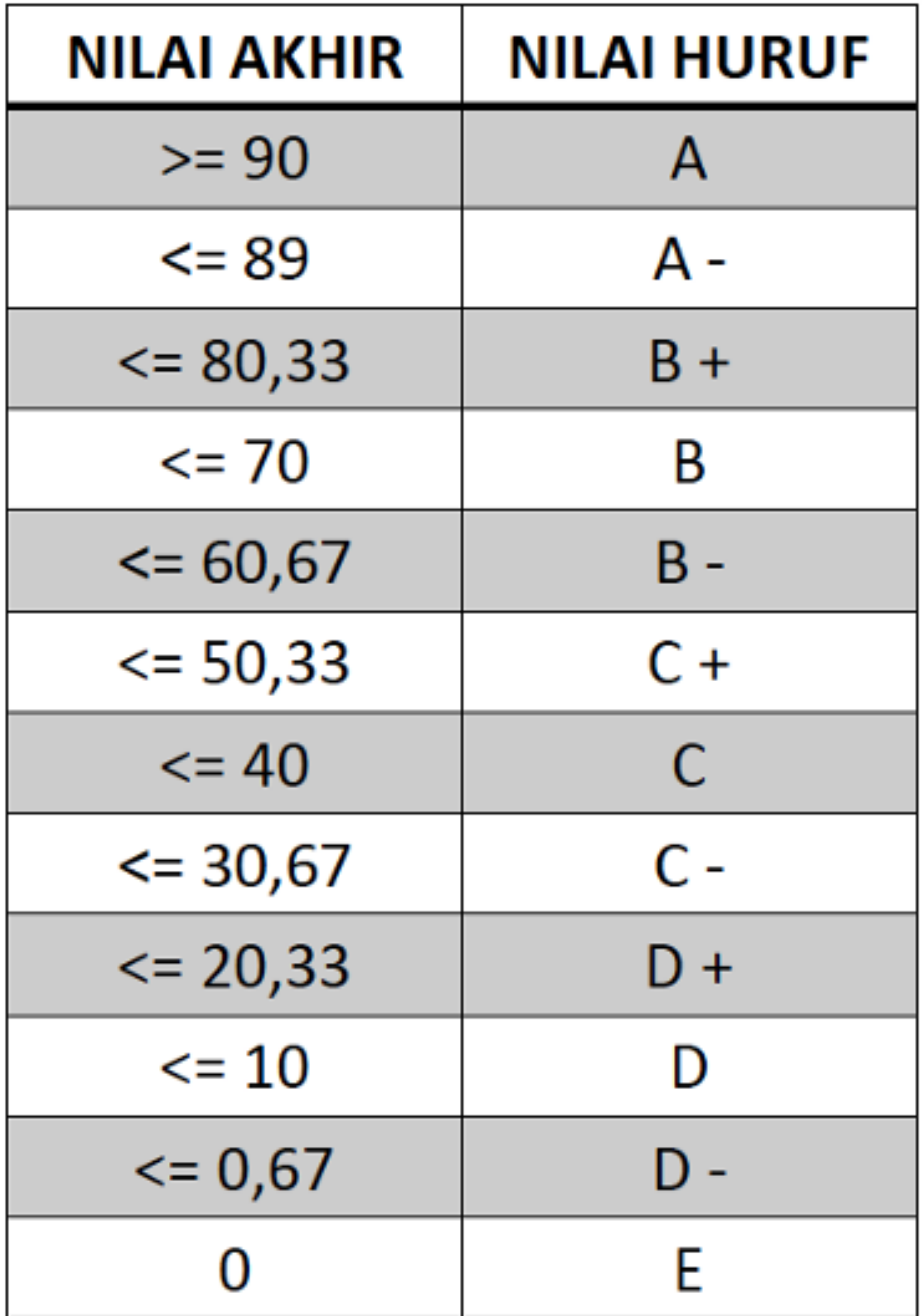
Klasifikasi Nilai



NILAI AKHIR	NILAI HURUF
>= 90	A
<= 89	A -
<= 80,33	B +
<= 70	B
<= 60,67	B -
<= 50,33	C +
<= 40	C
<= 30,67	C -
<= 20,33	D +
<= 10	D
<= 0,67	D -
0	E

BOBOT NILAI

Absen	10 %
Tugas	20 %
Kuis	15 %
UTS	25 %
UAS	30 %

Double-click or press Enter to edit text

$$\begin{tikzcd} \text{Computer} & \text{Program} & \text{useable} \\ & \text{calculate the speed} \arrow[u] & \end{tikzcd}$$

number of distance
number of time
divide both -> save memory
display in readable format

language that comp understain

language - tool for record, expres thought
our everyday language is mother tongue

comp mother tongue is machine language

need a bridge for
this diference word

human can write their program
computer use to execute prog
more compelx than macine language
bur simple that natural language

$$\begin{tikzcd} \text{compile, exe} & \text{hight level programing language} \arrow[l] \arrow[dl] \\ \text{interpreter} & \end{tikzcd}$$

program writen in this called
source code

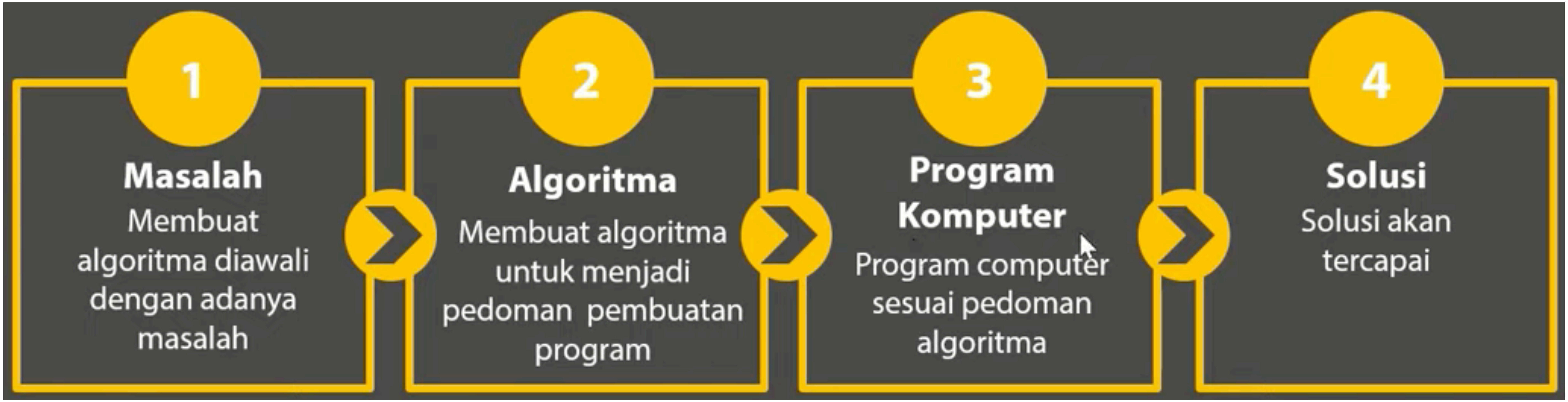
1.1 Algorithms as opposed to programs

An *algorithm* for a particular task can be defined as "a finite sequence of instructions, each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time". As such, an *algorithm* must be precise enough to be understood by *human beings*. However, in order to be *executed* by a *computer*, we will generally need a *program* that is written in a rigorous formal language; and since computers are quite inflexible compared to the human mind, programs usually need to contain more details than algorithms. Here we shall ignore most of those programming details and concentrate on the design of algorithms rather than programs.

The task of *implementing* the discussed algorithms as computer programs is important, of course, but these notes will concentrate on the theoretical aspects and leave the practical programming aspects to be studied elsewhere. Having said that, we will often find it useful to write down segments of actual programs in order to clarify and test certain theoretical aspects of algorithms and their data structures. It is also worth bearing in mind the distinction between different programming paradigms: *Imperative Programming* describes computation in terms of instructions that change the program/data state, whereas *Declarative Programming*

THE MAN
WHO INVENTED
ALGORITHMS
BBC

Park



1

Masalah

Membuat algoritma diawali dengan adanya masalah

2

Algoritma

Membuat algoritma untuk menjadi pedoman pembuatan program

3

Program Komputer

Program computer sesuai pedoman algoritma

4

Solusi

Solusi akan tercapai

BUATLAH NOTASI KALIMAT DESKRIPTIF PADA ALGORITMA

- CASE STUDY : BEBAS

1.3 Data structures, abstract data types, design patterns

For many problems, the ability to formulate an efficient algorithm depends on being able to organize the data in an appropriate manner. The term *data structure* is used to denote a particular way of organizing data for particular types of operation. These notes will look at numerous data structures ranging from familiar arrays and lists to more complex structures such as trees, heaps and graphs, and we will see how their choice affects the efficiency of the algorithms based upon them.

Often we want to talk about data structures without having to worry about all the implementational details associated with particular programming languages, or how the data is stored in computer memory. We can do this by formulating abstract mathematical models of particular classes of data structures or data types which have common features. These are called *abstract data types*, and are defined only by the operations that may be performed on them. Typically, we specify how they are built out of more *primitive data types* (e.g., integers or strings), how to extract that data from them, and some basic checks to control the flow of processing in algorithms. The idea that the implementational details are hidden from the user and protected from outside access is known as *encapsulation*. We shall see many examples of abstract data types throughout these notes.

Programming Language

An Introduction

Edy Haryanto, S.Kom., M.Eng.

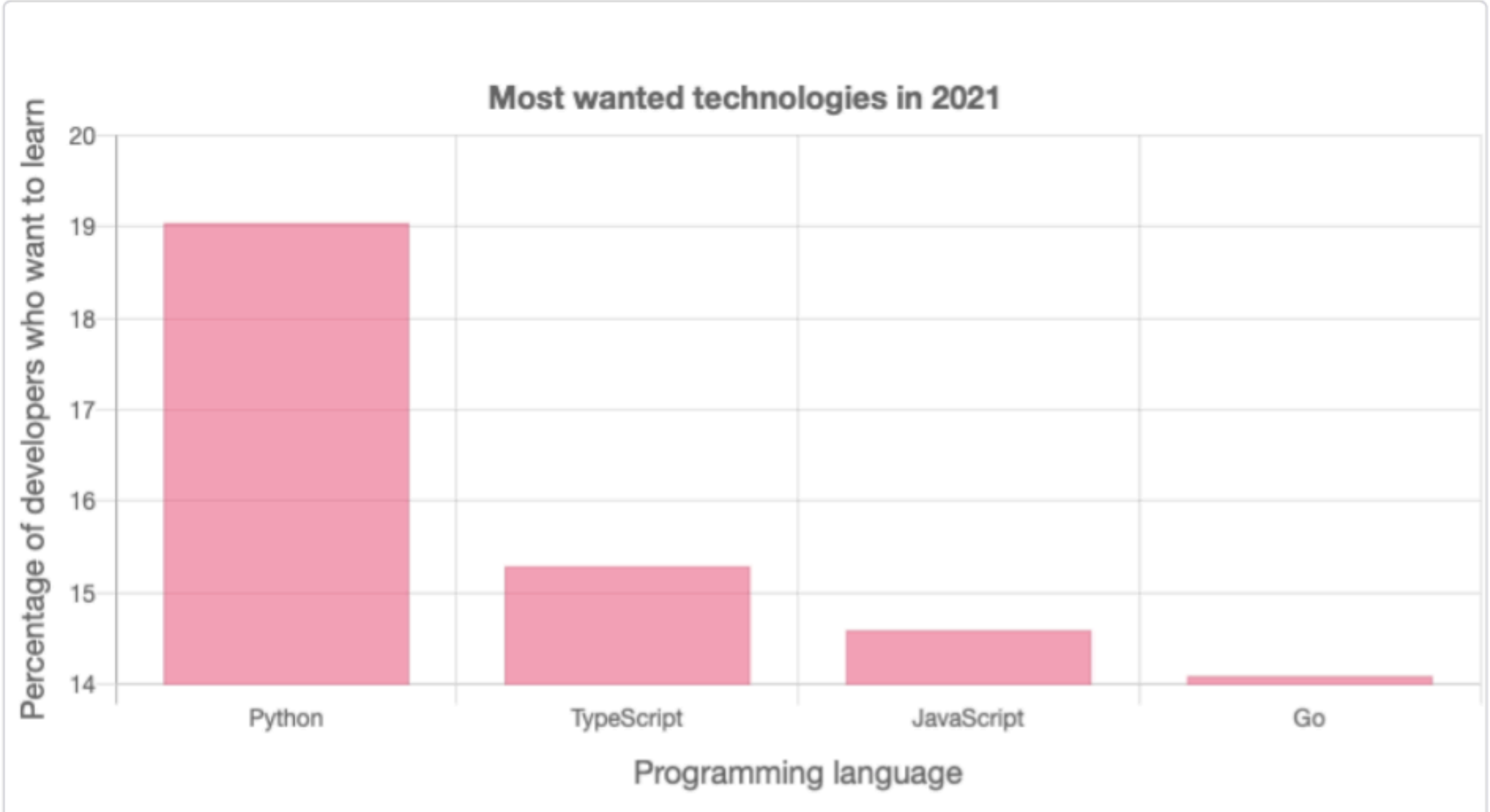
PYTHON

Python adalah bahasa pemrograman interpretatif yang multiplatform dan multiguna, *Python* termasuk di dalam bahasa pemrograman tingkat tinggi karya Guido van Rossum. *Python* sendiri telah banyak digunakan untuk membuat berbagai macam aplikasi dan program, seperti: Program GUI (desktop), Aplikasi Mobile, Web, IoT, Game, bahkan Program untuk Hacking.

Python was first released in 1991. Over 30 years later, Python remains one of the most popular programming languages among hobbyist and professional developers worldwide. TIOBE declared Python the **programming language of the year** in 2021. This was the third time Python won these honors in the last five years alone.

Large developer community

Python is one of the most popular programming languages in the world. In Stack Overflow's 2021 Developer Survey, 48% of respondents said they work with Python. When other respondents were asked which technology they had a desire to learn, Python ranked first as the most wanted technology among developers.



2. Extensive libraries

Python offers a wide range of libraries that can be used across various applications. Libraries are collections of resources that help us streamline application development. Instead of writing every piece of code from scratch, we can use libraries, which contain many pre-written functions and classes.

Some popular Python libraries include:

- **Numpy and SciPy**: For scientific computing, with a wide range of functions, algorithms, and optimizations, including linear algebra and statistics
- **Keras, Seaborn, TensorFlow, and SciKit-Learn**: For machine learning, artificial intelligence, natural language processing, and neural networks
- **Scrapy**: For data science, allows you to make an effective web crawler and data scraper
- **Pandas**: For data analysis, including data cleaning and manipulating both relational and labeled data
- **Matplotlib and Plotly**: For data visualization and plotting graphs

3. Write less, do more

Python has very concise syntax. This is noticeably true even when compared to other high-level programming languages, such as Java.

By comparing the "Hello World" program in Python to Java, we can see that **Python's syntax is much more concise**.

```java
class HelloWorld {
    public static void main( String args[] ) {
        System.out.println( "Hello World!" );
    }
}
```

"Hello World" in Java

3. Write less, do more

Python has very concise syntax. This is noticeably true even when compared to other high-level programming languages, such as Java.

By comparing the “Hello World” program in Python to Java, we can see that **Python’s syntax is much more concise**.

```
print "Hello World"
```

"Hello World" in Python

4. Portability

Portability is another one of Python’s strengths. Portability refers to an application’s ability to run across various operating systems (OS). Unless your program contains system-specific calls, **you can run your Python program across Windows, Mac OS, and Linux without modifying the program code**. All you have to do is to use the Python interpreter that’s appropriate for your chosen platform.

5. Wide range of use cases

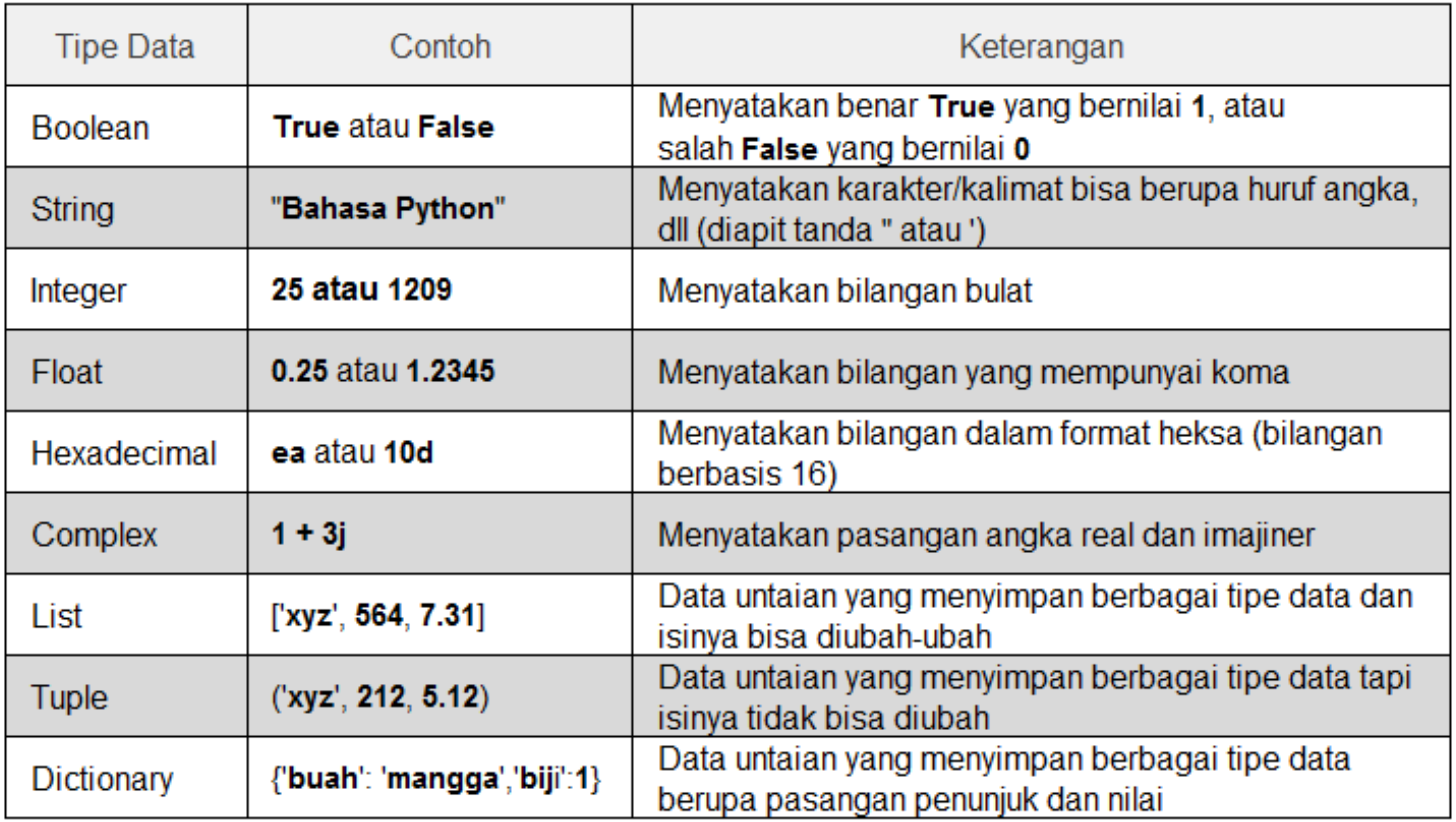
The Python programming language has various use cases in many growing fields, including:

- Data science
- Machine learning
- Statistics
- Cybersecurity
- Game development
- Back-end web application development
- Embedded applications

- **Fun community**: Python's community is built around the intention to make the language fun to use. In fact, its name isn't a reference to the serpent, but rather, to the British comedy group *Monty Python*.
- **Career opportunities**: Learning Python can help you get a developer job you love. Python developers are in demand in both big companies (such as Netflix) and smaller organizations and startups.
- **Industry use cases**: Python has applications across various industries. The language is particularly a strong choice if you're interested in areas such as machine learning, data science, game development, and back-end web development.
- **Learning curve**: Python has very human-like syntax compared to other programming languages. This can make the learning curve quite easy for beginners. As previously mentioned though, you may initially experience the opposite effect if you're used to other programming languages.
- **Flexibility of a multi-paradigm language**: Python supports multiple programming paradigms. You can use Python for functional programming, object-oriented programming, and procedural programming.
- **Resources and tools**: Because it's so widely used, there's an abundance of resources you can leverage to learn Python. There's also a wide selection of tools, frameworks, libraries, and Python IDEs to help streamline your development.

TYPE DATA PYTHON

Tipe data adalah suatu memori atau media pada komputer yang digunakan untuk menampung informasi atau data sementara. *Python* sendiri mempunyai tipe data yang cukup unik bila kita bandingkan dengan bahasa pemrograman yang lain.



Tipe Data	Contoh	Keterangan
Boolean	**True** atau **False**	Menyatakan benar **True** yang bernilai **1**, atau salah **False** yang bernilai **0**
String	"**Bahasa Python**"	Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')
Integer	**25 atau 1209**	Menyatakan bilangan bulat
Float	**0.25** atau **1.2345**	Menyatakan bilangan yang mempunyai koma
Hexadecimal	**ea** atau **10d**	Menyatakan bilangan dalam format heksa (bilangan berbasis 16)
Complex	**1 + 3j**	Menyatakan pasangan angka real dan imajiner
List	['**xyz**', **564**, **7.31**]	Data untaian yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah
Tuple	('**xyz**', **212**, **5.12**)	Data untaian yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah
Dictionary	{'**buah**': '**mangga**','**biji**':**1**}	Data untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai

Operator	**Contoh**	**Penjelasan**
Penjumlahan+	1 + 3 = 4	Menjumlahkan nilai dari masing-masing operan atau bilangan
Pengurangan -	4 - 1 = 3	Mengurangi nilai operan di sebelah kiri menggunakan operan di sebelah kanan
Perkalian *	2 * 4 = 8	Mengalikan operan/bilangan
Pembagian /	10 / 5 = 2	Untuk membagi operan di sebelah kiri menggunakan operan di sebelah kanan
Sisa Bagi%	11 % 2 = 1	Mendapatkan sisa pembagian dari operan di sebelah kiri operator ketika dibagi oleh operan di sebelah kanan
Pangkat **	8 ** 2 = 64	Memangkatkan operan disebelah kiri operator dengan operan di sebelah kanan operator
Pembagian Bulat //	10 // 3 = 3	Sama seperti pembagian. Hanya saja angka dibelakang koma dihilangkan

Operator	Contoh

Penjumlahan+	#Penjumlahan
	print(13 + 2)
	apel = 7
	jeruk = 9
	buah = apel + jeruk #
	print(buah)

1. Buatlah program yg menghitung jumlah sisa hutang dengan output sebagi berikut
Sisa hutang anda adalah 5000

1. Buatlah program yg menghitung luas persegi panjang dengan output sebagi berikut
Luas persegi panjang Adalah 50

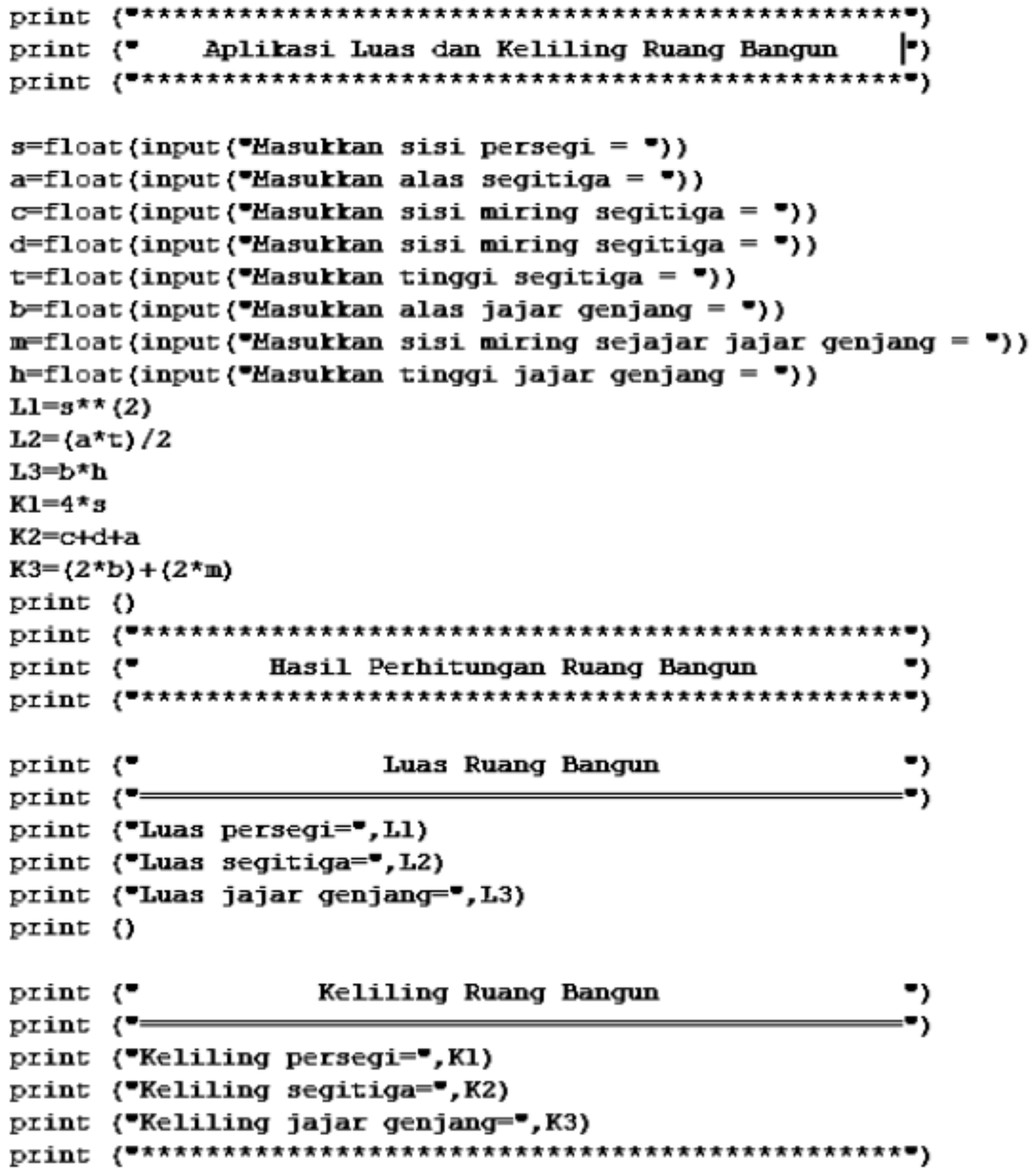
2. Buatlah program yg menghitung pembagian jumlah kue dan jumlah anak dengan output sebagi berikut
Tiap anak akan mendapatkan kue sebanyak 3

3. Buatlah program yg menghitung sisa bagi kue pada anak dengan output sebagi berikut
Setelah dibagi teryata kue tersisa sebanyak 3

4. Buatlah program yg menghitung perpangkatan dengan output sebagi berikut
Jika bilangan3 dipangkatkan dengan bilangan4 maka hasilnya adalah 64

5. Buatlah program yg menghitung pembulatan bilangan desimal dengan output sebagi berikut
Pembulatan dari hasil bagi bilangan5 dan blangan6 Adalah 3

Operator	Contoh	Penjelasan
and	a, b = True, True # hasil akan True print (a and b)	Jika kedua operan bernilai True, maka kondisi akan bernilai True. Selain kondisi tadi maka akan bernilai False.
or	a, b = True, False # hasil akan True print (a or b) print (b or a) print (a or a) # hasil akan False print (b or b)	Jika salah satu atau kedua operan bernilai True maka kondisi akan bernilai True. Jika keduanya False maka kondisi akan bernilai False.
not	a, b = True, False # hasil akan True print (not a) print (not b)	Membalikkan nilai kebeneran pada operan misal jika asalnya True akan menjadi False dan begitupun sebaliknya.



```
print ("************************************************")
print ("    Aplikasi Luas dan Keliling Ruang Bangun    ")
print ("************************************************")

s=float(input("Masukkan sisi persegi = "))
a=float(input("Masukkan alas segitiga = "))
c=float(input("Masukkan sisi miring segitiga = "))
d=float(input("Masukkan sisi miring segitiga = "))
t=float(input("Masukkan tinggi segitiga = "))
b=float(input("Masukkan alas jajar genjang = "))
m=float(input("Masukkan sisi miring sejajar jajar genjang = "))
h=float(input("Masukkan tinggi jajar genjang = "))
L1=s**(2)
L2=(a*t)/2
L3=b*h
K1=4*s
K2=c+d+a
K3=(2*b)+(2*m)
print ()
print ("************************************************")
print ("        Hasil Perhitungan Ruang Bangun          ")
print ("************************************************")

print ("               Luas Ruang Bangun               ")
print ("================================================")
print ("Luas persegi=",L1)
print ("Luas segitiga=",L2)
print ("Luas jajar genjang=",L3)
print ()

print ("           Keliling Ruang Bangun               ")
print ("================================================")
print ("Keliling persegi=",K1)
print ("Keliling segitiga=",K2)
print ("Keliling jajar genjang=",K3)
print ("************************************************")
```