Programming Languages are Not the Same

Thoughts and Discussion

Haitham A. El-Ghareeb

September 22, 2019

Faculty of Computers and Information Sciences
Mansoura University
Egypt
`helghareeb@mans.edu.eg`

Contacts

- https://www.haitham.ws
- https://youtube.com/helghareeb
- https://www.github.com/helghareeb
- http://eg.linkedin.com/in/helghareeb
- helghareeb@mans.edu.eg

Introduction

Introduction

Objectives

Session Covers

- Which Programming Language !
- How do we compare between PLs?
- What are the Criteria ?
- How do those Criteria Relate to me ?
- What Technology Leaders Think about this Question ?
- What is the "accurate" Question ?

Introduction

This is Not...

This is Not

This is Not

- Object Oriented Programming Course

This is Not

- Object Oriented Programming Course
- Functional Programming Course

This is Not

- Object Oriented Programming Course
- Functional Programming Course
- (Certain) Programming Language Course

This is Not

- Object Oriented Programming Course
- Functional Programming Course
- (Certain) Programming Language Course
- Even a Course !

Introduction

Prerequisites

Prerequisites

- Familiarity with Programming Concepts (Preferred)
- Familiarity with one or more Programming Languages (Preferred)

Introduction

Contents

Contents

Let’s Agree on..

Let’s Agree on..

Definitions

Programming Language

From Wikipedia [1]

[1] https://en.wikipedia.org/wiki/ProgrammingLanguage

Programming Language

From Wikipedia [1]

- A programming language is a **formal computer language** designed to **communicate instructions to a machine**, particularly a computer.

[1] https://en.wikipedia.org/wiki/ProgrammingLanguage

Programming Language

From Wikipedia [1]

- A programming language is a **formal computer language** designed to **communicate instructions to a machine**, particularly a computer.
- Programming languages can be **used to create programs** to **control the behavior** of a machine or to **express algorithms**.

[1] `https://en.wikipedia.org/wiki/ProgrammingLanguage`

Open Source

From Wikipedia [2]

- Computer software with its **source code made available with a license** in which the copyright holder provides the **rights to study, change, and distribute** the software to anyone and for any purpose.

[2] https://en.wikipedia.org/wiki/OpenSourceSoftware

Technical Standard - What

From Wikipedia[3]

- It is usually a **formal document** that establishes uniform engineering or **technical criteria, methods, processes and practices** related to **technical systems**.

[3] https://en.wikipedia.org/wiki/Technical_standard

Technical Standard - What

From Wikipedia[3]

- It is usually a **formal document** that establishes uniform engineering or **technical criteria, methods, processes and practices** related to **technical systems**.
- In contrast, a **custom, convention, company product, corporate standard**, and so forth that becomes **generally accepted and dominant** is often called a **de facto** standard.

[3] https://en.wikipedia.org/wiki/Technical_standard

Technical Standard - Who

From Wikipedia[4]

- A technical standard **may be developed privately** or unilaterally.

[4] `https://en.wikipedia.org/wiki/Technical_standard`

Technical Standard - Who

From Wikipedia[4]

- A technical standard **may be developed privately** or unilaterally.
- Standards can also be **developed by groups** such as trade unions, and trade associations.

[4]https://en.wikipedia.org/wiki/Technical_standard

Technical Standard - Who

From Wikipedia[4]

- A technical standard **may be developed privately** or unilaterally.
- Standards can also be **developed by groups** such as trade unions, and trade associations.
- Standards organizations often have more diverse input and **usually develop voluntary standards**.

[4] `https://en.wikipedia.org/wiki/Technical_standard`

Technical Standard - Who

From Wikipedia[4]

- A technical standard **may be developed privately** or unilaterally.
- Standards can also be **developed by groups** such as trade unions, and trade associations.
- Standards organizations often have more diverse input and **usually develop voluntary standards**.
- The standardization process may be by edict or **may involve the formal consensus of technical experts**.

[4]https://en.wikipedia.org/wiki/Technical_standard

Let’s Agree on..

Thoughts

Theory vs. Product

- Who leads: Academia vs. Standards vs. Industry ?

Theory vs. Product

- Who leads: Academia vs. Standards vs. Industry ?
- Who sticks with standards ?

Theory vs. Product

- Who leads: Academia vs. Standards vs. Industry ?
- Who sticks with standards ?
- Who wins ?

How Many Programming Languages ?

https://www.quora.com/How-many-programming-languages-are-there-in-the-world-of-software

How Many Programming Languages ?

- Hundreds
- New Ones Every Year
- Same Company Supports Different Languages !

Time-line of Programming Languages

https://en.wikipedia.org/wiki/Timeline_of_programming_languages

The Big List of 256 Programming Languages

`https://dzone.com/articles/big-list-256-programming`

Why so Many Programming Languages ?

- https://cs.stackexchange.com/questions/451/why-are-there-so-many-programming-languages
- https://www.reddit.com/r/explainlikeimfive/comments/1jk4jo/eli5_why_are_there_so_many_programming_languages/

Programming Languages Comparison

Programming Languages Comparison

Why We Need To Compare ?

Why we Need to Compare ?

- Eventually, we have to choose !
- We need to chose Only One... for the Task
- We don't have enough time to learn them all !

Programming Languages Comparison

How Do We Compare ?

Comparison Criteria

- Many
- No Standards !
- Even in Academia !!
- Industry Benchmarks !!!

Comparison Criteria - for us

1. Academic
2. Programming Paradigms
3. General Characteristics
4. Market Share / Adoption / Penetration

Academic

How Do Researchers Compare ?

https://scholar.google.com/

Programming Paradigms

Programming Paradigm

From Wikipedia [5]

- Way to classify programming languages based on their features.
- Languages can be classified into multiple paradigms.

[5] `https://en.wikipedia.org/wiki/ProgrammingParadigm`

Comparison of Programming Paradigms

https://en.wikipedia.org/wiki/Comparison_of_programming_paradigms

Programming Paradigms for Programmers
What Every Programmer Should Know

http://hiperc.buffalostate.edu/courses/ACM612-F15/uploads/ACM612/VanRoy-Programming.pdf

Examples of Programming Paradigms

1. Imperative
2. Structured
3. Procedural
4. Functional
5. Event-Driven
6. Object-Oriented
7. Declarative
8. Reactive
9. Others

Programming Paradigms

Imperative Programming

Imperative Programming

From Wikipedia [6]

- Programming paradigm that uses statements that change a program's state.
- Imperative program consists of commands for the computer to perform.

[6] `https://en.wikipedia.org/wiki/ImperativeProgramming`

State (Computer Science)

From Wikipedia [7]

- Program is described as **stateful** if it is designed to remember **preceding events** or **user interactions**; the remembered information is called the **state of the system**.
- The set of states a system can occupy is known as its **state space**.
- In a **discrete system**, the state space is countable and often finite

[7] https://en.wikipedia.org/wiki/State

Simple Statements in Python

https://docs.python.org/3.7/reference/simple_stmts.html

Expression (Computer Science - I)

From Wikipedia [8]

- **Expression** is a combination of one or more explicit values, constants, variables, operators, and functions that the programming language interprets and computes to produce (”to return”, in a stateful environment) another value.

[8] https://en.wikipedia.org/wiki/Expression

Expression (Computer Science - I)

From Wikipedia [8]

- **Expression** is a combination of one or more explicit values, constants, variables, operators, and functions that the programming language interprets and computes to produce (”to return”, in a stateful environment) another value.
- For example, $2 + 3$ is an arithmetic and programming expression which evaluates to 5.

[8] `https://en.wikipedia.org/wiki/Expression`

Expression (Computer Science - II)

- A variable is an expression because it denotes a value in memory, so $y + 6$ is an expression.

Expression (Computer Science - II)

- A variable is an expression because it denotes a value in memory, so $y + 6$ is an expression.
- An example of a relational expression is $4 \neq 4$, which evaluates to false.

Statement vs. Expression - I

From [9]

- **Statement** is a complete line of code that performs some action

[9] https://www.quora.com/Whats-the-difference-between-a-statement-and-an-expression-in-Python

Statement vs. Expression - I

From [9]

- **Statement** is a complete line of code that performs some action
- **Expression** is any section of the code that evaluates to a value

[9] https://www.quora.com/Whats-the-difference-between-a-statement-and-an-expression-in-Python

Statement vs. Expression - I

From [9]

- **Statement** is a complete line of code that performs some action
- **Expression** is any section of the code that evaluates to a value
- **Statements** can only be combined “vertically” by writing one after another, or with block constructs.

[9] https://www.quora.com/Whats-the-difference-between-a-statement-and-an-expression-in-Python

Statement vs. Expression - I

From [9]

- **Statement** is a complete line of code that performs some action
- **Expression** is any section of the code that evaluates to a value
- **Statements** can only be combined “vertically” by writing one after another, or with block constructs.
- **Expressions** can be combined “horizontally” into larger expressions using operators

[9] https://www.quora.com/Whats-the-difference-between-a-statement-and-an-expression-in-Python

Statement vs. Expression - II

- Every **expression** can be used as a statement (whose effect is to evaluate the expression and ignore the resulting value)
- Most **statements** cannot be used as expressions

Imperative Programming Languages - Examples

From [10]

- Most of the mainstream languages, including object-oriented programming (OOP) languages such as C#, Visual Basic, C++, and Java, were designed to primarily support imperative (procedural) programming.

[10] https://stackoverflow.com/questions/17826380/what-is-difference-between-functional-and-imperative-programming-languages

Programming Paradigms

Structured Programming

Structured Programming - I

From Wikipedia [11]

- Programming paradigm aimed at improving the clarity, quality, and development time of a computer program

[11] `https://en.wikipedia.org/wiki/StructuredProgramming`

Structured Programming - I

From Wikipedia [11]

- Programming paradigm aimed at improving the clarity, quality, and development time of a computer program
- Making extensive use of subroutines, block structures, for and while loops—in contrast to using simple tests and jumps such as the **goto** statement

[11] `https://en.wikipedia.org/wiki/StructuredProgramming`

Structured Programming - II

- It emerged in the late 1950s with the appearance of the ALGOL 58 and ALGOL 60 programming languages

[12] https://en.wikipedia.com/wiki/ProgrammingParadigm

Structured Programming - II

- It emerged in the late 1950s with the appearance of the ALGOL 58 and ALGOL 60 programming languages
- C, C++, Java, Python are Structured Programming Languages [12]

[12] https://en.wikipedia.com/wiki/ProgrammingParadigm

Programming Paradigms

Procedural Programming

Procedural Programming

From Wikipedia [13]

- Derived from Structured programming

[13] https://en.wikipedia.org/wiki/ProceduralProgramming

Procedural Programming

From Wikipedia [13]

- Derived from Structured programming
- Based upon the concept of the **procedure call**.

[13] https://en.wikipedia.org/wiki/ProceduralProgramming

Procedural Programming

From Wikipedia [13]

- Derived from Structured programming
- Based upon the concept of the **procedure call**.
- Procedures, also known as routines, subroutines, or functions simply contain a **series of computational steps to be carried out**.

[13] `https://en.wikipedia.org/wiki/ProceduralProgramming`

Procedural Programming

From Wikipedia [13]

- Derived from Structured programming
- Based upon the concept of the **procedure call**.
- Procedures, also known as routines, subroutines, or functions simply contain a **series of computational steps to be carried out**.
- Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

[13]`https://en.wikipedia.org/wiki/ProceduralProgramming`

Procedural Programming

From Wikipedia [13]

- Derived from Structured programming
- Based upon the concept of the **procedure call**.
- Procedures, also known as routines, subroutines, or functions simply contain a **series of computational steps to be carried out**.
- Any given procedure might be called at any point during a program's execution, including by other procedures or itself.
- Pascal, C, C++, Ada, Lisp, PHP, Python, and Go

[13] `https://en.wikipedia.org/wiki/ProceduralProgramming`

Programming Paradigms

Functional Programming

Functional Programming - I

From Wikipedia [14]

- Treats computation as the **evaluation of mathematical functions** and **avoids changing-state** and **mutable data**.

[14] `https://en.wikipedia.org/wiki/FunctionalProgramming`

Functional Programming - I

From Wikipedia [14]

- Treats computation as the **evaluation of mathematical functions** and **avoids changing-state** and **mutable data**.
- Programming is done with expressions or declarations instead of statements.

[14] https://en.wikipedia.org/wiki/FunctionalProgramming

Functional Programming - I

From Wikipedia [14]

- Treats computation as the **evaluation of mathematical functions** and **avoids changing-state** and **mutable data**.
- Programming is done with expressions or declarations instead of statements.
- In functional code, the output value of a function depends only on the arguments that are passed to the function, so calling a function *f* twice with the same value for an argument *x* will produce the same result *f(x)* each time.

[14] `https://en.wikipedia.org/wiki/FunctionalProgramming`

Functional Programming - II

- This is in contrast to procedures depending on a **local or global state**, which may produce different results at different times when called with the **same arguments but a different program state**.

Functional Programming - Must Read

https://wiki.haskell.org/Functional_programming

Some Features of Functional Languages - I

From [15]

- **Higher-order functions**, functions that take other functions as their arguments.

[15] https://wiki.haskell.org/Functional_programming

Some Features of Functional Languages - I

From [15]

- **Higher-order functions**, functions that take other functions as their arguments.
- **Purity**, some functional languages allow expressions to yield actions in addition to return values. These actions are called **side effects**. Languages that prohibit side effects are called **pure**.

[15] `https://wiki.haskell.org/Functional_programming`

Some Features of Functional Languages - I

From [15]

- **Higher-order functions**, functions that take other functions as their arguments.
- **Purity**, some functional languages allow expressions to yield actions in addition to return values. These actions are called **side effects**. Languages that prohibit side effects are called **pure**.
    - **Immutable data**, Instead of altering existing values, altered copies are created and the original is preserved.

[15] `https://wiki.haskell.org/Functional_programming`

Some Features of Functional Languages - II

- **Lazy Evaluation**, computations can be performed at any time and still yield the same result. This makes it possible to defer the computation of values until they are needed.

Some Features of Functional Languages - II

- **Lazy Evaluation**, computations can be performed at any time and still yield the same result. This makes it possible to defer the computation of values until they are needed.
- **Recursion**, often the only way to iterate. Implementations will often include tail call optimization.

Examples of Functional Programming Languages

C++, Clojure, Coffeescript, Elixir, Erlang, F#, Haskell, Lisp, Python, Ruby, Scala, SequenceL, Standard ML, JavaScript

Programming Paradigms

Event-Driven Programming

Event-Driven Programming - I

From Wikipedia [16]

- Flow of the program is determined by **events** such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs / threads.

[16] `https://en.wikipedia.org/wiki/EventDrivenProgramming`

Event-Driven Programming - I

From Wikipedia [16]

- Flow of the program is determined by **events** such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs / threads.
- Dominant paradigm used in graphical user interfaces (GUI) and other applications (e.g. JavaScript web applications) that are centered on performing certain actions in response to user input.

[16] `https://en.wikipedia.org/wiki/EventDrivenProgramming`

Event-Driven Programming - I

From Wikipedia [16]

- Flow of the program is determined by **events** such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs / threads.
- Dominant paradigm used in graphical user interfaces (GUI) and other applications (e.g. JavaScript web applications) that are centered on performing certain actions in response to user input.
- This is also true of programming for Device Drivers, Game Programming

[16] `https://en.wikipedia.org/wiki/EventDrivenProgramming`

Event-Driven Programming - II

- There is generally a main loop that listens for events, and then triggers a callback function when one of those events is detected.

Programming Paradigms

Object Oriented Programming

Object Oriented Programming

From Wikipedia [17]

- Based on the concept of **objects**, which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.

[17] `https://en.wikipedia.org/wiki/ObjectOrientedProgramming`

Object Oriented Programming

From Wikipedia [17]

- Based on the concept of **objects**, which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.
- Programs are designed by making them out of objects that interact with one another.

[17] `https://en.wikipedia.org/wiki/ObjectOrientedProgramming`

Object Oriented Programming

From Wikipedia [17]

- Based on the concept of **objects**, which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.
- Programs are designed by making them out of objects that interact with one another.
- Java, C++, C#, Python, PHP, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, and Smalltalk.

[17] `https://en.wikipedia.org/wiki/ObjectOrientedProgramming`

Object Orientation Characteristics I

- **Encapsulation**, concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse.

[18] https://en.wikipedia.org/wiki/InformationHiding

Object Orientation Characteristics I

- **Encapsulation**, concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse.
- **Data/Information Hiding** [18], ability to prevent certain aspects of a class or software component from being accessible to its clients.

[18]https://en.wikipedia.org/wiki/InformationHiding

Object Orientation Characteristics I

- **Encapsulation**, concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse.
- **Data/Information Hiding** [18], ability to prevent certain aspects of a class or software component from being accessible to its clients.
- **Composition**, Objects can contain other objects in their instance variables.

[18] https://en.wikipedia.org/wiki/InformationHiding

Object Orientation Characteristics II

- **Inheritance**, This allows classes to be arranged in a hierarchy that represents *is-a-type-of* relationships.

Object Orientation Characteristics II

- **Inheritance**, This allows classes to be arranged in a hierarchy that represents *is-a-type-of* relationships.
    - All the data and methods available to the parent class also appear in the child class with the same names.

Object Orientation Characteristics II

- **Inheritance**, This allows classes to be arranged in a hierarchy that represents *is-a-type-of* relationships.
    - All the data and methods available to the parent class also appear in the child class with the same names.
    - Allows easy re-use of the same procedures and data definitions, in addition to potentially mirroring real-world relationships in an intuitive way.

Object Orientation Characteristics III

- **Polymorphism** [19], provision of a single interface to entities of different types.

[19] https://en.wikipedia.com/wiki/Polymorphism

Object Orientation Characteristics III

- **Polymorphism** [19], provision of a single interface to entities of different types.
- A polymorphic type is one whose operations can also be applied to values of some other type, or types.

[19] https://en.wikipedia.com/wiki/Polymorphism

Object Orientation Characteristics - IV

- Several kinds of polymorphism:

Object Orientation Characteristics - IV

- Several kinds of polymorphism:
    - **Ad hoc polymorphism**, function denotes different and potentially heterogeneous implementations depending on a limited range of individually specified types and combinations (*function overloading*).

Object Orientation Characteristics - IV

- Several kinds of polymorphism:
    - **Ad hoc polymorphism**, function denotes different and potentially heterogeneous implementations depending on a limited range of individually specified types and combinations (*function overloading*).
    - **Parametric polymorphism**, code is written without mention of any specific type and can be used transparently with any number of new types (*generics*).

Object Orientation Characteristics - IV

- Several kinds of polymorphism:
    - **Ad hoc polymorphism**, function denotes different and potentially heterogeneous implementations depending on a limited range of individually specified types and combinations (*function overloading*).
    - **Parametric polymorphism**, code is written without mention of any specific type and can be used transparently with any number of new types (*generics*).
    - **Subtyping**, name denotes instances of different classes related by some common superclass (*polymorphism*).

Object Orientation Characteristics V

- **Dynamic Binding**, linking procedure call to a specific sequence of code (method) at run-time.

Object Orientation Characteristics V

- **Dynamic Binding**, linking procedure call to a specific sequence of code (method) at run-time.
    - It means that the code to be executed for a specific procedure call is not known until run-time.

Object Orientation Characteristics V

- **Dynamic Binding**, linking procedure call to a specific sequence of code (method) at run-time.
    - It means that the code to be executed for a specific procedure call is not known until run-time.
    - Dynamic binding is also known as late binding or run-time binding.

Object Orientation Characteristics V

- **Dynamic Binding**, linking procedure call to a specific sequence of code (method) at run-time.
    - It means that the code to be executed for a specific procedure call is not known until run-time.
    - Dynamic binding is also known as late binding or run-time binding.
- All predefined types are Objects

Object Orientation Characteristics V

- **Dynamic Binding**, linking procedure call to a specific sequence of code (method) at run-time.
    - It means that the code to be executed for a specific procedure call is not known until run-time.
    - Dynamic binding is also known as late binding or run-time binding.
- All predefined types are Objects
- All operations performed by sending messages to Objects

Object Orientation Characteristics V

- **Dynamic Binding**, linking procedure call to a specific sequence of code (method) at run-time.
    - It means that the code to be executed for a specific procedure call is not known until run-time.
    - Dynamic binding is also known as late binding or run-time binding.
- All predefined types are Objects
- All operations performed by sending messages to Objects
- All user defined types are Objects

Programming Paradigms

Declarative Programming

Declarative Programming

From Wikipedia [20]

- Style of building the structure and elements of computer programs — that expresses the logic of a computation without describing its control flow.

[20] https://en.wikipedia.com/wiki/DeclarativeProgramming

Declarative Programming

From Wikipedia [20]

- Style of building the structure and elements of computer programs — that expresses the logic of a computation without describing its control flow.
- SQL, regular expressions, CSS, Prolog, OWL, SPARQL

[20] https://en.wikipedia.com/wiki/DeclarativeProgramming

Programming Paradigms

Reactive Programming

Reactive Programming

- `https://en.wikipedia.org/wiki/Reactive_programming`

Programming Paradigms

Others

Other Programming Paradigms

List continues, to include (*not only*)

- Automata based programming
- Logic
- Symbolic

Further Reading `http://cs.lmu.edu/~ray/notes/paradigms/`

General Characteristics

General Characteristics

- What are the features ?
- Are they equivalent ?
- Do we care about all of them ?

Selected Features

Compiled vs. Interpreted
Garbage Collection
Object Oriented Features
Multi-threading / Concurrency
Design by Contract
Language Integration
Market Share / Adoption
Standard Programming Language
Type System
Functional Programming Support
Pointer Arithmetic
Regular Expressions
Built-in Security

General Characteristics

Compiled vs. Interpreted

Compiled vs. Interpreted Programming Languages

- **Compiled**, implementations are typically compilers (translators that generate machine code from source code), and not interpreters

Compiled vs. Interpreted Programming Languages

- **Compiled**, implementations are typically compilers (translators that generate machine code from source code), and not interpreters
- **Interpreted**, step-by-step executors of source code, where no pre-runtime translation takes place.

Compiled vs. Interpreted Programming Languages

- **Compiled**, implementations are typically compilers (translators that generate machine code from source code), and not interpreters
- **Interpreted**, step-by-step executors of source code, where no pre-runtime translation takes place.
- Compiled then Interpreted

General Characteristics

Standardized Programming Languages

Standardized Programming Languages

- ANSI / ISO Standard

Standardized Programming Languages

- ANSI / ISO Standard
- Important ?

Standardized Programming Languages

- ANSI / ISO Standard
- Important ?
- Different Implementations for the same Programming Language

General Characteristics

Garbage Collection

Garbage Collection - I

- Form of automatic memory management.

Garbage Collection - I

- Form of automatic memory management.
- The garbage collector attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program.

Garbage Collection - I

- Form of automatic memory management.
- The garbage collector attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program.
- Invented by John McCarthy to simplify manual memory management in Lisp.

Garbage Collection - II

Strategies include [21]

[21] https://en.wikipedia.com/wiki/GarbageCollection

Garbage Collection - II

Strategies include [21]

- **Tracing**, strategy consists of determining which objects should be garbage collected by tracing which objects are reachable by a chain of references from certain root objects, and considering the rest as garbage and collecting them.

[21] https://en.wikipedia.com/wiki/GarbageCollection

Garbage Collection - II

Strategies include [21]

- **Tracing**, strategy consists of determining which objects should be garbage collected by tracing which objects are reachable by a chain of references from certain root objects, and considering the rest as garbage and collecting them.
- **Reference Counting**, each object has a count of the number of references to it. Garbage is identified by having a reference count of zero.

[21] https://en.wikipedia.com/wiki/GarbageCollection

Garbage Collection - III

Strategies include (cont.)

- **Escape Analysis**, used to convert heap allocations to stack allocations, thus reducing the amount of work needed to be done by the garbage collector. This is done using a compile-time analysis.

Garbage Collection - III

Strategies include (cont.)

- **Escape Analysis**, used to convert heap allocations to stack allocations, thus reducing the amount of work needed to be done by the garbage collector. This is done using a compile-time analysis.
- **Mark and Sweep**, `http://www.geeksforgeeks.org/mark-and-sweep-garbage-collection-algorithm/`

Garbage Collection - III

Strategies include (cont.)

- **Escape Analysis**, used to convert heap allocations to stack allocations, thus reducing the amount of work needed to be done by the garbage collector. This is done using a compile-time analysis.
- **Mark and Sweep**, `http://www.geeksforgeeks.org/mark-and-sweep-garbage-collection-algorithm/`
- **Generational**, `http://wiki.c2.com/?GenerationalGarbageCollection`

General Characteristics

Type System

Type System Classification

From [22]

- Static vs. Dynamic
- Strong vs. Weak

[22] https://stackoverflow.com/questions/2351190/static-dynamic-vs-strong-weak

Static vs. Dynamic Type Checking

- about **when** type information is acquired
    - **Static**, variables are checked at compile-time, (should) remain the same, and requires well-defined type system (variables adhere to restrictions). No possibility of run-time error.
    - **Dynamic**, variables change, and does not require a specific type system. Type checking happens at run-time.

Strong vs. Weak Typed

- about **how strictly** types are distinguished.
    - **Strong**, Programming Language raises errors when data types are not compatible.
    - **Weak**, Programming Language tries to do implicit conversions.

General Characteristics

Object Oriented Programming Features Support

Access Control

Ability for a modules implementation to remain hidden behind its public interface

Generic Classes

- aka Parametric Type
- ex. Stack Class Parameterized with what it Contains
- Allows statically typed languages to retain their compile-time type safety yet remain nearly as flexible as dynamically typed languages.
- Dynamically typed languages support generic programming inherently

Inheritance

- Multiple Inheritance
- **Prototypal Inheritance**, objects inherit from objects.
- `http://javascript.crockford.com/prototypal.html`

Feature Renaming

- Attribute / Method
- Provide a feature with a more natural name for its new context
- Resolve naming ambiguities when a name is inherited from multiple inheritance paths

Operator Overloading - Polymorphism

- Define an operator (such as $+$ or *) for user-defined types.

Uniform Access

- All services offered by a module should be available through a uniform notation
- 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 by 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 and manipulate various pieces of information about an object at run-time.
- Most object oriented programming languages support some form of reflection.
- This includes ability to determine
  - Object type
  - Object inheritance structure
  - Object methods, including number and types of parameters, and return types
  - Object attributes names and types (optional)

Introspection vs. Reflection

- **Type Introspection** is the ability of a program to examine the type or properties of an object at runtime.
- **Reflection**, ability for a program to manipulate the values, meta-data, properties and/or functions of an object at runtime [23].

[23] https://stackoverflow.com/questions/25198271/what-is-the-difference-between-introspection-and-reflection

Object Oriented Programming Language

- Pure Object Oriented Programming Languages
- Hybrid Object Oriented Programming Languages
- Otherwise (None Object Oriented)

General Characteristics

Functional Programming Features Support

Higher Order Functions

- Functions can be treated as if they were data objects
    - 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 Closures

- Bundling up the lexical (static) scope surrounding the function with the function itself
- Function carries its surrounding environment around with it wherever it may be used

General Characteristics

Multithreading / Concurrency

Multithreading / Concurrency

- **Multithreading,** ability for a single process to process two or more tasks concurrently
- **Concurrency,** the decomposability property of a program, algorithm, or problem into order-independent or partially-ordered components or units [24]

[24] https://en.wikipedia.com/wiki/Concurrency

General Characteristics

Pointer Arithmetic

Pointer Arithmetic

Ability for a language to directly manipulate memory addresses and their contents

General Characteristics

Design by Contract

Design By Contract

- Ability to incorporate important aspects of a specification into the software that is implementing it.
- Important features are:
    - **Pre-conditions,** conditions that must be true before a method is invoked
    - **Post-conditions,** conditions guaranteed to be true after the invocation of a method
    - **Invariant,** conditions guaranteed to be true at any stable point during the lifetime of an object

General Characteristics

Regular Expression

Regular Expressions

Pattern matching constructs capable of recognizing the class of languages known as *regular languages* [25]

[25] https://en.wikipedia.com/wiki/RegularLanguages

General Characteristics

Language Integration

Language Integration

Seaming-less integration with other programming languages

General Characteristics

Built-In Security

Built-In Security

- Programming language's 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

Market Share / Adoption / Penetration

Interactive: The 2018 Top Programming Languages - IEEE Spectrum

https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018

TIOBE Index

`https://www.tiobe.com/tiobe-index/`

Final Thoughts

Final Thoughts

Programming Languages Philosophy

Philosophy

Programming Languages that are Optimized for

- Concurrency
- Readability
- Overcome what (they think shortages) in other languages (mainly C/C++)
- Other

Final Thoughts

What Experts Think ?

Are All PLs the Same?

https://www.coursera.org/learn/programming-languages/lecture/fbcb7/are-all-pls-the-same

Final Thoughts

What Shall I Do ?

Teach Yourself Programming in 10 Years

`http://norvig.com/21-days.html`

Experts' Opinion in Programming Languages Comparison

- `https://www.quora.com/What-are-the-best-programming-languages-to-learn-today`
- `https://www.quora.com/What-programming-languages-should-a-modern-day-programmer-have-ir`

Summary

Summary

https://www.haitham.ws