Supplementary Material 2 (SM2) for "Punctuated equilibrium in the large scale evolution of programming languages"

Sergi Valverde and Ricard V. Solé

ICREA-Complex Systems Lab
Universitat Pompeu Fabra
Barcelona

1. Influence Database

The dataset studied here was extracted from the English version of the Wikipedia web site. We wrote a simple internet/web robot that downloaded all the Wikipedia pages describing programming languages in May 2012. All the web sites were parsed and translated to simple format amenable for automatic reconstruction and network analysis (see below). We would like to thank the Santa Fe Institute for providing the facilities for obtaining this dataset.

2. Dataset

We use a simple relational text format to store the influence data of programming languages. The format was made as simple as possible for maximum transparency and flexibility. The file contains information about two different relations (1) the influence graph and (2) release year of the PL. Each line in the file represents information about these relations. For example, the following text line:

```
Cite "flpl" "ipl"
```

represents an influence link between "IPL" and "FLPL". Links are directed, i.e., the first language is influenced by the second language. Also, lines that start with the keyword 'Year' indicate the release data of the corresponding PL. For example:

```
Year "ipl"     "1954"
```

means that the programming language "IPL" was released in 1954. Lines starting with the symbol # represent comments and they are discarded for the network analysis. We found useful to add comments for describing any situation when additional sources had to be checked. We also provide the URL of any web site providing additional information when the corresponding Wikipedia source was unclear.

FULL LISTING

```
Year "ipl"     "1954"
# @Author: Allen Newell, Cliff Shaw, Herbert A. Simon
# Information Processing Language (IPL). IPL-V was the first- and also a major-
# language for doing list  processing. Ideas of list processing in the IPL family
# included the linked representation of list structures and the use of a stack
# (push-down list) to implement recursion.
# LISP prehistory (1956-1958): http://www-formal.stanford.edu/jmc/history/lisp/node2.html
Year "flpl" "1956"
# @Author: Herbert Gerlernter, Nathaniel Rochester and John McCarthy
# In the summer of 1956, John McCarthy heard a description of the ipl2 programming
# language at the 1st major workshop on artificial intelligence, held at Dartmouth.
# McCarthy realized that an algebraic list-processing language, on the style of the
# recently announced Fortran I system, would be very useful.  The summer of 1956,
# Gerlernter and Gerberich of IBM were working, with the advice of McCarthy, on a
# geometry problem. As a tool they developed FLPL, the Fortran List-Processing Language,
# by writing a set of list-processing subprograms for use with Fortran programs.
# In connection with IBM's plane geometry project, Nathaniel Rochester and
# Herbert Gelernter (on the advice of McCarthy) decided to implement a
# list processing language within FORTRAN, because this seemed to the the
#  easiest way to get started, and, in those days, writing a compiler for a
# new language was believed to take many man-years. This work was undertaken
```

```
# by Herbert Gelernter and Carl Gerberich at IBM and led to FLPL, standing
# for FORTRAN List Processing Language. Gelernter and Gerberich noticed that
# cons should be a function, not just a subroutine, and that its value should
#  be the location of the word that had been taken from the free storage list.
# This permitted new expressions to be constructed out of subsubexpressions by
# composing occurrences of cons.
Cite "flpl" "ipl"
Cite "flpl" "fortran"
# One result of [the FLPL] work was the development of the basic list-processing
# primitives that were later incorporated into Lisp.
Cite "lisp" "flpl"
Year "comit"  "1957"
# @Class: string manipulation, pattern-matching language
# COMIT was the first realistic string handling and pattern matching language; most of its
# features appear (although with different syntax) in any other language attempting to
# do any string manipulation
Year "fact"       "1959"
# FACT was an early computer programming language, created by the Datamatic Division
# of Minneapolis Honeywell for its model 800 series business computers in 1959.
# FACT was an acronym for "Fully Automated Compiling Technique". Some of the design of
# FACT was based on the linguistic project Basic English, developed about 1925 by
# C.K. Ogden.
# http://en.wikipedia.org/wiki/FACT_computer_language
Year "trac"      "1959"
# @Author: Calvin Mooers
# @Class: string/text processing
# TRAC is a purely text-based language—a kind of macro language.
# In most respects, TRAC is a case of pure functional programming.
# The acronym TRAC stands for "text reckoning and compiling"
# http://en.wikipedia.org/wiki/TRAC_(programming_language)
# http://www.thocp.net/software/languages/trac.html
# http://web.archive.org/web/20050324000031/http://tracfoundation.org/
# http://web.archive.org/web/20050206175120/http://www.tracfoundation.org/trac64/handling.htm
Cite "trac" "lisp"
# TRAC has in common with LISP a syntax that generally involves the presence of many levels of
nested parentheses.
Cite "trac" "comit"
# TRAC had a number of sources of inspiration; COMIT and LISP provided the main challenge to devise
a system
# which could do at least as much as they could, yet would not have their limitations
# http://web.archive.org/web/20050206175120/http://www.tracfoundation.org/trac64/handling.htm
# @Author: Victor H. Yngve
# @Class: String/list processing language
# Implementation: 1961
# COMIT was the first string processing language (compare SNOBOL, TRAC,
# and Perl), developed on the IBM 700/7000 series computers by Dr.
#  Victor Yngve and collaborators at MIT from 1957-1965. Yngve
# created the language for supporting computerized research in the
# field of linguistics, and more specifically, the area of machine
#  translation for natural language processing. The creation of COMIT
#  led to the creation of SNOBOL.
# http://en.wikipedia.org/wiki/COMIT
# Yngve, V.H., "COMIT", Comm. ACM 6, 83 (1963).
Year "snobol" "1962"
# http://en.wikipedia.org/wiki/SNOBOL
# @Class: string processing, symbol processing
# @Author: https://en.wikipedia.org/wiki/Ralph_Griswold
Cite "snobol" "comit"
Cite "snobol" "fortran"
# snobol fortran ii
# http://en.wikipedia.org/wiki/Timeline_of_programming_languages
Year "gap"  "1964"
# Farber, D. J., 635 Assembly System - GAP. Bell Telephone Laboratories Computation Center (1964)
# @Class:  macro processor
Year "gpm"     "1965"
# @Author: Christopher Strachey
# @Class: general-purpose macro processor
# One of the earliest macro processors was GPM (the General Purpose Macrogenerator). It is
# a symbol-stream processor. GPM takes as its input a stream of characters and produces as its
# output another stream of character which is produced from the input by direct copying except
# in the case of macro-calls in the input stream. It operates by a form of substitution which is
# completely general in its application in that it is allowed everywhere. GPM  was developed
# at the University of Cambridge, UK, in the mid 1960s, under the direction of
# Christopher Strachey.  GPM was originally devised for  the very practical purpose of
#  helping to write a compiler for CPL.
Year "refal"  "1966"
# @Author: V. F Turchin
# @Class: metaalgorithmic language
# V.F.Turchin. The Language Refal--The Theory of Compilation and  Metasystem Analysis.  Curant
Institute of Mathematics. Technical report 018, NY, 1980.
```

```
# @Author: Valentin Turchin
# Refal (Recursive functions algorithmic language) "is a functional programming
# language oriented toward symbol manipulation", including "string processing,
# translation, [and] artificial intelligence". It is one of the oldest members
# of this family, first conceived in 1966 as a theoretical tool with the first
# implementation appearing in 1968. Unlike Lisp, Refal is based on pattern matching.
# Compared to Prolog, Refal is conceptually simpler. Its pattern matching works
# in the forward direction rather than backwards (starting from the goal) as in Prolog.
# http://en.wikipedia.org/wiki/REFAL
Cite "refal" "lisp"
Cite "refal" "comit"
# "In the creation of Refal the ideas embodied in LISP and COMIT were used..
# Refal shares some ideas with SNOBOL and a striking resemblance to CONVERT
# can be seen, though in 1965-66, when Refal was designed, the author was
# not acquainted with either of these languages"
# From: V.F.Turchin. The Language Refal--The Theory of Compilation and  Metasystem Analysis. Curant
Institute of Mathematics. Technical report 018, NY, 1980.
Year "ml/i" "1966"
# @Class: general-purpose macro processor
# @Author: Peter Brown
#  Peter looked at various kinds of macro processors, and evaluated their various features,
# strengths and weaknesses before ML/I was developed.
# http://www.ml1.org.uk/dissertation.html
# http://www.ml1.org.uk/history.html
# http://en.wikipedia.org/wiki/ML/I
Cite "ml/i"  "gpm"
# The basic concept of a macro as in ML/I is not new and in this respect ML/i belongs to
# the family of macro processor consisting of GPM, TRAC and 803 Macro-generator. Of these,
# ML/I has leaned most heavily on GPM.
# Macro-time statements are not new and represent a cross between the ideas of GPM and
# the PL/I macro processor
# http://www.ml1.org.uk/pdf/ml1diss_part2.pdf
Year "crt_rps" "1967"
# @Class: report generator
# @Author: Louis Schlueter
# CRT Report Generating System, a database report generator, was developed by Louis Schlueter in
1967.
# Initially developed for the 418 line of computers as the CRT Report Generating System, the
# name had to be changed to 418 CRT Report Processing System (418 CRT RPS) to avoid confusion
#  with an existent RPG software package.
# http://special.lib.umn.edu/findaid/xml/cbi00121.xml
Year "ttm"  "1968"
# http://en.wikipedia.org/wiki/TTM_(programming_language)
# @Class:  macro processor
Cite "ttm"     "gap"
# http://en.wikipedia.org/wiki/General-purpose_macro_processor
# Farber, D. J., 635 Assembly System - GAP. Bell Telephone Laboratories Computation Center (1964
Cite "ttm"     "gpm"
# http://en.wikipedia.org/wiki/General-purpose_macro_processor
# http://comjnl.oxfordjournals.org/content/8/3/225
Cite "ttm" "trac"
# http://en.wikipedia.org/wiki/TTM_(programming_language)
# The exact relationship between TTM and TRAC is unknown. The TTM documentation indicates
# that it was derived from GAP[3] and GPM.[4] In any case, the description of the
# characteristics of TRAC also apply to TTM. However, by removing the syntactic
# distinction between built-in and user-defined function, TTM would appear to be
# a much cleaner language.
# http://en.wikipedia.org/wiki/TTM_(programming_language)
Cite "dibol" "cobol"
# DIBOL shares the COBOL program structure of data and procedure divisions.
# http://en.wikipedia.org/wiki/DIBOL
Year "ppl"     "1969"
# http://en.wikipedia.org/wiki/Polymorphic_Programming_Language
# @Author: Thomas A. Standish at Harvard University
Year "edinburgh_imp" "1969"
# http://en.wikipedia.org/wiki/Edinburgh_IMP
# @Author: Edinburgh University
Cite "edinburgh_imp" "algol_60"
Cite "edinburgh_imp" "autocode"
Cite "edinburgh_imp" "atlas_autocode"
# http://ecosse.org/jack/history/autocodes/
# The Ferranti Manchester Mark 1, operational from about 1949, was the first British general purpose
computer, and had at first to be programmed in this way. However, RA Brooker, around 1953, developed
the first (fairly) high level language, Manchester Autocode, which he described [2,3] in 1955/6.
Year "autocode" "1955"
# Atlas Autocode (AA)[1][2] was a programming language developed around 1965 at Manchester
University for the Atlas Computer. It was developed by Tony Brooker and Derrick Morris as a variant
of the ALGOL programming language, removing some Algol features such as "passing parameters by name"
(which in Algol 60 means passing the address of a short subroutine to recalculate the parameter each
time it was mentioned).
```

```
# http://en.wikipedia.org/wiki/Atlas_Autocode
Year "atlas_autocode" "1965"
Year "sl5" "1975"
# @Author: https://en.wikipedia.org/wiki/Ralph_Griswold
# @Class: string processing
# SL5 is a programming language developed for experimental work in generalized pattern-matching
# and high-level data structuring and access mechanisms.
Cite "sl5" "snobol"
# @Info: "SL5 derives many of its characteristics from SNOBOL4" but there are several important
differences.
# http://drhanson.s3.amazonaws.com/storage/documents/sl5datastructures.pdf
# Cite "sl5" "algol_68"
# @Info: the syntax of SL5 is expression-oriented, and the control structures are similar to
# those in Algol 68. http://drhanson.s3.amazonaws.com/storage/documents/popl3.pdf
Year "mapper" "1975"
# @Class: report generator
# @Author:  Louis Schlueter
# http://en.wikipedia.org/wiki/MAPPER
# "There are similarities between the development history of MAPPER and that of UNIX."
# "MAPPER invented client/server programming before the industry defined it"
# "The earliest external use of MAPPER was in the Santa Fe railroad for tracking its piggyback
trailers in Chicago."
# http://web.archive.org/web/20080823061841/http://www.enterprisenetworksandservers.com/monthly/
art.php?402
# Louis Schlueter, "The History of MAPPER: The Birth of User Designed Computing," presented at UUA/
SUAE, Spring 1988.
# Michael Nicoll-Griffith, "MAPPER Was the First User-command Language" USE Proceed. 1983
# Louis Schlueter, User-Designed Computing: The Next Generation, 1988.
Cite "mapper" "crt_rps"
# The 418 CRT RPS software became MAPPER in 1975 when the 418 line was terminated
#  and the software was copied to the 1100 line.
# ------------------------ core ---------------------
Year    ".net_framework"       "2002"
Year    "a+"    "1988"
Year    "abap"  "1983"
Year    "abc"   "1987"
Year    "actionscript" "1998"
Year    "ada"   "1980"
Year    "ada_95"   "1995"
Year    "ada_2005"     "2005"
Year    "agentsheets"  "1991"
Year    "agilent_vee"  "1991"
Year    "aimms" "1993"
Year    "alef"  "1992"
Year    "algebraic_logic_functional"  "1995"
# Year  "algol" "1958"
Year    "algol_58"     "1958"
Year    "algol_60"     "1960"
Year    "algol_68"     "1968"
Year    "algol_w"      "1966"
Year    "alice" "2000"
Year    "alma-0"       "1997"
Year    "ampl"  "1985"
Year    "antlr" "1992"
Year    "apache_ant"   "2000"
Year    "apl"   "1964"
# @Class: array-oriented language
# http://en.wikipedia.org/wiki/APL_(programming_language)
# APL (named after the book A Programming Language) is an interactive array-oriented language
# and integrated development environment. APL is a concise, highly mathematical language
# designed to deal with array-structured data.
Year    "apple_developer_tools#quartz_composer"      "2005"
Year    "applescript"  "1993"
Year    "argus" "1982"
# @Author: Barbara Liskov
# @Class: Distributed programming
# Argus is an extension of the CLU programming language designed to support the creation
# of distributed programs by encapsulating related procedures within objects called 'guardians'
# and by supporting atomic operation called 'actions'
Year    "as/400_control_language"     "1988"
Year    "aspectj"      "2001"
Year    "ateji_px"     "2010"
Year    "autohotkey"   "2003"
Year    "autoit"       "1999"
Year    "autolisp"     "1986"
Year    "awk"   "1977"
Year    "b"     "1969"
Year    "bash_(unix_shell)"    "1989"
Year    "basic" "1964"
Year    "bcpl"  "1966"
```

```
Year    "beanshell"    "2000"
Year    "beta"  "1993"
Year    "bistro"       "1999"
Year    "bliss" "1970"
Year    "boo"   "2003"
Year    "bourne_shell" "1977"
Year    "brainfuck"    "1993"
Year    "c"     "1972"
Year    "c++"   "1983"
Year    "c--"   "1997"
Year    "c_omega"      "2003"
Year    "c_sharp"      "2001"
Year    "c_shell"      "1978"
Year    "cakewalk_(sequencer)#features"      "1987"
Year    "cal_(quark_framework)#cal"   "2004"
Year    "caml"  "1985"
Year    "candle"       "2005"
Year    "cel_(programming_language)"  "1998"
Year    "charity"      "1992"
Year    "charm_(language)"    "1996"
Year    "chuck" "2003"
Year    "cilk"  "1994"
Year    "clacl" "2000"
Year    "clarion"      "1986"
Year    "clean" "1987"
Year    "clipper"      "1985"
Year    "clips" "1985"
Year    "clojure"      "2007"
Year    "clu"   "1974"
Year    "cms_exec"     "1966"
Year    "co-array_fortran"    "1998"
Year    "cobol" "1959"
Year    "cobra" "2006"
Year    "coco/r"       "1983"
Year    "coffeescript" "2009"
Year    "coldfusion"   "1995"
Year    "colorforth"   "1992"
Year    "comal" "1973"
Year    "common_lisp"  "1984"
Year    "communicating_sequential_processes"  "1978"
Year    "component_pascal"    "1991"
Year    "comtran"      "1957"
Year    "concurrent_pascal"   "1975"
Year    "corvision"    "1986"
Year    "cowsel"       "1964"
Year    "cpl"   "1963"
Year    "csound"       "1985"
Year    "curl"  "1998"
Year    "curry" "1997"
Year    "cyclone"      "2006"
Year    "d"     "2001"
Year    "dart"  "2011"
Year    "datalog"      "1977"
Year    "dbase" "1979"
Year    "design_by_numbers"   "1999"
Year    "dibol" "1970"
Year    "digital_command_language"    "1972"
Year    "dylan" "1990"
Year    "e"     "1997"
Year    "e_(verification_language)"   "1992"
Year    "ecmascript"   "1997"
Year    "eel_(extensible_embeddable_language)"       "2005"
Year    "eicaslab"     "2001"
Year    "eiffel"       "1986"
Year    "embarcadero_delphi"  "1995"
Year    "epigram"      "2004"
Year    "erlang"       "1986"
Year    "escher"       "1995"
Year    "etoys" "1996"
Year    "euclid"       "1975"
Year    "eulisp"       "1990"
Year    "exec_2"       "1972"
Year    "f-script"     "2006"
Year    "f_sharp"      "2005"
Year    "factor"       "2003"
Year    "falcon"       "2003"
Year    "false" "1993"
Year    "fancy" "2010"
Year    "fantom"       "2005"
Year    "fargo" "1959"
```

```
Year    "fl"    "1989"
Year    "flow-matic"    "1955"
# The first language suitable for business data processing and the first to have heavy
# emphasis on an "english-like" syntax.
Year    "focal" "1968"
Year    "form"  "1984"
Year    "forth" "1971"
# Fortran was the first higher level language to be widely used. It opened the door
# to practical usage of computers by large numbers of scientific and engineering
# personnel.
# http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/fortran/team/
# 1953  Backus (of IBM) sends a letter to his boss asking that he be allowed to search
# for a better way of programming.
# http://inventors.about.com/od/computersoftware/a/Fortran.htm
# 1954 Backus invents fortran.
# 1957 Fortran is released commercially
Year    "fortran"       "1954"
Year    "fortress"      "2011"
Year    "fp"    "1977"
Year    "fpr"   "2011"
Year    "free_pascal"   "1997"
Year    "fril"  "1980"
Year    "frink" "2001"
Year    "gambas"        "1999"
Year    "game_maker_language"   "1999"
Year    "gamemonkey_script"     "2002"
Year    "general_algebraic_modeling_system"     "1978"
# Generic java is a dialect of Java
Year    "generic_java"  "2001"
Year    "genie" "2008"
Year    "gml"   "1969"
Year    "gnu_assembler"         "1988"
Year    "gnu_bison"     "1986"
Year    "gnu_e" "1991"
Year    "gnu_guile"     "1995"
Year    "go"    "2009"
Year    "gofer" "1994"
Year    "goo"   "2003"
Year    "grass" "1974"
Year    "groovy"        "2003"
Year    "hamilton_c_shell"      "1988"
Year    "harbour_(software)"    "1999"
Year    "hartmann_pipeline"     "1986"
Year    "haskell"       "1987"
Year    "high_level_assembly"   "1999"
Year    "hop_(software)"        "2006"
Year    "hope"  "1980"
Year    "html"  "1990"
Year    "hypercard"     "1987"
Year    "hypertalk"     "1987"
Year    "ibm_netrexx"   "1996"
Year    "ibm_rpg_iv"    "1994"
Year    "ibm_rpg"       "1959"
Year    "ibm_rpg_ii"    "1969"
Year    "ibm_rpg_iii"   "1978"
Year    "ibm_visualage"         "1984"
Year    "icon"  "1977"
# @Class: string processing
Year    "id"    "1979"
Year    "idl"   "1977"
# INFORM created by Lou Santoro and Mike Lowery
# for the time-sharing company Standard Information Systems (SIS).
# http://en.wikipedia.org/wiki/CorVision
Year    "inform_sis"    "1972"
Year    "inform"        "1993"
Year    "adl"   "1987"
Year    "ddl"   "1981"
Year    "gluxl" "????"
Year    "hugo"  "1995"
Year    "zil"   "1979"
Year    "interactive_ruby_shell"        "1996"
Year    "io"    "2002"
Year    "ioke"  "2008"
Year    "iswim" "1966"
Year    "j"     "1990"
Year    "java"  "1995"
Year    "javacc"        "1996"
Year    "javascript"    "1995"
Year    "jess"  "1995"
Year    "job_control_language"  "1964"
```

```
Year    "join_java"    "2000"
Year    "joss"  "1963"
# Joss was developed in beta version in 1963 http://en.wikipedia.org/wiki/JOSS
Year    "joule" "1996"
Year    "jovial"       "1960"
Year    "joy"   "2001"
Year    "joyce" "1987"
Year    "jscript"      "1996"
Year    "k"     "1993"
Year    "kaya"  "2004"
Year    "kent_recursive_calculator"   "1981"
Year    "korn_shell"   "1983"
Year    "labview"      "1986"
Year    "lasso" "1996"
Year    "limbo" "1995"
Year    "lingo" "1988"
Year    "linoleum"     "1996"
Year    "lisaac"       "2003"
Year    "lisp"  "1958"
# @Class: list processing, functional programming
# Lisp is developed in the 1950s for AI programming. In these applications, complex
# interrelationships among data must be represented. Natural data structures are:
# pointers and linked lists.
Year    "little_b"     "2004"
Year    "logo"  "1967"
Year    "lolcode"      "2007"
Year    "lolpython"    "2007"
Year    "lpc"   "1989"
Year    "lua"   "1993"
Year    "lucid" "1976"
Year    "luna"  "2011"
Year    "m4"    "1977"
# @Class: macro processor
# @Author: Brian Kernighan, Dennis Ritchie
# Developed in 1977 based of ideas by Cristopher Stratchey
# http://en.wikipedia.org/wiki/M4_(computer_language)
Cite     "m4"   "gpm"
# m4 derives from GPM
Year    "macsyma"      "1968"
Year    "maple_(software)"     "1980"
Year    "mathematica"  "1988"
Year    "matlab"       "1978"
# MATrix LABoratory. There was no explicit tie to APL.
Year    "max_(software)"       "1985"
Year    "mdl"   "1971"
Year    "mercury"      "1995"
# Mesa was first implemented on the Alto [Thacker79] in 1976. The development
# environment was the Alto operating system [Lampson79] whose user interface
# was the Executive, a BCPL program that operated much like the time-sharing
# executives of the day. From "The Mesa Programming Environment"
# http://www.digibarn.com/collections/papers/dick-sweet/xdepaper.pdf
Year    "mesa"  "1976"
Year    "metafont"     "1979"
Year    "metal" "2001"
Year    "metaocaml"    "1996"
Year    "metapost"     "1994"
Year    "microsoft_dynamics_ax"       "1998"
Year    "minid" "2006"
Year    "mirah" "2008"
# Miranda was developed between 1983-1985
# Orwell released in 1984 as an alternative to miranda,
# so we have to assume that miranda was already well-defined in 1983.
Year    "miranda"      "1983"
Year    "mirc_scripting_language"     "1995"
Year    "ml"    "1973"
Year    "modelica"     "1997"
Year    "modula"       "1975"
Year    "modula-2"     "1978"
Year    "modula-2+"    "1984"
Year    "modula-3"     "1986"
Year    "moo"   "1990"
Year    "muf"   "1990"
Year    "mumps" "1966"
Year    "music-n"      "1957"
Year    "nemerle"      "2003"
Year    "netlogo"      "1999"
Year    "netwide_assembler"    "1996"
Year    "newsqueak"    "1989"
Year    "newtonscript" "1993"
Year    "npl"   "1977"
```

```
Year    "nu"    "2007"
Year    "oaklisp"       "1986"
Year    "oberon"        "1986"
Year    "oberon-2"      "1991"
Year    "obix_programming_language"    "2011"
Year    "object_pascal"         "1986"
Year    "object_rexx"   "1988"
Year    "objective-c"   "1983"
Year    "objective-j"   "2008"
Year    "objvlisp"      "1984"
Year    "obliq" "1995"
Year    "ocaml" "1996"
Year    "occam" "1983"
Year    "opa"   "2010"
Year    "opencl"        "2008"
Year    "openlaszlo"    "2002"
Year    "ops5"  "1977"
Year    "optimj"        "2006"
Year    "oriel_(scripting_language)"  "1991"
Year    "orwell"        "1984"
# Oxygene was first introduced for the .NET platform in 2005. Since its first
# release in 2005, Oxygene has attracted an ever-growing number of developers
# who enjoy its power and flexibility, and the benefit of being able to use
# a single programming language to target any platform they need to develop
# to, without compromise.
#Year   "oxygene"       "1986"
Year    "oxygene"       "2005"
Year    "oz"    "1991"
Year    "p''"   "1964"
Year    "parrot_virtual_machine"       "2009"
Year    "pascal"        "1970"
Year    "pcastl"        "2008"
Year    "perl"  "1987"
Year    "perl_data_language"    "1996"
Year    "perl_shell"    "1999"
Year    "php"   "1995"
Year    "pico"  "1997"
Year    "pike"  "1994"
Year    "pikt"  "1998"
Year    "pizza" "2001"
Year    "pl/c"  "1973"
Year    "pl/i"  "1964"
Year    "pl/sql"        "1988"
Year    "plus"  "1976"
Year    "pop-1" "1968"
Year    "pop-2" "1970"
Year    "postscript"    "1982"
Year    "processing"    "2001"
Year    "prograph"      "1983"
Year    "progress_4gl" "1984"
Year    "progress_4gl_10.1"    "2006"
Year    "prolog"        "1972"
Year    "pure"  "2008"
Year    "pure_data"     "1996"
Year    "python"        "1991"
Year    "q_(equational_programming_language)" "1991"
Year    "q_(programming_language_from_kx_systems)"    "2003"
Year    "qore"  "2007"
Year    "quickbasic"    "1985"
Year    "r"     "1993"
Year    "racket"        "1994"
Year    "reduce"        "1968"
Year    "rebol" "1997"
Year    "refal" "1968"
Year    "reia"  "2008"
Year    "relax_ng"      "2003"
Year    "revolution"    "1993"
# Rexx became reality on March 20th, 1979.  Why is that day considered Rexx's birthday?
# According to Mike Cowlishaw, "'twas the day I woke up at 3am with a clear idea of
# what was needed, and by the end of the day had the initial specification off around
# the world for comment."
# http://www.rexxla.org/rexxlang/
Year    "rexx"  "1979"
Year    "rpl"   "1984"
Year    "rtl/2" "1972"
Year    "ruby"  "1995"
Year    "rust"  "2010"
Year    "s"     "1976"
Year    "sac_programming_language"     "1994"
Year    "salsa" "1998"
```

```
# Year  "sas_system"   "1976"
Year    "sasl"  "1972"
Year    "sather"        "1990"
Year    "scala" "2003"
Year    "scheme"        "1975"
Year    "schoonschip"   "1967"
Year    "scratch"       "2006"
Year    "sed"   "1973"
Year    "seed7" "2005"
Year    "self"  "1987"
Year    "setl"  "1969"
Year    "sgml"  "1986"
Year    "simula"        "1967"
Year    "simulink"      "1984"
Year    "sisal" "1983"
Year    "smalltalk"     "1972"
Year    "sp/k"  "1974"
Year    "speedcoding"   "1953"
Year    "split-c"       "1993"
Year    "sequel"        "1974"
# In 1979, Relational Software, Inc. (now Oracle) introduced the first
# commercially available implementation of SQL.
# http://docs.oracle.com/cd/E11882_01/server.112/e26088/intro001.htm#SQLRF50932
Year    "sql"   "1979"
Year    "squeak"        "1996"
Year    "squirrel"      "2003"
Year    "standard_ml"   "1984"
Year    "starlogo"      "1995"
Year    "sue"   "1971"
Year    "supercollider"         "1996"
Year    "superpascal"   "1993"
Year    "symbolic_manipulation_program"       "1979"
Year    "t"     "1982"
Year    "tads"  "1987"
Year    "tcl"   "1988"
Year    "tea"   "1997"
Year    "telcomp"       "1965"
Year    "thinbasic"     "2011"
Year    "toontalk"      "1995"
Year    "turing"        "1982"
Year    "tutor" "1969"
Year    "typescript"    "2012"
Year    "ucsd_pascal"   "1978"
Year    "unified_parallel_c"    "1999"
Year    "unrealscript" "1998"
Year    "val"   "1979"
Year    "vala"  "2006"
Year    "vbscript"      "1996"
Year    "vhdl"  "1987"
Year    "vimscript"     "1991"
Year    "vissim"        "1989"
Year    "visual_basic" "1991"
Year    "visual_basic_.net"     "2001"
Year    "visual_basic_for_applications"       "1993"
Year    "visual_prolog"         "1986"
Year    "vvvv"  "1998"
Year    "webmethods_flow"       "2000"
Year    "windev"        "1993"
Year    "windows_powershell"    "2006"
Year    "x10"   "2004"
Year    "xc"    "2005"
Year    "xl"    "2000"
Year    "xml"   "1996"
Year    "xpath" "1999"
Year    "xpl"   "1967"
# XQuery 1.0 first release was 2007 but its history goes
# at least back to 2002 when the first public draft was
# released. Languages like Candle (2005) where based on
# XQuery so we have to consider that XQuery existed (and what
# is more important, influenced) languages before 2007.
# check: http://www.w3.org/standards/history/xquery
Year    "xquery"        "2002"
Year    "xslt"  "1999"
Year    "yacc"  "1970"
Year    "yorick"        "1996"
Year    "z_notation"    "1977"
Year    "zgrass"        "1978"
Year    "zpl"   "1993"
Cite    "a+"    "apl"
Cite    "abap"  "cobol"
```

```
Cite    "abap"  "objective-c"
Cite    "abap"  "sql"
Cite    "abc"   "algol_68"
Cite    "abc"   "setl"
Cite    "actionscript" "java"
Cite    "actionscript" "javascript"
Cite    "ada"   "algol_68"
Cite    "ada"   "clu"
Cite    "ada"   "pascal"
Cite    "ada_95"  "ada"
Cite    "ada_95"  "smalltalk"
Cite    "ada_95"   "c++"
Cite    "ada_2005"     "ada_95"
Cite    "ada_2005"     "java"
Cite    "agentsheets"  "lisp"
Cite    "agentsheets"  "logo"
Cite    "agentsheets"  "smalltalk"
Cite    "aimms" "general_algebraic_modeling_system"
Cite    "alef"  "c"
Cite    "alef"  "newsqueak"
Cite    "algebraic_logic_functional"  "prolog"
# Cite  "algol" "algol_58"
Cite    "algol_58"     "fortran"
Cite    "algol_60"     "algol_58"
Cite    "algol_68"     "algol_60"
Cite    "algol_w"      "algol_60"
Cite    "alice" "ml"
Cite    "alice" "oz"
Cite    "alma-0"       "modula-2"
Cite    "ampl"  "awk"
Cite    "ampl"  "c"
Cite    "antlr" "yacc"
Cite    "applescript"  "hypertalk"
Cite     "argus" "clu"
# Argus is an extension of the CLU programming language
Cite    "as/400_control_language"     "job_control_language"
Cite    "as/400_control_language"     "pl/i"
Cite    "aspectj"      "java"
Cite    "ateji_px"     "java"
Cite    "autohotkey"   "autoit"
Cite    "autohotkey"   "basic"
Cite    "autohotkey"   "c"
Cite    "autoit"       "basic"
Cite    "autolisp"     "lisp"
Cite    "awk"   "bourne_shell"
Cite    "awk"   "c"
Cite    "awk"   "snobol"
Cite    "b"     "bcpl"
Cite    "b"     "pl/i"
Cite    "bash_(unix_shell)"    "bourne_shell"
Cite    "bash_(unix_shell)"    "c_shell"
Cite    "basic" "algol_60"
Cite    "basic" "fortran"
Cite    "basic" "joss"
Cite    "bcpl"  "cpl"
# Check "How BCPL Evolved from CPL" by Martin Richards (2011)
Cite    "beanshell"    "java"
Cite    "beta"  "simula"
Cite    "bistro"       "java"
Cite    "bistro"       "smalltalk"
# In the paper "Bliss: a language for systems programming" they mention B5500 Extended Algol (1966)
# as an "extant language that fits our definition" (and thus we conclude it was a reference). The
# Burroughs B5500 implements virtually all of Algol-60.
Cite    "bliss" "algol_60"
Cite    "boo"   "python"
Cite    "bourne_shell" "algol_68"
Cite    "brainfuck"    "false"
Cite    "brainfuck"    "p''"
Cite    "c"     "algol_68"
Cite    "c"     "b"
Cite    "c"     "bcpl"
Cite    "c"     "fortran"
Cite    "c"     "pl/i"
Cite    "c++"   "ada"
Cite    "c++"   "algol_68"
Cite    "c++"   "c"
Cite    "c++"   "clu"
Cite    "c++"   "ml"
Cite    "c++"   "simula"
Cite    "c--"   "c"
```

```
Cite    "c_omega"       "c_sharp"
Cite    "c_sharp"       "c++"
Cite    "c_sharp"       "eiffel"
Cite    "c_sharp"       "java"
Cite    "c_sharp"       "modula-3"
Cite    "c_sharp"       "object_pascal"
Cite    "c_shell"       "c"
Cite    "cal_(quark_framework)#cal"   "haskell"
Cite    "caml"  "ml"
Cite    "candle"        "relax_ng"
Cite    "candle"        "xquery"
Cite    "candle"        "xslt"
Cite    "cel_(programming_language)"  "forth"
Cite    "cel_(programming_language)"  "newtonscript"
Cite    "cel_(programming_language)"  "objective-c"
Cite    "cel_(programming_language)"  "python"
Cite    "cel_(programming_language)"  "self"
Cite    "cel_(programming_language)"  "smalltalk"
Cite    "charm_(language)"      "c"
Cite    "charm_(language)"      "pascal"
Cite    "charm_(language)"      "rtl/2"
Cite    "chuck" "c"
Cite    "chuck" "music-n"
Cite    "cilk"  "c"
Cite    "clacl" "c"
Cite    "clacl" "prolog"
Cite    "clarion"       "basic"
Cite    "clarion"       "cobol"
Cite    "clarion"       "pascal"
Cite    "clean" "haskell"
Cite    "clean" "miranda"
Cite    "clipper"       "c"
Cite    "clipper"       "dbase"
Cite    "clips" "lisp"
Cite    "clips" "ops5"
Cite    "clojure"       "common_lisp"
Cite    "clojure"       "erlang"
Cite    "clojure"       "haskell"
Cite    "clojure"       "java"
Cite    "clojure"       "ml"
Cite    "clojure"       "prolog"
Cite    "clojure"       "ruby"
Cite    "clojure"       "scheme"
Cite    "clu"   "algol_60"
Cite    "clu"   "lisp"
Cite    "clu"   "simula"
Cite    "co-array_fortran"      "fortran"
Cite    "cobol" "comtran"
# http://en.wikipedia.org/wiki/COBOL
Cite    "cobol" "flow-matic"
# http://en.wikipedia.org/wiki/COBOL
Cite     "cobol" "fact"
# FACT was an influence on the design of the COBOL  prog lang
# http://en.wikipedia.org/wiki/FACT_computer_language
# http://en.wikipedia.org/wiki/COBOL
Cite    "cobra" "c_sharp"
Cite    "cobra" "eiffel"
Cite    "cobra" "objective-c"
Cite    "cobra" "python"
Cite    "coffeescript" "haskell"
Cite    "coffeescript" "javascript"
Cite    "coffeescript" "perl"
Cite    "coffeescript" "python"
Cite    "coffeescript" "ruby"
Cite    "coldfusion"    "html"
Cite    "coldfusion"    "javascript"
Cite    "colorforth"    "forth"
Cite    "comal" "basic"
Cite    "comal" "pascal"
Cite    "common_lisp"   "lisp"
Cite    "common_lisp"   "scheme"
Cite    "component_pascal"      "oberon-2"
Cite    "comtran"       "flow-matic"
Cite     "communicating_sequential_processes" "simula"
Cite    "corvision"     "inform_sis"
Cite    "cowsel"        "cpl"
Cite    "cowsel"        "lisp"
Cite    "cpl"   "algol_60"
Cite    "csound"        "c"
Cite    "csound"        "music-n"
```

```
Cite    "curl"  "html"
Cite    "curl"  "javascript"
Cite    "curl"  "lisp"
Cite    "curry" "haskell"
Cite    "cyclone"       "c"
Cite    "d"     "c"
Cite    "d"     "c++"
Cite    "d"     "c_sharp"
Cite    "d"     "eiffel"
Cite    "d"     "java"
Cite    "d"     "python"
Cite    "d"     "ruby"
Cite    "dart"  "java"
Cite    "dart"  "javascript"
Cite    "datalog"       "prolog"
Cite    "dibol" "basic"
Cite    "dibol" "fortran"
Cite    "dylan" "algol_68"
Cite    "dylan" "common_lisp"
Cite    "dylan" "eulisp"
Cite    "dylan" "scheme"
Cite    "e"     "java"
Cite    "e"     "joule"
Cite    "ecmascript"    "awk"
Cite    "ecmascript"    "c"
Cite    "ecmascript"    "hypertalk"
Cite    "ecmascript"    "java"
Cite    "ecmascript"    "javascript"
Cite    "ecmascript"    "perl"
Cite    "ecmascript"    "python"
Cite    "ecmascript"    "scheme"
Cite    "ecmascript"    "self"
Cite    "eel_(extensible_embeddable_language)"       "c"
Cite    "eel_(extensible_embeddable_language)"       "lua"
Cite    "eel_(extensible_embeddable_language)"       "pascal"
Cite    "eiffel"        "ada"
Cite    "eiffel"        "simula"
Cite    "eiffel"        "z_notation"
Cite    "epigram"       "haskell"
Cite    "erlang"        "communicating_sequential_processes"
Cite    "erlang"        "ml"
Cite    "erlang"        "prolog"
Cite    "escher"        "haskell"
Cite    "etoys" "logo"
Cite    "etoys" "smalltalk"
Cite    "etoys" "starlogo"
Cite    "euclid"        "pascal"
Cite    "eulisp"        "common_lisp"
Cite    "eulisp"        "dylan"
Cite    "eulisp"        "oaklisp"
Cite    "eulisp"        "objvlisp"
Cite    "eulisp"        "scheme"
Cite    "eulisp"        "t"
Cite    "exec_2"        "cms_exec"
Cite    "f-script"      "apl"
Cite    "f-script"      "smalltalk"
Cite    "f_sharp"       "c_sharp"
Cite    "f_sharp"       "erlang"
Cite    "f_sharp"       "haskell"
Cite    "f_sharp"       "ml"
Cite    "f_sharp"       "ocaml"
Cite    "f_sharp"       "python"
Cite    "factor"        "forth"
Cite    "factor"        "joy"
Cite    "factor"        "lisp"
Cite    "factor"        "self"
Cite    "falcon"        "c++"
Cite    "falcon"        "lisp"
Cite    "falcon"        "lua"
Cite    "falcon"        "perl"
Cite    "falcon"        "php"
Cite    "falcon"        "python"
Cite    "falcon"        "ruby"
Cite    "falcon"        "smalltalk"
Cite    "false" "forth"
Cite    "fancy" "erlang"
Cite    "fancy" "io"
Cite    "fancy" "ruby"
Cite    "fancy" "smalltalk"
Cite    "fantom"        "c_sharp"
```

```
Cite    "fantom"        "erlang"
Cite    "fantom"        "java"
Cite    "fantom"        "ruby"
Cite    "fantom"        "scala"
Cite    "adl"   "c"
Cite    "adl"   "ddl"
Cite    "adl"   "lisp"
Cite    "ddl"   "lisp"
# Cite  "glulx" "inform"
# Cite  "glulx" "zil"
Cite    "hugo"  "basic"
Cite    "hugo"  "c"
Cite    "hugo"  "inform"
Cite    "inform"        "c"
Cite    "inform"        "zil"
Cite    "mdl"   "lisp"
Cite    "zil"   "mdl"
Cite    "fl"    "fp"
Cite    "focal" "joss"
Cite    "form"  "schoonschip"
Cite    "forth" "apl"
Cite    "forth" "lisp"
Cite    "fortran"       "speedcoding"
Cite    "fortress"      "fortran"
Cite    "fortress"      "haskell"
Cite    "fortress"      "scala"
Cite    "fp"    "apl"
Cite    "fpr"   "lisp"
Cite    "free_pascal"   "pascal"
Cite    "fril"  "prolog"
Cite    "frink" "java"
Cite    "frink" "perl"
Cite    "frink" "smalltalk"
Cite    "gambas"        "java"
Cite    "gambas"        "visual_basic"
Cite    "game_maker_language"   "c"
Cite    "gamemonkey_script"     "c"
Cite    "gamemonkey_script"     "lua"
Cite    "generic_java" "java"
Cite    "generic_java" "pizza"
Cite    "genie" "boo"
Cite    "genie" "d"
Cite    "genie" "object_pascal"
Cite    "genie" "python"
Cite    "gnu_bison"     "yacc"
Cite    "gnu_e" "c++"
Cite    "gnu_guile"     "scheme"
# Gofer ("Good For Equational Reasoning") is an implementation of the programming language
# Haskell intended for educational purposes and supporting a language based on version 1.2
# of the Haskell report. It was replaced by Hugs.[1]
# http://en.wikipedia.org/wiki/Gofer_(software)
Cite    "gofer"         "haskell"
# Gofer syntax is closer to the earlier commercial language Miranda.
# However, gofer is NOT Miranda - they are similar, but they are not the same
Cite    "gofer"   "miranda"
Cite    "go"    "c"
Cite    "go"    "communicating_sequential_processes"
Cite    "go"    "limbo"
Cite    "go"    "modula"
Cite    "go"    "newsqueak"
Cite    "go"    "oberon"
Cite    "go"    "pascal"
Cite    "go"    "python"
Cite    "goo"   "dylan"
Cite    "goo"   "scheme"
Cite    "grass" "basic"
Cite    "groovy"        "java"
Cite    "groovy"        "objective-c"
Cite    "groovy"        "perl"
# Groovy was motivated by the desire to bring the Python
# design philosophy to Java
# See: http://radio-weblogs.com/0112098/2003/08/29.html
Cite    "groovy"        "python"
Cite    "groovy"        "ruby"
Cite    "groovy"        "smalltalk"
Cite    "hamilton_c_shell"      "c_shell"
Cite    "harbour_(software)"    "clipper"
Cite    "hartmann_pipeline"     "apl"
Cite    "haskell"       "apl"
Cite    "haskell"       "clean"
```

```
Cite    "haskell"       "fp"
# gofer is a dialect of haskell
# Cite  "haskell"       "gofer"
Cite    "haskell"       "hope"
Cite    "haskell"       "id"
Cite    "haskell"       "iswim"
Cite    "haskell"       "kent_recursive_calculator"
Cite    "haskell"       "lisp"
Cite    "haskell"       "miranda"
Cite    "haskell"       "ml"
Cite    "haskell"       "orwell"
Cite    "haskell"       "sasl"
Cite    "haskell"       "scheme"
Cite    "haskell"       "sisal"
Cite    "haskell"       "standard_ml"
Cite    "high_level_assembly"  "ada"
Cite    "high_level_assembly"  "c++"
Cite    "high_level_assembly"  "modula-2"
Cite    "high_level_assembly"  "pascal"
Cite    "hop_(software)"       "scheme"
Cite    "hope"  "npl"
Cite    "html"  "sgml"
Cite    "ibm_netrexx"  "basic"
Cite    "ibm_netrexx"  "exec_2"
Cite    "ibm_netrexx"  "java"
Cite    "ibm_netrexx"  "pl/i"
Cite    "ibm_rpg"       "fargo"
Cite    "ibm_rpg_ii"    "ibm_rpg"
Cite    "ibm_rpg_iii"   "ibm_rpg_ii"
Cite    "ibm_rpg_iv"    "ibm_rpg_iii"
Cite    "ibm_rpg_iv"    "sql"
# The Icon language is derived from the ALGOL-class of structured programming
#  languages, and thus has syntax similar to C or Pascal. Icon is most similar
# to Pascal, using := syntax for assignments, the procedure keyword and similar
#  syntax. On the other hand, Icon uses C-style brackets for structuring execution
# groups, and programs start by running a procedure called "main".
Cite    "icon"  "algol_60"
Cite    "icon"  "snobol"
Cite    "icon"  "sl5"
# @Info: Icon combined the backtracking of SNOBOL4 pattern matching with more standard ALGOL-like
structuring, as well as adding some features of their own.
# http://en.wikipedia.org/wiki/SNOBOL
Cite    "idl"   "fortran"
Cite    "interactive_ruby_shell"      "ruby"
Cite    "io"    "lisp"
Cite    "io"    "lua"
Cite    "io"    "newtonscript"
Cite    "io"    "self"
Cite    "io"    "smalltalk"
Cite    "ioke"  "io"
Cite    "ioke"  "lisp"
Cite    "ioke"  "ruby"
Cite    "ioke"  "smalltalk"
Cite    "iswim" "algol_60"
Cite    "iswim" "lisp"
Cite    "j"     "apl"
Cite    "j"     "fl"
Cite    "j"     "fp"
Cite    "java"  "ada"
Cite    "java"  "c++"
# ava 5.0 added several new language features (the enhanced
# for loop, autoboxing, varargs and annotations), after
# they were introduced in the similar (and competing) C# language
# Cite  "java"  "c_sharp"
Cite    "java"  "eiffel"
Cite    "java"  "mesa"
Cite    "java"  "modula-3"
Cite    "java"  "oberon"
Cite    "java"  "objective-c"
Cite    "java"  "smalltalk"
Cite    "java"  "ucsd_pascal"
Cite    "javacc"        "yacc"
Cite    "javascript"    "c"
Cite    "javascript"    "java"
Cite    "javascript"    "perl"
Cite    "javascript"    "python"
Cite    "javascript"    "scheme"
Cite    "javascript"    "self"
Cite    "jess"  "clips"
Cite    "join_java"     "java"
```

```
Cite    "joss"  "algol_58"
# JOVIAL is an acronym for "Jules Own Version of the International Algorithmic Language."
# The "International Algorithmic Language" was a name originally proposed for ALGOL 58.
# It was developed by Jules Schwartz in 1959 to compose software for the electronics of
# military aircraft.
Cite    "jovial" "algol_58"
Cite    "joy"   "c"
Cite    "joy"   "scheme"
Cite    "joyce" "communicating_sequential_processes"
Cite    "joyce" "concurrent_pascal"
Cite    "joyce" "pascal"
Cite    "jscript"       "javascript"
Cite    "k"     "a+"
Cite    "k"     "scheme"
Cite    "kaya"  "c"
Cite    "kaya"  "haskell"
Cite    "kaya"  "ocaml"
Cite    "kent_recursive_calculator"   "sasl"
Cite    "korn_shell"   "awk"
Cite    "korn_shell"   "bourne_shell"
Cite    "lasso" "html"
Cite    "limbo" "alef"
Cite    "limbo" "c"
Cite    "limbo" "communicating_sequential_processes"
Cite    "limbo" "newsqueak"
Cite    "limbo" "pascal"
Cite    "lingo" "hypertalk"
Cite    "lisaac"        "eiffel"
Cite    "lisaac"        "self"
Cite    "lisaac"        "smalltalk"
Cite    "lisp"  "ipl"
Cite    "little_b"      "lisp"
Cite    "logo"  "lisp"
Cite    "lolpython"     "lolcode"
Cite    "lolpython"     "python"
Cite    "lpc"   "c"
Cite    "lpc"   "c++"
Cite    "lpc"   "lisp"
Cite    "lpc"   "perl"
Cite    "lua"   "awk"
Cite    "lua"   "c++"
Cite    "lua"   "clu"
Cite    "lua"   "modula-2"
Cite    "lua"   "scheme"
Cite    "lua"   "snobol"
Cite    "lucid" "iswim"
Cite    "luna"  "io"
Cite    "luna"  "javascript"
Cite    "luna"  "lua"
Cite    "luna"  "python"
Cite    "luna"  "ruby"
Cite    "macsyma"       "lisp"
Cite    "maple_(software)"      "macsyma"
Cite    "maple_(software)"      "pascal"
Cite    "mathematica"  "symbolic_manipulation_program"
Cite    "mathematica"  "lisp"
Cite    "mathematica"  "macsyma"
Cite    "matlab"       "fortran"
# The first version of matlab was implemented in fortran and based on some fortan
# libraries for matrix manipulation (matlab means MATrix LABoratory). Arrays
# in matlab start in '1'      like fortran.  Later on, matlab was rewritten in C.
Cite    "max_(software)"       "music-n"
Cite    "mdl"   "lisp"
Cite    "mercury"      "haskell"
Cite    "mercury"      "prolog"
# "Mesa is derived from Pascal"
# http://research.microsoft.com/en-us/um/people/blampson/32a-CedarLang/32a-CedarLangOCR.htm
Cite    "mesa"  "algol_60"
Cite    "mesa"  "pascal"
Cite    "metal" "xml"
Cite    "metaocaml"     "standard_ml"
Cite    "metapost"      "metafont"
Cite    "microsoft_dynamics_ax"       "c++"
Cite    "minid" "d"
Cite    "minid" "io"
Cite    "minid" "javascript"
Cite    "minid" "lua"
Cite    "minid" "python"
Cite    "minid" "squirrel"
Cite    "mirah" "boo"
```

```
Cite    "mirah" "java"
Cite    "mirah" "ruby"
Cite    "miranda"       "hope"
Cite    "miranda"       "kent_recursive_calculator"
Cite    "miranda"       "ml"
Cite    "miranda"       "sasl"
Cite    "ml"    "iswim"
Cite    "modula"        "pascal"
# Modula-2 was understood by Niklaus Wirth as a successor to his earlier
# programming language Pascal.[2] The language design was also influenced
# by the Mesa programming language and the new programming possibilities
# of the early personal computer Xerox Alto, both from Xerox, that Wirth
# saw during his 1976 sabbatical year at Xerox PARC.
Cite    "modula-2"      "mesa"
Cite    "modula-2"      "algol_w"
Cite    "modula-2"      "modula"
Cite    "modula-2"      "pascal"
# the difference between modula-2+ and modula-2 was: concurrency,
# exception handling and garbage collector.
# Cite  "modula-2+"     "algol_w"
# NOT SURE IF Algol-60 or Algol-W
Cite    "modula-2+"     "modula-2"
Cite    "modula-2+"     "pascal"
# Modula-3 is mostly GC (Garbage Collector), like Algol-68.
# Cite  "modula-3"      "algol_w"
# NOT SURE IF Algol-60 or Algol-W
Cite    "modula-3"      "modula-2"
Cite    "modula-3"      "modula-2+"
Cite    "modula-3"      "oberon"
Cite    "modula-3"      "pascal"
Cite    "moo"   "ada"
Cite    "moo"   "c"
Cite    "moo"   "muf"
Cite    "moo"   "scheme"
Cite    "moo"   "self"
Cite    "moo"   "smalltalk"
Cite    "muf"   "forth"
Cite    "mumps" "joss"
Cite    "nemerle"       "c_sharp"
Cite    "nemerle"       "lisp"
Cite    "nemerle"       "ml"
Cite    "netlogo"       "logo"
Cite    "netlogo"       "starlogo"
Cite    "newsqueak"     "c"
Cite    "newsqueak"     "communicating_sequential_processes"
Cite    "newtonscript" "dylan"
Cite    "newtonscript" "self"
Cite    "nu"    "lisp"
Cite    "nu"    "objective-c"
Cite    "nu"    "ruby"
Cite    "oaklisp"       "scheme"
Cite    "oaklisp"       "smalltalk"
Cite    "oberon"        "modula-2"
Cite    "oberon-2"      "oberon"
Cite    "obix_programming_language"   "java"
Cite    "object_pascal"         "pascal"
Cite    "object_pascal"         "smalltalk"
Cite    "object_rexx"   "rexx"
Cite    "objective-c"   "c"
Cite    "objective-c"   "smalltalk"
Cite    "objective-j"   "javascript"
Cite    "objective-j"   "objective-c"
Cite    "objvlisp"      "lisp"
Cite    "obliq" "modula-3"
Cite    "obliq" "oberon"
Cite    "obliq" "self"
Cite    "ocaml" "caml"
Cite    "ocaml" "standard_ml"
Cite    "occam" "communicating_sequential_processes"
Cite    "opa"   "erlang"
Cite    "opa"   "javascript"
Cite    "opa"   "ocaml"
Cite    "opencl"        "c"
Cite    "openlaszlo"    "javascript"
Cite    "openlaszlo"    "xml"
Cite    "ops5"  "lisp"
Cite    "optimj"        "java"
Cite    "orwell"        "miranda"
# Oxygene is a dialect of Object Pascal
Cite    "oxygene"       "object_pascal"
```

```
Cite    "oxygene"       "c_sharp"
Cite    "oz"    "erlang"
Cite    "oz"    "lisp"
Cite    "oz"    "prolog"
Cite    "pascal"        "algol_w"
Cite    "pcastl"        "c"
Cite    "pcastl"        "r"
Cite    "perl"  "awk"
Cite    "perl"  "c"
Cite    "perl"  "c++"
Cite    "perl"  "lisp"
Cite    "perl"  "pascal"
Cite    "perl"  "sed"
Cite    "perl"  "smalltalk"
Cite    "perl_data_language"    "apl"
Cite    "perl_data_language"    "perl"
Cite    "perl_shell"    "bash_(unix_shell)"
Cite    "perl_shell"    "perl"
Cite    "php"   "c"
Cite    "php"   "c++"
Cite    "php"   "java"
Cite    "php"   "perl"
Cite    "php"   "tcl"
Cite    "pico"  "scheme"
Cite    "pike"  "c"
Cite    "pike"  "c++"
Cite    "pike"  "lpc"
Cite    "pizza" "java"
Cite    "pl/c"  "pl/i"
# IT can't be algol-68 because PL/1 appeared before in 1964.
Cite    "pl/i"  "algol_60"
Cite    "pl/i"  "cobol"
Cite    "pl/i"  "fortran"
Cite    "pl/sql"        "sql"
Cite    "pl/sql"        "ada"
Cite    "pl/sql"        "pascal"
Cite    "plus"  "pascal"
Cite    "plus"  "sue"
Cite    "pop-1" "cowsel"
Cite    "pop-2" "algol_60"
Cite    "pop-2" "lisp"
Cite    "pop-2" "pop-1"
Cite    "postscript"    "forth"
Cite    "processing"    "c"
Cite    "processing"    "design_by_numbers"
Cite    "processing"    "java"
Cite    "processing"    "postscript"
Cite    "progress_4gl" "sql"
Cite    "progress_4gl_10.1"     "progress_4gl"
Cite    "pure"  "q_(equational_programming_language)"
Cite    "pure_data"     "music-n"
Cite    "python"        "abc"
Cite    "python"        "algol_68"
Cite    "python"        "c"
Cite    "python"        "c++"
Cite    "python"        "dylan"
Cite    "python"        "haskell"
Cite    "python"        "icon"
# The interaction between python and java occurred much later
# I think Python 2.4a3 (released 2004)
# see: http://legacy.python.org/dev/peps/pep-0318/
# There is some history in Java using @ initially as a marker
# in Javadoc comments [23] and later in Java 1.5 for annotations
# [10], which are similar to Python decorators.
# Cite  "python"        "java"
Cite    "python"        "lisp"
Cite    "python"        "modula-3"
Cite    "python"        "perl"
Cite    "q_(equational_programming_language)" "haskell"
Cite    "q_(equational_programming_language)" "ml"
Cite    "q_(programming_language_from_kx_systems)"    "a+"
Cite    "q_(programming_language_from_kx_systems)"    "k"
Cite    "q_(programming_language_from_kx_systems)"    "scheme"
Cite    "qore"  "c++"
Cite    "qore"  "d"
Cite    "qore"  "java"
Cite    "qore"  "perl"
Cite    "r"     "s"
Cite    "r"     "scheme"
Cite    "racket"        "eiffel"
```

```
Cite    "racket"        "scheme"
Cite    "reduce"        "lisp"
Cite    "rebol" "forth"
Cite    "rebol" "lisp"
Cite    "rebol" "logo"
Cite    "rebol" "self"
Cite    "reia"  "erlang"
Cite    "reia"  "python"
Cite    "reia"  "ruby"
Cite    "relax_ng"      "xml"
Cite    "revolution"    "hypertalk"
Cite    "rexx"  "algol_60"
Cite    "rexx"  "exec_2"
Cite    "rexx"  "pl/i"
Cite    "rpl"   "forth"
Cite    "ruby"  "ada"
Cite    "ruby"  "c++"
Cite    "ruby"  "clu"
Cite    "ruby"  "dylan"
Cite    "ruby"  "eiffel"
Cite    "ruby"  "lisp"
Cite    "ruby"  "perl"
Cite    "ruby"  "python"
Cite    "ruby"  "smalltalk"
Cite    "rust"  "alef"
Cite    "rust"  "c++"
Cite    "rust"  "c_sharp"
Cite    "rust"  "common_lisp"
Cite    "rust"  "cyclone"
Cite    "rust"  "erlang"
Cite    "rust"  "newsqueak"
Cite    "rust"  "ruby"
Cite    "rust"  "sather"
Cite    "rust"  "standard_ml"
Cite    "s"     "c"
Cite    "s" "ppl"
# http://en.wikipedia.org/wiki/S_programming_language
Cite    "s" "apl"
# http://en.wikipedia.org/wiki/S_programming_language
Cite    "s" "fortran"
# http://en.wikipedia.org/wiki/S_programming_language
Cite    "sac_programming_language"      "apl"
Cite    "sac_programming_language"      "c"
Cite    "sac_programming_language"      "sisal"
# Cite  "sas_system"    "pl/i"
# Cite  "sas_system"    "sql"
Cite    "sasl"  "iswim"
Cite    "sather"        "clu"
Cite    "sather"        "common_lisp"
Cite    "sather"        "eiffel"
Cite    "sather"        "scheme"
Cite    "scala" "eiffel"
Cite    "scala" "erlang"
Cite    "scala" "haskell"
Cite    "scala" "java"
Cite    "scala" "ocaml"
Cite    "scala" "pizza"
Cite    "scala" "scheme"
Cite    "scala" "smalltalk"
Cite    "scala" "standard_ml"
# The "Revised5 Report on the Algorithmic Language Scheme" only mentions Algol-60
Cite    "scheme"        "algol_60"
Cite    "scheme"        "lisp"
Cite    "scheme"        "mdl"
Cite    "scratch"       "agentsheets"
Cite    "scratch"       "etoys"
Cite    "scratch"       "logo"
Cite    "scratch"       "smalltalk"
Cite    "scratch"       "starlogo"
Cite    "seed7" "ada"
Cite    "seed7" "algol_68"
Cite    "seed7" "c"
Cite    "seed7" "c++"
Cite    "seed7" "java"
Cite    "seed7" "modula-2"
Cite    "seed7" "pascal"
Cite    "self"  "smalltalk"
Cite    "sgml"  "gml"
Cite    "simula"        "algol_60"
# Dahl and Nygaard, "Simula, an Algol based simulation language" CACM 9: 671-678 (1966).
```

```
Cite    "sisal" "c"
Cite    "sisal" "fortran"
Cite    "sisal" "pascal"
Cite    "sisal" "val"
Cite    "smalltalk"    "lisp"
Cite    "smalltalk"    "logo"
Cite    "smalltalk"    "simula"
# Smalltalk-71 language was very influenced by FLEX, PLANNER, LOGO and META II
# See: "The Early History of Smalltalk" by Alan C. Kay
Cite    "sp/k"  "pl/i"
Cite    "split-c"      "c"
Cite    "sql"   "sequel"
Cite    "sql"   "datalog"
Cite    "squeak"       "smalltalk"
Cite    "squirrel"     "c++"
Cite    "squirrel"     "javascript"
Cite    "squirrel"     "lua"
Cite    "squirrel"     "python"
Cite    "standard_ml"  "hope"
Cite    "standard_ml"  "ml"
Cite    "starlogo"     "logo"
Cite    "sue"   "pascal"
Cite    "supercollider"        "c"
Cite    "supercollider"        "music-n"
Cite    "supercollider"        "smalltalk"
Cite    "superpascal"  "communicating_sequential_processes"
Cite    "superpascal"  "concurrent_pascal"
Cite    "superpascal"  "joyce"
Cite    "superpascal"  "occam"
Cite    "superpascal"  "pascal"
Cite    "symbolic_manipulation_program"       "macsyma"
Cite    "symbolic_manipulation_program"       "schoonschip"
Cite    "t"     "scheme"
Cite    "tads"  "c"
Cite    "tads"  "pascal"
Cite    "tcl"   "awk"
Cite    "tcl"   "lisp"
Cite    "tea"   "java"
Cite    "tea"   "scheme"
Cite    "tea"   "tcl"
Cite    "telcomp"      "joss"
Cite    "thinbasic"    "basic"
Cite    "turing"       "euclid"
Cite    "turing"       "pascal"
Cite    "turing"       "sp/k"
Cite    "tutor" "cobol"
Cite    "typescript"   "c_sharp"
Cite    "typescript"   "java"
Cite    "typescript"   "javascript"
Cite    "ucsd_pascal"  "pascal"
Cite    "unified_parallel_c"   "c"
Cite    "unified_parallel_c"   "split-c"
Cite    "unrealscript" "c++"
Cite    "unrealscript" "java"
Cite    "val"   "clu"
Cite    "vala"  "c"
Cite    "vala"  "c++"
Cite    "vala"  "c_sharp"
Cite    "vala"  "d"
Cite    "vala"  "java"
Cite    "vbscript"     "visual_basic"
Cite    "vhdl"  "ada"
Cite    "vhdl"  "pascal"
Cite    "vissim"       "c"
Cite    "visual_basic" "basic"
Cite    "visual_basic_.net"    "haskell"
Cite    "visual_basic_.net"    "visual_basic"
Cite    "visual_basic_for_applications"       "quickbasic"
Cite    "visual_basic_for_applications"       "visual_basic"
Cite    "visual_prolog"        "prolog"
Cite    "vvvv"  "apl"
Cite    "vvvv"  "pure_data"
Cite    "webmethods_flow"      "java"
Cite    "windows_powershell"   "as/400_control_language"
Cite    "windows_powershell"   "c_sharp"
Cite    "windows_powershell"   "digital_command_language"
Cite    "windows_powershell"   "korn_shell"
Cite    "windows_powershell"   "perl"
Cite    "windows_powershell"   "sql"
Cite    "windows_powershell"   "tcl"
```

```
Cite    "x10"   "java"
Cite    "xc"    "c"
Cite    "xc"    "communicating_sequential_processes"
Cite    "xc"    "occam"
Cite    "xl"    "ada"
Cite    "xl"    "c++"
Cite    "xml"   "sgml"
Cite    "xpath" "xslt"
Cite    "xpl"   "pl/i"
Cite    "xquery"        "sql"
Cite    "xquery"        "xpath"
Cite    "xquery"        "xslt"
Cite    "xslt"  "xml"
Cite    "zgrass"        "grass"
Cite    "zpl"   "c"
```