Programming Languages are Not the Same

Dr.Haitham A. El-Ghareeb

Information Systems Department
Faculty of Computers and Information Sciences
Mansoura University

helghareeb@gmail.com

April 22, 2013

Peter Norvig

- 1996-1998: Junglee Corp.
  - Chief Scientist
- 1994-1996: Harlequin, Inc.
  - Chief Designer
- 1991-1994: Sun Microsystems Labs
  - Senior Scientist
- 1986-1991: University of California, Berkeley
  - Research Faculty Member
- 1985-1986: University of Southern California
  - Assistant Professor
- 1978-1980: Higher Order Software, Inc.
  - Member of Technical Staff
- 1977-1977: Woods Hole Oceanographic Institute
  - Summer Programming Intern

Teach Yourself Programming in Ten Years

Peter Norvig

- 2001-now: Google
  - Director of Research (2006-now);
  - formerly Director of Search Quality (2002-2006)
  - and Machine Learning (2001)
- 1998-2001: NASA Ames Research Center
  - Division Chief, Computational Sciences

Why is everyone in such a rush?

Why is everyone in such a rush?

- Learn ANY Programming Language in HOURS / DAYS

Why is everyone in such a rush?

- Learn ANY Programming Language in HOURS / DAYS
- There are no books on how to learn Beethoven, or Quantum Physics, or even Dog Grooming in a few days.

Why is everyone in such a rush?

- Learn ANY Programming Language in HOURS / DAYS
- There are no books on how to learn Beethoven, or Quantum Physics, or even Dog Grooming in a few days.
- Let's analyze what a title like Learn C++ in Three Days could mean:

Learn

- In 3 days you won't have time to write several significant programs, and learn from your successes and failures with them.
- You won't have time to work with an experienced programmer and understand what it is like to live in a C++ environment.
- In short, you won't have time to learn much.
- So the book can only be talking about a superficial familiarity, not a deep understanding.
- As Alexander Pope said, a little learning is a dangerous thing.

C++

- In 3 days you might be able to learn some of the syntax of C++ (if you already know another language), but you couldn't learn much about how to use the language.
- In short, if you were, say, a Basic programmer, you could learn to write programs in the style of Basic using C++ syntax, but you couldn't learn what C++ is actually good (and bad) for.
- So what's the point? Alan Perlis once said: "A language that doesn't affect the way you think about programming, is not worth knowing".

3 Days

- in Three Days: Unfortunately, this is not enough, as the next section shows.

Recipe


كلية الحاسبات والمعلومات
CiS
Faculty of Computers and Information Sciences


جامعة المنصورة
Mansoura University

Recipe

- Get interested in programming, and do some because it is fun.

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.
- Learn at least a half dozen programming languages.

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.
- Learn at least a half dozen programming languages.
  - Include one language that supports class abstractions (like Java or C++),

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.
- Learn at least a half dozen programming languages.
  - Include one language that supports class abstractions (like Java or C++),
  - one that supports functional abstraction (like Lisp or ML),

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.
- Learn at least a half dozen programming languages.
  - Include one language that supports class abstractions (like Java or C++),
  - one that supports functional abstraction (like Lisp or ML),
  - one that supports syntactic abstraction (like Lisp),

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.
- Learn at least a half dozen programming languages.
  - Include one language that supports class abstractions (like Java or C++),
  - one that supports functional abstraction (like Lisp or ML),
  - one that supports syntactic abstraction (like Lisp),
  - one that supports declarative specifications (like Prolog or C++ templates),

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.
- Learn at least a half dozen programming languages.
  - Include one language that supports class abstractions (like Java or C++),
  - one that supports functional abstraction (like Lisp or ML),
  - one that supports syntactic abstraction (like Lisp),
  - one that supports declarative specifications (like Prolog or C++ templates),
  - one that supports coroutines (like Icon or Scheme),

Recipe

- Get interested in programming, and do some because it is fun.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- Work on projects with other programmers. Be the best programmer on some projects; be the worst on some others.
- Learn at least a half dozen programming languages.
  - Include one language that supports class abstractions (like Java or C++),
  - one that supports functional abstraction (like Lisp or ML),
  - one that supports syntactic abstraction (like Lisp),
  - one that supports declarative specifications (like Prolog or C++ templates),
  - one that supports coroutines (like Icon or Scheme),
  - and one that supports parallelism (like Sisal).

Recipe


كلية الحاسبات والمعلومات
CiS
Faculty of Computers and Information Sciences


جامعة المنصورة
Mansoura University

Recipe

- Remember that there is a "computer" in "computer science".

Recipe

- Remember that there is a "computer" in "computer science".
  - Know how long it takes your computer to execute an instruction,

Recipe

- Remember that there is a "computer" in "computer science".
  - Know how long it takes your computer to execute an instruction,
  - fetch a word from memory (with and without a cache miss),

Recipe

- Remember that there is a "computer" in "computer science".
  - Know how long it takes your computer to execute an instruction,
  - fetch a word from memory (with and without a cache miss),
  - read consecutive words from disk,

Recipe

- Remember that there is a "computer" in "computer science".
  - Know how long it takes your computer to execute an instruction,
  - fetch a word from memory (with and without a cache miss),
  - read consecutive words from disk,
  - and seek to a new location on disk.

Recipe

- Remember that there is a "computer" in "computer science".
  - Know how long it takes your computer to execute an instruction,
  - fetch a word from memory (with and without a cache miss),
  - read consecutive words from disk,
  - and seek to a new location on disk.
- Get involved in a language standardization effort. It could be the ANSI C++ committee, or it could be deciding if your local coding style will have 2 or 4 space indentation levels.

C# OR Python

JAVA May be!

or whatever!

Programming Languages

Programming Languages

- used for controlling the behavior of a machine (often a computer).

Programming Languages

- used for controlling the behavior of a machine (often a computer).
- programming languages conform to rules for syntax and semantics.

Programming Languages

- used for controlling the behavior of a machine (often a computer).
- programming languages conform to rules for syntax and semantics.
- There are thousands of programming languages and new ones are created every year.

Programming Languages

- used for controlling the behavior of a machine (often a computer).
- programming languages conform to rules for syntax and semantics.
- There are thousands of programming languages and new ones are created every year.
- Few languages ever become sufficiently popular that they are used by more than a few people

Programming Languages

- used for controlling the behavior of a machine (often a computer).
- programming languages conform to rules for syntax and semantics.
- There are thousands of programming languages and new ones are created every year.
- Few languages ever become sufficiently popular that they are used by more than a few people
- professional programmers may use dozens of languages in a career.

Basis for Comparing Programming Languages

1. Object Orientation
2. Static vs. Dynamic Typing
3. Generic Classes
4. Inheritence
5. Feature Renaming
6. Method Overloading
7. Operator Overloading
8. Higher Order Functions & Lexical Closures
9. Garbage Collection

Basis for Comparing Programming Languages (Cont.)

10. Uniform Access
11. Class Variables/Methods
12. Reflection
13. Access Control
14. Design by Contract
15. Multithreading
16. Regular Expressions
17. Pointer Arithmetic
18. Language Integration
19. Built-In Security

Object Orientation

There are several qualities for Object Oriented Languages:

Object Orientation

There are several qualities for Object Oriented Languages:

- Encapsulation / Information Hiding

Object Orientation

There are several qualities for Object Oriented Languages:

- Encapsulation / Information Hiding
- Inheritance

Object Orientation

There are several qualities for Object Oriented Languages:

- Encapsulation / Information Hiding
- Inheritance
- Polymorphism/Dynamic Binding

Object Orientation

There are several qualities for Object Oriented Languages:

- Encapsulation / Information Hiding
- Inheritance
- Polymorphism/Dynamic Binding
- All pre-defined types are Objects

Object Orientation

There are several qualities for Object Oriented Languages:

- Encapsulation / Information Hiding
- Inheritance
- Polymorphism/Dynamic Binding
- All pre-defined types are Objects
- All operations performed by sending messages to Objects

Object Orientation

There are several qualities for Object Oriented Languages:

- Encapsulation / Information Hiding
- Inheritance
- Polymorphism/Dynamic Binding
- All pre-defined types are Objects
- All operations performed by sending messages to Objects
- All user-defined types are Objects

Static vs. Dynamic Typing

Static vs. Dynamic Typing

- The debate between static and dynamic typing has raged in Object-Oriented circles for many years with no clear conclusion.

Static vs. Dynamic Typing

- The debate between static and dynamic typing has raged in Object-Oriented circles for many years with no clear conclusion.
- Proponents of dynamic typing contend that it is more flexible and allows for increased productivity.

Static vs. Dynamic Typing

- The debate between static and dynamic typing has raged in Object-Oriented circles for many years with no clear conclusion.
- Proponents of dynamic typing contend that it is more flexible and allows for increased productivity.
- Those who prefer static typing argue that it enforces safer, more reliable code, and increases efficiency of the resulting product.

Static vs. Dynamic Typing

- The debate between static and dynamic typing has raged in Object-Oriented circles for many years with no clear conclusion.
- Proponents of dynamic typing contend that it is more flexible and allows for increased productivity.
- Those who prefer static typing argue that it enforces safer, more reliable code, and increases efficiency of the resulting product.
- A dynamic type system does not require variables to be declared as a specific type. Any variable can contain any value or object.

Static vs. Dynamic Typing

- The debate between static and dynamic typing has raged in Object-Oriented circles for many years with no clear conclusion.
- Proponents of dynamic typing contend that it is more flexible and allows for increased productivity.
- Those who prefer static typing argue that it enforces safer, more reliable code, and increases efficiency of the resulting product.
- A dynamic type system does not require variables to be declared as a specific type. Any variable can contain any value or object.
- Statically-typed languages require that all variables are declared with a specific type.

Generic Classes

Generic Classes

- Refer to the ability to parameterize a class with specific data types.

Generic Classes

- Refer to the ability to parameterize a class with specific data types.
- A common example is a stack class that is parameterized by the type of elements it contains.

Generic Classes

- Refer to the ability to parameterize a class with specific data types.
- A common example is a stack class that is parameterized by the type of elements it contains.
- it allows statically typed languages to retain their compile-time type safety

Generic Classes

- Refer to the ability to parameterize a class with specific data types.
- A common example is a stack class that is parameterized by the type of elements it contains.
- it allows statically typed languages to retain their compile-time type safety
- Dynamically typed languages do not need parameterized types in order to support generic programming

Inheritence

Inheritence

- the ability for a class or object to be defined as an extension or specialization of another class or object.

Inheritence

- the ability for a class or object to be defined as an extension or specialization of another class or object.
- Most object-oriented languages support class-based inheritance

Inheritence

- the ability for a class or object to be defined as an extension or specialization of another class or object.
- Most object-oriented languages support class-based inheritance
- others such as SELF and JavaScript support object-based inheritance

Inheritence

- the ability for a class or object to be defined as an extension or specialization of another class or object.
- Most object-oriented languages support class-based inheritance
- others such as SELF and JavaScript support object-based inheritance
- A few languages, notably Python and Ruby, support both class and object-based inheritance, in which a class can inherit from another class and individual objects can be extended at run time

Inheritence

- the ability for a class or object to be defined as an extension or specialization of another class or object.
- Most object-oriented languages support class-based inheritance
- others such as SELF and JavaScript support object-based inheritance
- A few languages, notably Python and Ruby, support both class and object-based inheritance, in which a class can inherit from another class and individual objects can be extended at run time
- Though there are identified and classified as many as 17 different forms of inheritance. Even so, most languages provide only a few syntactic constructs for inheritance

Inheritence

- the ability for a class or object to be defined as an extension or specialization of another class or object.
- Most object-oriented languages support class-based inheritance
- others such as SELF and JavaScript support object-based inheritance
- A few languages, notably Python and Ruby, support both class and object-based inheritance, in which a class can inherit from another class and individual objects can be extended at run time
- Though there are identified and classified as many as 17 different forms of inheritance. Even so, most languages provide only a few syntactic constructs for inheritance
- Multiple Inheritence

Feature Renaming

Feature renaming is the ability for a class or object to rename one of its features (a term used to collectively refer to attributes and methods) that it inherited from a super class.

Method Overloading

- Method overloading (also referred to as parametric polymorphism) is the ability for a class, module, or other scope to have two or more methods with the same name.
- Calls to these methods are disambiguated by the number and/or type of arguments passed to the method at the call site.

Operator Overloading

ability for a programmer to define an operator (such as +, or *) for user-defined types.

Higher Order Functions and Lexical Closures

Higher Order Functions:

- functions that can be treated as if they were data objects.
  - they can be bound to variables (including the ability to be stored in collections)
  - can be passed to other functions as parameters
  - can be returned as the result of other functions

Lexical Closure: bundling up the lexical (static) scope surrounding the function with the function itself, so that the function carries its surrounding environment around with it wherever it may be used.

Garbage Collection

mechanism allowing a language implementation to free memory of unused objects on behalf of the programmer

- Reference Counting
- Mark and Sweep
- Generational

Uniform Access

All services offered by a module should be available through a uniform notation, which does not betray whether they are implemented through storage or through computation.

Class Variables/Methods

Class variables and methods are owned by a class, and not any particular instance of a class. This means that for however many instances of a class exist at any given point in time, only one copy of each class variable/method exists and is shared by every instance of the class.

Reflection

ability for a program to determine various pieces of information about an object at run-time. Most object-oriented languages support some form of reflection. This includes the ability to determine:

- the type of the object,
- its inheritance structure,
- and the methods it contains, including the number and types of parameters and return types.
- It might also include the ability for determining the names and types of attributes of the object.

Access Control

ability for a modules implementation to remain hidden behind its public interface.

Design by Contract

Design by Contract (DBC) is the ability to incorporate important aspects of a specification into the software that is implementing it. The most important features of DBC are:

- Pre-conditions, which are conditions that must be true before a method is invoked
- Post-conditions, which are conditions guaranteed to be true after the invocation of a method
- Invariants, which are conditions guaranteed to be true at any stable point during the lifetime of an object

Multithreading

ability for a single process to process two or more tasks concurrently.

Regular Expressions

pattern matching constructs capable of recognizing the class of languages known as regular languages.

Pointer Arithmetic

ability for a language to directly manipulate memory addresses and their contents.

Language Integration

It is important for a high level language (particularly interpreted languages) to be able to integrate seamlessly with other languages.

Built-In Security

language implementations ability to determine whether or not a piece of code comes from a trusted source (such as the users hard disk), limiting the permissions of the code if it does not.