Unit 2a: Classification of Programming Languages

1 Introduction

The first part of the module involves gaining an understanding of the features of programming languages which help you to make a choice of language for specific app development.

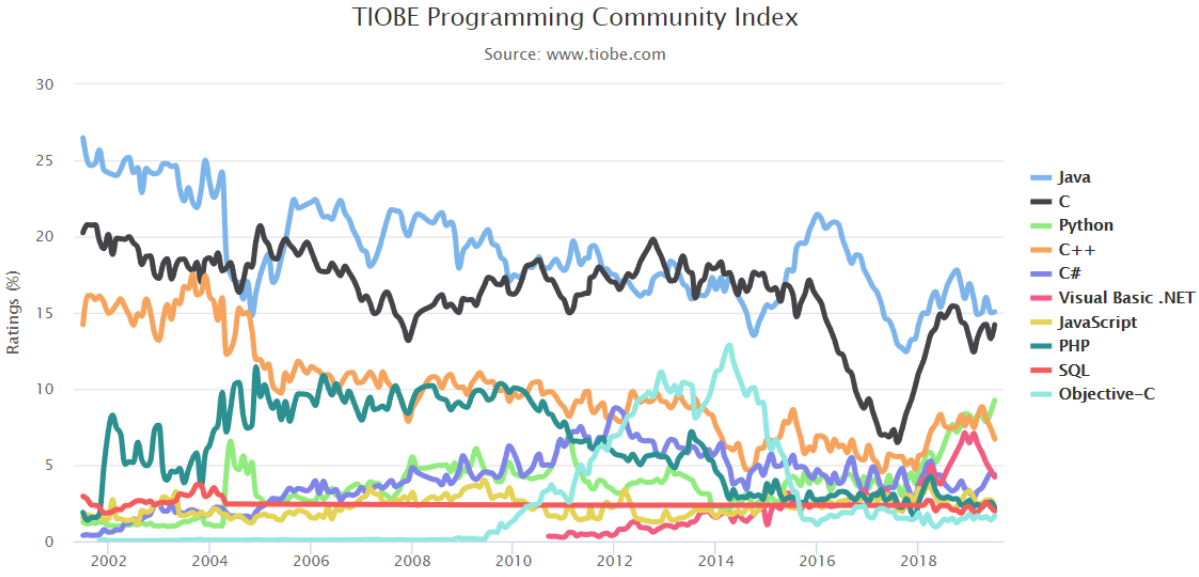
Programming languages are often developed to suit one particular programming approach (model/paradigm) though may be able to be used in other ways e.g. Java is essentially an Object Oriented language but has Functional support.

A software developer will, typically, learn and develop with a number of programming languages so it is important to learn how to learn a new programming language. The TIOBE index measures usage of programming languages worldwide.

1.1 Popular programming languages

Jul 2019	Jul 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.058%	-1.08%
2	2		C	14.211%	-0.45%
3	4	↑	Python	9.260%	+2.90%
4	3	↓	C++	6.705%	-0.91%
5	6	↑	C#	4.365%	+0.57%
6	5	↓	Visual Basic .NET	4.208%	-0.04%
7	8	↑	JavaScript	2.304%	-0.53%
8	7	↓	PHP	2.167%	-0.67%
9	9		SQL	1.977%	-0.36%
10	10		Objective-C	1.686%	+0.23%
11	12	↑	Ruby	1.636%	+0.43%
12	13	↑	Assembly language	1.390%	+0.24%
13	11	↓	Swift	1.121%	-0.29%
14	15	↑	MATLAB	1.078%	-0.05%
15	81	⇈	Groovy	1.070%	+0.96%
16	18	↑	Go	1.016%	+0.05%
17	19	↑	Visual Basic	1.009%	+0.12%

27	COBOL	0.434%
28	Scala	0.410%
29	Fortran	0.378%



January Headline: Python is TIOBE's programming language of the year 2018!

The Python programming language has won the title "programming language of the year"! Python has received this title because it has gained most ranking points in 2018 if compared to all other languages. The Python language has won 3.62%, followed by Visual Basic .NET and Java. Python has now definitely become part of the big programming languages. For almost 20 years, C, C++ and Java are consistently in the top 3, far ahead of the rest of the pack. Python is joining these 3 languages now. It is the most frequently taught first language at universities nowadays, it is number one in the statistical domain, number one in AI programming, number one in scripting and number one in writing system tests. Besides this, Python is also leading in web programming and scientific computing (just to name some other domains). In summary, Python is everywhere.

Other interesting positive moves of 2018 are MATLAB (#18 to #11), Kotlin (#39 to #31), Rust (#46 to #33), Julia (#47 to #37) and TypeScript (#167 to #49). The following languages had a hard time in 2018: Ruby (#11 to #18), Erlang (#23 to #50), F# (#40 to #64) and Alice (#26 to #66). Let's do one prediction for 2019: Kotlin will enter the top 20. We see a fast adoption in the industrial mobile app market of this language.

IMPORTANT NOTE. SQL has been added again to the TIOBE index since February 2018. The reason for this is that SQL appears to be Turing complete. As a consequence, there is no recent history for the language and thus it might seem that the SQL language is rising very fast. This is not the case.

1.2 Why are there so many languages?

Sometimes a language is developed to suit a new programming paradigm or application area; sometimes to address the issues surrounding existing languages. Often general-purpose languages can become quite bloated and be a victim of decisions made in the past which compromise the way in which new additions can be made.

Some languages suit a particular way of viewing business systems e.g. COBOL and file-based systems, and, time overtakes them as newer storage mechanisms become popular e.g. relational databases.

1.3 Purpose of programming languages

First of all we need to ensure that we understand the purpose of programming languages, particularly, high level languages i.e. languages with an English-like syntax. A programming language is a tool for humans to express ideas to computers; different programming paradigms express these ideas in very different ways - there are many choices of good programming languages, you can select the one that “works the way I think”. Sometimes thinking about the solution to a problem in a different way produces a 'better' solution e.g. thinking functionally rather than imperatively may be more appropriate.

1.4 Application Domain

Sometimes a particular application domain will require a particular type of programming language e.g. scripting languages for client-side web development. Sometimes a particular language is aimed at a specific domain, and restricted to that domain e.g. SQL and relational databases.

Most of the popular programming languages noted above are general-purpose languages and are widely used across a range of domains. The choice of a general purpose language rather than a specific application domain language might be based on trade-offs in experience, performance, developer productivity, security, robustness, etc.

1.5 People and communities

Often an organization will have a preference for one or a small number of programming languages based on legacy systems and the maintenance of these. The choice of a language may be based on what you know, or what your team knows.

Sometimes a good ecosystem (developer community and organisations) can make the individual developer more successful.

1.6 Why learn new languages?

Learning different languages allows you to be in a position to choose the right tool for the job.

> *"As I worked through the languages in this book, I often kicked myself, knowing that through the years, I've driven many a screw with a sledgehammer." Bruce Tate, in Seven Languages in Seven Weeks*

It is important that you learn *how to learn* new languages – you'll have to do it sometime as a developer.

> *"Learn at least one new language every year. Different languages solve the same problems in different ways. By learning several different approaches, you can help broaden your thinking and avoid getting stuck in a rut." Andrew Hunt, David Thomas, in The Pragmatic Programmer*

This allows you to become a better and more employable developer and be more able to form your own opinions on the advantages and disadvantages of specific languages and the characteristics of those languages – developers tend to have strong opinions and rarely agree with each other!

In this module we will consider the following languages i.e. learn enough about a few languages to be able to do something useful with them:

- **Java** (you know this one already)

- **Scala** www.scala-lang.org

  "Scala is an acronym for "Scalable Language". This means that Scala grows with you. You can play with it by typing one-line expressions and observing the results. But you can also rely on it for large mission critical systems, as many companies, including Twitter, LinkedIn, or Intel do. At the root, the language's scalability is the result of a careful integration of object-oriented and functional language concepts."

- **Groovy** http://www.groovy-lang.org

  "Groovy is a powerful, optionally typed and dynamic language aimed at improving developer productivity thanks to a concise, familiar and easy to learn syntax."

2 Classifying programming languages

There are a number of features of programming languages which dictate their appopriateness fo a particular app development.

2.1 Programming Model/Paradigm

The underlying programming model or paradigm is important: object oriented, functional, procedural, imperative, declarative, etc. This is to do with the way in we think about and develop a solution and dictates the basic structure of an app's code.

2.2 Typing Model

Typing involves specifying how the language works with data and the constraints involved. A number of typing models exist: static or dynamic; strong or weak; manifest or implicit; and, language-specific idiosyncrasies.

2.3 Execution Model

We need to think about how we interact with the language i.e. how is the code converted from the English-like syntax of a high level programming language into executable machine code: compiled, interpreted, interactive mode, VM.

2.4 Memory Management

The way in which a run-time environment manages the (limited) memory used by an app is a consideration.

2.5 Decision constructs and control structures

Most languages have the usual *if*, *while* and *for* statements with minor variations. A variety of other constructs exist, however, some specific to one language, some common to a number of languages, for example:

2.5.1 Pattern matching

A construct for choosing which variant of a function or expression is the correct one to apply; common in functional languages and fits in with their declarative nature.

2.5.2 Comprehensions

Allow procedures, typically involving transformations from one data structure to another, to be expressed in a declarative way rather than with traditional loops; common in, but not exclusive to, functional languages, e.g. also implemented in Python. Note *for* comprehensions in Scala are much more powerful than the usual *for* loop.

You will learn more about these later with examples in Scala.

2.6 Data structures

Data structures, or collections play a vital role in just about any language; they are defined in the language or provided as libraries or modules. Some are very well organised and integrated with the language, others less so (C#, for example). Typically you have arrays, lists, sets, tuples, and, dictionaries.

A Collections API defines which functions/methods can be called on a collection. Functional languages usually have rich APIs supporting powerful filtering and transformation capabilities.

2.7 Unique features

Some languages have features that make the language unique in some way or particularly useful in a certain type of scenario. This can be related to the whole philosophy of the language, or simply features that are just useful and/or unusual e.g.

- advanced features for concurrent or parallel programming (e.g. Scala, Erlang)
- advanced features for statistics and data analysis (e.g. R)
- source code indentation which is syntactically significant enforcing readable code (Python)
- close integration with JSON allowing easy data exchange (JavaScript)
- checked exceptions to enforce exception handling (Java)