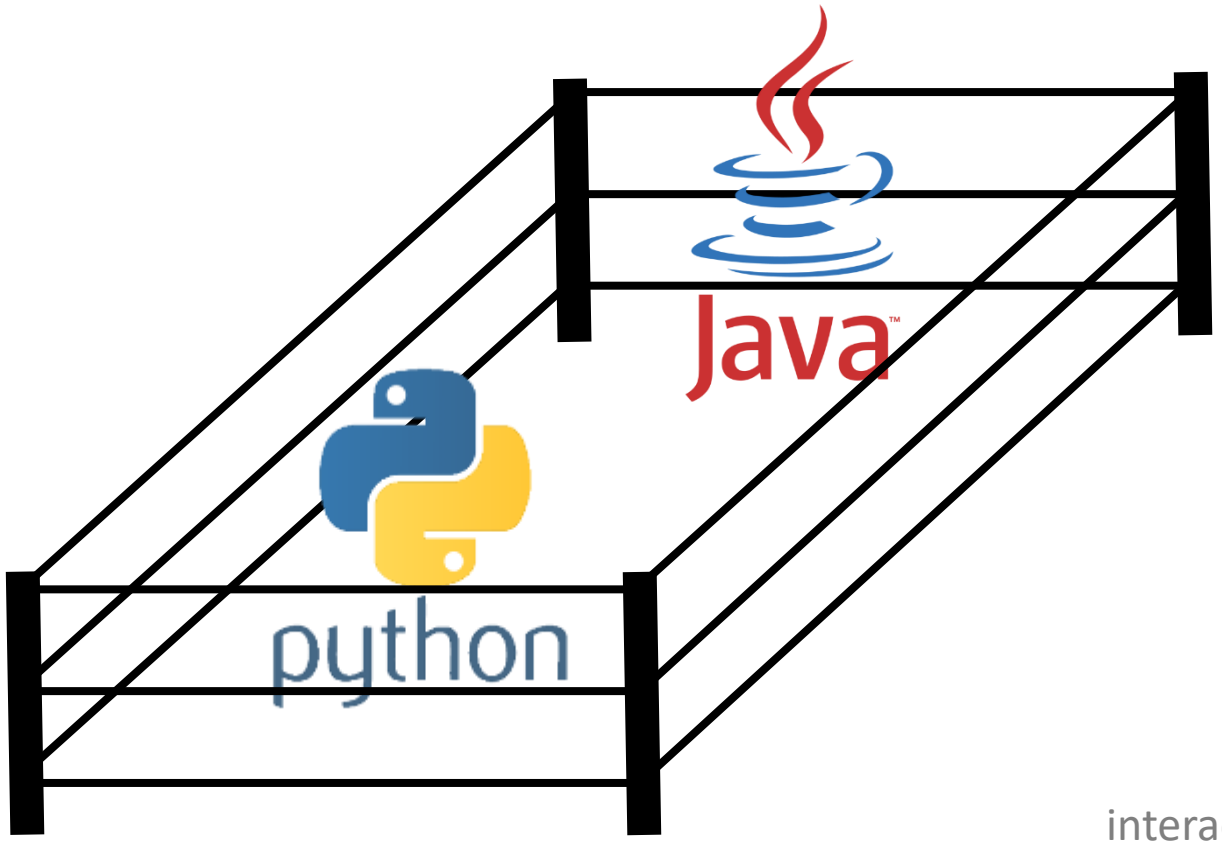
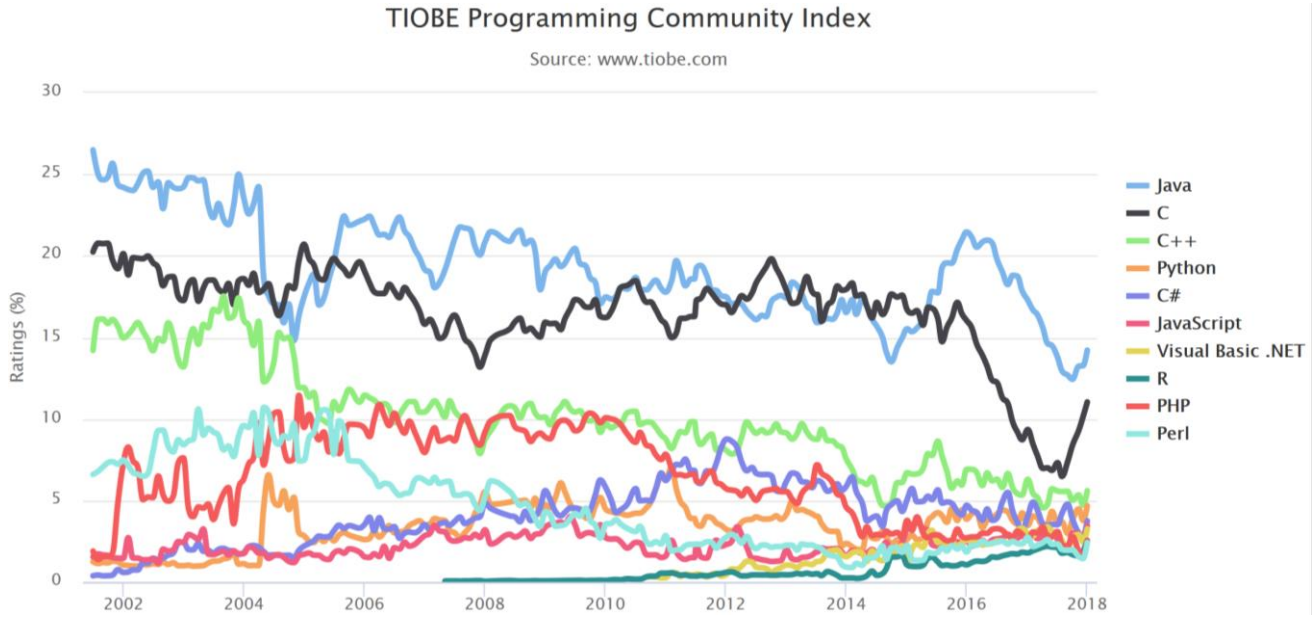


Java vs Python

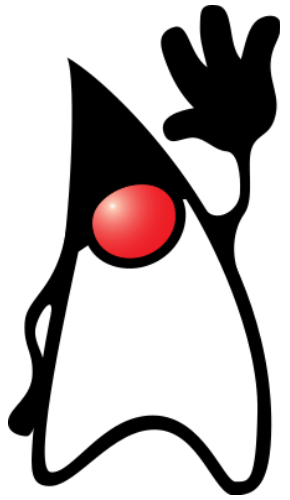
interactivepython.org/runestone/static/java4python/index.html

Why should you know something about Java?

- Java is an example of a statically typed language (like C and C++) opposed to Python's dynamic
- One of the most widespread used programming languages
- Used in other courses at the Department of Computer Science



Java history



- Java 1.0 released 1995 by Sun Microsystems (acquired by Oracle 2010)
- ”Write Once, Run Anywhere”
- 1999 improved performance by the **Java HotSpot Performance Engine**
- Current version Java 10 (released March 2018)
- Java compiler generates **Java bytecode** that is executed on a **Java virtual machine** (**JVM**)

PyPy is adopting the same ideas to Python (Just-in-Time compilation)

Installing Java

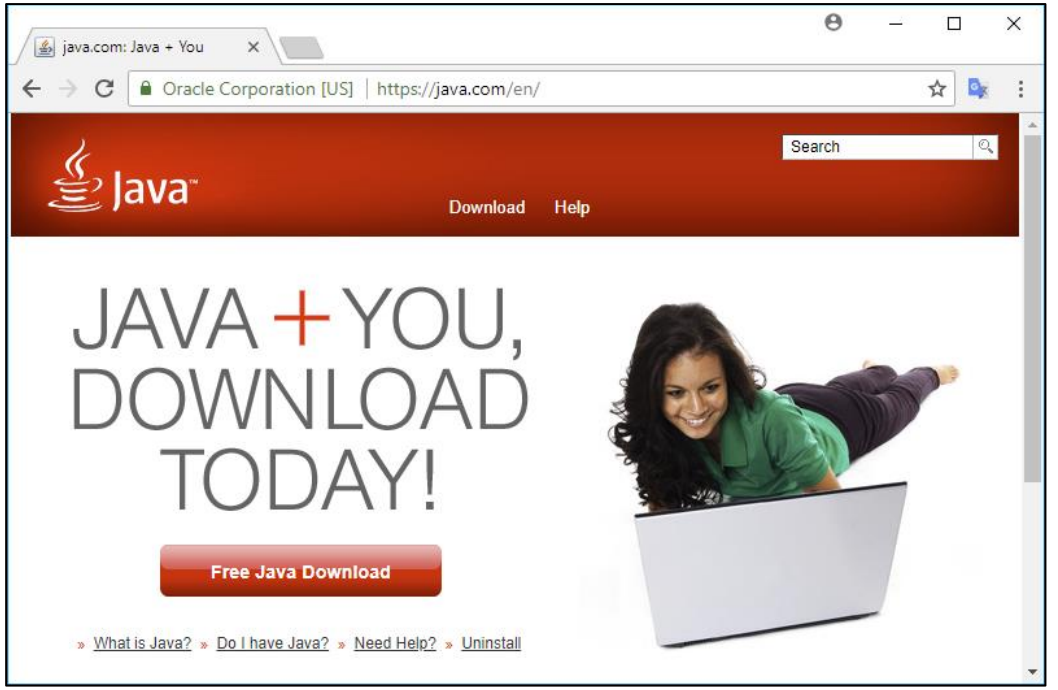
- Run compiled Java programs:

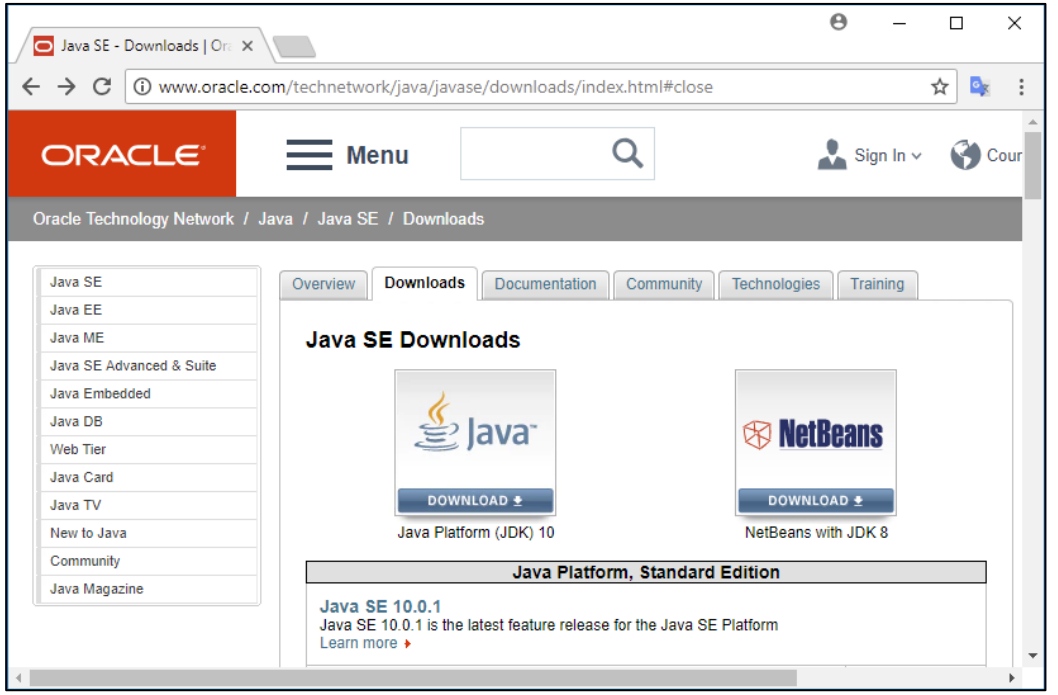
  java.com/download

- To compile Java programs into **bytecode** you need a compiler, e.g. from Java SE Development Kit (**JDK**):

  www.oracle.com/technetwork/java/javase/downloads/

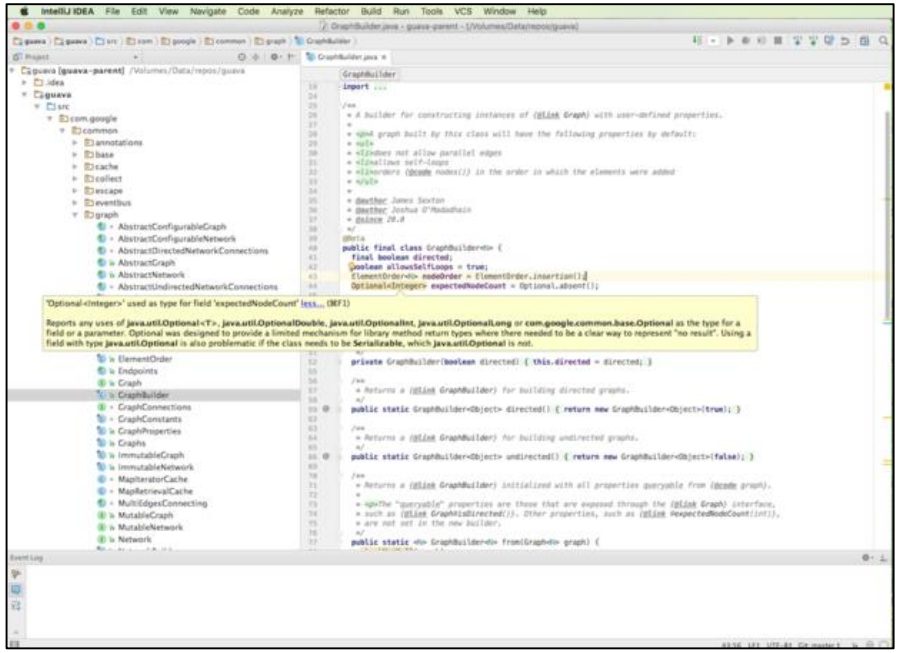
  (you might need to add the JDK directory to your PATH, e.g. C:\Program Files\Java\jdk-10.0.1\bin)

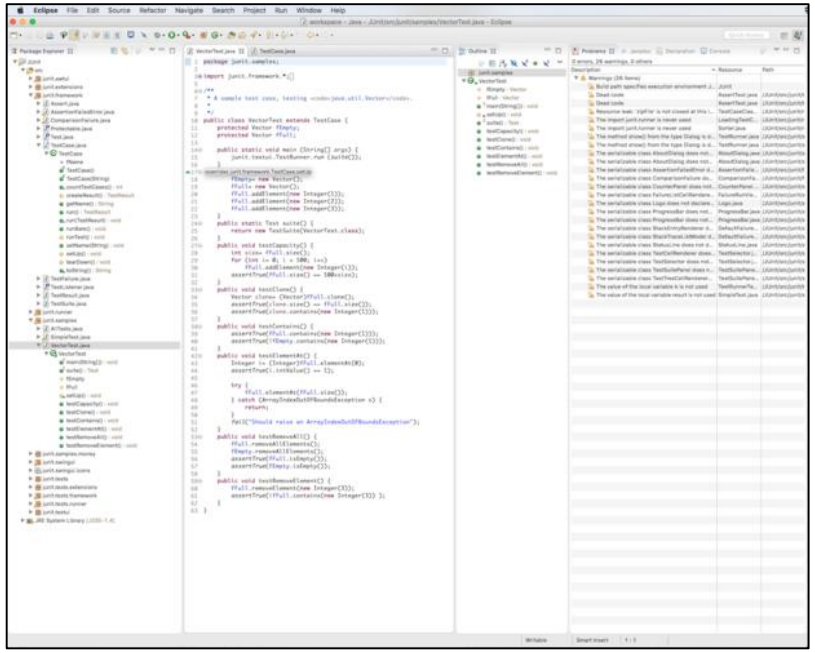


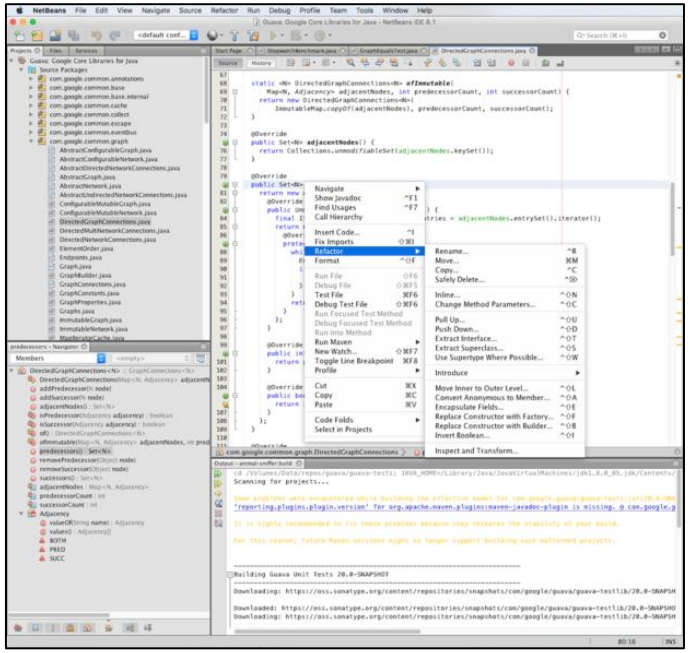


Java IDE

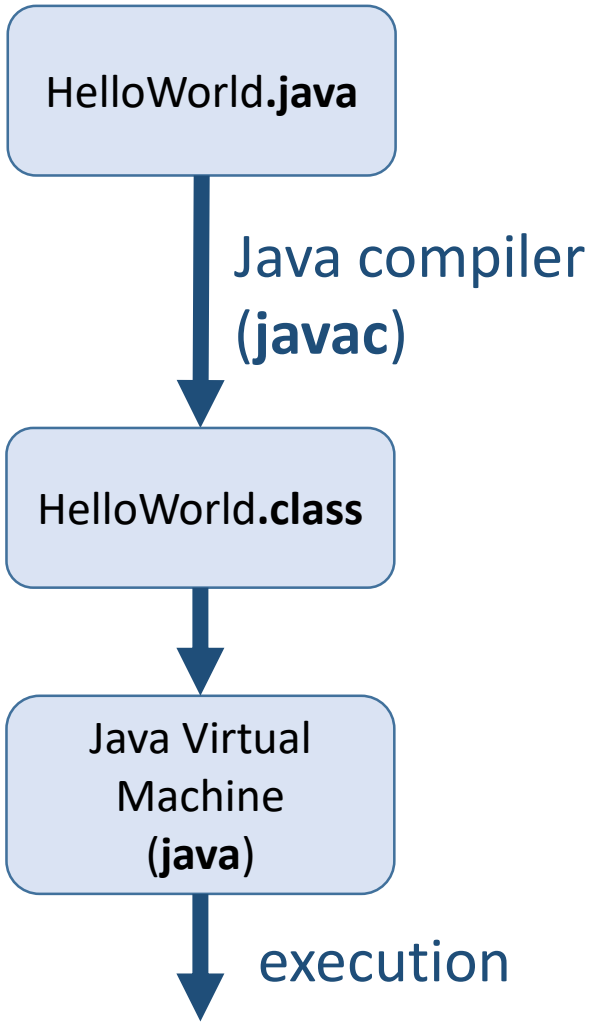
- Many available, some popular: IntelliJ IDEA, Eclipse, and NetBeans
- An IDE for beginners: BlueJ





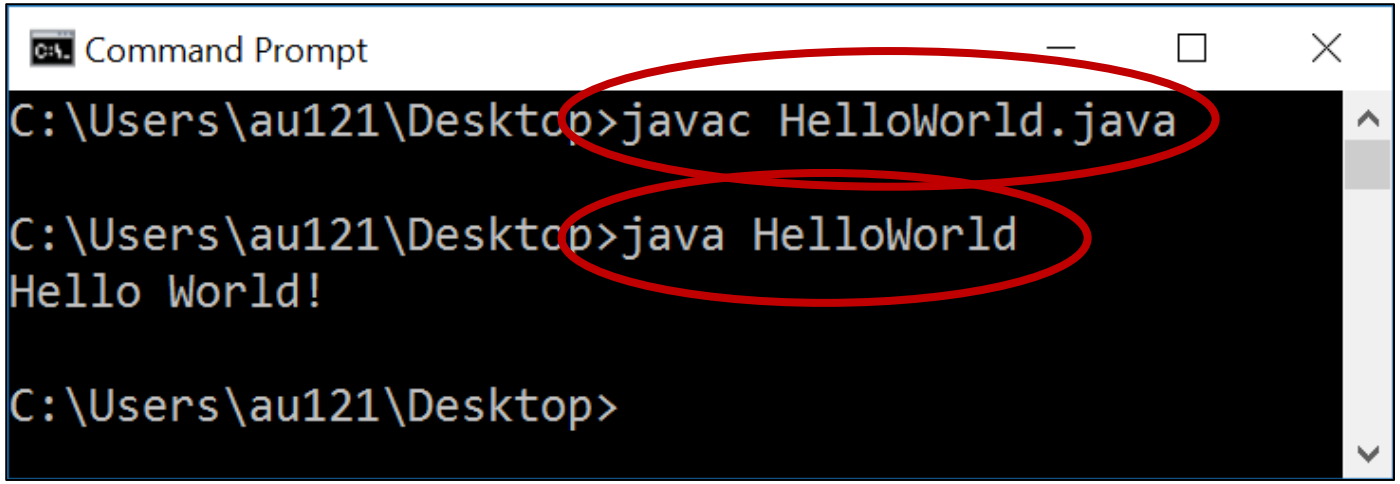


Compiling and running a Java program



HelloWorld.java

```
public class HelloWorld {
    public static void main( String[] args ) {
        System.out.println( "Hello World!" );
        System.exit( 0 ); //success
    }
}
```



Java : `main`

- *name*`.java` must be equal to the `public class` *name*
- A class can only be excuted using `java` *name* (without `.class`) if the class has a class method `main` with signature

```
public static void main(String[] args)
```

- (this is inherited from C and C++ sharing a lot of syntax with Java)

print_arguments.java

```
public class print_arguments {
    public static void main( String[] args ) {
       for (int i=0; i<args.length; i++)
           System.out.println( args[i] );
    }
}
```

shell

```
> java print_arguments x y z
| x
| y
| z
```

primitive.java

```
/**
 * A Java docstring to be processed using 'javadoc'
 */
// comment until end-of-line
public class primitive {
    public static void main( String[] args ) {
        int x;  // type of variable must be declared before used
        x = 1;  // remember ';' after each statement
           int y=2;  // indentation does not matter
    int a=3, b=4;  // multiple declarations and initialization
        System.out.println(x + y + a + b);
        int[] v={1, 2, 42, 3};  // array of four int
        System.out.println(v[2]);  // prints 42, arrays 0-indexed
/* multi-line comment
       that continues until here */
        v = new int[3];  // new array of size three, containing zeros
        System.out.println(v[2]);
        if (x == y) {  // if-syntax '(' and ')' mandatory
             a = 1;
             b = 2;
        } else {  // use '{' and '}' to create block of statements
             a = 4; b = 3;  // two statements on one line
        }
}}
```

Why state types – Python works without…

- Just enforcing a different programming style
- Helps users to avoid mixing up values of different types
- (Some) type errors can be caught at compile time

type_error.py

```
x = 3
y = "abc"
print("program running...")
print(x / y)
```

Python shell

```
| program running...
...
| ----> 4 print(x / y)
| TypeError: unsupported operand type(s) for
  /: 'int' and 'str'
```

type_error.java

```
public class type_error {
    public static void main( String[] args ) {
        int x = 3;
        String y = "abc";
        System.out.println(x / y);
    }
}
```

shell

```
> javac type_error.java
| javac type_error.java
| type_error.java:5: error: bad operand types for
  binary operator '/'
|         System.out.println(x / y);
|                            ^
|   first type:  int
|   second type: String
| 1 error
```

Basic Java types

Type	Values
`boolean`	`true` or `false`
`byte`	8 bit integer
`char`	character (16-bit UTF)
`short`	16 bit integer
`int`	32 bit integer
`long`	64 bit integer
`float`	32 bit floating bout
`double`	64 bit floating point
`class BigInteger`	arbitrary precision integers
`class String`	strings

biginteger.java

```
import java.math.*;

public class biginteger {
    public static void main( String[] args ) {
        BigInteger x = new BigInteger("2");
        while (true) {
            // BigIntegers are immutable
            x = x.multiply(x);
            // java.math.BigInteger.toString()
            System.out.println(x);
        }
    }
}
```

shell

```
4
16
256
65536
4294967296
18446744073709551616
340282366920938463463374607431768211456
...
```

Java arrays

- The size of a builtin Java array can not be modified when first created. If you need a bigger array you have to instantiate a new array.

- Or better use a standard collection class like **ArrayList**

concatenate_array_lists.java

```
import java.util.*;  // java.util contains ArrayList

public class concatenate_array_list {
    public static void main( String[] args ) {
        // ArrayList is a generic container
        ArrayList<String> A = new ArrayList<String>();
        ArrayList<String> B = new ArrayList<String>();
        ArrayList<String> C = new ArrayList<String>();

        A.add("A1");
        A.add("A2");
        B.add("B1");

        C.addAll(A);
        C.addAll(B);

        for (String e : C) {  // foreach over iterator
            System.out.println(e);
        }
    }
}
```

shell

```
A1
A2
B1
```

Function arguments

- Must declare the number of arguments and the types, and the return type
- The argument types are part of the signature of the function
- Several functions can have the same name, but different type signatures
- Python keyword arguments, * and ** do not exist in Java 

functions.java

```
public class functions {
    private static int f(int x) {
        return x * x;
    }
    private static int f(int x, int y) {
        return x * y;
    }
    private static String f(String a, String b) {
        return a + b; // string concatenation
    }

    public static void main( String[] args ) {
        System.out.println(f(7));
        System.out.println(f(3, 4));
        System.out.println(f("abc", "def"));
    }
}
```

shell

```
49
12
abcdef
```

functions.py

```
def f(x, y=None):
    if y == None:
        y = x
    if type(x) is int:
        return x * y
    else:
        return x + y
print(f(7), f(3, 4), f('abc', 'def'))
```

Class

- Constructor = method with name equal to class name
- `this` = referes to current object (Python ”self”)
- Use `private`/`public` on attributes/methods to give access outside class
- Use `new` *name*(arguments) to create new objects

- There can be multiple constructors, but with distinct type signatures

A_class.java

```
class rectangle {
    private int width, height;  // declare attributes

    // constructor, class name, no return type
    public rectangle(int width, int height) {
       this.width = width; this.height = height;
    }

    public int area() {
       return width * height;
    }
}

public class A_class {
    public static void main( String[] args ) {
       rectangle R = new rectangle(6, 7);
       System.out.println(R.area());
    }
}
```

shell

```
| 42
```

Inheritance

- Java supports single inheritance using `extends`
- Attributes and methods that should be accessible in a subclass must be declared protected (or public)
- Constructors are not inherited but can be called using `super`

inheritance.java

```
class basic_rectangle {
    // protected allows subclass to access attributes
    protected int width, height;
    public basic_rectangle(int width, int height) {
        this.width = width; this.height = height;
    }
}
class rectangle extends basic_rectangle {
    public rectangle(int width, int height) {
        // call constructor of super class
        super(width, height);
    }
    public int area() {
        return width * height;
    }
}
public class inheritance {
    public static void main( String[] args ) {
        rectangle R = new rectangle(6, 7);
        System.out.println(R.area());
    }
}
```

shell

```
42
```

Generic class

- Class that is parameterized by one or more types
- Primitive types cannot by type parameters
- Instead use *wrappers*, like Integer for int

generic_pair.java

```
class pair<element> {
    private element x, y;
    public pair(element x, element y) {
        this.x = x; this.y = y;
    }
    element first() { return x; }
    element second() { return y; }
}
public class generic_pair {
    public static void main( String[] args ) {
        pair<Integer> p = new pair<Integer>(6, 7);
        System.out.println(p.first() * p.second());
    }
}
```

shell

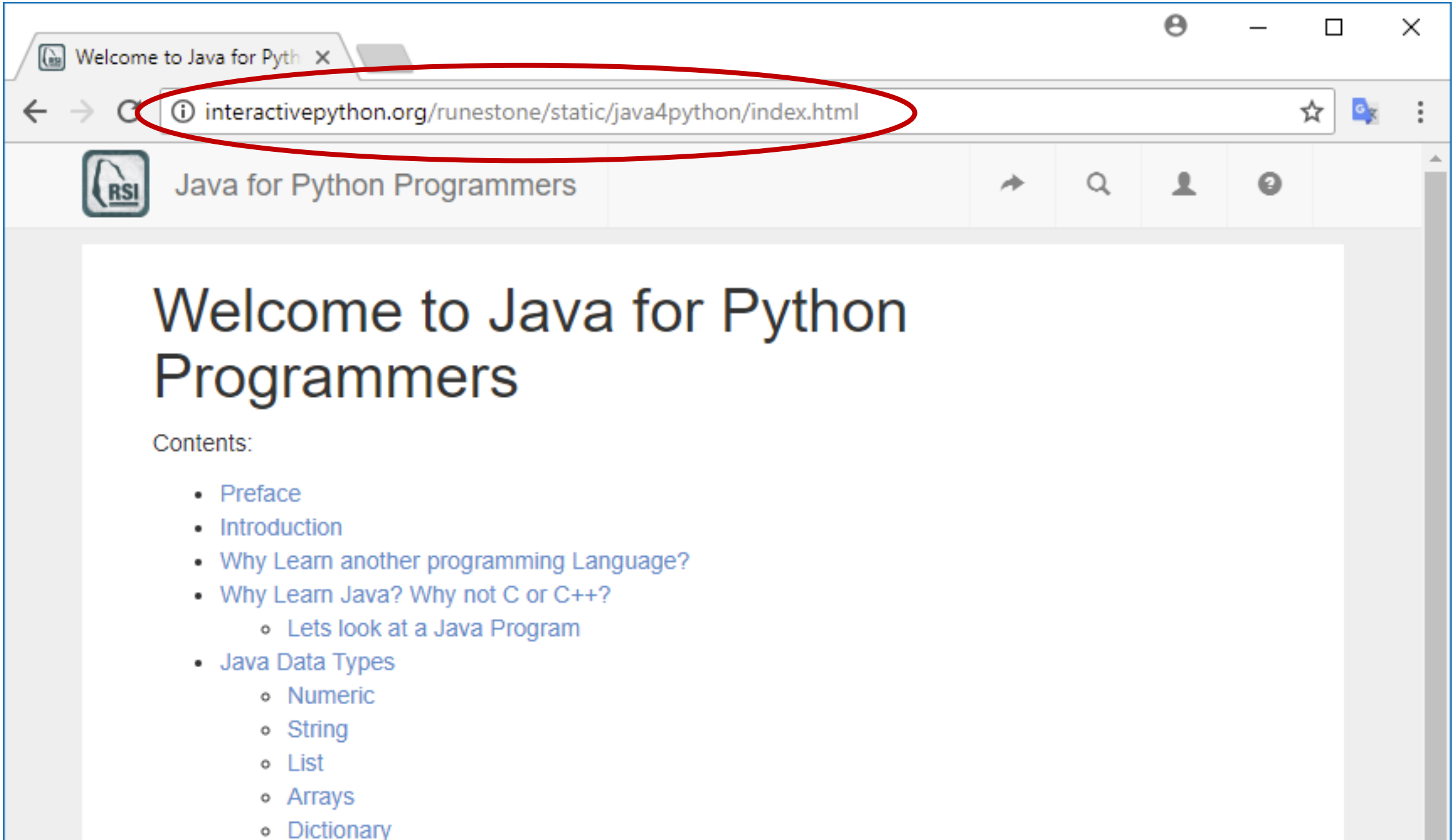
```
| 42
```

Interface

- Java does not support multiple inheritance like Python
- But a class can implement an arbitrary number of interfaces
- An interface specifices a set of attributes and methods a class must have
- The type of a variable can be an interface, and the variable can hold any object where the class is stated to implement the interface

interface.java

```
interface shape {
    public int area(); // method declaration
}
class rectangle implements shape {
    private int width, height;
    // constructor, class name, no return type
    public rectangle(int width, int height) {
        this.width = width; this.height = height;
    }
    public int area() {
        return width * height;
    }
}
public class interfaces {
    public static void main( String[] args ) {
        shape R = new rectangle(6, 7);
        System.out.println(R.area());
    }
}
```


Welcome to Java for Pyth
interactivepython.org/runestone/static/java4python/index.html
Java for Python Programmers
Welcome to Java for Python Programmers
Contents:
Preface
Introduction
Why Learn another programming Language?
Why Learn Java? Why not C or C++?
Lets look at a Java Program
Java Data Types
Numeric
String
List
Arrays
Dictionary