CS213: Chapter 1 (Preliminaries) By Ramas

1- Reasons for Studying Concepts of Programming Languages [ما الهدف من دراسة مبادئ لغات البرمجة؟]

- Increase the ability to express ideas.

[القدرة على التعبير عن الأفكار من خلال إستخدام اللغة المناسبة، مثل إستخدام لغة Prolog لأنها تخدم مجال الذكاء الإصطناعي]

- Improve background for choosing the appropriate language.

[تصبح لديك خبرة فـ بالتالي سوف تستطيع إختيار اللغة المناسبة]

- Increased ability to learn new languages.

[كل ما تعلمت لغة برمجية كل ما كان أسهل عليك تعلم لغة أخرى]

- Better use of languages that are already known.

[كل ما تعلمت لغة برمجية جديدة كل ما توسع إدراكك للغة البرمجية التي سبق تعلمتها؛ لأن اللغات البرمجية لديها نفس الأساس نوعًا ما]

- Overall advancement of computing.

[التقدم الشامل للحوسبة]

2- Programming Domains [مجالات البرمجة]

Domain	Description	Example	Notes
Scientific Applications [مجال علمي]	Heavy floating-point math, arrays	Fortran	[للتوضيح: إجراء عمليات معقدة بإستخدام الأعداد العشرية]
Business Applications [مجال أعمال]	Reports, records, decimals/characters	COBOL	[للتوضيح: التعامل مع الأعداد العشرية بدقة عالية للأموال، التعامل مع بيانات ضخمة وسجلات، مثل البنوك والشركات]
Artificial Intelligence [الذكاء الإصطناعي]	Symbols & linked lists, pattern matching	LISP, Prolog	[للتوضيح: التعامل مع الرموز والكلمات بدلًا من الأرقام]
Systems Programming [أنظمة التشغيل]	Efficiency, hardware access	C	[للتوضيح: تتطلب دقة وكفاءة عالية]
Web Software [برامج الويب]	Mix of markup, scripting, and backend languages	HTML, JavaScript, PHP, Python, Java, C#	[للتوضيح: الجمع بين عدة لغات وتقنيات في نفس المشروع، مثل مواقع التواصل الإجتماعي]

3- Language Evaluation Criteria [أسس تقييم اللغة]

3.1 Readability قابلية القراءة

Programs can be read and understood easily.

- **Overall Simplicity**: Limited features, minimal repetition, few overloaded operators

[اللغة بسيطة وتحتوي على ميزات محدودة لا تعقد على المبرمج]

- **Orthogonality**: A small set of primitive constructs can be combined in a small number of ways

[يمكن دمج ميزات مع بعض دون أن يحصل تعارض، مثلًا الفنكشنز والمصفوفات]

- **Data Types**: Predefined types, flexible naming

[أنواع بيانات تم تعريفها مسبقًا، أسماء متغيرات مرنة وواضحة]

- **Syntax Considerations**: Clear statements, meaningful keywords

[مثلًا جملة الـif else، جملة واضحة حتى غير المبرمج يستطيع تخمين المعنى]

3.2 Writability قابلية الكتابة

Programs can be written easily.

- **Simplicity & Orthogonality:** Few constructs, simple rules, easy to combine features.
- **Support for Abstraction**: Can define/use complex structures without worrying about details

[توفر فنكشنز من المكتبات بدلًا من كتابة برامج لتعريف هذه الفنكشنز]

- Expressivity: Convenient ways to specify operations; powerful operators

[مثلًا المعامل + له إستخدامات كثيرة: 1+2 أو "bc" + "ab" جمع عددين وجمع نصين]

3.3 Reliability الموثوقية أو قابلية الإعتماد عليها

- **Type checking**: Testing for type errors

[تأكد أن نوع المتغير صحيح]

- **Exception handling**: Intercept run-time errors and take corrective measures

[تدعم خاصية catch and try]

- **Aliasing**: Two or more references to the same memory location

[تدعم خاصية متغيرين يشيرون لنفس الموقع بالذاكرة]

3.4 Cost التكلفة

- **Training**: Cost of teaching programmers to use the language

[تدريب المبرمجين على إستخدام اللغة لتقليل التكلفة]

- **Writing Programs**: Effort needed to write programs, including suitability for applications

[إستخدام اللغة المناسبة]

- **Compiling**: Cost of compiling programs
- **Executing**: Cost of running programs
- **Language Implementation**: Availability of compilers [توفر كومبايلرز]
- **Reliability**: Poor reliability leads to high cost [قابلية الإعتماد على البرنامج]
- **Maintenance**: Cost of fixing programs

3.5 Other Criteria

- **Portability**: Programs can be moved from one system to another

[البرنامج الذي نفذته في جهاز بنظام تشغيل أستطيع نقله وتنفيذه في جهاز آخر]

- **Generality**: Language can run on any system

[البرنامج ينفذ في أي نظام تشغيل]

- **Well-definedness**: Language has a reference for learning

[البرنامج لديه مرجع للتعلم]

4- Influences on Language Design

Computer Architecture:

Programming languages are designed to match the widespread computer architecture. Most early and many modern languages are based on the **von Neumann architecture.**

Program Design Methodologies:

The way software is developed influences language design. Examples:

- Object-oriented development → Java, C++
- Traditional methods → C, Pascal

[الخلاصة: تصميم اللغة يتأثر بالهاردوير وطريقة تطوير البرامج]

5- Language Categories فئات اللغات البرمجية

Category	Description	Examples
Imperative / Procedural [إجرائية]	Variables, assignments, loops, OOP, scripting	C, Java, Perl, JavaScript, C++
Functional [وظيفية]	Computation by functions applied to parameters	LISP, Scheme, ML, F#
Logic [منطقية]	Rule-based programming like AI	Prolog
Markup/Hybrid	Markup languages with programming features	JSTL, XSLT

6- Language Design Trade-Offs مقايضات

✓ **Reliability vs. Cost of Execution**
- More reliability checks → higher execution cost.
- **Example:** Java checks array bounds, increasing runtime cost.

✓ **Readability vs. Writability**
- More powerful/compact features → harder to read.
- **Example:** APL allows complex computations in short code but reduces readability.

✓ **Writability (Flexibility) vs. Reliability**
- More flexible features → may be less safe.
- **Example:** C++ pointers are powerful but can lead to errors if misused.

[الخلاصة: المقايضات في تصميم اللغة تعني أن تحسين ميزة معينة غالبًا يؤثر على أخرى، مثال: زيادة الموثوقية (التحقق من الأخطاء) = زيادة مدة التنفيذ، والعكس سرعة مدة التنفيذ = التقليل من الموثوقية (عدم التحقق من الأخطاء)]

7- Implementation Methods & Compilation Terms طرق التنفيذ

Method / Term	How it Works	Typical Use / Key Feature
Compilation	Translates the whole program into machine code (includes JIT)	Large commercial applications; fast execution
Pure Interpretation	Executes the program one statement at a time	Small programs; flexible; easy debugging
Hybrid Implementation	Mix of compilation and interpretation	Small/medium systems; balance speed & flexibility
Load Module (Executable Image)	Final program combining user and system code	Ready to execute by the computer
Linking and Loading	Combines user code + system libraries and loads into memory	Needed to produce the load module

- **Preprocessors**
  - Definition: A program that processes code before compilation
  - Purpose: Expands macros and includes code from other files
  - Example: C Preprocessor: #include, #define, etc.
  - Key Idea: Prepares the source code so the compiler can translate it into machine code

[في المستوى السابق –برمجة ٣– كنا نفصل الملفات ثم نستدعي الملف المطلوب في الماين فنكشن بإستخدام#إنلكود]

8- Programming Environments بيئات البرمجة

Environment	Description	Typical Use / Key Feature
UNIX	Older OS and tool collection	Often used via GUI (CDE, KDE, GNOME)
Microsoft Visual Studio.NET	Large, complex visual environment	Build Web and non-Web applications in any .NET language
NetBeans	Similar to Visual Studio.NET	Build Java applications

[البيئة الي نبرمج فيها، مثل الـ C++ كنا نبرمج بالڤيجوال، وزي الجافا كنا نبرمج بالـ نيتبينز]