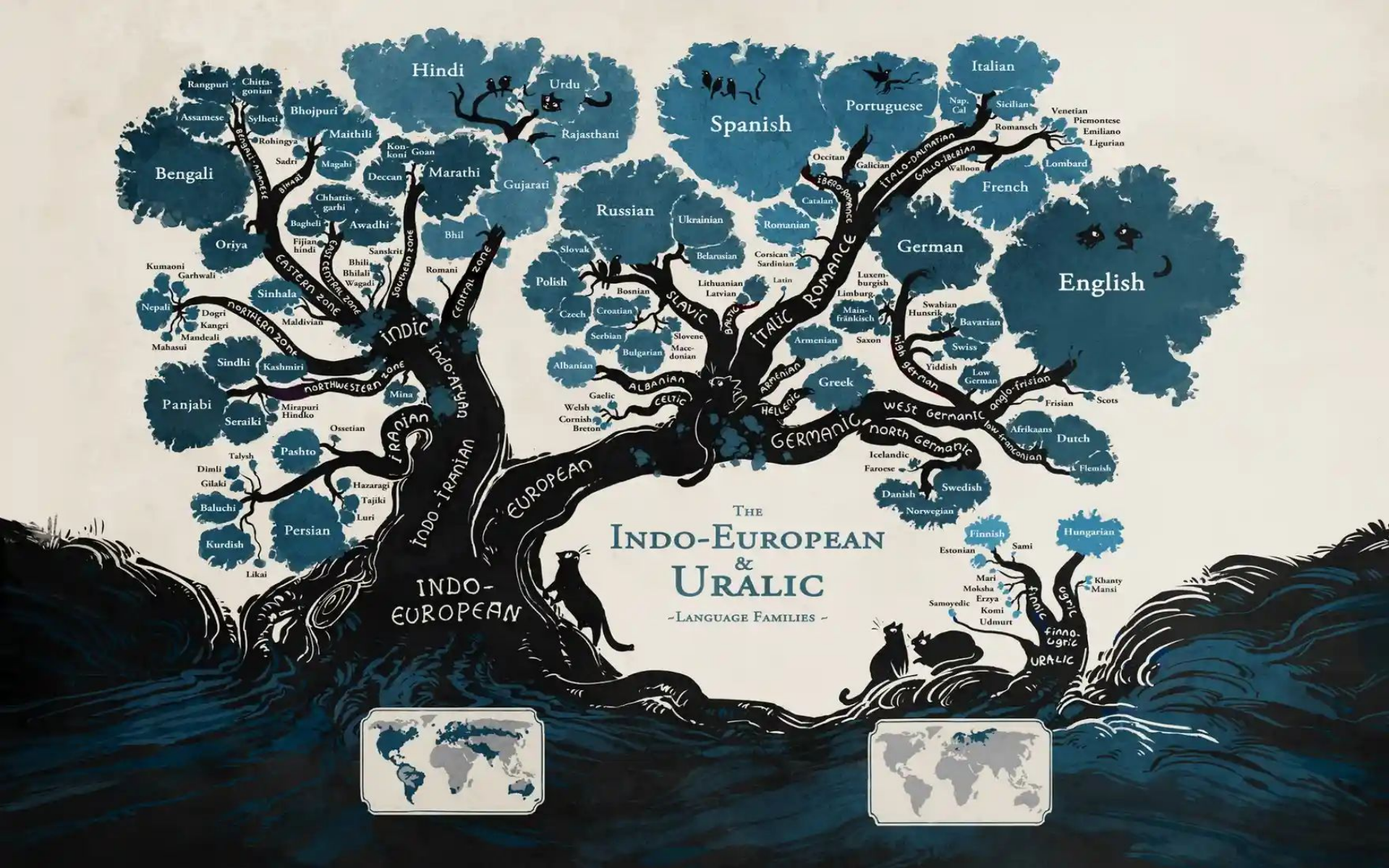

Hindi
Urdu
Rajasthani
Gujarati
Bhojpuri
Maithili
Magahi
Bengali
Rangpuri
Chitta-gonian
Assamese
Sylheti
Rohingy
Sadri
BENGALI-ASSAMESE
BIHARI
Konkoni
Goan
Deccan
Marathi
Chhattis-garhi
Bagheli
Awadhi
Fijian hindi
Oriya
EASTERN ZONE
EAST CENTRAL ZONE
SOUTHERN ZONE
Bhil
Sanskrit
Bhili
Bhilali
Wagadi
Romani
CENTRAL ZONE
Kumaoni
Garhwali
Sinhala
Maldivian
Nepali
Dogri
Kangri
Mandeali
Mahasu
NORTHERN ZONE
INDIC
Sindhi
Kashmiri
NORTHWESTERN ZONE
Panjabi
Seraiki
Mirapuri
Hindko
Mina
INDO-ARYAN
Ossetian
IRANIAN
Pashto
Talysh
Dimli
Gilaki
Hazaragi
Tajiki
Luri
Baluchi
Persian
Kurdish
Likai
INDO-IRANIAN
EUROPEAN
INDO-EUROPEAN
Spanish
Portuguese
Italian
Nap. Cal
Sicilian
Venetian
Piemontese
Emiliano
Ligurian
Romansch
ITALO-DALMATIAN
GALLO-IBERIAN
Occitan
Galician
Walloon
Lombard
French
IBERO-ROMANCE
Catalan
Romanian
Russian
Ukrainian
Belarusian
Slovak
Polish
Corsican
Sardinian
Latin
Lithuanian
Latvian
Bosnian
Czech
Croatian
Serbian
Slovene
Bulgarian
Macce-donian
SLAVIC
BALTIC
ITALIC
ROMANCE
Luxem-burgish
Limburg.
Main-fränkisch
Saxon
Armenian
German
English
Swabian
Hunsrik
Bavarian
Swiss
Yiddish
Low German
high german
Albanian
ALBANIAN
Greek
HELLENIC
ARMENIAN
Gaelic
Welsh
Cornish
Breton
CELTIC
Frisian
Scots
anglo-frisian
WEST GERMANIC
low franconian
Afrikaans
Dutch
Flemish
GERMANIC
NORTH GERMANIC
Icelandic
Faroese
Swedish
Danish
Norwegian
THE
INDO-EUROPEAN
&
URALIC
-LANGUAGE FAMILIES-
Finnish
Estonian
Sami
Hungarian
Khanty
Mansi
Mari
Moksha
Erzya
Komi
Udmurt
Samoyedic
FINNIC
UGRIC
FINNO-UGRIC
URALIC


What about programming languages?
Hindi
Urdu
Portuguese
Italian
Nap. Cal
Sicilian
Romansch
Venetian
Piemontese
Emiliano
Ligurian
Lombard
French
English
Rangpuri
Chitta-gonian
Assamese
Sylheti
Bhojpuri
Rohingya
Bengali
Oriya
Kumaoni
Garhwal
Sinhala
Nepali
Dogri
Kangri
Mandeali
Mahasui
Sindhi
Panjabi
Seraiki
Talysh
Dimli
Gilaki
Baluchi
Kurdish
Persian
Likai
anglo-frisian
Frisian
Scots
Afrikaans
Dutch
low franconian
Flemish
Finnish
Hungarian
Estonian
Sami
Khanty
Mansi
Mari
Moksha
Erzya
Samoyedic
Komi
Udmurt
finnic
ugric
finno-ugric
URALIC
INDO-EUROPEAN
URALIC
~LANGUAGE FAMILIES~

Fortran

From Wikipedia, the free encyclopedia

Fortran (/ˈfɔːrtræn/; formerly **FORTRAN**) is a general-purpose, compiled imperative programming language that is especially suited to numeric computation and scientific computing.

Fortran was originally developed by IBM[2] in the 1950s for scientific and engineering applications, and subsequently came to dominate scientific computing. It has been in use for over six decades in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, geophysics, computational physics, crystallography and computational chemistry. It is a popular language for high-performance computing[3] and is used for programs that benchmark and rank the world's fastest supercomputers.[4][5]

Fortran has had numerous versions, each of which has added extensions while largely retaining compatibility with preceding versions. Successive versions have added support for structured programming and processing of character-based data (FORTRAN 77), array programming, modular programming and generic programming (Fortran 90), High Performance Fortran (Fortran 95), object-oriented programming (Fortran 2003), concurrent programming (Fortran 2008), and native parallel computing capabilities (Coarray Fortran 2008/2018).

Fortran's design was the basis for many other programming languages. Among the better-known is BASIC, which is based on FORTRAN II with a number of syntax cleanups, notably better logical structures,[6] and other changes to work more easily in an interactive environment.[7]

Since August 2021 Fortran has ranked among the top 15 languages in the TIOBE index, a measure of the popularity of programming languages.[8]

Contents [hide]

Fortran

Paradigm	Multi-paradigm: structured, imperative (procedural, object-oriented), generic, array
Designed by	John Backus
Developer	John Backus and IBM
First appeared	1957; 65 years ago
Stable release	Fortran 2018 (ISO/IEC 1539-1:2018) / 28 November 2018; 3 years ago
Typing discipline	strong, static, manifest
Filename extensions	`.f`, `.for`, `.f90`
Website	fortran-lang.org
Major implementations	Absoft, Cray, GFortran, G95, IBM XL Fortran, Intel, Hitachi, Lahey/Fujitsu, Numerical Algorithms Group, Open Watcom, PathScale, PGI, Silverfrost, Oracle Solaris Studio, others
Influenced by	Speedcoding
Influenced	ALGOL 58, BASIC, C, Chapel,[1] CMS-2, DOPE, Fortress, PL/I, PACT I, MUMPS, IDL, Ratfor

Fortran

From Wikipedia, the free encyclopedia

Fortran (/ˈfɔːrtræn/; formerly **FORTRAN**) is a general-purpose, compiled imperative programming language that is especially suited to numeric computation and scientific computing.

Fortran was originally developed by IBM[2] in the 1950s for scientific and engineering applications, and subsequently came to dominate scientific computing. It has been in use for over six decades in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, geophysics, computational physics, crystallography and computational chemistry. It is a popular language for high-performance computing[3] and is used for programs that benchmark and rank the world's fastest supercomputers.[4][5]

Fortran has had numerous versions, each of which has added extensions while largely retaining compatibility with preceding versions. Successive versions have added support for structured programming and processing of character-based data (FORTRAN 77), array programming, modular programming and generic programming (Fortran 90), High Performance Fortran (Fortran 95), object-oriented programming (Fortran 2003), concurrent programming (Fortran 2008), and native parallel computing capabilities (Coarray Fortran 2008/2018).

Fortran's design was the basis for many other programming languages. Among the better-known is BASIC, which is based on FORTRAN II with a number of syntax cleanups, notably better logical structures,[6] and other changes to work more easily in an interactive environment.[7]

Since August 2021 Fortran has ranked among the top 15 languages in the TIOBE index, a measure of the popularity of programming languages.[8]

Contents [hide]

<table class="infobox vevent">

Fortran

Paradigm	Multi-paradigm: structured, imperative (procedural, object-oriented), generic, array
Designed by	John Backus
Developer	John Backus and IBM
First appeared	1957; 65 years ago
Stable release	Fortran 2018 (ISO/IEC 1539-1:2018) / 28 November 2018; 3 years ago
Typing discipline	strong, static, manifest
Filename extensions	`.f`, `.for`, `.f90`
Website	fortran-lang.org
Major implementations	
Absoft, Cray, GFortran, G95, IBM XL Fortran, Intel, Hitachi, Lahey/Fujitsu, Numerical Algorithms Group, Open Watcom, PathScale, PGI, Silverfrost, Oracle Solaris Studio, others	
Influenced by	
Speedcoding	
Influenced	
ALGOL 58, BASIC, C, Chapel,[1] CMS-2, DOPE, Fortress, PL/I, PACT I, MUMPS, IDL, Ratfor	

</table>

<table class="infobox vevent">

Fortran

Paradigm	Multi-paradigm: structured, imperative (procedural, object-oriented), generic, array
Designed by	John Backus
Developer	John Backus and IBM
First appeared	1957; 65 years ago
Stable release	Fortran 2018 (ISO/IEC 1539-1:2018) / 28 November 2018; 3 years ago
Typing discipline	strong, static, manifest
Filename extensions	`.f` , `.for` , `.f90`
Website	fortran-lang.org

Major implementations

Absoft, Cray, GFortran, G95, IBM XL Fortran, Intel, Hitachi, Lahey/Fujitsu, Numerical Algorithms Group, Open Watcom, PathScale, PGI, Silverfrost, Oracle Solaris Studio, others

Influenced by

Speedcoding

Influenced

ALGOL 58, BASIC, C, Chapel,[1] CMS-2, DOPE, Fortress, PL/I, PACT I, MUMPS, IDL, Ratfor

C++

Logo endorsed by the C++ standards committee

Paradigms	Multi-paradigm: procedural, imperative, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC 1 (Joint Technical Committee 1) / SC 22 (Subcommittee 22) / WG 21 (Working Group 21)
First appeared	1985; 37 years ago
Stable release	C++20 (ISO/IEC 14882:2020) / 15 December 2020; 21 months ago
Preview release	C++23 / 17 March 2022; 6 months ago
Typing discipline	Static, nominative, partially inferred
OS	Cross-platform
Filename extensions	.C, .cc, .cpp, .cxx, .c++, .h, .H, .hh, .hpp, .hxx, .h++
Website	isocpp.org

Major implementations

GCC, LLVM Clang, Microsoft Visual C++, Embarcadero C++Builder, Intel C++ Compiler, IBM XL C++, EDG

Influenced by

Ada, ALGOL 68,[1] BCPL,[2] C, CLU,[1] ML, Mesa,[1] Modula-2,[1] Simula, Smalltalk[1]

Influenced

Ada 95, C#,[3] C99, Chapel,[4] Clojure,[5] D, Java,[6] JS++,[7] Lua, Nim,[8] Objective-C++, Perl, PHP, Python,[9] Rust, Seed7

C++ Programming at Wikibooks

MATLAB (programming language)

Paradigm	multi-paradigm: functional, imperative, procedural, object-oriented, array
Designed by	Cleve Moler
Developer	MathWorks
First appeared	late 1970s
Stable release	R2022b[1] / September 15, 2022; 4 days ago
Typing discipline	dynamic, weak
Filename extensions	.m, .p,[2] .mex*,[3] .mat,[4] .fig,[5] .mlx,[6] .mlapp,[7] .mltbx,[8] .mlappinstall,[9] .mlpkginstall[10]
Website	mathworks.com

Influenced by

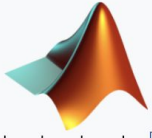
APL · EISPACK · LINPACK · PL/0 · Speakeasy[11]

Influenced

Julia[12] · Octave[13] · Scilab[14] · INTLAB[15][16][17][18]

MATLAB Programming at Wikibooks

MATLAB (software)



L-shaped membrane logo[19]

Developer(s)	MathWorks
Initial release	1984; 38 years ago
Stable release	R2022b[1] / September 15, 2022; 4 days ago
Written in	C/C++, MATLAB
Operating system	Windows, macOS, and Linux[20][21]
Platform	IA-32, x86-64
Type	Numerical computing
License	Proprietary commercial software
Website	mathworks.com

Python

Paradigm	Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	20 February 1991; 31 years ago[2]
Stable release	3.10.7[3] / 7 September 2022; 12 days ago
Preview release	3.11.0rc2[4] / 12 September 2022; 7 days ago
Typing discipline	Duck, dynamic, strong typing;[5] gradual (since 3.5, but ignored in CPython)[6]
OS	Windows, macOS, Linux/UNIX, Android[7][8] and more[9]
License	Python Software Foundation License
Filename extensions	.py, .pyi, .pyc, .pyd, .pyw, .pyz (since 3.5),[10] .pyo (prior to 3.5)[11]
Website	python.org

Major implementations

CPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, Jython

Dialects

Cython, RPython, Starlark[12]

Influenced by

ABC,[13] Ada,[14] ALGOL 68,[15] APL,[16] C,[17] C++,[18] CLU,[19] Dylan,[20] Haskell,[21][16] Icon,[22] Lisp,[23] Modula-3,[15][18] Perl,[24] Standard ML[16]

Influenced

Apache Groovy, Boo, Cobra, CoffeeScript,[25] D, F#, Genie,[26] Go, JavaScript,[27][28] Julia,[29] Nim, Ring,[30] Ruby,[31] Swift[32]

Python Programming at Wikibooks

julia

Paradigm	Multi-paradigm: multiple dispatch (primary paradigm), procedural, functional, meta, multistaged[1]
Designed by	Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah
Developer	Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and other contributors[2][3]
First appeared	2012; 10 years ago[4]
Stable release	1.8.1[5] / 6 September 2022; 13 days ago and 1.6.7 LTS[8][9] / 19 July 2022; 2 months ago
Preview release	Being worked on: 1.8.2[6] and 1.9.0-DEV with daily updates[7]
Typing discipline	Dynamic,[10] strong,[10] nominative, parametric, optional
Implementation language	Julia, C, C++, Scheme, LLVM[11]
Platform	Tier 1: x86-64, IA-32, CUDA 10.1+[12]/Nvidia GPUs (for Linux and Windows) Tier 2: 64-bit Arm (e.g. Apple M1 Macs, while they also have tier 1 support using Rosetta[13]), 32-bit Windows (64-bit is tier 1) Tier 3: 32-bit Arm, PowerPC, AMD (ROCm) GPUs. Also supports oneAPI/Intel's GPUs and Google's TPUs,[14] and has web browser support (for JavaScript and WebAssembly),[15] and can work in Android. For more details see "supported platforms".
OS	Linux, macOS, Windows and FreeBSD
License	MIT (core),[2] GPL v2;[16][17] a makefile option omits GPL libraries[18]
Filename extensions	.jl
Website	JuliaLang.org

Influenced by

C[4] · Dylan[19] · Lisp[4] · Lua[20] · Mathematica[4] (strictly its Wolfram Language[21][22]) · MATLAB[4] · Perl[20] · Python[20] · R[4] · Ruby[20] · Scheme[23]

data_wikigraph.json U

```
        ],
        "influenced": []
    },
    "Haskell": {
        "influenced_by": [
            "Clean_(programming_language)",
            "FP_(programming_language)",
            "Gofer_(programming_language)",
            "Hope_(programming_language)",
            "Id_(programming_language)",
            "ISWIM",
            "Kent_Recursive_Calculator",
            "Lisp_(programming_language)",
            "Miranda_(programming_language)",
            "ML_(programming_language)",
            "Standard_ML",
            "Orwell_(programming_language)",
            "SASL_(programming_language)",
            "Scheme_(programming_language)",
            "SISAL"
        ],
        "influenced": [
            "Agda_(programming_language)",
            "Bluespec,_Inc.",
            "C++11",
            "Concepts_(C++)",
            "C_Sharp_(programming_language)",
            "Language_Integrated_Query",
            "Wikipedia:Citation_needed",
            "Cayenne_(programming_language)",
            "Clean_(programming_language)",
            "Clojure",
            "CoffeeScript",
            "Curry_(programming_language)",
            "Elm_(programming_language)",
            "Epigram_(programming_language)",
            "Wikipedia:Citation_needed",
            "Escher_(programming_language)",
            "F_Sharp_(programming_language)",
            "Hack_(programming_language)",
            "Idris_(programming_language)",
            "Isabelle_theorem_prover",
            "Java_(programming_language)",
            "Generics_in_Java",
            "LiveScript",
            "Mercury_(programming_language)",
            "\u03a9mega",
            "PureScript",
            "Python_(programming_language)",
            "Raku_(programming_language)",
            "Rust_(programming_language)",
            "Scala_(programming_language)",
            "Swift_(programming_language)",
            "Visual_Basic_.NET"
        ]
    },
```

wikigraph_json.py U

```python
# by Matias I. Bofarull Oddo - 2022.09.26

import json
import requests
from bs4 import BeautifulSoup

rest_API = "https://en.wikipedia.org/api/rest_v1/page/html/"

dict_wikigraph = {}


def wikiscrape_infobox(page_href):

    try:

        if page_href not in dict_wikigraph:
            dict_wikigraph[page_href] = {}

        sesh = requests.Session()
        page = sesh.get(rest_API + page_href, timeout=10)
        soup = BeautifulSoup(page.content, "html.parser")
        infobox_rows = [row.prettify() for row in soup.find_all("tr")]
        section = 0
        row_index = 0
        data_strings = ["", ""]
        for row in infobox_rows:
            if "Influenced" in row:
                data_strings[section] += infobox_rows[row_index]
                data_strings[section] += infobox_rows[row_index + 1]
                section += 1
            row_index += 1

        data_influenced_by = BeautifulSoup(data_strings[0], "html.parser")
        list_influenced_by = [
            str(a["href"])[2:]
            for a in data_influenced_by.find_all(
                "a",
                {"rel": True},
            )
        ]
        data_influenced = BeautifulSoup(data_strings[1], "html.parser")
        list_influenced = [
            str(a["href"])[2:]
            for a in data_influenced.find_all(
                "a",
                {"rel": True},
            )
        ]

        list_href = sorted(set(list_influenced_by + list_influenced))

        if page_href in dict_wikigraph:
            dict_wikigraph[page_href]["influenced_by"] = list_influenced_by
            dict_wikigraph[page_href]["influenced"] = list_influenced
            print()
            print(page_href)
            print(list_influenced_by)
```

```
    ],
    "influenced": []
},
"Haskell": {
    "influenced_by": [
        "Clean_(programming_language)",
        "FP_(programming_language)",
        "Gofer_(programming_language)",
        "Hope_(programming_language)",
        "Id_(programming_language)",
        "ISWIM",
        "Kent_Recursive_Calculator",
        "Lisp_(programming_language)",
        "Miranda_(programming_language)",
        "ML_(programming_language)",
        "Standard_ML",
        "Orwell_(programming_language)",
        "SASL_(programming_language)",
        "Scheme_(programming_language)",
        "SISAL"
    ],
    "influenced": [
        "Agda_(programming_language)",
        "Bluespec,_Inc.",
        "C++11",
        "Concepts_(C++)",
        "C_Sharp_(programming_language)",
        "Language_Integrated_Query",
        "Wikipedia:Citation_needed",
        "Cayenne_(programming_language)",
        "Clean_(programming_language)",
        "Clojure",
        "CoffeeScript",
        "Curry_(programming_language)",
        "Elm_(programming_language)",
        "Epigram_(programming_language)",
        "Wikipedia:Citation_needed",
        "Escher_(programming_language)",
        "F_Sharp_(programming_language)",
        "Hack_(programming_language)",
        "Idris_(programming_language)",
        "Isabelle_theorem_prover",
        "Java_(programming_language)",
        "Generics_in_Java",
        "LiveScript",
        "Mercury_(programming_language)",
        "\u03a9mega",
        "PureScript",
        "Python_(programming_language)",
        "Raku_(programming_language)",
        "Rust_(programming_language)",
        "Scala_(programming_language)",
        "Swift_(programming_language)",
        "Visual_Basic_.NET"
    ]
},
```


Haskell

Haskell

Paradigm	Purely functional
Designed by	Lennart Augustsson, Dave Barton, Brian Boutel, Warren Burton, Joseph Fasel, Kevin Hammond, Ralf Hinze, Paul Hudak, John Hughes, Thomas Johnsson, Mark Jones, Simon Peyton Jones, John Launchbury, Erik Meijer, John Peterson, Alastair Reid, Colin Runciman, Philip Wadler
First appeared	1990; 32 years ago[1]
Stable release	Haskell 2010[2] / July 2010; 12 years ago
Preview release	Haskell 2020 announced[3]
Typing discipline	Inferred, static, strong
OS	Cross-platform
Filename extensions	.hs, .lhs
Website	www.haskell.org

Major implementations

GHC, Hugs, NHC, JHC, Yhc, UHC

Dialects

Gofer

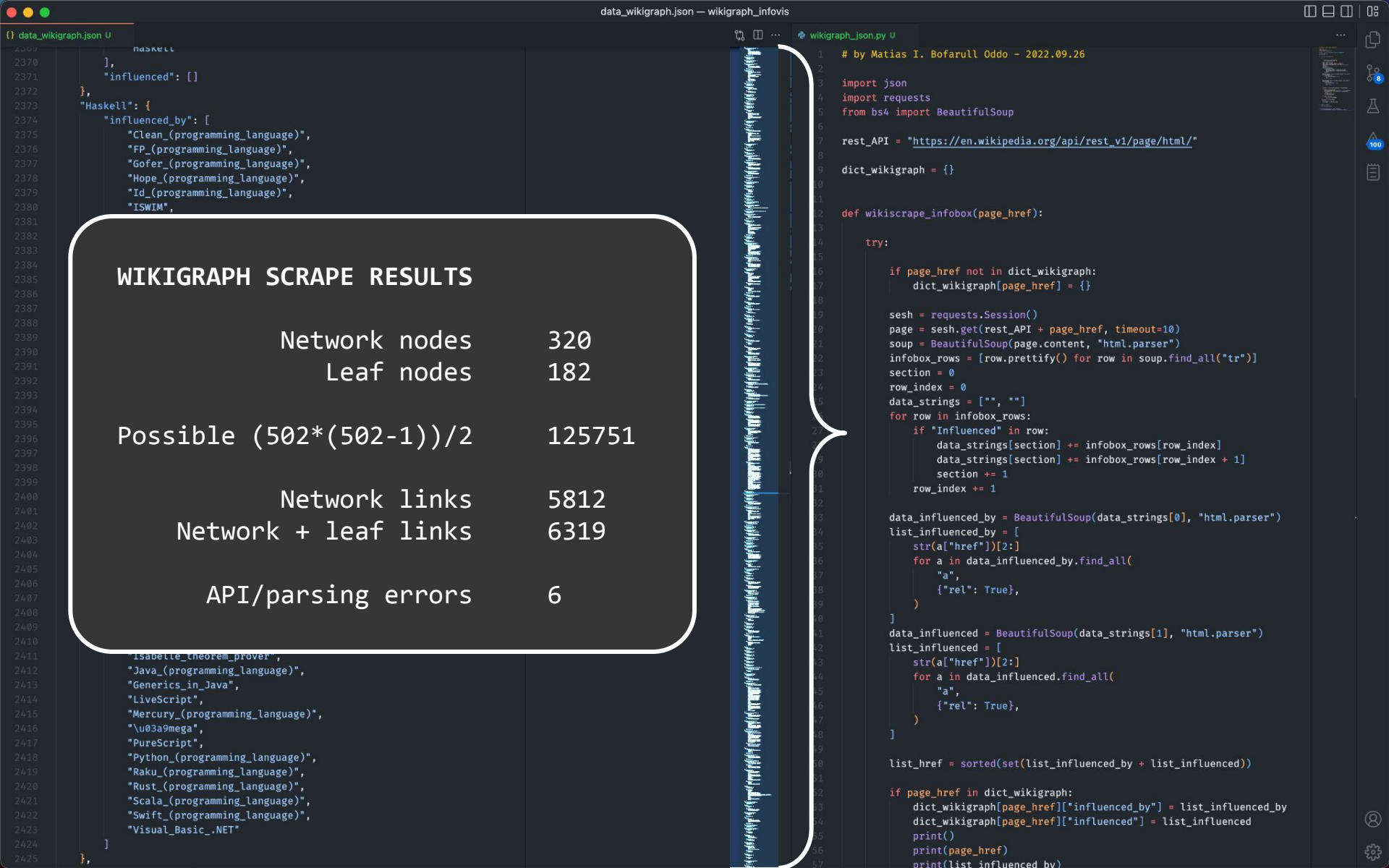
Influenced by

Clean,[4] FP,[4] Gofer,[4] Hope and Hope+,[4] Id,[4] ISWIM,[4] KRC,[4] Lisp,[4] Miranda,[4] ML and Standard ML,[4] Orwell, SASL,[4] Scheme,[4] SISAL[4]

Influenced

Agda,[5] Bluespec,[6] C++11/Concepts,[7] C#/LINQ,[8][9][10][11] CAL,[*citation needed*] Cayenne,[8] Clean,[8] Clojure,[12] CoffeeScript,[13] Curry,[8] Elm, Epigram,[*citation needed*] Escher,[14] F#,[15] Hack,[16] Idris,[17] Isabelle,[8] Java/Generics,[8] LiveScript,[18] Mercury,[8] Ωmega, PureScript,[19] Python,[8][20] Raku,[21] Rust,[22] Scala,[8][23] Swift,[24] Visual Basic 9.0[8][9]

```python
# by Matias I. Bofarull Oddo - 2022.09.26

import json
import requests
from bs4 import BeautifulSoup

rest_API = "https://en.wikipedia.org/api/rest_v1/page/html/"

dict_wikigraph = {}


def wikiscrape_infobox(page_href):

    try:

        if page_href not in dict_wikigraph:
            dict_wikigraph[page_href] = {}

        sesh = requests.Session()
        page = sesh.get(rest_API + page_href, timeout=10)
        soup = BeautifulSoup(page.content, "html.parser")
        infobox_rows = [row.prettify() for row in soup.find_all("tr")]
        section = 0
        row_index = 0
        data_strings = ["", ""]
        for row in infobox_rows:
            if "Influenced" in row:
                data_strings[section] += infobox_rows[row_index]
                data_strings[section] += infobox_rows[row_index + 1]
                section += 1
            row_index += 1

        data_influenced_by = BeautifulSoup(data_strings[0], "html.parser")
        list_influenced_by = [
            str(a["href"])[2:]
            for a in data_influenced_by.find_all(
                "a",
                {"rel": True},
            )
        ]
        data_influenced = BeautifulSoup(data_strings[1], "html.parser")
        list_influenced = [
            str(a["href"])[2:]
            for a in data_influenced.find_all(
                "a",
                {"rel": True},
            )
        ]

        list_href = sorted(set(list_influenced_by + list_influenced))

        if page_href in dict_wikigraph:
            dict_wikigraph[page_href]["influenced_by"] = list_influenced_by
            dict_wikigraph[page_href]["influenced"] = list_influenced
            print()
            print(page_href)
            print(list_influenced_by)
```


data_wikigraph.json — wikigraph_infovis
data_wikigraph.json U
wikigraph_json.py U
"influenced": []
"Haskell": {
"influenced_by": [
"Clean_(programming_language)",
"FP_(programming_language)",
"Gofer_(programming_language)",
"Hope_(programming_language)",
"Id_(programming_language)",
"ISWIM",
"Isabelle_(theorem_prover)",
"Java_(programming_language)",
"Generics_in_Java",
"LiveScript",
"Mercury_(programming_language)",
"\u03a9mega",
"PureScript",
"Python_(programming_language)",
"Raku_(programming_language)",
"Rust_(programming_language)",
"Scala_(programming_language)",
"Swift_(programming_language)",
"Visual_Basic_.NET"
]
},
WIKIGRAPH SCRAPE RESULTS
Network nodes 320
Leaf nodes 182
Possible (502*(502-1))/2 125751
Network links 5812
Network + leaf links 6319
API/parsing errors 6
# by Matias I. Bofarull Oddo - 2022.09.26
import json
import requests
from bs4 import BeautifulSoup
rest_API = "https://en.wikipedia.org/api/rest_v1/page/html/"
dict_wikigraph = {}
def wikiscrape_infobox(page_href):
try:
if page_href not in dict_wikigraph:
dict_wikigraph[page_href] = {}
sesh = requests.Session()
page = sesh.get(rest_API + page_href, timeout=10)
soup = BeautifulSoup(page.content, "html.parser")
infobox_rows = [row.prettify() for row in soup.find_all("tr")]
section = 0
row_index = 0
data_strings = ["", ""]
for row in infobox_rows:
if "Influenced" in row:
data_strings[section] += infobox_rows[row_index]
data_strings[section] += infobox_rows[row_index + 1]
section += 1
row_index += 1
data_influenced_by = BeautifulSoup(data_strings[0], "html.parser")
list_influenced_by = [
str(a["href"])[2:]
for a in data_influenced_by.find_all(
"a",
{"rel": True},
)
]
data_influenced = BeautifulSoup(data_strings[1], "html.parser")
list_influenced = [
str(a["href"])[2:]
for a in data_influenced.find_all(
"a",
{"rel": True},
)
]
list_href = sorted(set(list_influenced_by + list_influenced))
if page_href in dict_wikigraph:
dict_wikigraph[page_href]["influenced_by"] = list_influenced_by
dict_wikigraph[page_href]["influenced"] = list_influenced
print()
print(page_href)
print(list_influenced_by)

main 1 branch 0 tags

dirediredock finalized JSON export of scraped infobox data (#1) 040d42e 1 minute ago 13 commits

.gitignore	finalized JSON export of scraped infobox data (#1)	1 minute ago
README.md	Update README.md	7 hours ago
data_wikigraph.json	finalized JSON export of scraped infobox data (#1)	1 minute ago
wikigraph_API_scrape.py	finalized JSON export of scraped infobox data (#1)	1 minute ago
wikigraph_json.py	finalized JSON export of scraped infobox data (#1)	1 minute ago
wikigraph_list.py	finalized JSON export of scraped infobox data (#1)	1 minute ago

README.md

wikigraph_infovis

A network infographic of programming languages (PLs) - this project explores how to create an all-at-once visualization of PLs, how they influence each other, and how they develop over time.

The data comes from Wikipedia, specifically from PL pages (.en) that have an "infobox" HTML tag, which has href links that can be recursively Python scraped to construct an adjacency matrix, which in turn translates into a graph. In this network graph ach unqiue PL is a node, and relationships between nodes is determined by how PLs relate to one another. Supplementary data can include year of introduction, programming paradigm, and typing discipline.

Git Setup Cheatsheet

First download the repo through CLI and use `checkout -b` to create and name a new branch.

This project is about creating a network infographic of programming languages, a visualization of how languages influence each other and develop over time. Project for 2022 CPSC547 (Information Visualization) at UBC.

0 stars
1 watching
0 forks

Python 100.0%