Python

There are many reasons for the spread of Python, in particular:

 Ease of use. For those who are new to coding and programming, Python can be a great first step. It's relatively easy to learn, and it's a great way to start developing your programming knowledge.

 Simple Syntax.

Python is relatively easy to read and understand because its syntax is more like regular English

What is Python as a programming language

In general, Python can be described in one sentence:

A high-level, interpretable, object-oriented, imperative, strictly typed general-purpose language that has dynamic typing.

High-Level .

Programming languages are divided into high-level and low-level languages. Low-level languages are languages that are close to machine code or its constructs (such as bytecode). The classics of such languages are C, Assembler, Forth.

High-level languages are designed for usability and speed of program writing, respectively. They use certain abstractions - data structures, a set of auxiliary functions and so on. These languages such as Python, JS, PHP, Go,

Interpreted

Languages are divided into interpreted (Python, JS, PHP, R, Ruby) and compiled (C, C++, Pascal). In the first case, the program is executed by a special program - the interpreter, in the second case, the program is first converted into executable files understandable to the computer.

Object-oriented .

All languages are also divided into procedural, functional and object-oriented, depending on how the program is created and how it is executed.

In object-oriented languages, the basis is classes and class instances, which is equivalent to a type and an object of that type. The execution of conditional tasks or just the work of the program is based on the interaction of different classes.

Although Python is an object-oriented language, it also It also supports procedural programming, which means that a program can be written without a single class.

Functional languages are based on a different computational system called lambda-calculus, which is nevertheless equivalent to Turing machine, about which the corresponding theorem is proved

Imperative

Programming languages can also be divided into imperative and declarative languages. In an imperative language, the programmer will specify a sequence of commands to execute (these are all the languages

programming languages that we consider to be "programming languages" )

Declarative languages, on the other hand, expect us to describe the result we want to get during the execution of a query. A prime example of a declarative language is SQL (Stucted Query Language) or Structured Query Language. It's in this language we describe a specific result of the program execution rather than a sequence of commands. Other declarative languages are HTML, CSS, SVG, VRML, SQL, lex/VACC.

To understand the definitions of imperative and declarative language, let's set a problem: build a house.

Imperative: Go to the store -> Buy materials -> Prepare the site -> Build -> Move in.

Declarative: I want a new house.

Strictly (heavily) typified.

Regarding typification. In a strongly typed language, the interpreter, when executing commands, will not implicitly cast types, unlike in weakly typed languages, where type conversions can be done implicitly.

General Purpose

Languages are divided into general-purpose languages (Python, Java, Kotlin, C, Ruby) and specialized (for example, subject oriented languages or DSL)

DSL (subject oriented language) - a computer language, specialized for a particular area of application (as opposed to general-purpose language, which applies to a wide range of areas and does not take into account the peculiarities of specific knowledge areas). The construction of such a language and / or its data structure reflects the specifics of tasks that it solves. It is a key concept of language-oriented programming. Perl, SQL, HTML, Haskell, Verilog, AutoLisp are examples of such languages.

Dynamic typification

Dynamic typing implies that during the execution of a command, a variable can contain objects of different types. In other words, we declare a variable without explicitly specifying what type of data it will contain, and in the process of program execution one variable can contain both text and number, or even a Boolean value.

Static typing means that the type of data it can contain is specified explicitly.

The distinctive features of the Python language

Introspection .

In programming, this is the ability of a program to get different information about objects while the program is running. Specifically, you can find out what class an object belongs to, what data type you're working with at the moment (remember dynamic typing?), or the list of attributes and methods available for the object

Examples of useful introspection functions in Python:

dir();

type();

hasattr();

id();

isinstance().

Multi-platform

No, well, Java too, yes. But Python is not lagging behind, it comes in a box with Linux and SQL Server, the Python interpreter can be easily put on Windows, macOS, and iOS. It works everywhere, 8 times slower than Swift on iOS, but it works.

Built-in design patterns

Python has its own built-in design patterns, the best examples to understand are the Decorator and Iterator.

Decorator allows you to extend functionality without inheritance. To understand the Iterator pattern, just learn how iterators and generators work in the language (I wrote about them here).

Extensive set of standard libraries

When you use Core Python, you get a box of useful standard libraries. Examples include SQLite database library, but also json, math, re, random, csv, html and many others.

Readability of the language

Formatting text in Python is part of its syntax, and yes, it's those indentations (spaces, tabs whichever you like) that make code more readable.

What can a beginner do in Python?

Five simple Python project ideas

Random number generator

Guess the Number

Text generator

Text Quest

"Guess the word" ("Hangman").

Why developers use Python.

Python is one of the most popular general-purpose languages. That means you can write pretty much anything in it - from web applications to serious low-level system stuff. As it happens, I have a few students who don't really have much to do with programming but wanted to learn Python. For example, to automate some daily routine tasks, or to better understand the research that the analytics department in their company is doing. The threshold of entry is not very high, which allows you to achieve meaningful results in a fairly short time, and it's always nice to see the results of their work. Despite the fact that the language has existed for quite some time, it is actively developing, absorbing all the new features and flaunting a giant set of extension modules. In addition, quite a number of different courses, ranging from low-level networking to Data Science, have moved from other languages to Python in new versions. You can see this in the projects on Coursera, for example.

Even though Python isn't a clear leader in any specific areas, it's usually good enough to solve problems in almost any subject area. And on the horizon off the top of my head, I can't think of any other language that is that versatile.

Python is currently experiencing its peak popularity, a lot of companies are developing in it and hundreds of developers are going to conferences and mitups about the language. All this makes the Python language not only in demand, but also easy to learn. In this regard, the Python language is worth learning.

 In general, the prospects for the Python language are excellent, at the moment it is the second best language for any task. "Python is actively developing, as evidenced by the reduction in release time for new versions from eighteen to nine months.

That's about as long as it takes to learn the basic basics of Python. Although everyone is different and they need different time to learn the same skills. Most importantly, there are no uniform requirements for a Junieur in the marketplace. Because of this, the learning period can vary from

Conclusion

Python is a beautiful, concise, and versatile language. It is used in completely disparate fields, is able to create projects alone and does not look at the possibilities of competitive languages.

A common problem encountered among amateur developers is superficiality in learning Python. The tutorials you find on the Internet are paraphrased hundreds of times and hide important, useful features and features of programming languages.

It's important to fill in the blanks of your knowledge not only at the beginning of your training but throughout your career, otherwise tomorrow the phrase "make way for the young" will be relevant to you and you'll find yourself out of the loop.