UNIVERSITÉ DE GENÈVE

Introduction to Programming

March 4, 2021

Administrative Information

- Please enrol to the Nuvolos course webpage, if you have not done it yet. A link has been sent by email.
- Please regularly check the schedule folder for an up-to-date information on the next lecture/seminar. You will be notified by email on any change in schedule.

Outline

1. Programming in Finance
2. Excel
3. MATLAB
4. R
5. Implementation
    - Compiled
        - C/C++
    - Interpreted
        - Java
        - Python
    - Just-in-Time compilation
6. Summary
    - Relative performance
    - Learning curve
    - Purpose

Why learning programming in finance?

Increasing demand for IT skills in the financial industry mostly due to increasing complexity in the models and the quantity of data available.

- Fundamental Analyst: Use statistical analysis to make forecast.
  - Commodities: Supply and demand models.
  - Fixed Income - Currency (FIC): Macro-economic model are derived from statistical theory.
  - Equity: Financial modelling (Price-to-Earning, Dividend yield, ...)
- Trading: 60% of market volume of US trading comes from algorithms.
  - Execution: Most of the trading volume is now executed by algorithm (VWAP, TWAP, PVOL, ...).
  - Strategies: Fundamental (see above) and technicals (Trend-Following, Mean-Reverting, Statistical Arbitrage, ...) are implemented through programming languages.

Why learning programming in finance?

- Risk Analyst: Determine the aggregate risk and stress test of a portfolio using i.e. Extreme Value Theory and/or Copula.
- Derivative Analyst: Increasing complexity in derivatives contract makes Excel useless to price them.
- Strategist: Asset allocation done via quantitative rule (Risk-Parity, Black-Litterman, Constant-Weighting, ...) are automated.

Excel

- Microsoft Excel is the primary choice in the financial industry.
- Excel is used for data storage, data modeling, data computations, charts and graphs, etc.
- Visual Basic for Applications (VBA) is the programming language of Excel.
- VBA enables building customized functions, automating processes (so-called macros) and interacting with other programs.

Macros: An Example

1 On the Developer tab, click Insert.

2 In the ActiveX Controls group, click Command Button.

3 Right click CommandButton1 (make sure Design Mode is selected). Click View Code. The Visual Basic Editor appears.

4 Place your cursor between Private Sub CommandButton1_Click() and End Sub. Add the code line shown below:

```
Private Sub CommandButton1_Click()
Range("A1").Value = "Hello"
End Sub
```

5 Close the VB Editor. Click the command button on the sheet (make sure Design Mode is deselected).

- A common and easy way to generate VBA code is by using the Macro Recorder.

Threads

- Excel works with a single thread. This means that instructions are executed in a single sequence.
- Threads are a way for a program to divide split itself into two or more simultaneously running tasks.
- The opposite of single-threaded processes are multi-threaded processes. These processes allow the execution of multiple parts of a program at the same time.
- Multi-threaded processes enable maximum utilization of computer processors which leads to lower execution times.
- For example, calculating the price of a derivative with no closed-form solution by MC simulations can be tedious under a single thread.

Adding functionalities

- In order to increase available functions, one can call MATLAB and R functions inside the Excel environment (spreadsheet).
- This requires the Spreadsheet Link toolbox (for MATLAB) or the BERT toolkit (for R).
- With this approach, the user can access the powerful function libraries of MATLAB and R while still working in the excel environment.
- You also benefit from the the speed of MATLAB and R functions.
- For highly complex tasks and/or very large data sets, one has no choice but to work outside the Excel environment, i.e. using another programming language, and then write the results back in Excel.

MATLAB

- MATLAB is a programming language for engineers and scientists and is intended primarily for numerical computing.
- MATLAB features so-called toolboxes. Toolboxes provide a set of functions that solves a specific problems.
- Advantages: natural LA syntax, fast, easy to use, trusted functions, lots of good toolboxes.
- Disadvantages: Paying licence is needed from mathworks.com, proprietary (closed-source), expensive compiler and coder, dynamically-typed (see below).

Statically- and Dynamically-typed languages

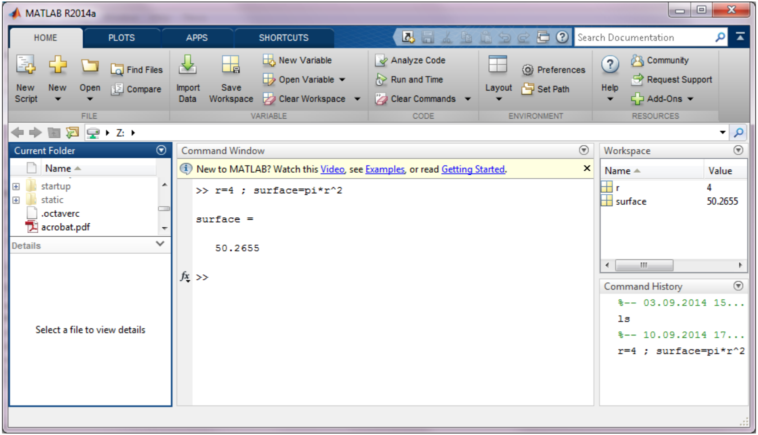
A language is **statically-typed** if the type of variables is known at the compile time.

- A lot of trivial errors may be caught at an early stage by the compiler
- A programmer must specify the type of each variable

A language is **dynamically-typed** if the type of variables is associated with run-time values.

- Possible errors due to misinterpreting the type of a variable
- A programmer does not need to specify the type of each variable

MATLAB Interface



startup
static
.octaverc
acrobat.pdf

Select a file to view details

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>> r=4 ; surface=pi*r^2

surface =

    50.2655

>>
```

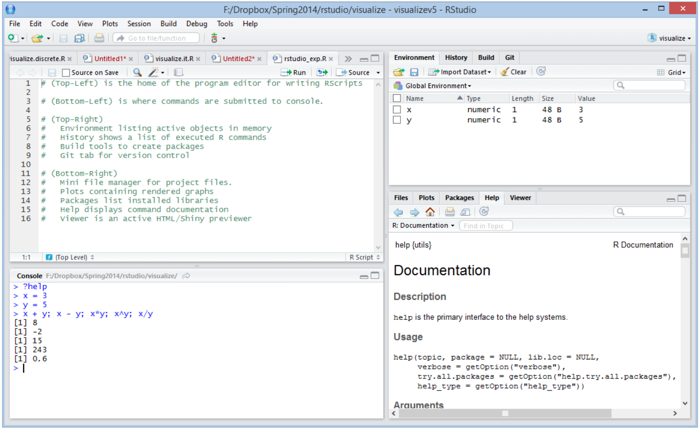
Name	Value
r	4
surface	50.2655

```
%-- 03.09.2014 15...
ls
%-- 10.09.2014 17...
r=4 ; surface=pi*r^2
```

R

- R is designed specifically for statistical computing and graphics.
- R features so-called packages. Packages provide functions, compiled code and sample data.
- Advantages: Open-source (freely available from r-project.org.), more than $7,000$ packages, extensive support on r-bloggers.com, easy to learn.
- Disadvantages: slower than other programming languages, difficult to read, packages/functions have to be checked, purely functional.

R Interface


F:/Dropbox/Spring2014/rstudio/visualize - visualizev5 - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
visualize
isualize.discrete.R
Untitled1*
visualize.it.R
Untitled2*
rstudio_exp.R
□ Source on Save
Run
Source
1:1
(Top Level)
R Script
Environment History Build Git
Import Dataset
Clear
Grid
Global Environment
Files Plots Packages Help Viewer
R: Documentation
Find in Topic
Console F:/Dropbox/Spring2014/rstudio/visualize/

```
# (Top-Left) is the home of the program editor for writing RScripts

# (Bottom-Left) is where commands are submitted to console.

# (Top-Right)
#   Environment listing active objects in memory
#   History shows a list of executed R commands
#   Build tools to create packages
#   Git tab for version control

# (Bottom-Right)
#   Mini file manager for project files.
#   Plots containing rendered graphs
#   Packages list installed libraries
#   Help displays command documentation
#   Viewer is an active HTML/Shiny previewer
```

```
> ?help
> x = 3
> y = 5
> x + y; x - y; x*y; x^y; x/y
[1] 8
[1] -2
[1] 15
[1] 243
[1] 0.6
>
```

□	Name	Type	Length	Size	Value
□	x	numeric	1	48 B	3
□	y	numeric	1	48 B	5

help {utils} R Documentation

Documentation

Description

`help` is the primary interface to the help systems.

Usage

```
help(topic, package = NULL, lib.loc = NULL,
     verbose = getOption("verbose"),
     try.all.packages = getOption("help.try.all.packages"),
     help_type = getOption("help_type"))
```

Arguments

Prototypes

- MATLAB and R are often used by quant teams to create prototypes, e.g. a first version of a new trading algorithm.
- The goal when creating a prototype is to develop a solution fast and to be able to implement/test it at low costs.
- Prototyping happens mostly in hedge funds and in quant trading groups within banks.
- Assuming everything went well during the initial stages, prototypes are then translated into faster languages (such as C++) by quant developers in order to be implemented across the firm.
- This is why a good understanding of lower level languages is also important in the quant industry.

Other applications

So far we have considered programming tools needed mainly for finance (risk analyst or portfolio manager).

Let us now consider situations when we value

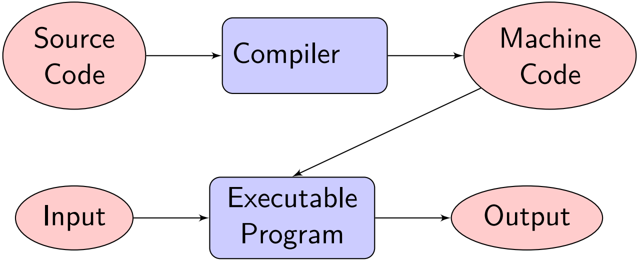
- Speed of execution: high-frequency trader
    - C++: very efficient execution
- Advanced tools: data scientist
    - Python: offers good libraries for machine learning

Why does C++ have high performance?

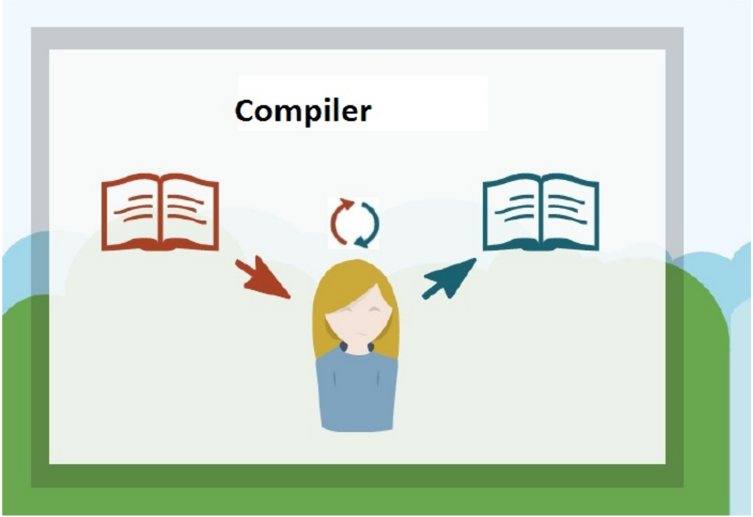
Compiled languages

Parsing and execution of a source code occur in two distinct steps:

1. The source code is translated (by a compiler) into the machine code of a computer (once)
2. The machine code is then executed (it may be executed several times, no need to compile each time)



Compiled languages



Compiled languages

Advantages:

- Fast execution

Disadvantages:

- Less flexibility
- Debugging is less explicit

Debugging

Debugging is the process of identifying and removing errors from a computer program.

Errors:

- Static
- Dynamic

Tools:

- Breakpoints
- Line by line execution
- Exception handling mechanisms

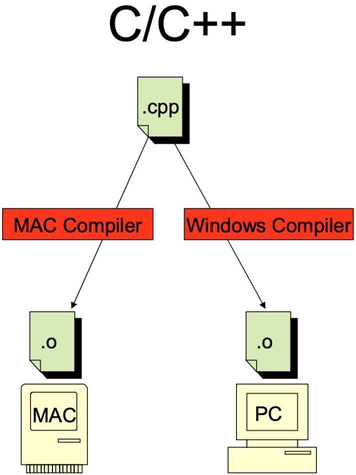
Compiled languages: C/C++

- Very efficient execution
- Procedural (C) or fully object-oriented (C++)

But,

- Hard to learn
- No garbage collection (manual memory management)
- Little errors checking built-in

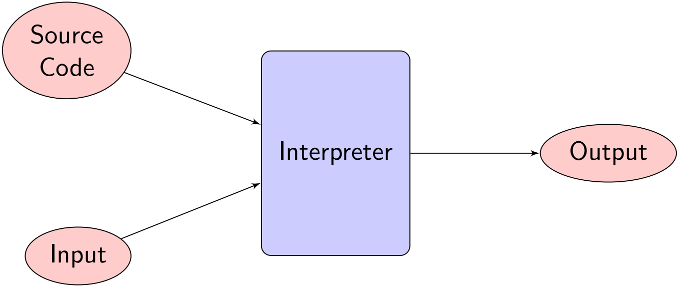
C/C++ compiler



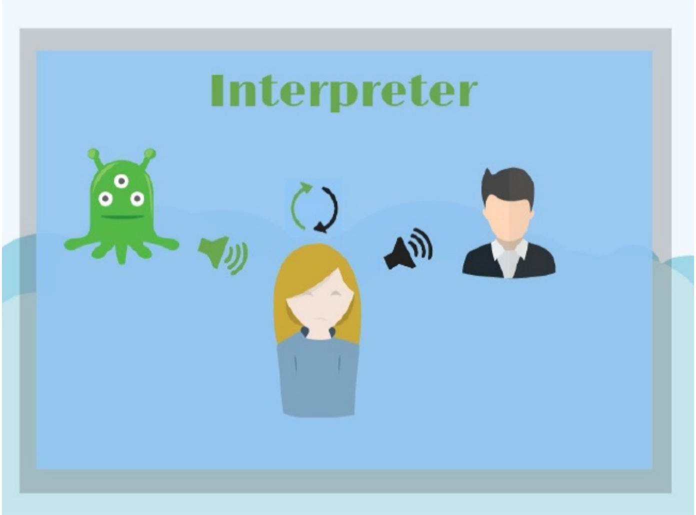
Interpreted languages

An **interpreter** is a program that translates a high-level language into a low-level one (some efficient intermediate representation) and immediately execute this.

- A source code is translated and executed on the fly (line by line)
- The source code has to be interpreted at each execution
- A program can be launched on any environment without a need to recompile (given a previously installed interpreter)



Interpreted languages



Interpreted languages

Advantages:

- Flexibility
- Advanced tools for debugging

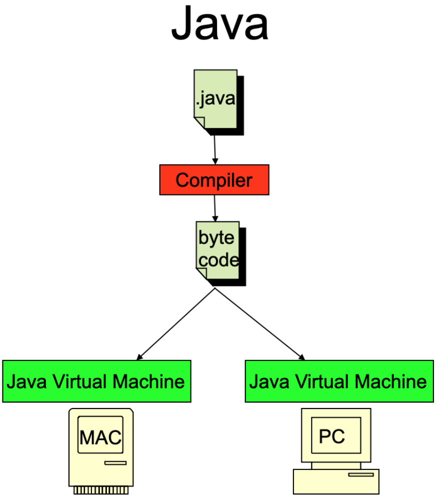
Disadvantages:

- Slower compared to compiled languages

Example: Java

- Syntax very similar to C/C++ (but simpler!)
- “Compile once, run anywhere”
- Automatic garbage collection

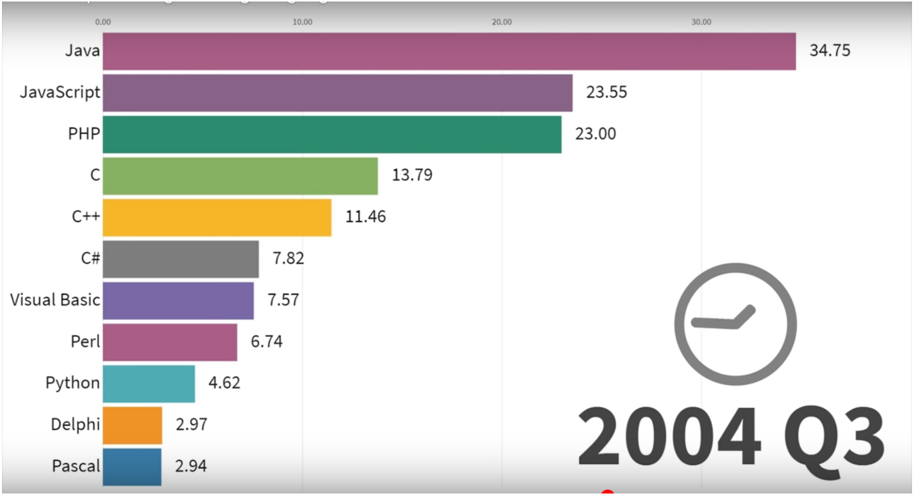
Java Virtual Machine (JVM)



Example: Python

- Open source (no license is needed)
- Both functional and object-oriented
- Increasing number of packages available (with continuous development and support)
- It becomes popular within the financial industry
- Great for big projects (since it supports OOP)

Most popular programming languages 2004-2019



Source: https://www.youtube.com/watch?v=Og847HVwRSI

```
#object-oriented way to implement sum of 2 numbers
#define a class
class BasicMaths:
    #init method
    def __init__(self, a, b):
        self.a=a
        self.b=b
    #define a method
    def sum(self):
        return self.a+self.b

#we need to instantiate an object
obj=BasicMaths(3,4)
#perform a method
obj.sum() #output is 7

#functional way to implement sum of 2 numbers
#define a function
def sum(a,b):
    return a+b
#call the function
sum(3,4) #output is 7
```

Example: Python

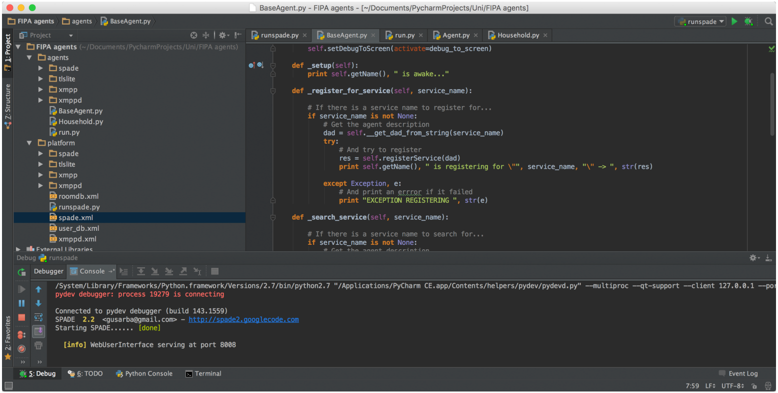
But,

- Quite difficult to learn (harder than R, but easier than C++)
- Be careful with libraries

```python
#import OLS from statsmodels
from statsmodels.api import OLS
#import linear_model from scikit-learn
from sklearn import linear_model

#y is a response vector (dependent variable)
#X is feature matrix (independent variables)

#Fit the model using scikit-learn
lm = linear_model.LinearRegression(X,y)
model = lm.fit()

#Fit the model using statsmodels
#Note the difference in argument order
lm= OLS(y,X)
model = lm.fit()
```

Environments for Python

- **PyCharm** is widely used for big scale projects (available on https://www.jetbrains.com/pycharm)
- **Spyder** is included in the Anaconda package
- **Jupyter**



```python
        self.setDebugToScreen(activate=debug_to_screen)

    def _setup(self):
        print self.getName(), " is awake..."

    def _register_for_service(self, service_name):

        # If there is a service name to register for...
        if service_name is not None:
            # Get the agent description
            dad = self.__get_dad_from_string(service_name)
            try:
                # And try to register
                res = self.registerService(dad)
                print self.getName(), " is registering for \"", service_name, "\" -> ", str(res)

            except Exception, e:
                # And print an errror if it failed
                print "EXCEPTION REGISTERING ", str(e)

    def _search_service(self, service_name):

        # If there is a service name to search for...
        if service_name is not None:
```

```
/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7 "/Applications/PyCharm CE.app/Contents/helpers/pydev/pydevd.py" --multiproc --qt-support --client 127.0.0.1 --por
pydev debugger: process 19279 is connecting

Connected to pydev debugger (build 143.1559)
SPADE  2.2  <gusarba@gmail.com> - http://spade2.googlecode.com
Starting SPADE...... [done]

  [info] WebUserInterface serving at port 8008
```

Just-in-time compiled languages

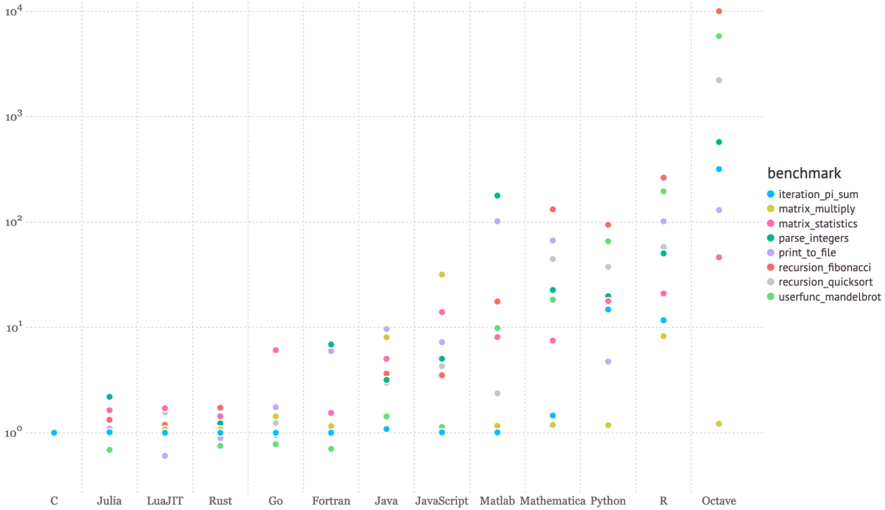
It is a combination of the two abovementioned traditional approaches

1. A high-level language is first compiled to a bytecode (low level language)
2. A dynamic compilation (not just an interpretation) of the bytecode (in part) to the machine code

The goal is to reach performance of static compilation while maintaining the advantages of bytecode interpretation.

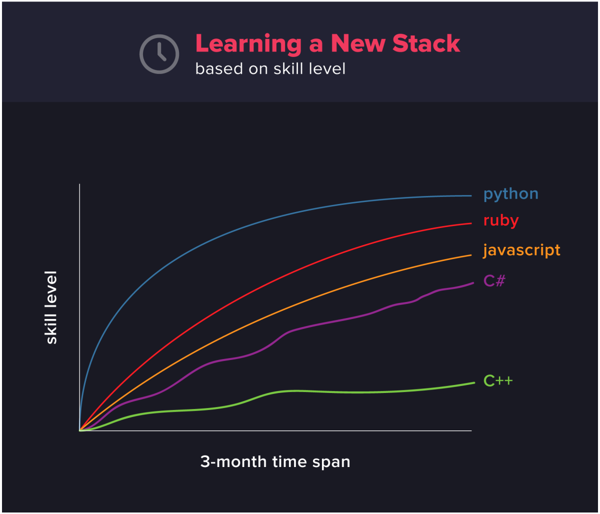
Examples: Java, Julia.

Relative performance



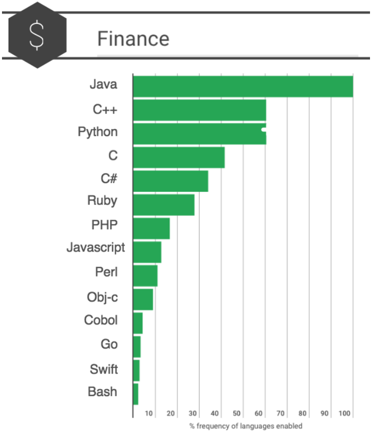
Source: https://julialang.org/benchmarks/

Learning curve



Source: codingdojo.com

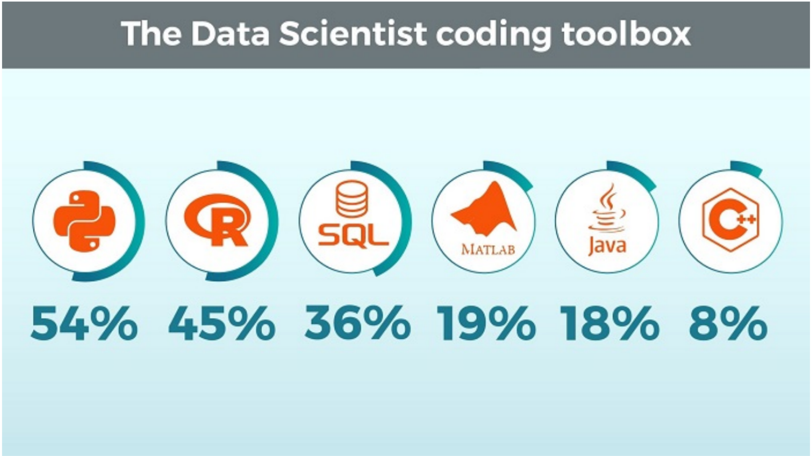
Purpose



1,000+ coding challenges across 20+ financial services companies

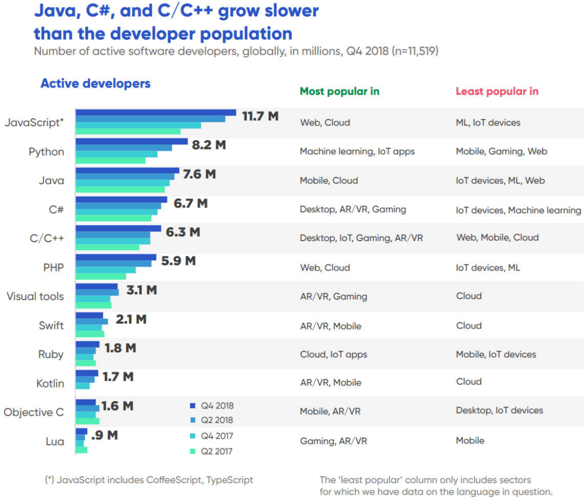
Source: https://blog.hackerrank.com/emerging-languages-still-overshadowed-by-incumbents-java-python-in-coding-interviews/

Purpose



Source: https://www.kdnuggets.com/2019/03/typical-data-scientist-2019.html

Purpose



Java, C#, and C/C++ grow slower than the developer population

Number of active software developers, globally, in millions, Q4 2018 (n=11,519)

	Active developers	Most popular in	Least popular in
JavaScript*	11.7 M	Web, Cloud	ML, IoT devices
Python	8.2 M	Machine learning, IoT apps	Mobile, Gaming, Web
Java	7.6 M	Mobile, Cloud	IoT devices, ML, Web
C#	6.7 M	Desktop, AR/VR, Gaming	IoT devices, Machine learning
C/C++	6.3 M	Desktop, IoT, Gaming, AR/VR	Web, Mobile, Cloud
PHP	5.9 M	Web, Cloud	IoT devices, ML
Visual tools	3.1 M	AR/VR, Gaming	Cloud
Swift	2.1 M	AR/VR, Mobile	Cloud
Ruby	1.8 M	Cloud, IoT apps	Mobile, IoT devices
Kotlin	1.7 M	AR/VR, Mobile	Cloud
Objective C	1.6 M	Mobile, AR/VR	Desktop, IoT devices
Lua	.9 M	Gaming, AR/VR	Mobile

(*) JavaScript includes CoffeeScript, TypeScript

The 'least popular' column only includes sectors for which we have data on the language in question.

Source: zdnet.com