

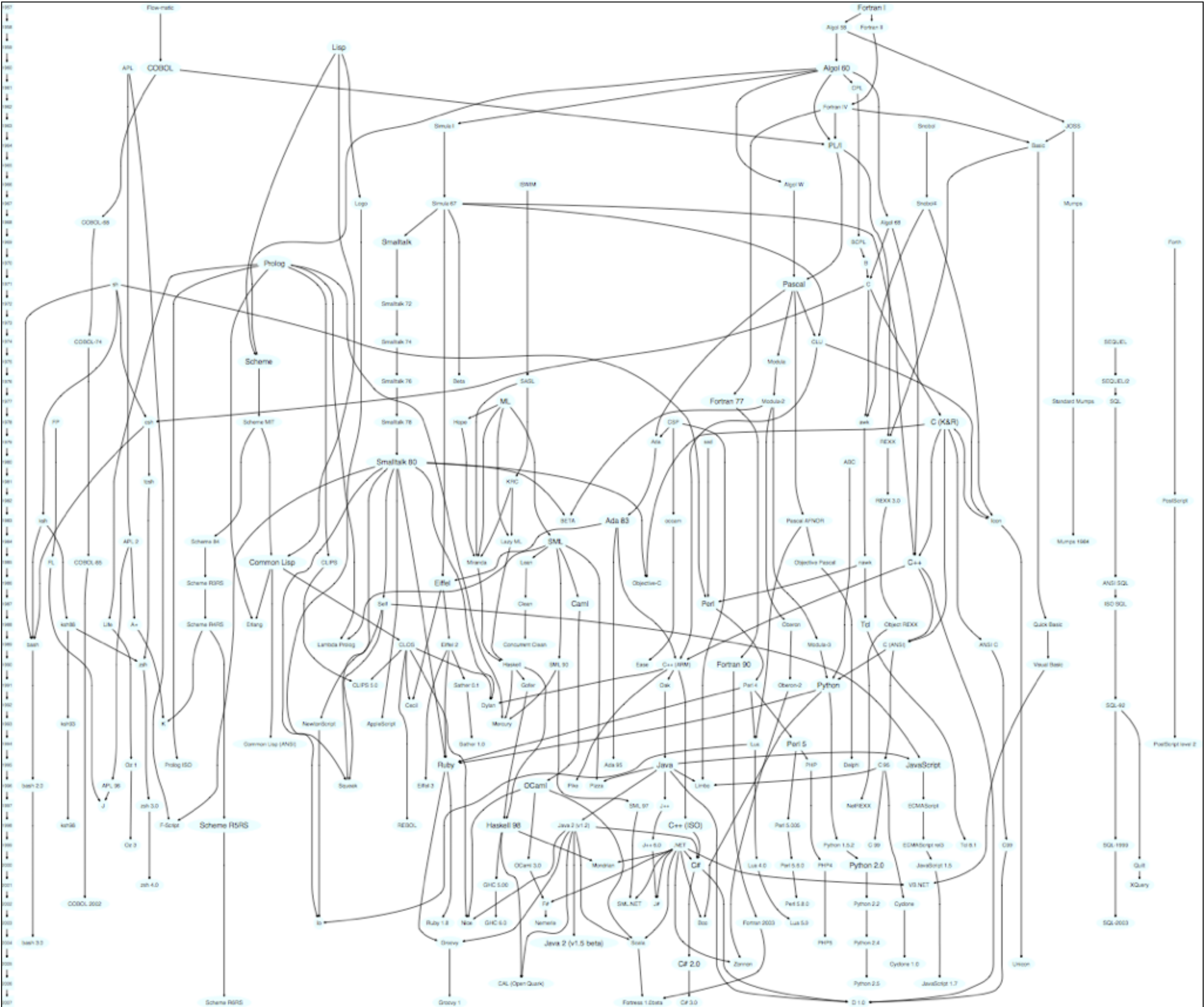
# Convergence

## Convergence



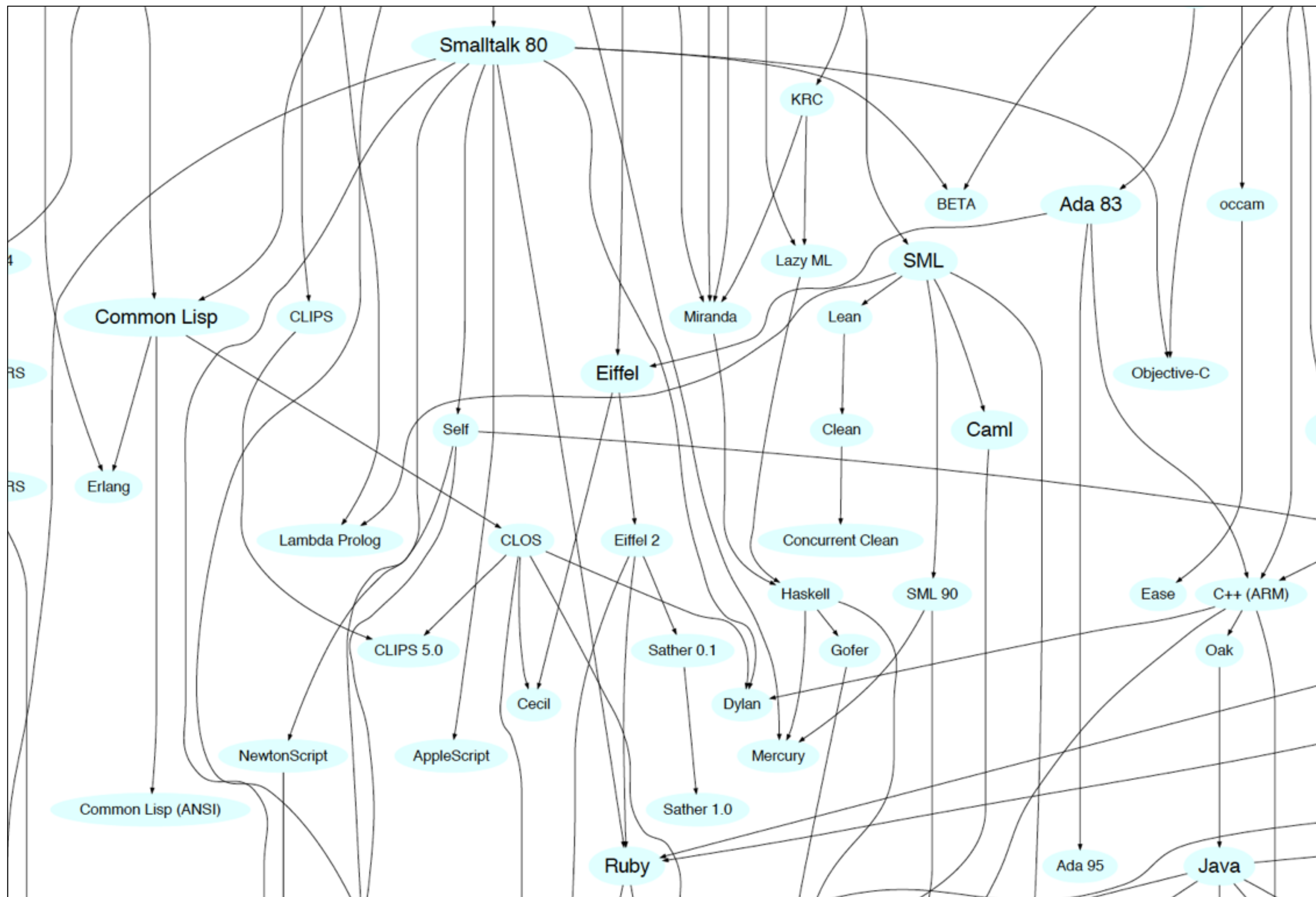
A simple algorithm  
expressed in Java, Groovy,  
Swift and Kotlin

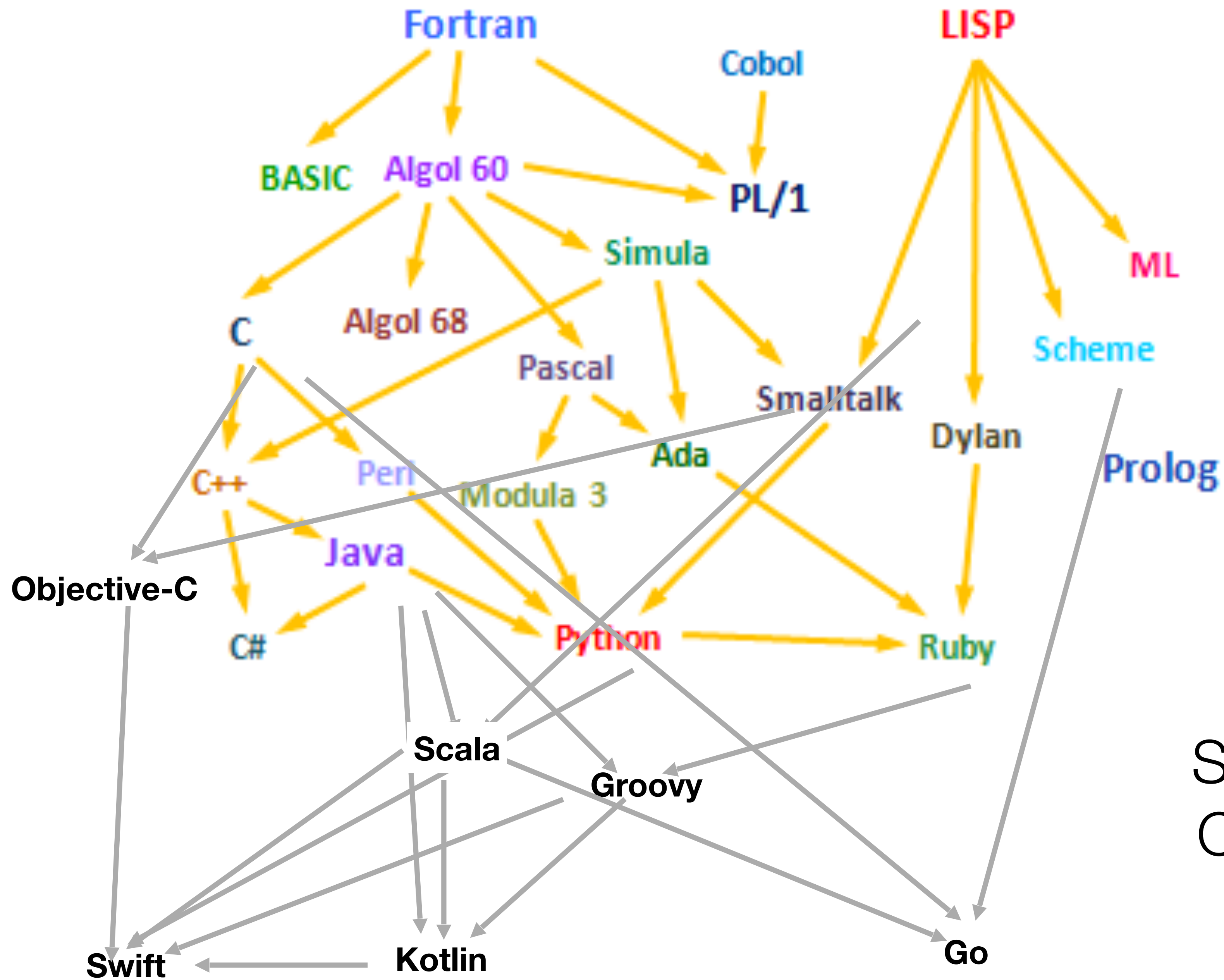
# Programming Language Family Tree





# Smalltalk Cluster





Scala / Groovy /  
C# / Swift/ Go /  
Kotlin Cluster

# Java Example

- Java algorithm to filter a list of strings
- Only printing those with 3 or less characters (in this test case).

```
import java.util.ArrayList;
import java.util.List;

class Erase{
    public static void main(String[] args)
    {
        List<String> names = new ArrayList<String>();
        names.add("Ted");
        names.add("Fred");
        names.add("Jed");
        names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List<String> short_names = e.filterLongerThan(names, 3);
        System.out.println(short_names.size());
        for (String s : short_names)
        {
            System.out.println(s);
        }
    }

    public List<String> filterLongerThan(List<String> strings, int length)
    {
        List<String> result = new ArrayList<String>();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s);
            }
        }
        return result;
    }
}
```

# Groovy 1

---

- Also a valid Groovy program...

```
import java.util.ArrayList;
import java.util.List;

class Erase{
    public static void main(String[] args)
    {
        List<String> names = new ArrayList<String>();
        names.add("Ted");
        names.add("Fred");
        names.add("Jed");
        names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List<String> short_names = e.filterLongerThan(names, 3);
        System.out.println(short_names.size());
        for (String s : short_names)
        {
            System.out.println(s);
        }
    }

    public List<String> filterLongerThan(List<String> strings,int length)
    {
        List<String> result = new ArrayList<String>();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s);
            }
        }
        return result;
    }
}
```



# Groovy 1

---

- Do we need generics?
- What about semicolons?
- Should standard libraries be imported?

```
import java.util.ArrayList;
import java.util.List;

class Erase{
    public static void main(String[] args)
    {
        List<String> names = new ArrayList<String>();
        names.add("Ted");
        names.add("Fred");
        names.add("Jed");
        names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List<String> short_names = e.filterLongerThan(names, 3);
        System.out.println(short_names.size());
        for (String s : short_names)
        {
            System.out.println(s);
        }
    }

    public List<String> filterLongerThan(List<String> strings,int length)
    {
        List<String> result = new ArrayList<String>();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s);
            }
        }
        return result;
    }
}
```

# Groovy 2

- *ArrayList not given a generic type.*
- *No need for semicolons.*
- *No need to import libraries.*

```
class Erase
{
    public static void main(String[] args)
    {
        List names = new ArrayList()
        names.add("Ted")
        names.add("Fred")
        names.add("Jed")
        names.add("Ned")
        System.out.println(names)
        Erase e = new Erase()
        List short_names = e.filterLongerThan(names, 3)
        System.out.println(short_names.size())
        for (String s : short_names)
        {
            System.out.println(s)
        }
    }

    public List filterLongerThan(List strings, int length)
    {
        List result = new ArrayList();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s)
            }
        }
        return result
    }
}
```



# Groovy 2

---

- Do we need the static types?
- Must we always have a main method and class definition?

```
class Erase
{
    public static void main(String[] args)
    {
        List names = new ArrayList()
        names.add("Ted")
        names.add("Fred")
        names.add("Jed")
        names.add("Ned")
        System.out.println(names)
        Erase e = new Erase()
        List short_names = e.filterLongerThan(names, 3)
        System.out.println(short_names.size())
        for (String s : short_names)
        {
            System.out.println(s)
        }
    }

    public List filterLongerThan(List strings, int length)
    {
        List result = new ArrayList();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s)
            }
        }
        return result
    }
}
```

# Groovy 3

- *Types removed in method signature.*
- *main method and class definition removed.*

```
def filterLongerThan(strings, length)
{
    List result = new ArrayList();
    for (String s : strings)
    {
        if (s.length() < length + 1)
        {
            result.add(s)
        }
    }
    return result
}

List names = new ArrayList()
names.add("Ted")
names.add("Fred")
names.add("Jed")
names.add("Ned")
System.out.println(names)
List short_names = filterLongerThan(names, 3)
System.out.println(short_names.size())
for (String s : short_names)
{
    System.out.println(s)
}
```

# Groovy 3

- Should we have a special notation for lists?
- And special facilities for list processing?

```
def filterLongerThan(strings, length)
{
    List result = new ArrayList();
    for (String s : strings)
    {
        if (s.length() < length + 1)
        {
            result.add(s)
        }
    }
    return result
}

List names = new ArrayList()
names.add("Ted")
names.add("Fred")
names.add("Jed")
names.add("Ned")
System.out.println(names)
List short_names = filterLongerThan(names, 3)
System.out.println(short_names.size())
for (String s : short_names)
{
    System.out.println(s)
}
```

# Groovy 4

---

- *special notation for lists used*
- *list processing closures used.*

```
def filterLongerThan(strings, length)
{
    return strings.findAll {it.size() <= length}
}

names = ["Ted", "Fred", "Jed", "Ned"]
System.out.println(names)
List short_names = filterLongerThan(names, 3)
System.out.println(short_names.size())
short_names.each {System.out.println(it)}
```



# Groovy 4

---

- Method needed any longer?
- Is there an easier way to use common methods (e.g. println)?
- Are brackets always needed?

```
def filterLongerThan(strings, length)
{
    return strings.findAll {it.size() <= length}
}

names = ["Ted", "Fred", "Jed", "Ned"]
System.out.println(names)
List short_names = filterLongerThan(names, 3)
System.out.println(short_names.size())
short_names.each {System.out.println(it)}
```

# Groovy 5

---

- *Method removed*
- *Used common method notation*
- *Removed non necessary brackets.*

```
names = ["Ted", "Fred", "Jed", "Ned"]  
println names  
short_names = names.findAll{it.size() <= 3}  
println short_names.size()  
short_names.each {println it}
```

```

import java.util.ArrayList;
import java.util.List;

class Erase{
    public static void main(String[] args)
    {
        List<String> names = new ArrayList<String>();
        names.add("Ted");
        names.add("Fred");
        names.add("Jed");
        names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List<String> short_names = e.filterLongerThan(names, 3);
        System.out.println(short_names.size());
        for (String s : short_names)
        {
            System.out.println(s);
        }
    }

    public List<String> filterLongerThan(List<String> strings, int length)
    {
        List<String> result = new ArrayList<String>();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s);
            }
        }
        return result;
    }
}

```

```

names = ["Ted", "Fred", "Jed", "Ned"]
println names
short_names = names.findAll{it.size() <= 3}
println short_names.size()
short_names.each {println it}

```

## Java vs Groovy?

# Another Approach to Types?

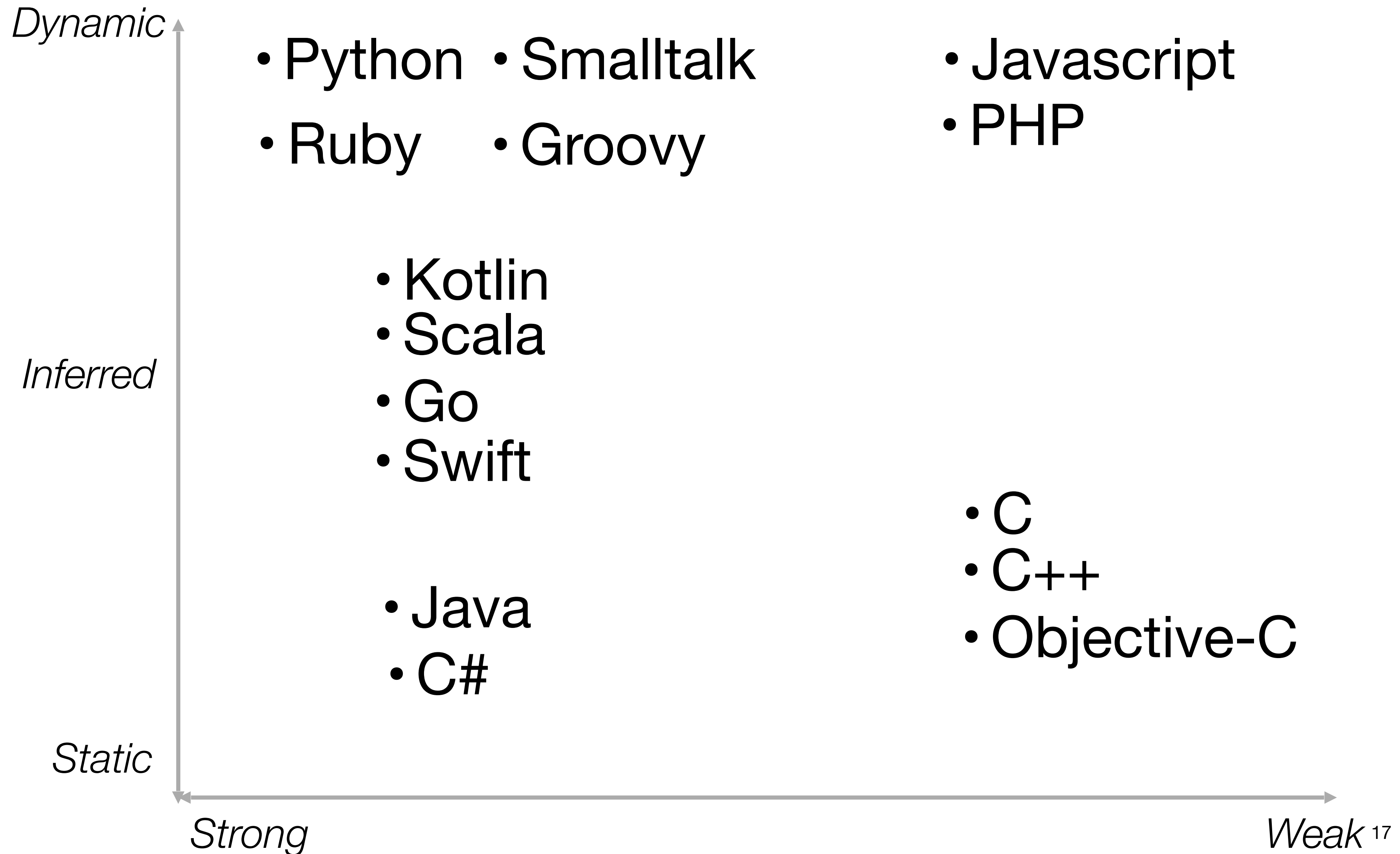
---

- *Type Inference* : the compiler draws conclusions about the types of variables based on how programmers use those variables.
  - Yields programs that have some of the conciseness of Dynamically Typed Languages
  - But - decision made at *compile time*, not at *run time*
  - More information for static analysis - refactoring tools, complexity analysis, bug checking etc...
- Haskell, Scala, **Kotlin**  
Java (from 7 onwards)

```
object InferenceTest1 extends Application {  
    val x = 1 + 2 * 3           // the type of x is int  
    val y = x.toString()       // the type of y is String  
    def succ(x: int) = x + 1    // method succ returns int values  
}
```



# Typing Spectrum



# Back to our Java Example

---

- Java algorithm to filter a list of strings
- Only printing those with 3 or less characters (in this test case).

```
import java.util.ArrayList;
import java.util.List;

class Erase{
    public static void main(String[] args)
    {
        List<String> names = new ArrayList<String>();
        names.add("Ted");
        names.add("Fred");
        names.add("Jed");
        names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List<String> short_names = e.filterLongerThan(names, 3);
        System.out.println(short_names.size());
        for (String s : short_names)
        {
            System.out.println(s);
        }
    }

    public List<String> filterLongerThan(List<String> strings, int length)
    {
        List<String> result = new ArrayList<String>();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s);
            }
        }
        return result;
    }
}
```

# Swift

---

```
import Foundation

class Erase
{
    func main()
    {
        var names:String[] = String[]()
        names.append ("ted")
        names.append ("fred")
        names.append ("jed")
        names.append ("ned")
        println(names)
        var short_names:String[] = filterLongerThan(names, length:3)
        for name:String in short_names
        {
            println (name)
        }
    }
    func filterLongerThan (strings : String[], length : Int) -> String[]
    {
        var result:String[] = String[]()
        for s:String in strings
        {
            if countElements(s) < length + 1
            {
                result.append(s)
            }
        }
        return result
    }
}

var erase:Erase = Erase()
erase.main()
```

# Swift

- Type Inference

```
import Foundation

class Erase
{
    func main()
    {
        var names = String[]()
        names.append ("ted")
        names.append ("fred")
        names.append ("jed")
        names.append ("ned")
        println(names)
        var short_names = filterLongerThan(names, length:3)
        for name in short_names
        {
            println (name)
        }
    }
    func filterLongerThan (strings : String[], length : Int) -> String[]
    {
        var result = String[]()
        for s in strings
        {
            if countElements(s) < length + 1
            {
                result.append(s)
            }
        }
        return result
    }
}

var erase = Erase()
erase.main()
```



# Swift

- Literals

```
import Foundation

class Erase
{
    func main()
    {
        var names = ["ted", "fred", "jed", "ned"]
        var short_names = filterLongerThan(names, length:3)
        for name in short_names
        {
            println (name)
        }
    }

    func filterLongerThan (strings : String[], length : Int) -> String[]
    {
        var result = String[]()
        for s in strings
        {
            if countElements(s) < length + 1
            {
                result.append(s)
            }
        }
        return result
    }
}

var erase = Erase()
erase.main()
```

# Swift

---

- Closures (Lambdas)

```
import Foundation

class Erase
{
    func main()
    {
        var names = ["ted", "fred", "jed", "ned"]
        var short_names = names.filter { countElements($0) < 4 }
        for name in short_names
        {
            println (name)
        }
    }
}

var erase = Erase()
erase.main()
```

# Swift

---

- Script

```
import Foundation

var names = ["ted", "fred", "jed", "ned"]
println(names)
var short_names = names.filter { countElements($0) < 4 }
println(short_names)
```

```

import java.util.ArrayList;
import java.util.List;

class Erase{
    public static void main(String[] args)
    {
        List<String> names = new ArrayList<String>();
        names.add("Ted");
        names.add("Fred");
        names.add("Jed");
        names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List<String> short_names = e.filterLongerThan(names, 3);
        System.out.println(short_names.size());
        for (String s : short_names)
        {
            System.out.println(s);
        }
    }

    public List<String> filterLongerThan(List<String> strings, int length)
    {
        List<String> result = new ArrayList<String>();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s);
            }
        }
        return result;
    }
}

```

Java

```

names = ["Ted", "Fred", "Jed", "Ned"]
println names
short_names = names.findAll{it.size() <= 3}
short_names.each {println it}

```

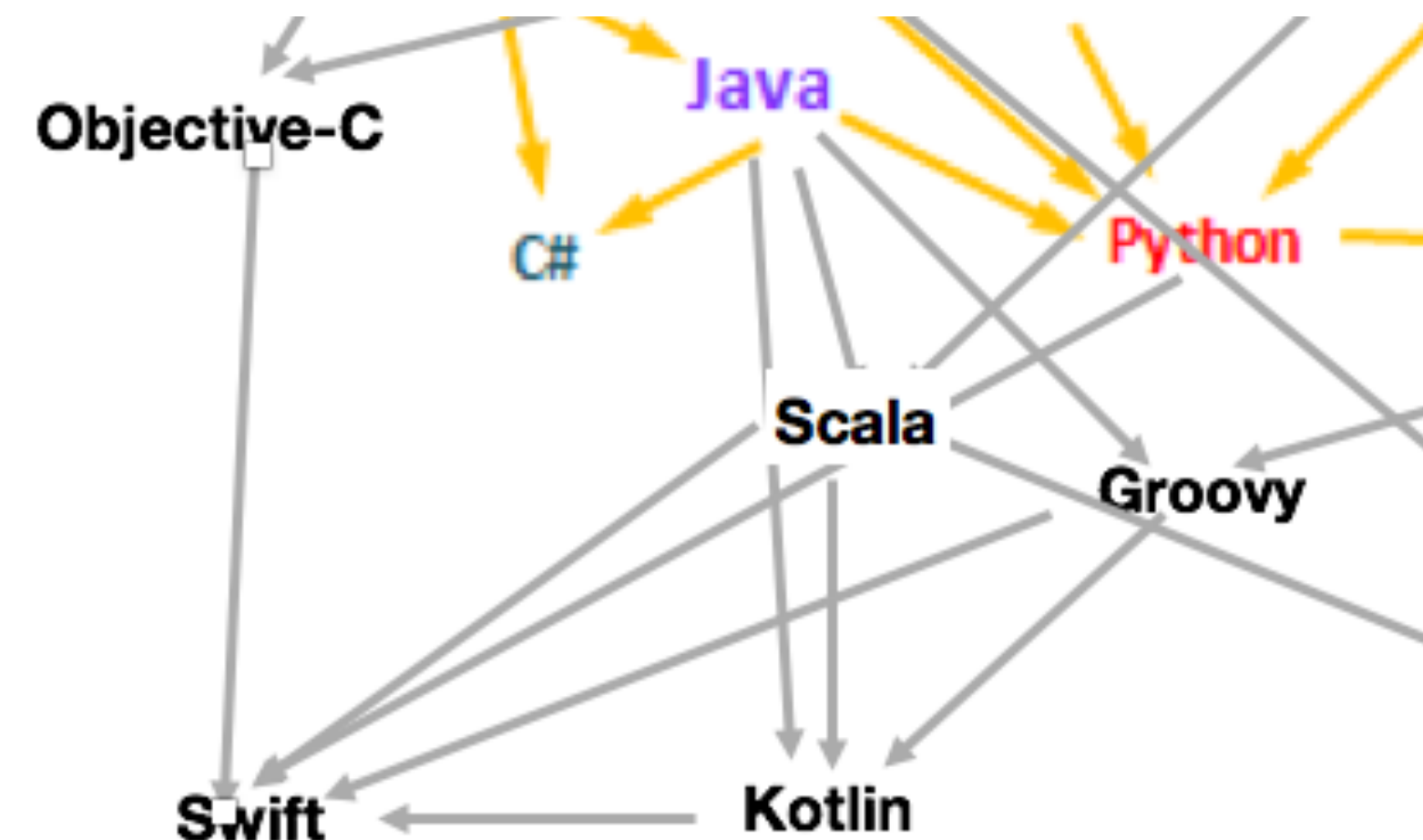
Groovy

```

var names = ["ted", "fred", "jed", "ned"]
println(names)
var short_names = names.filter { countElements($0) < 4 }
println(short_names)

```

Swift





```
package wordfilter
import java.util.ArrayList;

fun main(args: Array<String>) {
    val names: MutableList<String> = ArrayList<String>();
    names.add("Ted");
    names.add("Fred");
    names.add("Jed");
    names.add("Ned");
    System.out.println(names);
    val e = Erase();
    val short_names = e.filterLongerThan(names, 3);
    System.out.println(short_names.size);
    for (s: String in short_names) {
        System.out.println(s);
    }
}

class Erase {
    fun filterLongerThan(strings: MutableList<String>, length: Int): MutableList<String> {
        val result: MutableList<String> = ArrayList<String>();
        for (s: String in strings) {
            if (s.length < length + 1) {
                result.add(s)
            }
        }
        return result
    }
}
```

```
package wordfilter
import java.util.ArrayList;

fun main(args: Array<String>) {
    val names: MutableList<String> = ArrayList<String>();
    names.add("Ted");
    names.add("Fred");
    names.add("Jed");
    names.add("Ned");
    System.out.println(names);
    val e = Erase();
    val short_names = e.filterLongerThan(names, 3);
    System.out.println(short_names.size);
    for (s: String in short_names) {
        System.out.println(s);
    }
}

fun filterLongerThan(strings: MutableList<String>, length: Int): MutableList<String> {
    val result: MutableList<String> = ArrayList<String>();
    for (s: String in strings) {
        if (s.length < length + 1) {
            result.add(s)
        }
    }
    return result
}
```

```
package wordfilter
import java.util.ArrayList;

fun main(args: Array<String>) {
    val names: MutableList<String> = mutableListOf("Ted", "Fred", "Jed", "Ned");
    println(names);
    val e = Erase();
    val short_names = e.filterLongerThan(names, 3)
    println(short_names.size)
    for (s: String in short_names) {
        println(s);
    }
}

fun filterLongerThan1(strings: MutableList<String>, length: Int): List<String> {
    val result: List<String> = strings.filter { it.length < length + 1 }
    return result
}
```

```
package wordfilter
import java.util.ArrayList;

fun main(args: Array<String>) {
    val names: MutableList<String> = mutableListOf("Ted", "Fred", "Jed", "Ned");
    println(names);
    val short_names: List<String> = names.filter { it.length < 4 }
    println(short_names.size)
    println(short_names)
}
```

```
val names = mutableListOf("Ted", "Fred", "Jed", "Ned");  
println(names);  
val short_names = names.filter { it.length < 4 }  
println(short_names)
```

```

import java.util.ArrayList;
import java.util.List;

class Erase{
    public static void main(String[] args)
    {
        List<String> names = new ArrayList<String>();
        names.add("Ted");
        names.add("Fred");
        names.add("Jed");
        names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List<String> short_names = e.filterLongerThan(names, 3);
        System.out.println(short_names.size());
        for (String s : short_names)
        {
            System.out.println(s);
        }
    }

    public List<String> filterLongerThan(List<String> strings, int length)
    {
        List<String> result = new ArrayList<String>();
        for (String s : strings)
        {
            if (s.length() < length + 1)
            {
                result.add(s);
            }
        }
        return result;
    }
}

```

Java

```

names = ["Ted", "Fred", "Jed", "Ned"]
println names
short_names = names.findAll{ it.size() < 4 }
short_names.each {println it}

```

Groovy

```

let names = ["ted", "fred", "jed", "ned"]
println(names)
let short_names = names.filter { countElements($0) < 4 }
println(short_names)

```

Swift

```

val names = mutableListOf("Ted", "Fred", "Jed", "Ned");
println(names);
val short_names = names.filter { it.length < 4 }
println(short_names)

```

Kotlin

