,
integration



HASKELL



Java 8

Scala

OCaml

python



ERLANG

Functional programming example

A Python program that prints a tables of squares.

squares.py

a function that returns the
square of its argument

```
def square(x):  
    return x*x
```

a function that takes a *function* and
a range as arguments and prints a
table of values of the function for
every value in the range

```
def table(f, sequence):  
    for x in sequence:  
        print x,  
        print f(x)
```

print a table of the
squares of the numbers
from 0 to 9

```
table (square, range(10))
```

```
% python squares.py  
0 0  
1 1  
2 4  
3 9  
4 16  
5 25  
6 36  
7 49  
8 64  
9 81
```

Functions that operate on functions

Functions as first-class objects admit compact code for powerful operations.

Example 1. The **MAP** operation takes a function and a list as arguments.

$\text{MAP}(f, \text{sequence})$ is the result of replacing every x in sequence by $f(x)$.

map.py

```
def square(x):  
    return x*x  
  
def odd(x):  
    return 2*x + 1  
  
print map (odd, range(10))  
print map (square, range(10))
```

a function that returns the square of its argument →

print the squares of the numbers from 0 to 9 →

```
% python map.py  
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Functions that operate on functions

Functions as first-class objects admit compact code for powerful operations.

Example 2. The **REDUCE** operation takes a function and a list as arguments.

$\text{REDUCE}(f, L)$ is $f(\text{car}(L), \text{REDUCE}(f, \text{cdr}(L)))$.

↑
first entry on L

↑
all but first entry on L

python does it from
the right end

reduce.py

```
def plus(x, y):  
    return x + y  
  
def odd(x):  
    return 2*x + 1  
  
print reduce(plus, map(odd, range(10)))
```

```
% python reduce.py  
100
```

```
reduce(plus, [1, 3, 5, 7, 9, 11, 13, 15, 17, 19])  
= reduce(plus, [1, 3, 5, 7, 9, 11, 13, 15, 17]) + 19  
= reduce(plus, [1, 3, 5, 7, 9, 11, 13, 15]) + 17 + 19  
= reduce(plus, [1, 3, 5, 7, 9, 11, 13]) + 15 + 17 + 19  
= reduce(plus, [1, 3, 5, 7, 9, 11]) + 13 + 15 + 17 + 19  
= reduce(plus, [1, 3, 5, 7, 9]) + 11 + 13 + 15 + 17 + 19  
= reduce(plus, [1, 3, 5, 7]) + 9 + 11 + 13 + 15 + 17 + 19  
= reduce(plus, [1, 3, 5]) + 7 + 9 + 11 + 13 + 15 + 17 + 19  
= reduce(plus, [1, 3]) + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19  
= reduce(plus, [1]) + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19  
= 1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19  
= 100
```


Why learn functional programming?

Good reasons to learn a programming language

- Offers something new.
- Need to interface with co-workers.
- Better than Java for the application at hand.
- Provides an intellectual challenge
- Opportunity to learn something about computation.
- Introduces a new programming style.



Intro CS at MIT was taught in Scheme (a functional language) for decades

Modern applications

- Communications systems
- Financial systems
- Google MapReduce

Deep and direct connections to theoretical CS (stay tuned).



Warning. Functional programming may be addictive.



Functional Programming Jobs ?!?!?!?

A human being should be able to
change a diaper,
plan an invasion,
butcher a hog,
conn a ship,
design a building,
write a sonnet,
balance accounts,
build a wall,
set a bone,
comfort the dying,
take orders,
give orders,
cooperate,
act alone,
solve equations,
analyze a new problem,
pitch manure,
program a computer, ✓
cook a tasty meal,
fight efficiently, and
die gallantly.

Specialization is for insects.

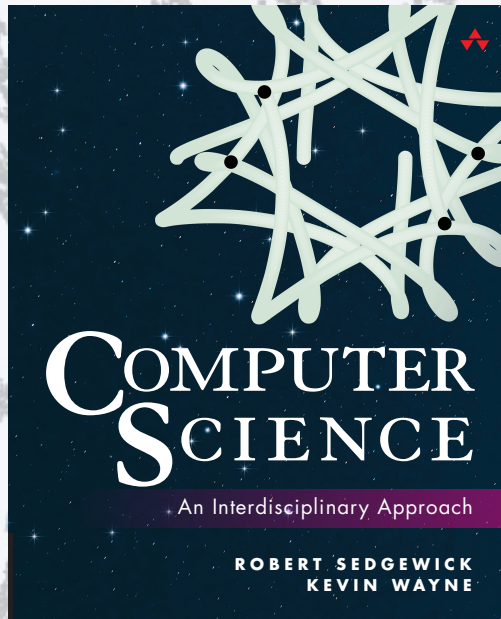
Robert A. Heinlein
Time Enough for Love (1973)



COMPUTER SCIENCE
S E D G E W I C K / W A Y N E
PART I: PROGRAMMING IN JAVA

CS.10.E.Languages.Functional

COMPUTER SCIENCE
SEDGEWICK / WAYNE
PART I: PROGRAMMING IN JAVA



<http://introcs.cs.princeton.edu>

10. Programming Languages