Language history, design

History
=======

Antikythera mechanism https://en.wikipedia.org/wiki/Antikythera_mechanism
~200BC, clockwork machine to show astronomical positions, eclypses even olympics

Abacus, napier bones, mechanical calculators (e.g. 1642: Blaise Pascal)

Jacquard loom https://en.wikipedia.org/wiki/Jacquard_machine
~1800 Joseph Marie Jacquard weaving with changed cards
- first programmable device

Analytical engine
1833- Charles Babbage
- arithmetic logic unit, control flow: conditional branching and loops
- integrated memory,
- first design for a general-purpose computer
- Turing-complete
- Ada Lovelace

Analog computers
1900-1940 military purposes

Electromechamical computers
1940 Conrad Zuse electromechanical relay computers Z3: first programmable EM
1940- Polish and Brit scientist at Bletchley Park (incl. Alan Turing)
"Bomb" to greak Enigma code

ENIAC (Electronic programmable)
EDVAC (stored program)

Programming paradigms
======================

In which way we split the program into manageable parts
Functional LISP, 1958
Logical Prolog, 1972
Relational SQL

50s: Automatic programming
70s: Structural/procedural programming
80s: Object oriented programming (Simula 67, Smalltalk-80)
2000: Generative (incl. Aspect-oriented, Generic, Intentional, DSLs)

Language structure
==========

- Syntax
  - The correct grammar
    - ; separator or closing part
    - dangling if
    - whitespace's role (like make: space or tab)
    - Algol68: begin = (
- Semantics
  - The meaning of the syntactically correct code
    - axiomatic: P{x<-y}Q vagy push(pop(x)) == x
    - denotational == functional way
    - operational == iterative way
    - textual
- Pragmatics
  - Programming practices
    - Scala: optimized for inmutable
    - C++: RAII, PIMPL

- Language specification
  - BNF: Fortran
  - EBNF: Pascal
  - Wan Vijgarden: Algol68
  - Abstract machine: C/C++
  - Standardization:
    - Draft
    - TR (technical report)
    - Standard
- Design concepts
  - Expressivity (Scala anonym lambda)
  - Orthogonality (Algol68 "skip", C++: public virtual default ...)
  - Generality (Template-s)
  - Modularity (namespace vs. Java package, Erlang flat module, C/C++ include)
  - Portability (source/bytecode/binary)
  - Learnability (Pascal,Python,C++)
- Language evolution:
  - New keywords/features
  - Backward compatibility
    - C/C++
      - C++11 vector
      - Python2 -> Python3
      - Java -> Scala,Closure
  - Depricated

- Execution
  - Interpreted
  - Compiled
  - Environment (COBOL, Java, C#)