Lecture 1

Chapter 1: Preliminaries

Reasons for studying concepts of programming

ايه الاسباب اللي تخلينا نتعلم concepts of programming ؟

1. Increased ability to express ideas

ازاي الidea اللي عندك تعملها express أو تعبر عنها عن طريق استخدام programming language.

2. Improved background for choosing appropriate languages

ازاي يبقي عندنا خلفيه جيده من خلالها نقدر نختار الprogramming language.

3. Increased ability to learn new languages

ان يبقي عندك الثقة إنك تتعامل مع أي new language طالما الconcept programming language معاك.

4. Better understanding of significance of implementation Scientific application: large computation

يعني ايه هي الطريقة اللي هستخدمها في عمل الimplementation هل هتبقي functional programming ولا oop ولا هستخدم الاتنين.

5. Better use languages that are already known

وطبعا يفضل إني استخدم لغات معروفه.

6. Overall advancement of computing

نقدر نعمل computing بشكل أفضل.

Programming Domains

بنستخدم الprogramming Domains في حاجات زي scientific applications مثال علي كده الناس اللي بتشتغل في مجال الchemistry بتحتاج تعمل معادلات معينه بس مش بيقدروا يعملوها لأنها computation عالية فبيستخدموا الprogramming language في انهم يعملوا scientific applications تساعدهم.

- Business Applications: product reports, use decimal numbers and characters

لو مثلا بنشتغل في شركة وعايزين نعمل product reports ال reports بيبقي فيها طبعا characters و numbers.

- Artificial intelligence

في الAI بنستخدم symbols rather than numbers, وطبعا دلوقتي بنستخدم لغات برمجه عاديه في AI زي python وبقي فيها functions كتير بتساعدنا في شغل الAi و ML.

- Systems programming: need efficiency because of continuous use

يعني لو عندنا systems programs نفسها هي اللي بنستخدمها نقدر نعملها programming ، وبتحتاج طبعا efficiency لان الsystem هو اللي بيستخدمها زي الـoperating systems

- Web software: Eclectic collection of languages: Markup (e.g., HTML), scripting (e.g., PHP), general purpose (e.g., Java)

عشان نقدر من خلالهم نعمل web applications أو web pages.

Language Evaluation Criteria

ازاي نعمل compare بين language و language تانيه عندنا أربع انواع من ال criteria:

- Readability: the ease with which programs can be read and understood

مدي السهولة والقدرة علي فهم وقراية اللغة.

- Writability: the ease with which a language can be used to create programs

ازاي تقدر تكتب الprogram بتاعك بطريقه سهله لان في البداية كان كانت الlanguage صعب تستخدمها لأنها كانت low level لكن كل ما نطلع ب heigh level language بتبقي أسهل في القراءة والكتابة.

- Reliability: conformance to specifications (i.e., performs to its specifications)

نقدر نعمل specifications من غير أخطاء عن طريق اننا نقدر نعمل tracking و trace للكود بتاعنا و exception handling وغيرها.

- Cost: the ultimate total cost

الtotal cost اللي بيستهلكه الprogram عشان يعمل run سواء الrunning time أو الmemory usage أو سواء الcost اللي هتبقي علي هيئه فلوس هتدفعها إنك مثلا تشتري حاجه في programming language أو انك هتشتري application معينه او تدفع فلوس عشان تدرب الديفيلوبرز اللي عندك على حاجه جديده.

Readability

Overall simplicity

- A manageable set of features and constructs
- Minimal feature multiplicity

- Minimal operator overloading

نحاول نقلل الoverloading ونقلل من الfeatures multiplicity ما تبقاش كتيرة دي كلها بتخلي الـ programming language تبقي simpler.

Orthogonality

- means that operations change just one thing without affecting others.
- A relatively small set of primitive constructs can be combined in a relatively small number of ways
- Every possible combination is legal

مثلا لو عندك attribute في class معين عايز تغيرها فأنت بتغييرها ومش بتأثر بردو علي باقي الclasses , لو انت عامل الprogram بتاعك main بس فلما بتيجي تعدل على حاجه بتضطر تعدل على حاجات تانيه في الprogram لكن لو انت عامل functions ومحتاج تعدل في واحده منهم تعديل بسيط فانت مش محتاج تعدل على الباقي وبردو تقدر تعمل combination للfunctions بتاعتك ومن خلالها نقدر نستخدمها في applications كتير زي الimplementation اللي كنا بنعمله في data structure.

Data types

- Adequate predefined data types

بيبقي في data type متعرفه من قبل كده وبالتالي انت مش محتاج تعمل data type من الأول.

Syntax considerations

- Identifier forms: flexible composition
- Special words and methods of forming compound statements
- Form and meaning: self-descriptive constructs, meaningful keywords

الsyntax طبعا يكون flexible وتبقي سهله في الفهم عن طريق مثلا إنك تسمي الvariable functions, classes بأسماء كويسه ومفهومه.

Writability

Simplicity and orthogonality

- Few constructs, a small number of primitives, a small set of rules for combining them

ما نستخدمش constructs كتير، احنا عارفين أن في class تقدر تشيل عدد لا نهائي من الconstructs فانت مش هتعرف تستخدم انهي واحد لما تعمل initialize للclass فالأحسن انك تستخدم constructs قليله.

Support for abstraction

- The ability to define and use complex structures or operations in ways that allow details to be ignored

لما تيجي تعمل implementation لproblem إنك ما تعملش كل التفاصيل الخاصة بيها جنبها لا المفروض تعملها abstract.

Expressivity

- the breadth of ideas that can be represented and communicated in that language.
- The more expressive a language is, the greater the variety and quantity of ideas it can be used to represent.

إنك تقدر تعبر عن افكار كتير من خلال الlanguage دي كل ما اللغة تبقي more expressive كل ما تقدر تعمل بيها افكار اكتر.

Reliability

Type checking

- Testing for type errors

ان انت تقدر تعمل test للايرورز اللي عندك يعني ممكن تكون عامل variable من type معين ولما تيجي تدخل data فدخل حاجه ب type غلط فيعمل error في الprogram بتاعك فلازم تبقي حريص وتخلي بالك من حاجه زي كده.

Exception handling

- Intercept run-time errors and take corrective measures

إنك تعمل handle للايرورز معينه ممكن تحصل زي مثلا لو هتعمل access لdatabase أو هتكتب / تقرا من file لان ممكن الfile ده ميكنش موجود أصلا فلازم تعمل handle للحاجات دي.

Aliasing

- Presence of two or more distinct referencing methods for the same memory location

الموضوع ده بنشوفوا في لغات البرمجة اللي بتستخدم الpointers إنك تكون عامل مثلا variable وعايز تعمل access لنفس المكان ف ده طبعا بيسبب مشكله.

Readability and writability

- A language that does not support “natural” ways of expressing an algorithm will require the use of “unnatural” approaches, and hence reduced reliability

لازم اللغة بتاعتك تكون الreadability and writabilitiy كويسين عشان هما دول بيأثروكمان علي الprogram بتاعك يبقي more reliable.

Cost

- Training programmers to use the language
- Writing programs (closeness to particular applications)
- Compiling programs
- Executing programs
- Language implementation system: availability of free compilers
- Reliability: poor reliability leads to high costs
- Maintaining programs
  - يعني مثلا لو انت في شركه بتشتغل بالc# وعايز تعمل بروجيكت بالpython فالناس اللي عندك ما يعرفوش python فأنت لازم تدربهم وطبعا ده هيكلفك.
  - يعني هل ممكن الprogramming language اللي بتعامل بيها هي free ولا مش ببلاش ولازم ادفع عشان أقدر استخدمها.
  - ده يعتبر تكلفه في الوقت مش في الفلوس.
  - هل بردو اللغة اللي هستخدمها free ولا محتاجة license عشان أقدر استخدمها.
  - لو الreliability بتاعه الprogram بتاعتك قليله فده هيخلي تكلفه الtime تزيد عشان انت مثلا ممكن تكون مستخدم data structure بتاخد ميموري كبيرة فده هيعمل overload كبير ع الميموري والtime cost بتاعك هيزيد.
  - وبردو الmaintaining بيكلفني ممكن وقت وممكن فلوس فلازم أحسبه من ضمن الـcost.

Others

Portability

- The ease with which programs can be moved from one implementation to another

ان الprogram بتاعك يقدر يشتغل علي machines مختلفة زي مثلا في Java من أهم مميزاتها أنها portable أنها تقدر تشتغل علي operating systems مختلفة.

Generality

- The applicability to a wide range of applications

زي بردو مثال Java إنك تقدر تعمل بيها كذا حاجه زي desktop application أو web application عشان كده هي general purpose programming language.

Well-definedness

- The completeness and precision of the language's official definition

ان completeness و precision معروفين وان مايبقاش في confusion فيه.

Influences on Language Design

Program Design Methodologies

- New software development methodologies (e.g., object-oriented software development) led to new programming paradigms and by extension, new programming languages

ان ما بقاش في حاجه بتقف علي programming language معينه بقي كل شويه في software نازله جديده و extension للprogramming language وده بيخليك تغير الديزاين بتاع الprogram بتاعك لو انت بتشتغل بلغة تانية.

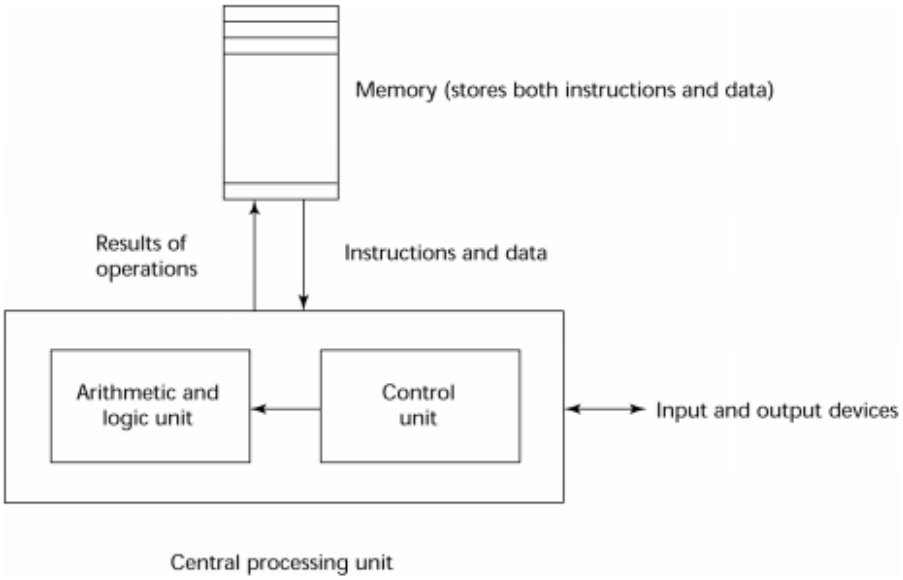
Computer Architecture Influence

- Well-known computer architecture: Von Neumann
- Imperative languages, most dominant, because of von Neumann computers
  - Data and programs stored in memory
  - Memory is separate from CPU
  - Instructions and data are piped from memory to CPU
  - Basis for imperative languages
    - Variables model memory cells
    - Assignment statements model piping
    - Iteration is efficient

بالنسبة للVon Neumann بتبقي stored in memory وطبعا الميموري دي موجودة في الCPU و الinstructions بتروح للميموري جوه الCPU وبعد كده بقي الprogramming language بتعامل مع الvariables دي وتروح رايحه

للmemory cells أو لو انت بتعمل assignment statement وبعد كده بتستخدم الiteration بتاعتك والصورة دي بتوضح instructions بتتحرك ازاي.

The von Neumann Architecture



- Fetch-execute-cycle (on a von Neumann architecture computer)

عشان الprogram يشتغل بعمل fetch execute cycle بتبع الخطوات دي:

- initialize the program counter
- repeat forever
- fetch the instruction pointed by the counter – increment the counter
- decode the instruction
- execute the instruction
- end repeat

Programming Methodologies Influences

- 1950s and early 1960s: Simple applications; worry about machine efficiency
- Late 1960s: People efficiency became important; readability, better control structures
  - structured programming
  - top-down design and step-wise refinement
- Late 1970s: Process-oriented to data-oriented

- data abstraction
- Middle 1980s: Object-oriented programming
  - Data abstraction + inheritance + polymorphism

بالنسبه للhistory بتاعه الprogramming language هنلاقي ان في أول الخمسينات والستينات كان عباره عن simple applications وكانت الmachine efficiency قليله جدا.

بعد كده في الستينات نهتم اكتر بالefficiency وإننا نحصل علي readability اعلي.

بعد كده في السبعينات دخلنا في oop عشان نقدر نعمل abstraction.

وفي التمانينات دخلنا اكتر في programming language واتعرفنا علي concepts جديده زي polymorphism و inheritance.

Language Categories

Imperative

- Central features are variables, assignment statements, and iteration
- Include languages that support object-oriented programming
- Include scripting languages
- Include the visual languages

Examples: C, Java, Perl, JavaScript, Visual BASIC .NET, C++

الimperative دي لغات زي الجافا والسي ودي بيبقي فيها variables و assignment statement و repetition و oop دي كلها تعتبر imperative language.

Functional

- Main means of making computations is by applying functions to given parameters

Examples: LISP, Scheme, ML, F#

هي بتبقي عبارة عن functions واحنا بنعمل apply عليها بـfunctions بردو as a parameter من اللغات زي Haskell و Ml و F#.

Logic

- Rule-based (rules are specified in no particular order)

Example: Prolog

لما بنحتاج نعمل Rule زي مثلا الAi فكنا بنستخدم لغة زي prolog لكن حاليا مابقتش بيتم استخدامها.

Markup/programming hybrid

- Markup languages extended to support some programming

Examples: JSTL, XSLT

دي اللي بنستخدم فيها XML و HTML.

Language Design Trade-Offs

لو انت حابب تقارن بين لغتين فأنت هتقارن بين reliability ولا writabilitiy ولا هتقارن الاتنين مع بعض؟

Reliability vs. cost of execution

Example: Java demands all references to array elements be checked for proper indexing, which leads to increased execution costs

في البداية لو هنقارن بين reliability و cost execution فأنا عندي حاجه زي الجافا بيبقي فيها references كتيرة.

Readability vs. Writability

Example: APL provides many powerful operators (and a large number of new symbols), allowing complex computations to be written in a compact program but at the cost of poor readability

Writability (flexibility) vs. Reliability

Example: C++ pointers are powerful and very flexible but are unreliable

بالنسبة للpointers في لغة cpp فهي powerful وكل حاجه بس من الاحسن أننا ما نستخدمش pointers كتير في الprogram بتاعنا عشان بيبقي unreliable.

Implementation Methods

Compilation

- Programs are translated into machine language; includes JIT systems

Use: Large commercial applications

في الcompiler بحول لماشين كود وبعدين الماشين كود دي بتترجم.

Pure Interpretation

- Programs are interpreted by another program known as an interpreter

Use: Small programs or when efficiency is not an issue

الinterpreter بيعمل run للكود line by line فمش بنحتاج أننا نعملها translation.

الcompiler بيبقي أحسن في الlarge applications أما لو small program فالinterpreter بيبقي أحسن.

Hybrid Implementation Systems

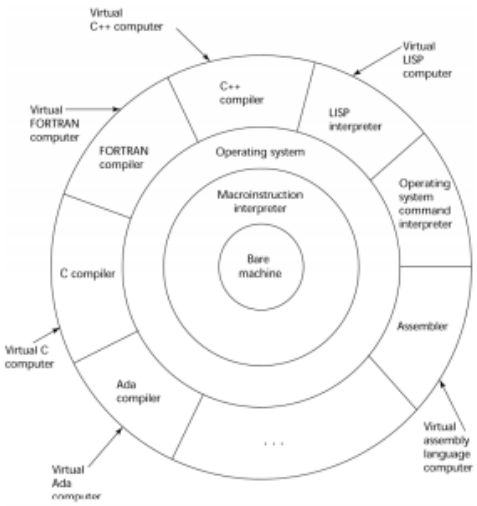
- A compromise between compilers and pure interpreters

Use: Small and medium systems when efficiency is not the first concern

الhybrid بيستخدم الاتنين الcompiler والinterpreter بيفضل في الsmall and medium applications.

Layered View of Computer

- The operating system and language implementation are layered over machine interface of a computer



Compilation

- Translate high-level program (source language) into machine code (machine language)
- Slow translation, fast execution
- Compilation process has several phases:
  - lexical analysis: converts characters in the source program into lexical units
  - syntax analysis: transforms lexical units into parse trees which represent the syntactic structure of program
  - Semantics analysis: generate intermediate code
  - code generation: machine code is generated

بيحول الsource code ل machine code وحتى لو التحويل ده بيبقي بطيء بس ال execution بيبقي سريع.

عموما الcompiler ليه كذا مرحلة اول حاجه lexical analysis إن الكود بتاعك بتقسمه لwords أو characters.

تاني حاجه syntax analysis إني باخد ال transformation lexical وادخلها علي trees

وبعدين بتخش علي semantics analysis.

واخر حاجه بيحصل code generation للmachine code.

The Compilation Process

Additional Compilation Terminologies

- Load module (executable image): the user and system code together
- Linking and loading: the process of collecting system program units and linking them to a user program

لو أنا عندي module زي الfunction مثلا او modules بتبقي ف classes تانية واحنا بنستخدمها دي من الterminologies الخاصة بالcompiler.

إنك تعمل linking لكل الclasses اللي عندك في الuser program بتاعك.

Von Neumann Bottleneck

- Connection speed between a computer's memory and its processor determines the speed of a computer
- Program instructions often can be executed much faster than the speed of the connection; the connection speed thus results in a bottleneck
- Known as the von Neumann bottleneck; it is the primary limiting factor in the speed of computers

الbottleneck معناها إن الprogram بيتعمله execution بسرعة جدا.

الprogram وهو بيتعمله execution بالسرعة دي ده بيؤدي لحدوث bottleneck يعني في جزء بيexecute أسرع من جزء تاني.

سرعة الconnection بين الmemory و الprocessor بتتحدد علي حسب سرعة الكمبيوتر عشان الـ program instructions بتexecute أسرع من الconnection والprogram بتاعي.

Pure Interpretation Process

- No translation
- Easier implementation of programs (run-time errors can easily and immediately be displayed)
- Slower execution (10 to 100 times slower than compiled programs)
- Often requires more space
- Now rare for traditional high-level languages
- Significant comeback with some Web scripting languages (e.g., JavaScript, PHP)

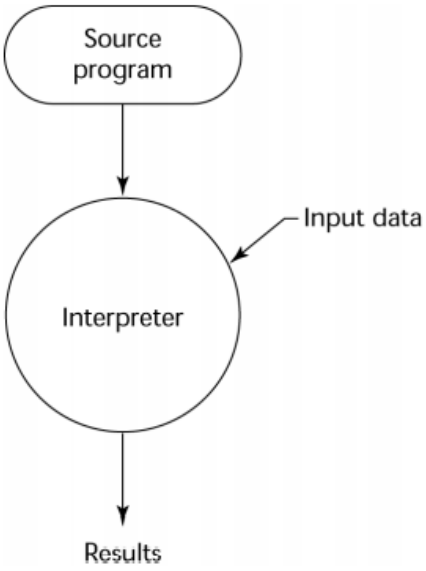
مفيهاش translation زي الـ compiler.

بتبقي أسهل في أنها توضح الerror لان ال run بيحصل line by line.

لكن الexecution بتاعها بيبقي ابطئ.

بتحتاج مساحة أكبر.

اغلبيه اللغات الhigh level بتستخدم الـinterpreter



Hybrid Implementation Systems

- A compromise between compilers and pure interpreters
- A high-level language program is translated to an intermediate language that allows easy interpretation
- Faster than pure interpretation

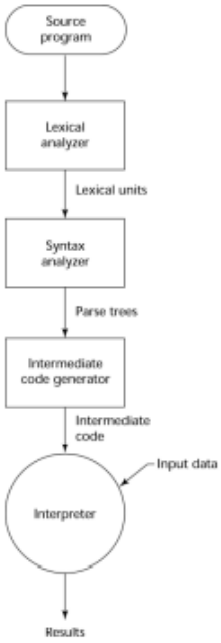
الhybrid بيستخدم الاتنين الcompiler والـinterpreter

أسرع وأحسن من الـ pure interpreter.

Examples:

- Perl programs are partially compiled to detect errors before interpretation
- Initial implementations of Java were hybrid; the intermediate form, **byte code**, provides portability to any machine that has a byte code interpreter and a run-time system (together, these are called **Java Virtual Machine**)

من الأمثلة المشهورة عليه الجافا عن طريق أن بيتم تحويل الكود لbyte code وده بيخليها تقدر تشتغل علي اي machine وده اللي بيتسمي Java virtual machine.



Preprocessors

- Preprocessor macros (instructions) are commonly used to specify that code from another file is to be included
- A preprocessor processes a program immediately before the program is compiled to expand embedded preprocessor macros
- A well-known example: C preprocessor
  - expands, #include, #define, and similar macros

الفكرة أننا بنعمل extending للprogram بتاعنا عشان نستخدم حاجات تانيه الpreprocessing من اول الحاجات اللي بتتعمل زي مثلا لما بعمل including لlibrary أو package معينه.

Programming Environments

- **A collection of tools used in software development**
- **UNIX**
  - An older operating system and tool collection
  - Nowadays often used through a GUI (e.g., CDE, KDE, or GNOM E) that runs on top of UNIX
- **Microsoft Visual Studio.NET**

- A large, complex visual environment
- **Used to build Web applications and non-Web applications in any .NET language**
- **NetBeans**
  - Related to Visual Studio .NET, except for applications in Java

الـenvironment اللي بنستخدم فيها الـprogramming language هي عباره عن مجموعه tools و operating system من خلالها بقدر ابني الـprogram بتاعي.

Summary

The study of programming languages is valuable for several reasons:

- Increase our capacity to use different constructs
- Enable us to choose languages more intelligently
- Makes learning new languages easier

Most important criteria for evaluating programming languages include:

- Readability, writability, reliability, cost

Major influences on language design have been machine architecture and software development methodologies

The major methods of implementing programming languages are compilation, pure interpretation, and hybrid implementation