What is Julia

A programming language created by Alan Edelman, Stefan Karpinski, Jeff Bezanson, and Viral Shah from MIT. It is optimized for scientific and numerical computing. Julia is free and open source software under MIT licensed.

Why Julia

There are several reasons for learning Julia as the first programming language for data science.

1. Relatively high performance
2. Dynamically typed language
3. High-level syntax and easy to learn

Installation

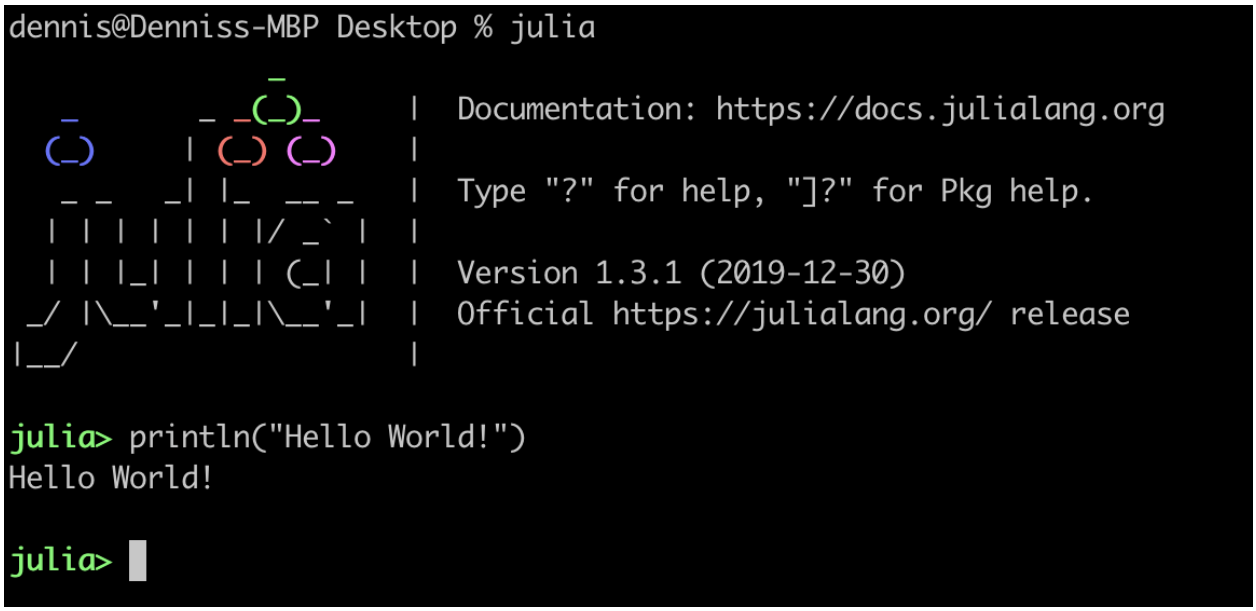
Julia installation is straightforward, whether using precompiled binaries or compiling from source. Download and install Julia by following the instructions at https://julialang.org/downloads/.

For windows users, you need to add the Julia to the PATH variable after installation.

1. Press the Windows and R key simultaneously. Enter “cmd”
2. Enter “ setx path "%path%;<your path>" “ (replace <your path> to your installation path) and run. The default path is C:\Users\<username>\AppData\Local\Julia-1.3.1\bin\

Julia interpreter

After installing Julia on your machine, open the command prompt / terminal, execute command “julia”(or double-clicking the Julia executable). This command starts the Julia interpreter, you can start to write some Julia code!



To exit the interactive session, type CTRL+D, or type exit().

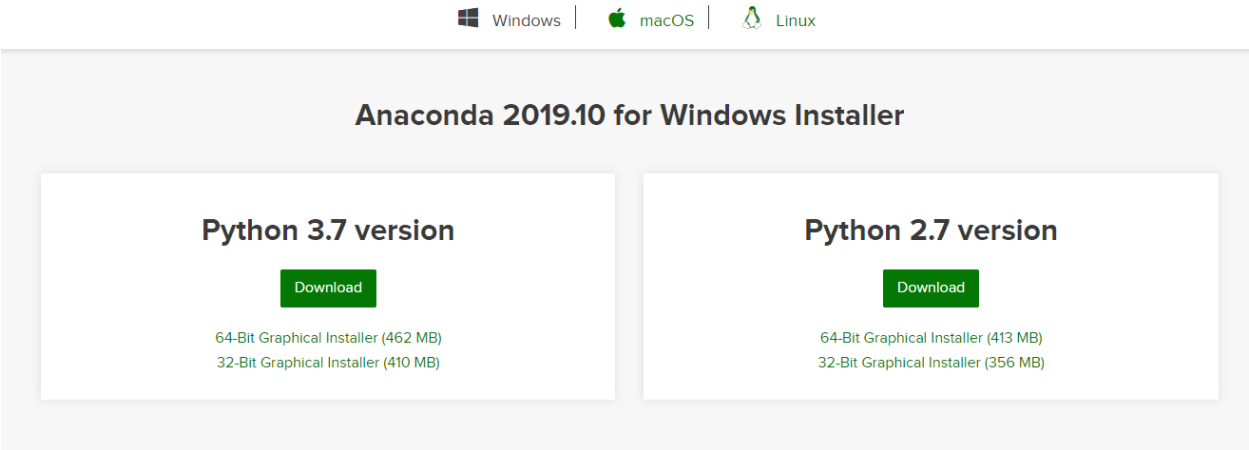
Install Jupyter Notebook

(Video:https://www.youtube.com/watch?v=oyx8M1yoboY)

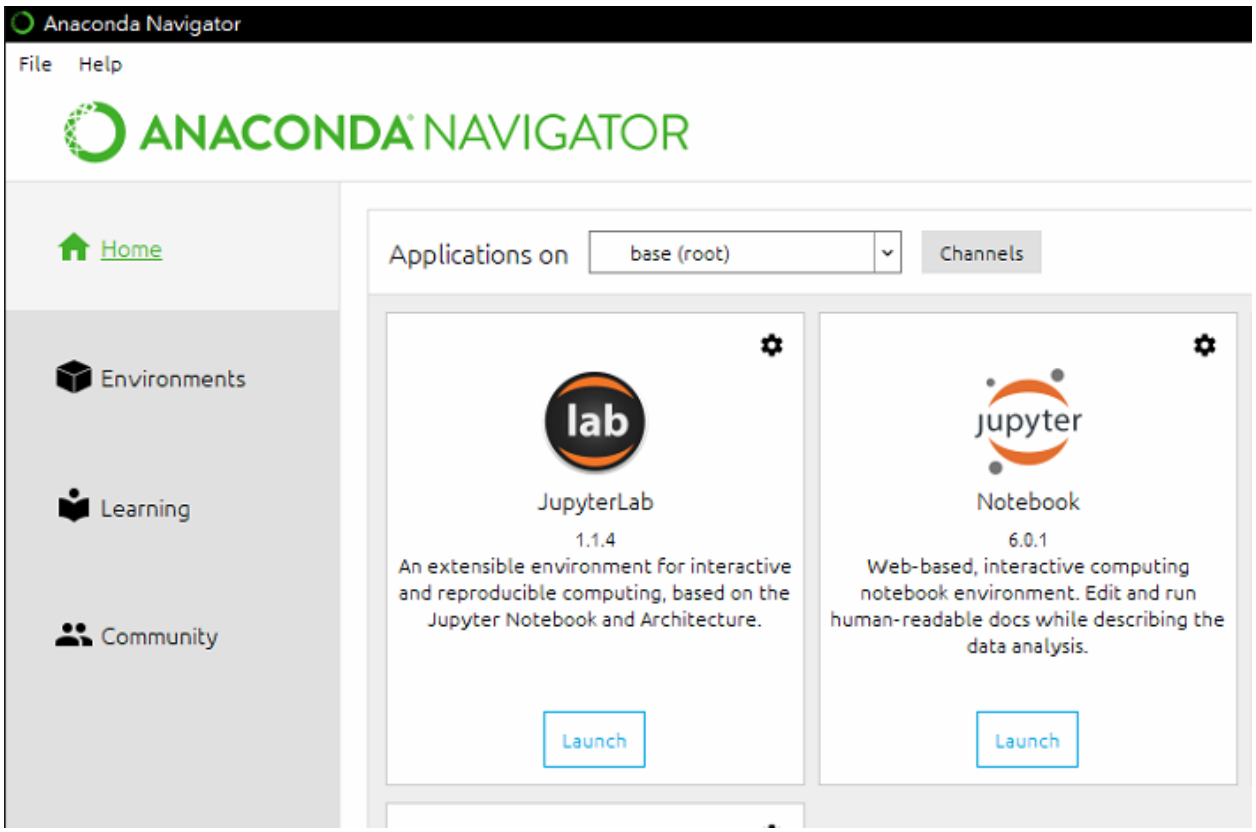
The following steps are based on the Windows 10 environment. For Mac users, please refer to the video.

Steps:

1. Download and Install Anaconda. Select Python 3.7 version 64-Bit Graphical Installer (URL: https://www.anaconda.com/distribution/)



2. Open Anaconda from your start menu. You should be able to see Jupyter Notebook is ready to launch

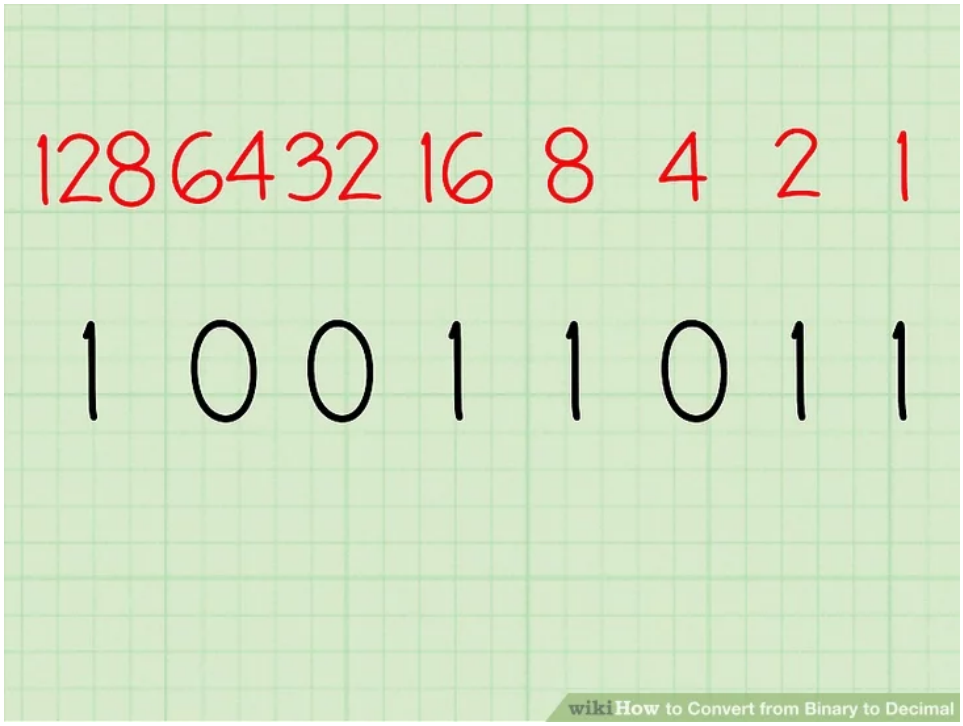


3. Open Julia and execute following command:
    a. using Pkg
    b. Pkg.add(“IJulia”)

4. Launch Jupyter Notebook. If you could see “Julia 1.3.1” when you are creating a new notebook, your environment is ready.

Mathematical Foundations for programming

Binary number system(Unsigned Integer)



Source: https://www.wikihow.com/Convert-from-Binary-to-Decimal

A Binary Number is made up of only 0s and 1s. "Bit" refers to a single binary digit. In computer, 1 byte = 8 bits.

✅ 1001_{bin} (4 bits) = 9_{dec}

✅ 1_{bin} (1 bits) = 1_{dec}

❌ 10123

Two's complement

Two's complement is a method for computers to represent an integer. With two's complement representation, the computer is able to perform calculation on integer numbers.

For a positive number, the two's complement representation of the number is its binary number.

Example: 9_{dec} = 0000 1001_{bin} = 0000 $1001_{Two's\ complement}$

For a negative number, you need to flip the binary number and add 1

Example: -9_{dec} = 0000 1001_{bin}

Step 1. Flip the number (it is called as one's complement)

0000 1001_{bin} -> 1111 $0110_{One's\ complement}$

Step 2. Add 1 to the One's complement

1111 $0110_{One's\ complement}$ -> $11111010_{Two's\ complement}$

Overflow Error

The integer usually takes 1 to 4 bytes to store in the world of computers and the range (i.e. maximum and minimum value) depends on the bytes used for storage.

Type	Range	Type	Range
Unsigned 8-bits Integer (Uint8)	0 to 255	Signed 8-bits Integer (Int8)	-128 to 127
Unsigned 16-bits Integer (Uint16)	0 to 65,535	Signed 16-bits Integer (Int8)	-32,768 to 32,767
Unsigned 32-bits Integer (Uint32)	0 to 4,294,967,295	Signed 32-bits Integer (Int8)	-2,147,483,648 to 2,147,483,647
Unsigned 64-bits Integer (Uint64)	0 to 18,446,744,073,709,551,615	Signed 64-bits Integer (Int8)	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Since the range is fixed, what if we are trying to create a number that is outside of the range? For example: 255 + 1 in unsigned 8-bits integer.

It is called (integer) overflow error. Overflow error occurs when an arithmetic operation attempts to create a number that is outside of the range, and leads to unintended behavior of the program. We should avoid and be cautious of the integer overflow error.

p.s. In some cases, we need to discard the carry bit(overflow). It does not necessarily lead to overflow error. For example in 8-bit integer: 2+(-1) =

		0	0	0	0	0	0	1	0
+	$_1$	1_1	1_1	1_1	1_1	1_1	1_1	1	1
	1	0	0	0	0	0	0	0	1

We will ignore the left-most bit. The result is 00000001, which is correct

Bitwise operation and Truth table

Bitwise operations are the operations that are based on the truth table. It will perform the operation on individual bits(bit-by-bit). For example:

5 AND 9

	0	1	0	1
AND	1	0	1	1
	0	0	0	1

The result of 5 AND 9 = 1

The following is the truth table of each common operation in bitwise operation

Bitwise AND:

Bit 1	Bit 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise OR:

Bit 1	Bit 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise XOR(exclusive or):

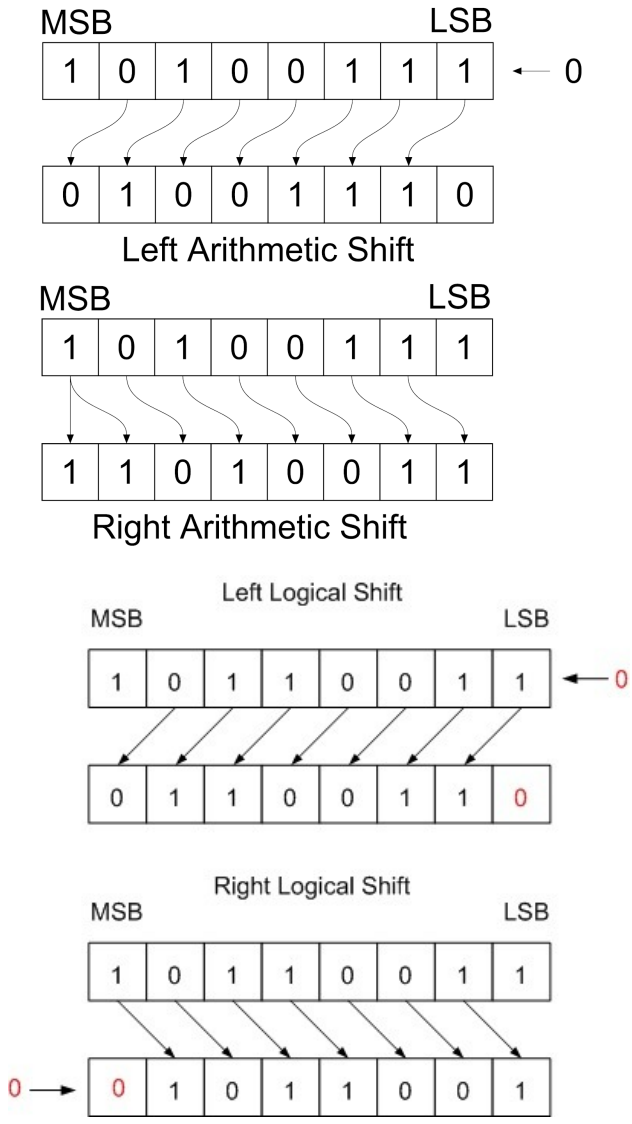
Bit 1	Bit 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise NOT:
Bitwise AND:

Bit 1	Output
0	1
1	0

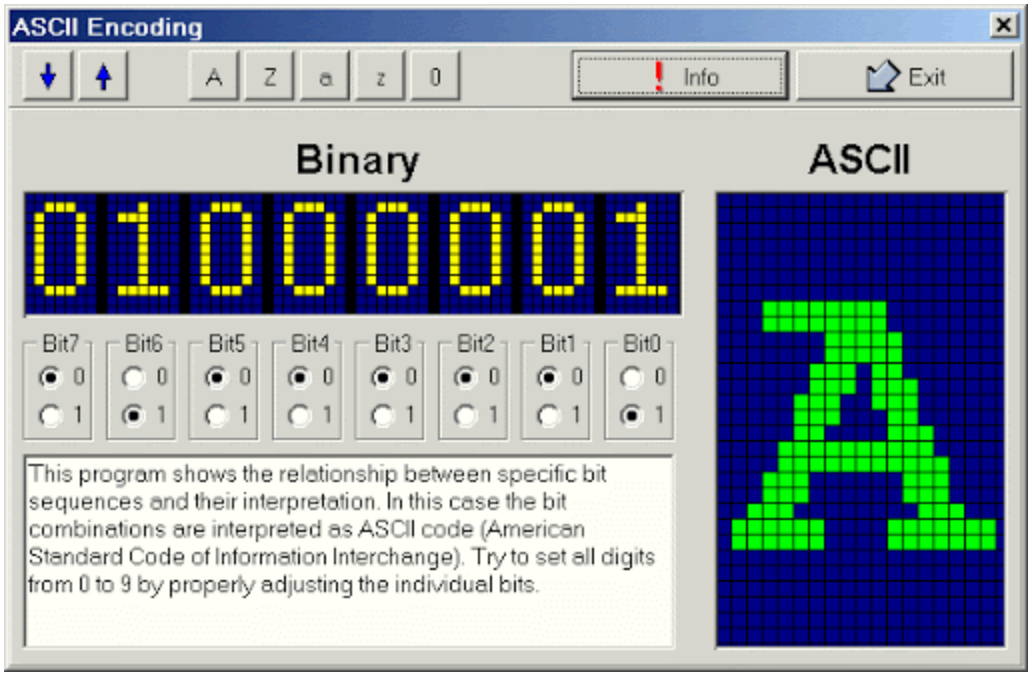
Logical and Arithmetic shift

Both the Logical and Arithmetic shift are shifting bits to the left or the right. The difference between them is arithmetic shift preserve sign bit, whereas Logical shift can not preserve sign bit in right shift.



Source: https://www.ahirlabs.com/difference/arithmetic-shift-and-logical-shift/

Represent a letter



Source: http://www.vias.org/simulations/simusoft_asciicode.html

In the view of the computer, each of the characters including english, chinese and many other languages are mapped to a value and it is called “encoding”. Except for those characters from the human language, there are some characters that are encoded such as “space”, “@”.

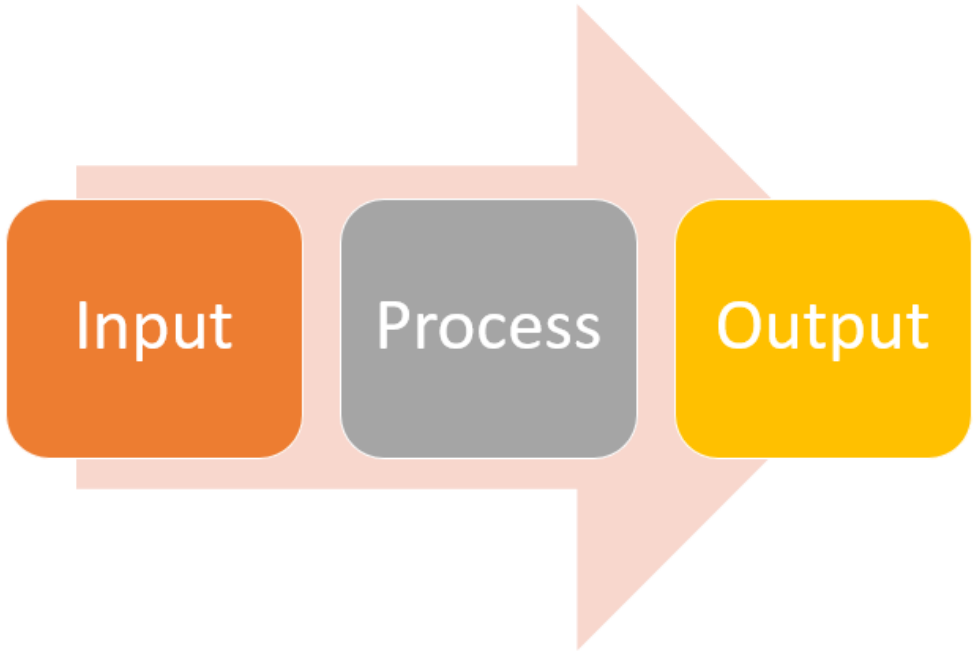
There are many encoding schemes for different languages as the standard for encoding characters. The most commonly used schemes are ASCII and Unicode.

American Standard Code for Information Interchange(ASCII) was first developed and published in 1963. The original purpose of ASCII was to encode characters in teleprinter. Therefore, there are some special characters (“control characters”) reserved.

ASCII scheme uses 7 bits to encode characters and there are 128 characters encoded in total. The range of control characters is 0-31 and these characters are not printable. The characters in ASCII scheme are case-sensitive: “A” to “Z” in ASCII are 65-90 and “a” to “z” are 97-122. Usually, you may do some “calculation” on character. For example, “a” + 1 = “b”.

Another common encoding scheme is Unicode. It is kind of an extension of ASCII. Unicode covers most of the characters in the world. The size of character in Unicode is variant. It uses 1-4 bytes to store a character.

IPO model, Algorithm and Implementation



In IPO model, there are three main phrases: Input, Process and Output. Computers are one of the examples of the IPO model.The computer (program) does some computations based on the inputs, and returns the results of the computations.

Algorithms are step-by-step methods for solving problems, which is the “process” phrase in the IPO model. A good algorithm should have the following features: execute time and store space efficient(time complexity and space complexity), easy to understand and implement and stable. Usually, we measure the complexity by the size of the input data. A fair algorithm should reach linear time(i.e. Linear relation between execute time and the input size).

Implementation is to describe the steps to convert from algorithm(theorcal) to executable program(practical). A good algorithm could be badly implemented and results in poor performance. Therefore, several rules of implementation are suggested such that we can guarantee the performance on problem solving.

Steps for problem-solving

Source: https://www.skipprichard.com/7-steps-to-problem-solving/

Problem solving is the process of identifying a problem generating ideas to solve the problem. Computers and programming are just tools for problem solving. We usually formalize the problem-solving process into 6 steps:

1. Problem Definition
2. Problem Analysis
3. Design of Algorithm
4. Implementation
5. Debugging and Testing
6. Documentation