Programming Language & Paradigm Selection

Mahmoud Ismail
Software Enginer, Peerialism

Agenda

* Introduction
* Selection Criteria
* Trending Topics
* Conclusion

Trending Topics

- Functional programming
- Concurrent and distributed programming
- Component oriented software development
- Aspect oriented programming
- Rise of JVM based languages (Jython, JRuby, ..)
- New Languages for web programming
  - Dart, Opa, CoffeeScript

Conclusion

- Overview about main programming paradigms
- How to select your programming language
- Advices:
  - don't stick with one language
  - try different paradigms especially functional

References

- http://www.slideshare.net/dnene/programming-language-selection
- http://rigaux.org/language-study/diagram.html
- http://www.ibm.com/developerworks/web/library/wa-optimal/index.html
- http://www.psl.cs.columbia.edu/classes/cs4156-f10/lectures/Programming%20Language%20Selection.pdf
- http://docs.python.org/2/howto/functional.html

Programming Language & Paradigm Selection

Mahmoud Ismail
Software Enginer, Peerialism

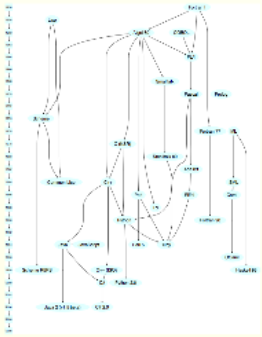
Agenda

* Introduction
* Selection Criteria
* Trending Topics
* Conclusion


ming Language & Paradigm Selection
Mahmoud Ismail
Software Enginer, Peerialism
Agenda
* Introduction
* Selection Criteria
* Trending Topics
* Conclusion

Introduction to Programming Language and Paradigms

Programming?!



Programming Paradigm?!

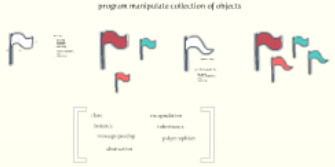
* Style or Way of programming
* Most languages are multi-paradigm
* Main Paradigms:
  - Imperative (How?)
  - Declarative (What?)
    - Database Query Languages (SQL)
    - Functional
    - Logic
  - Object Oriented

Imperative Programming

Computer is told how to solve the problem

Object Oriented Programming

program manipulate collection of objects

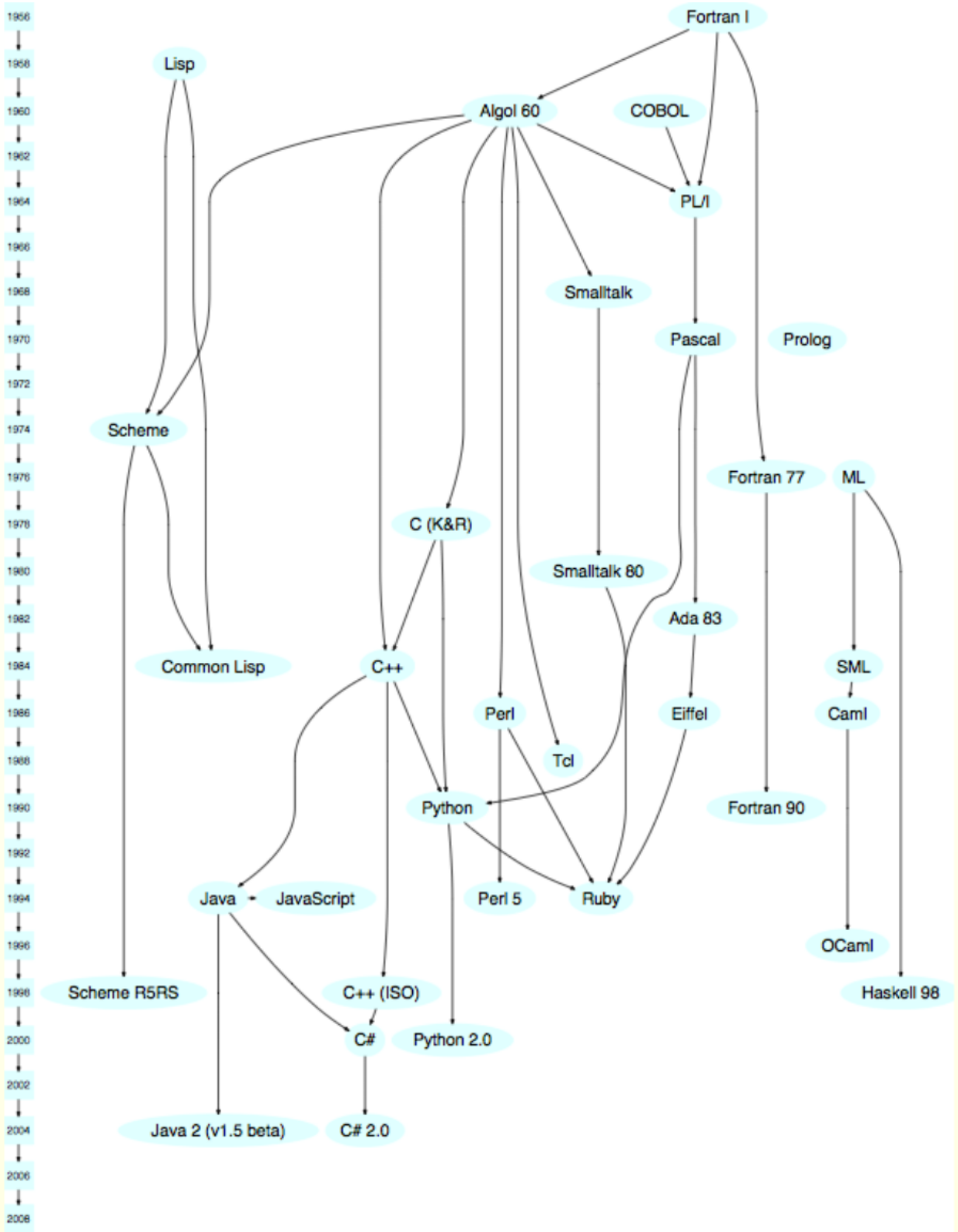


Functional Programming

a problem is decomposed into a set of functions
functions ==> mathematical functions

Selection Criteria

Programming?!


1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
Fortran I
Lisp
Algol 60
COBOL
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
Common Lisp
C++
SML
Perl
Eiffel
Caml
Tcl
Python
Fortran 90
Java
JavaScript
Perl 5
Ruby
OCaml
Scheme R5RS
C++ (ISO)
Haskell 98
C#
Python 2.0
Java 2 (v1.5 beta)
C# 2.0

Programming Paradigm?!

* Style or Way of programming
* Most languages are multi-paradigm
* Main Paradigms:
    - Imperative (How?)
    - Declarative (What?)
        - Database Query Languages (SQL)
        - Functional
        - Logic
    - Object Oriented

Imperative Programming

Computer is told how to solve the problem

mutable variables

assignment statement

Example: (C)

```
int i = 0;
int sum = 0;
```

mutable variables

assignment statement

control structures
if-then, for, ...

sub routines

Example: (C)

```
 int i = 0;
int sum = 0;
while (i < n) {
        i = i + 1;
        sum = sum + i;
}
```

Imperative Programming

Computer is told how to solve the problem

mutable variables

assignment statement

control structures
if-then, for, ...

sub routines

Example: (C)

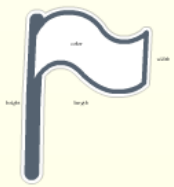
```
int i = 0;
int sum = 0;
while (i < n) {
    i = i + 1;
    sum = sum + i;
}
```

sub routines

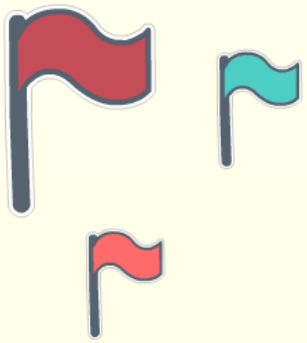
```
        sum = sum + i;
    }
```

Object Oriented Programming

program manipulate collection of objects

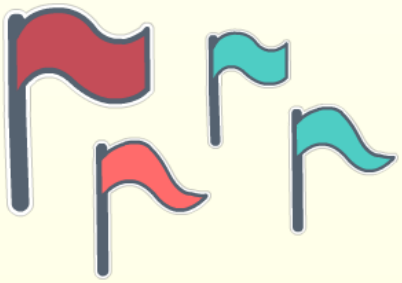


```
class Flag
{
    int width;
    int length;
    int height;
    Color color;

    void setLength(int l);
    ------
    void wave();
}
```



speed of waving

```
class TriFlag extends Flag
{
    int speed;
    void setLength(int l);
    ------
    void wave();
}
```

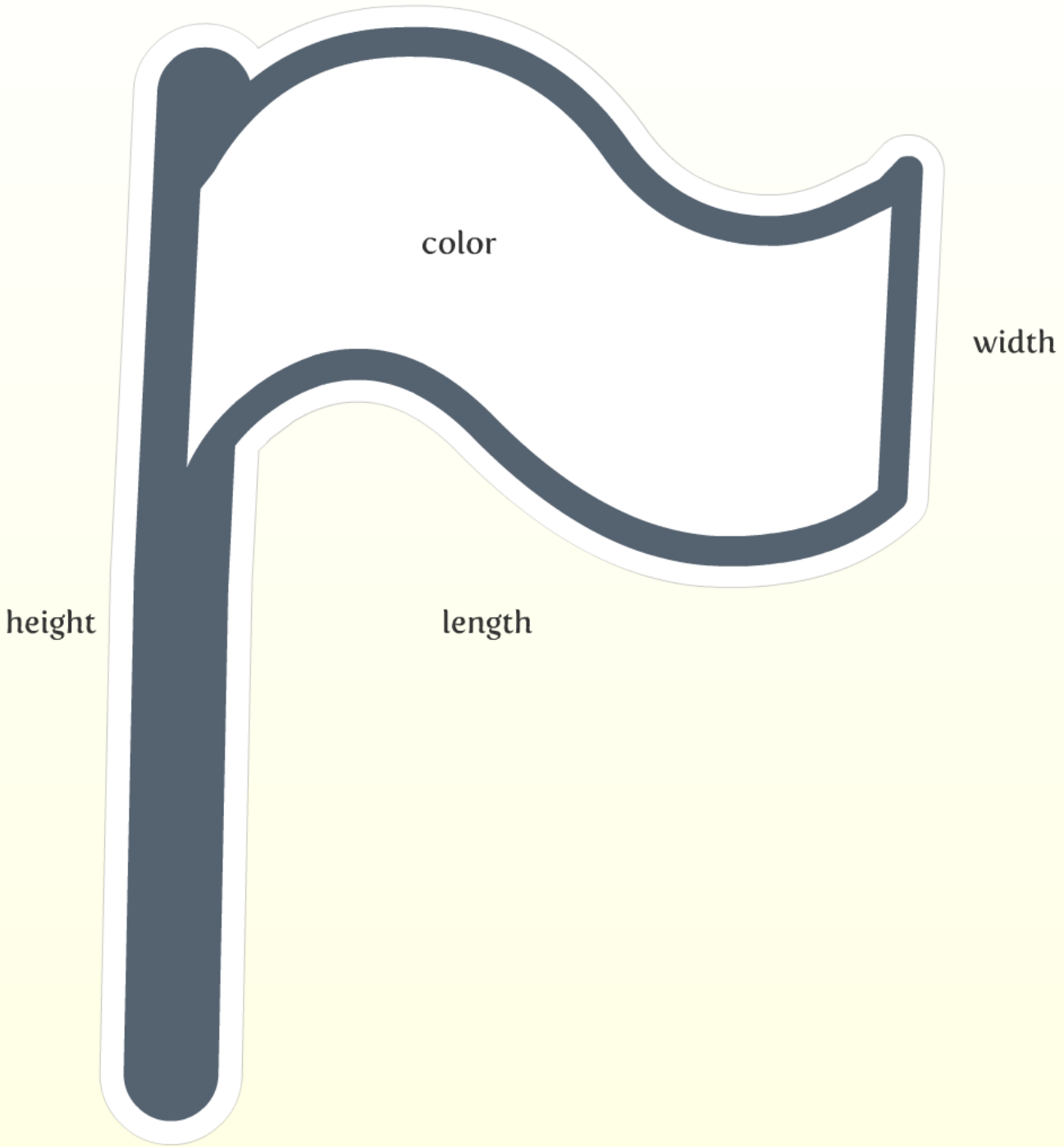


class

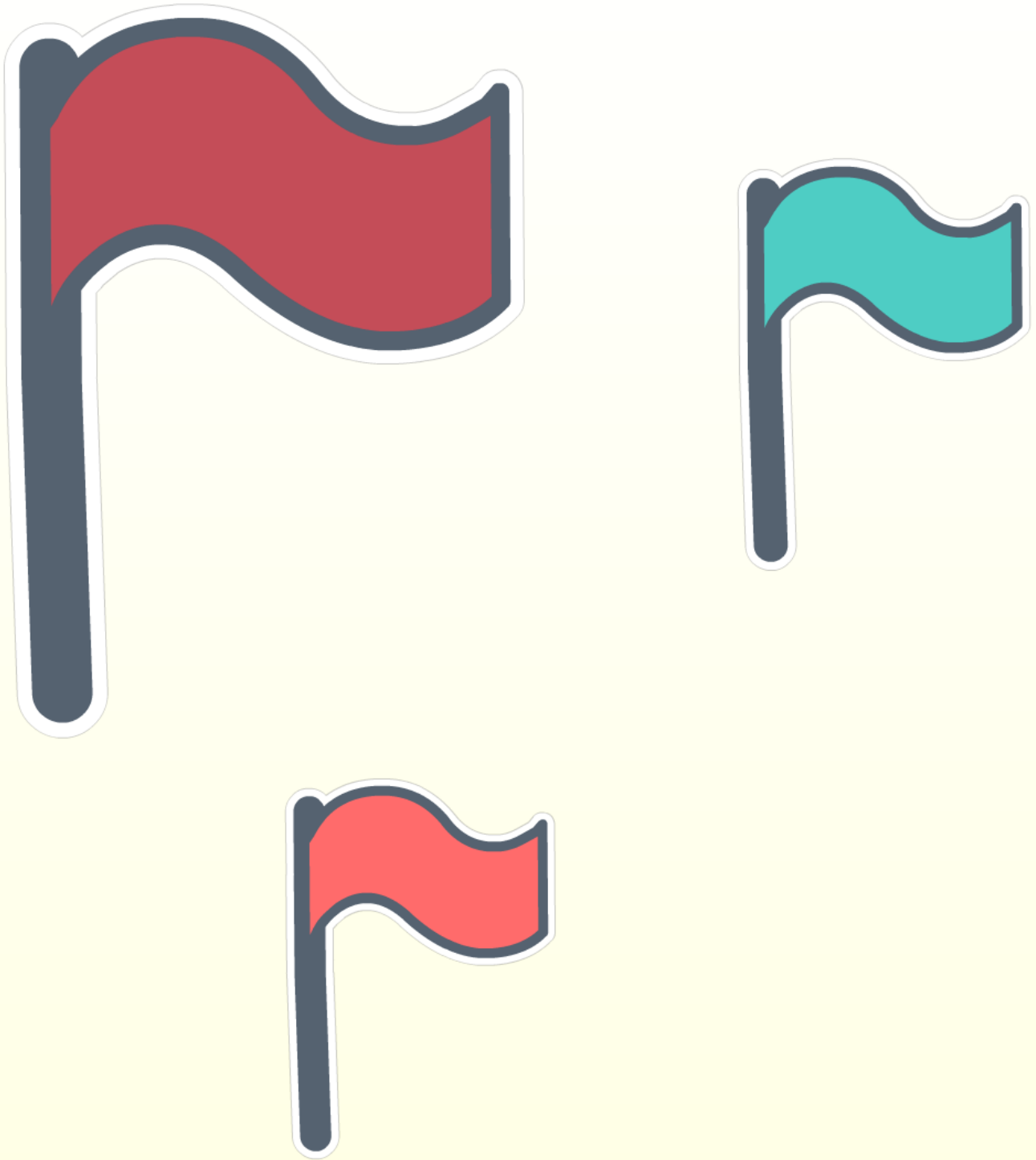
instance

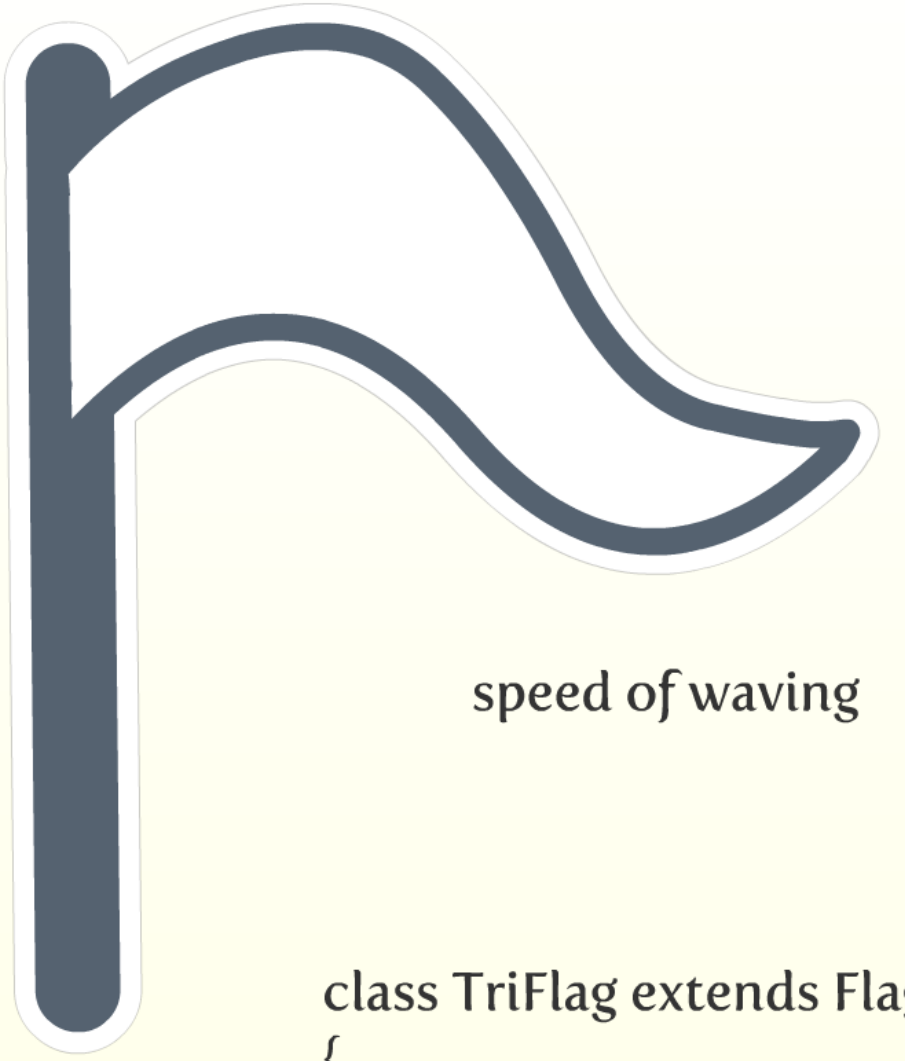
encapsulation

inheritance


color
width
height
length

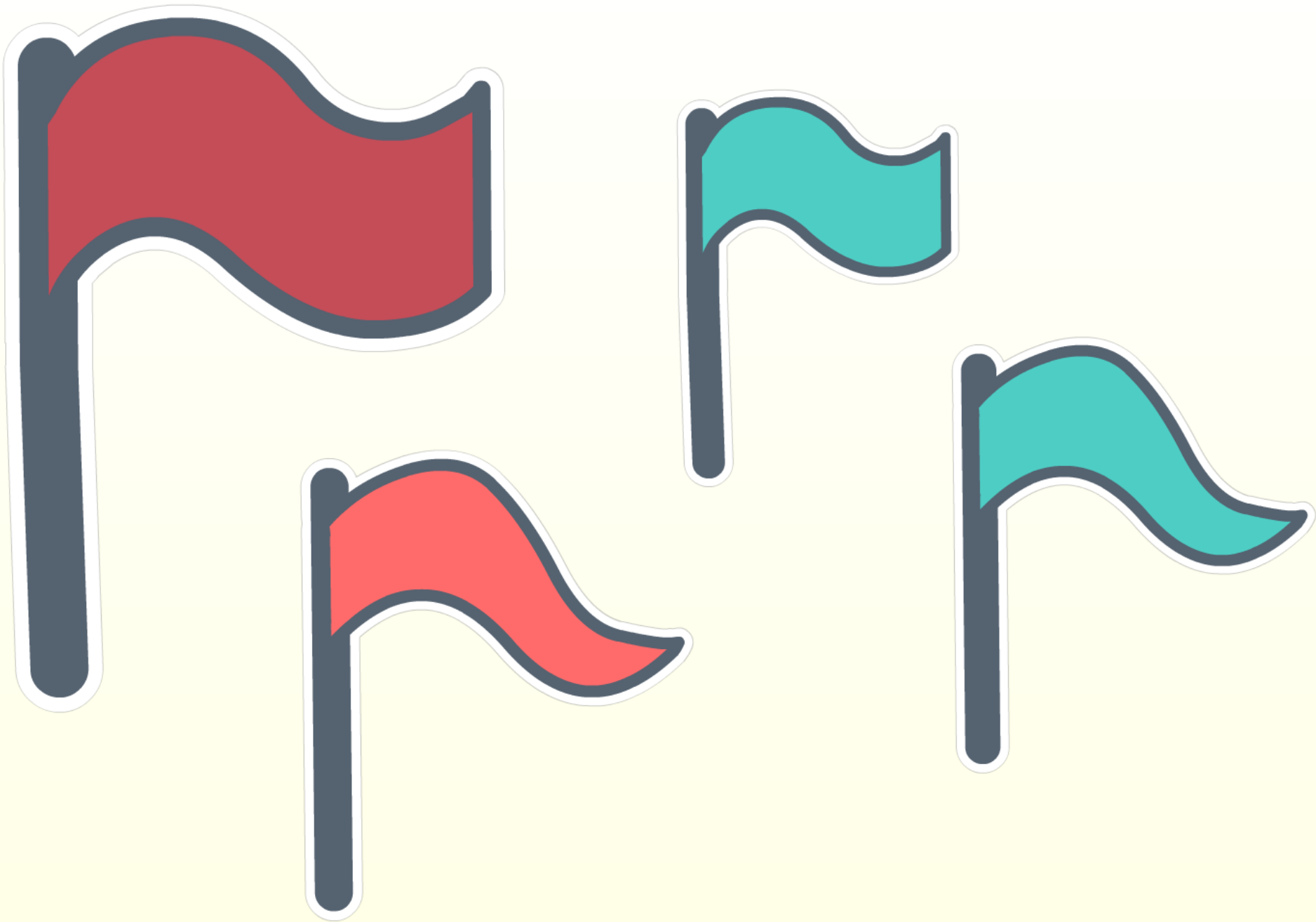
```
class Flag
{
      int width;
      int length;
      int height;
     Color color;

      void setLength(int l);
      ..........
      void wave();
}
```





speed of waving

```
class TriFlag extends Flag
{
     int speed;
     void setLength(int l);
     .....
     void wave();
}
```



}

class

instance

message passing

abstraction

encapsulation

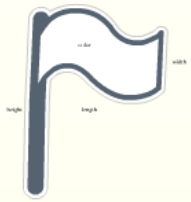
inheritance

polymorphism

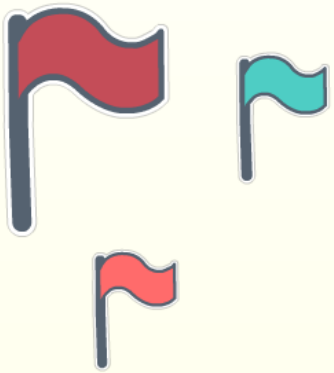
Gang of Four design patterns

Object Oriented Programming

program manipulate collection of objects

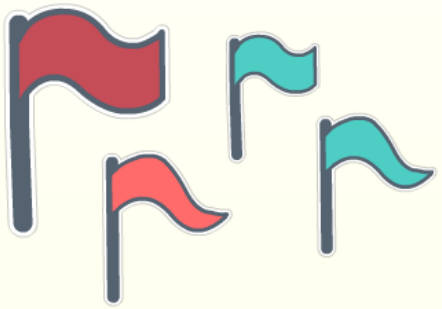


```
class Flag
{
    int width;
    int length;
    int height;
    Color color;

    void setLength(int l);
    ---------
    void wave();
}
```



speed of waving

```
class TriFlag extends Flag
{
    int speed;
    void setLength(int l);
    ---------
    void wave();
}
```



class

instance

message passing

abstraction

encapsulation

inheritance

polymorphism

message passing

abstraction

polymorphism

Functional Programming

a problem is decomposed into a set of functions
functions ==> mathematical functions

no side effects

recursion

immutable variables

$$x = x + 5$$

no side effects

functions don't modify internal state

calling the same function
should always give the same results

$$y = f(x) = x+5$$

recursion

```
int factorial(int n)
{
  if (n == 0)
    return 1;
  else
    return(n * factorial(n-1));
}
```

functions defined anywhere

Example: (python)

```
def fx(x):
    def fy(y):
        return y+2
    return fy(x+2)
```

high order functions

$$f(y) = y + 2$$

$$y = f(x) = x + 2$$

function that take function as argument

$$f(y) = y + 2$$
$$y = f(x) = x + 2$$

function that take
function as argument

Example: python

```
def fx(x):
  return x+2;

def fy(func, y):
  return func(y+2)

fy(fx, 2)
```

using lambda function

Example: python

```
def fy(func, y):
    return func(y+2)

fy(lambda x : x + 2, 2)
```

function that return functions

fx(0) ==> fy
fx(2) ==> fz

Example: python

```
def fx(x):
  def fy(y):
     return y+2;

  def fz(z):
    return z+3;

  if x == 0:
    return fy
  else:
    return fz
```

```
fy(lambda x : x + 2, 2)
```

Examples of some built-in high order functions in python:

- sum

```
sum([0,2,3,5])
```

-map

```
arr = [0, 2, 3, 5, 7]
def fx(x):
    return x+2;
```

```
newarr = map(fx, arr)
```

-reduce

```
arr = [0, 2, 3, 5, 7]
def fx(x, y):
    return x+y;
```

newarr = reduce(fx, arr)

Functional Programming

a problem is decomposed into a set of functions
functions ==> mathematical functions

no side effects

immutable variables

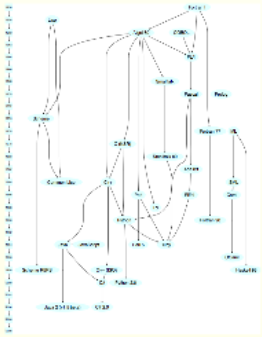
recursion

functions defined anywhere

high order functions

Introduction to Programming Language and Paradigms

Programming?!



Programming Paradigm?!

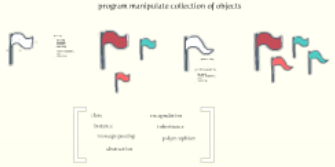
* Style or Way of programming
* Most languages are multi-paradigm
* Main Paradigms:
  - Imperative (How?)
  - Declarative (What?)
    - Database Query Languages (SQL)
    - Functional
    - Logic
  - Object Oriented

Imperative Programming

Computer is told how to solve the problem

Object Oriented Programming

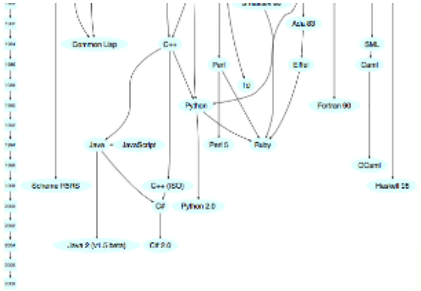
program manipulate collection of objects



Functional Programming

a problem is decomposed into a set of functions
functions ==> mathematical functions

Selection Criteria



Selection Criteria

ability

productivity

maintenance

understandability

time to production

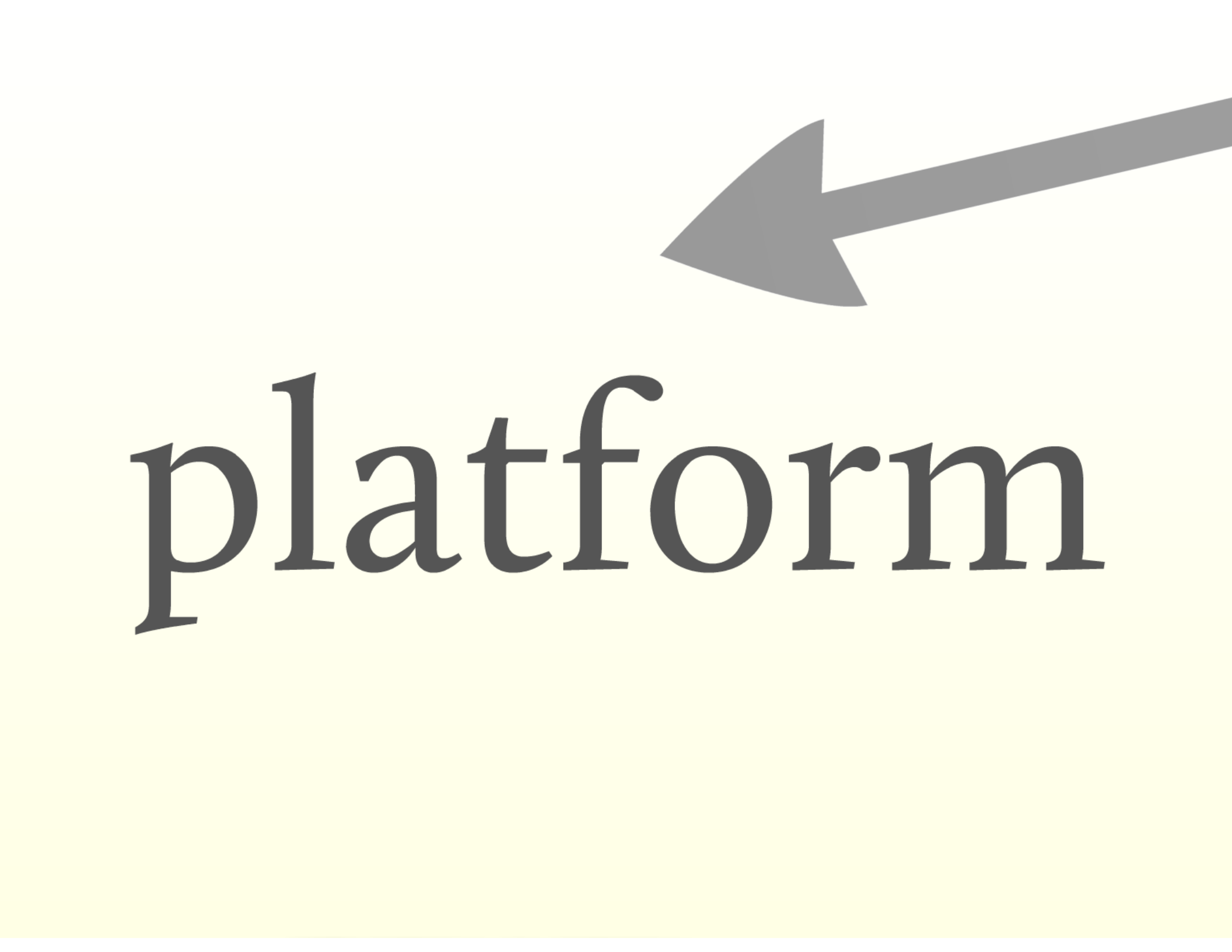
Ramp up time

c

ping

Others

Line of codes?
compiled or interpreted?
IDE availability?
refactoring ability?

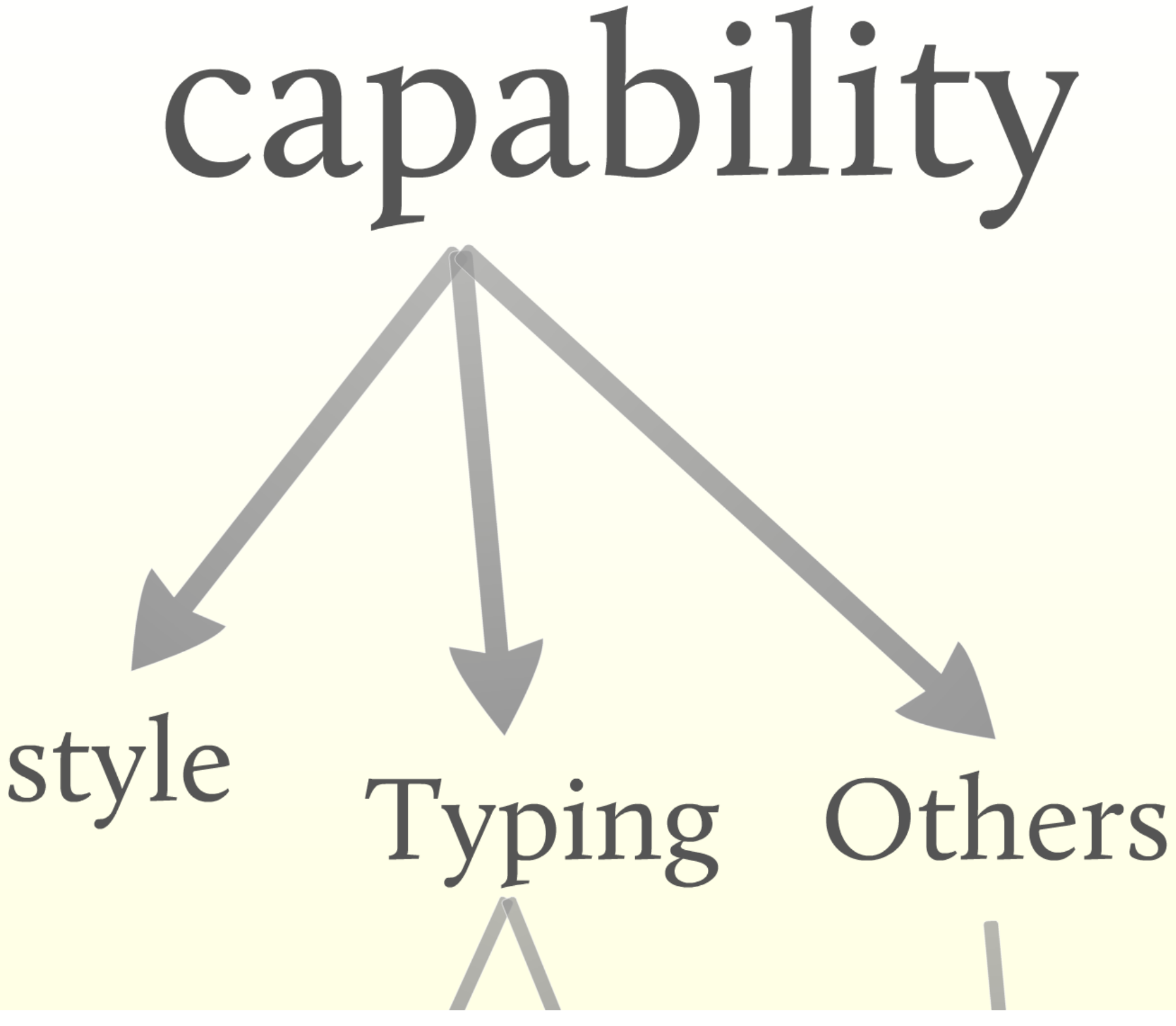

platform

Windows, Linux, Mac, Web, Android, iOS, ...

ex: Java binary is cross platform

C is cross platform, needs to be compiled for different OSs

C# only Windows, mono project for other OSs

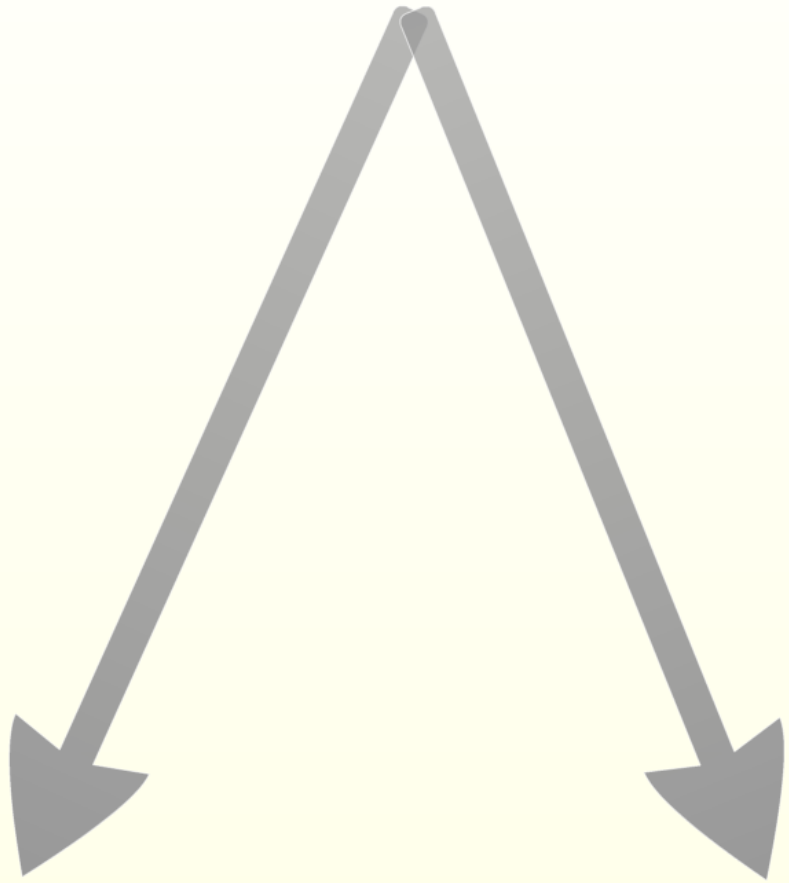

capability
style
Typing
Others


style

yle

Typing

Ot



static

ex: C/C++, Java, C#

```
int x = 0;
String x = "test";
```

dynamic

ex: Ruby, Python

```
x = 0
x = "test"
```

Ga

static

ex: C/C++, Java, C#

```
int x = 0;
String x = "test";
```

dynamic

ex: Ruby, Python

```
x = 0
x = "test"
```

Garbage Collected?

exception handling?

multi-threading?

generic classes?

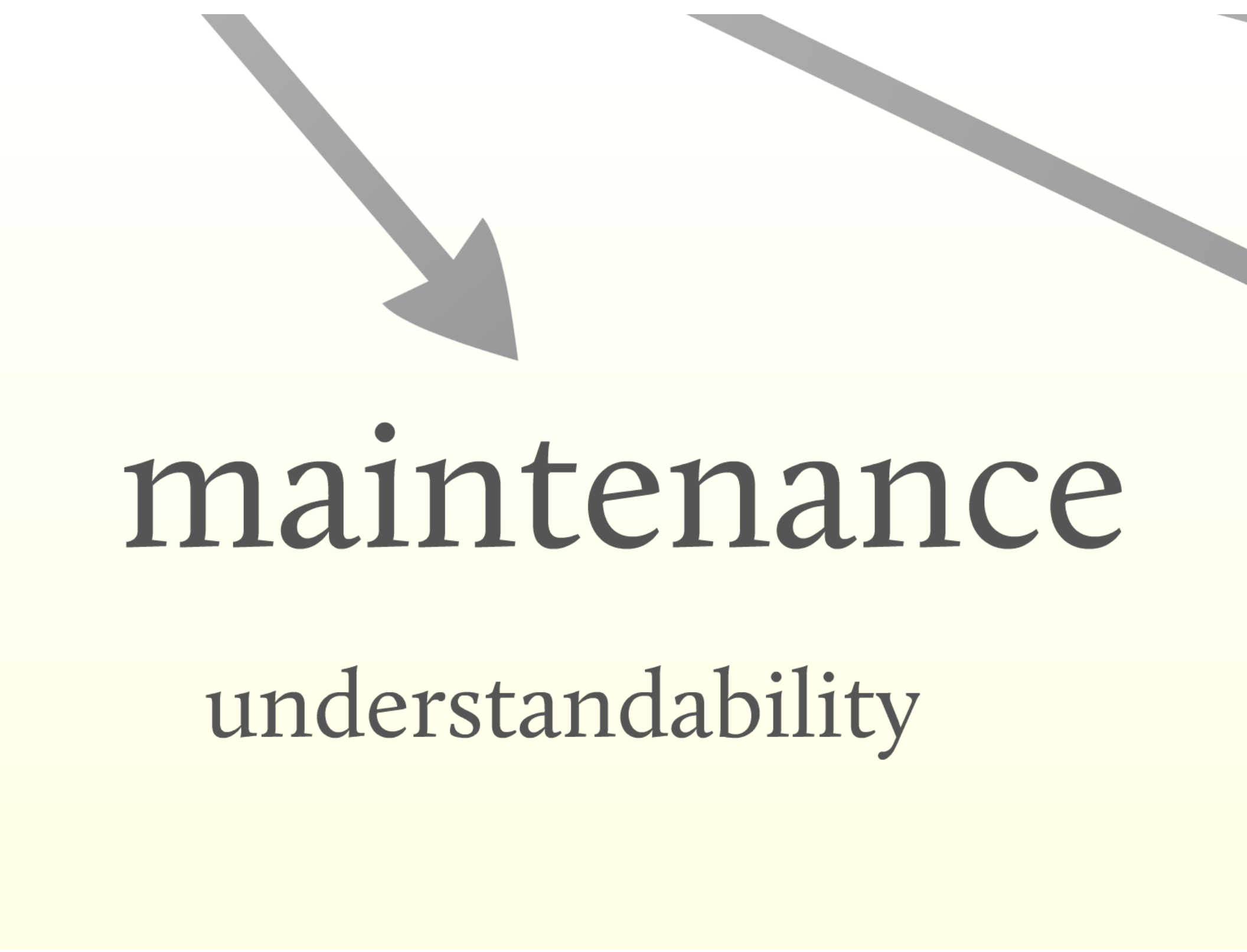
Reflection?

metaprogramming?

standard library?

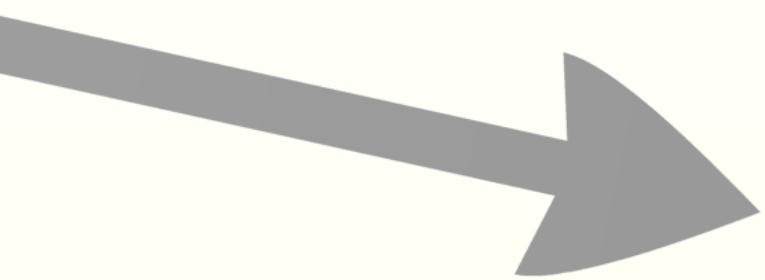

productivity

Line of codes?
compiled or interpreted?
IDE availability?
refactoring ability?

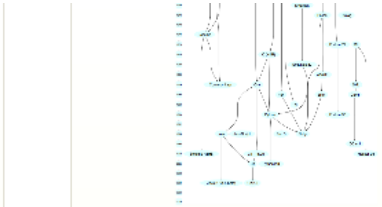

maintenance
understandability


time to
production
Ramp up time


community



performance



Selection Criteria

platform

capability

- style
- Typing
  - static
  - dynamic
- Others
  - Garbage Collected?
  - exception handling?
  - multi-threading?
  - generic classes?
  - Reflection?
  - metaprogramming?
  - standard library?

productivity

maintenance

- understandability

time to production

- Ramp up time

community

performance

Trending Topics

- Functional programming
- Concurrent and distributed programming
- Component oriented software development
- Aspect oriented programming
- Rise of JVM based languages (Jython, JRuby, ..)
- New Languages for web programming
    - Dart, Opa, CoffeeScript

Conclusion

- Overview about main programming paradigms
- How to select your programming language
- Advices:
  - don't stick with one language
  - try different paradigms especially functional

References

- http://www.slideshare.net/dnene/programming-language-selection
- http://rigaux.org/language-study/diagram.html
- http://www.ibm.com/developerworks/web/library/wa-optimal/index.html
- http://www.psl.cs.columbia.edu/classes/cs4156-f10/lectures/Programming%20Language%20Selection.pdf
- http://docs.python.org/2/howto/functional.html


ming Language & Paradigm Selection
Mahmoud Ismail
Software Enginer, Peerialism
Agenda
* Introduction
* Selection Criteria
* Trending Topics
* Conclusion