Expanded Research Question

Advanced programming

Albert MAS Compés

2020

# Language 1: Python

I) Programming models supported

Python is a high-level programming language that was built to be easily learned and to allow programmers to choose the programming model that best suited a specific task. For that, Python supports multiple programming models: object-oriented and procedural, which are imperative; and functional, which is declarative.

* Imperative
  + Object-Oriented

In object-oriented programming, we find “objects” which are basic entities that can contain data and methods to modify this data or perform certain operations. We can use this paradigm in Python. That enables the programmer to create more structured and complex code.

Example of object-oriented programming in Python:



* + Procedural

In procedural programming, the code is structured in procedures or functions. This allows, among other things, to call any available function (including itself) at any time and to reuse code fragments. Each function has its own scope, which prevents from accessing variables outside the current scope (other functions or other instances of this same function).

Example of procedural programming in Python:



* Declarative
  + Functional

In functional programming, operations are made in a mathematical function style. Instead of statements, we use expressions that are evaluated and a value is returned.

II) Typing Model

Python is classified as a strongly typed dynamic language. Variables do not have to be declared, but the programmer can do so if they want to force a type.

In the following first two examples ‘x’ will be an integer with value 5, whereas in the third one, you force the variable to be a float.



In Python, method and variable names are bind during program execution. Variables can also be assigned values of a different type from the ones that were previously assigned.

The following code:



Will return this:



As we can see, we changed the type of the value assigned to the variable from an integer to a string.

Still, Python is strongly typed, which means that operations like adding an integer to a string are not allowed, as they are not well defined.

Correct code indentation is needed for the code to compile, this way Python ensures a correct readability of the code.

III) Application types

Some applications python is good for:

* Web Development

Python offers some web frameworks like Django and Flask, that can be very useful for server-side coding of web development.

* Scripting

Python is a good program to create simple scripts due to easy syntax and fast coding.

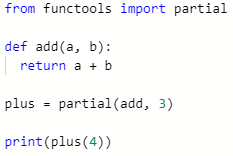
* Data Science

We can find several libraries and frameworks to develop programs related to Machine Learning (*scikit-learn* provides some popular machine learning algorithms and *TensorFlow* provides tools to build your own algorithms), Data Analysis/Visualisation (*Matplotlib*).

IV) Functional programming support

Even though Python is more oriented towards imperative programming, it also supports functional programming. Some of the functional aspects it provides are the filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. There are also the itertools and functools modules, implemented by the standard library, that offer more functional tools.

An example of this is the partial functions available with functools. A partially applied function is assigned to a variable:



Returns:



It also supports Lambda syntax and Decorators.

Python functional programming doesn’t go as far as fully functional languages and it won’t avoid all assignments and I/O. It will mostly show a functional appearance, while still using non-functional features internally.

V) Algebraic Data Types

Python includes the type Tuple, which is a collection two or more values of different type.



Returns:



We can also use Lists. More Algebraic Data Types are available importing certain libraries.

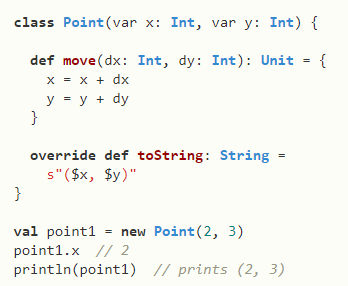
# Language 2: Scala

I) Programming models supported

Because Scala is strongly based on Java, it is a pure Object-Oriented language (every value is an object). It is also highly functional, as every function is a value. Also, it supports higher-order functions, nested functions and currying.

* Imperative
  + Object-oriented

Example of an object class in Scala, from the official Scala [documentation](https://docs.scala-lang.org/tour/classes.html):



* Declarative
  + Functional

Example of how to sum a list of floats using a functional approach in Scala:



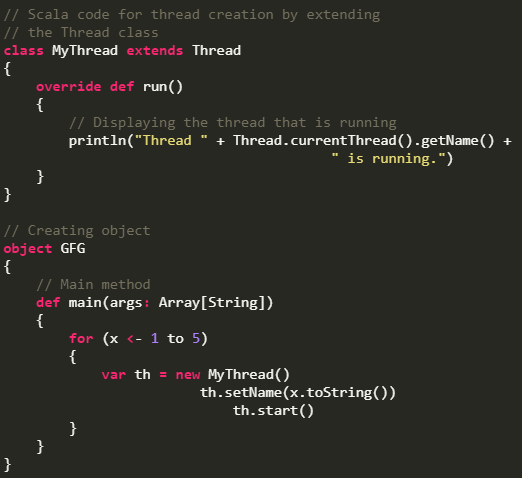
Example of a function assigned to a value of type Int:



* Concurrent

Concurrent computing allows to execute multiple calculations simultaneously instead of sequentially. Scala allows concurrent computation with the use of threads. Multithreading consists on creating multiple threads that will execute code concurrently, instead of always following a sequential path.

Example of thread creation extending the Thread class, from [geeksforgeeks.org](https://www.geeksforgeeks.org/) :

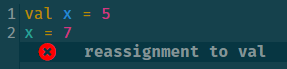


II) Typing model

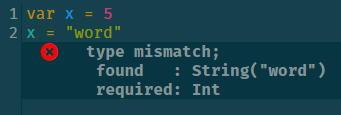
Scala is classified as a strongly typed static language. Variables have to be declared, but the type doesn’t have to be specified. In case you do want to specify it, the syntax goes as follows:



Variables can be declared as mutable with the key word *var* or immutable with *val*. Attempting to modify a *val* will trigger a compiler error:



After a value has been assigned to a variable, the data can be modified but its type cannot be changed.



III) Application types

Scala is a general-purpose programming language, which means that it can build almost all types of software. Still, there are areas where it will perform better than others.

Due to being a JVM language, Scala performs very well on android applications, similar to Java.

It can also be very useful for server-side web services.

Scala might not be the best choice for very demanding programs, as its performance cannot compete with other languages like C++.

IV) Functional programming support

Even though Scala provides all the OOP features from Java, it also provides lots of functional features that could allow programmers to code almost entirely using a functional style. Probably the most useful approach is to mix both styles and take advantage of the best features of each style.

Some of the functional features in Scala are:

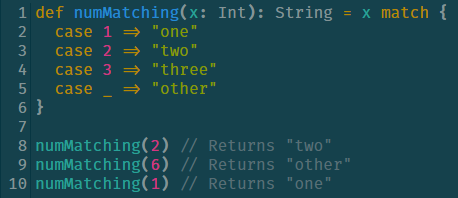
* Type inference
* Pattern matching
* Currying
* Immutable variables and objects
* Lazy evaluation
* Algebraic data types
* Tuples



* Higher-order and nested functions

V) Algebraic Data Types

Scala provides Algebraic Data Types functionality with its built-in pattern matching support.



Both Product and Sum types are implemented using classes in Scala, although Sum types require the use of subclasses.

We can use *for comprehensions* for formulating queries.

