

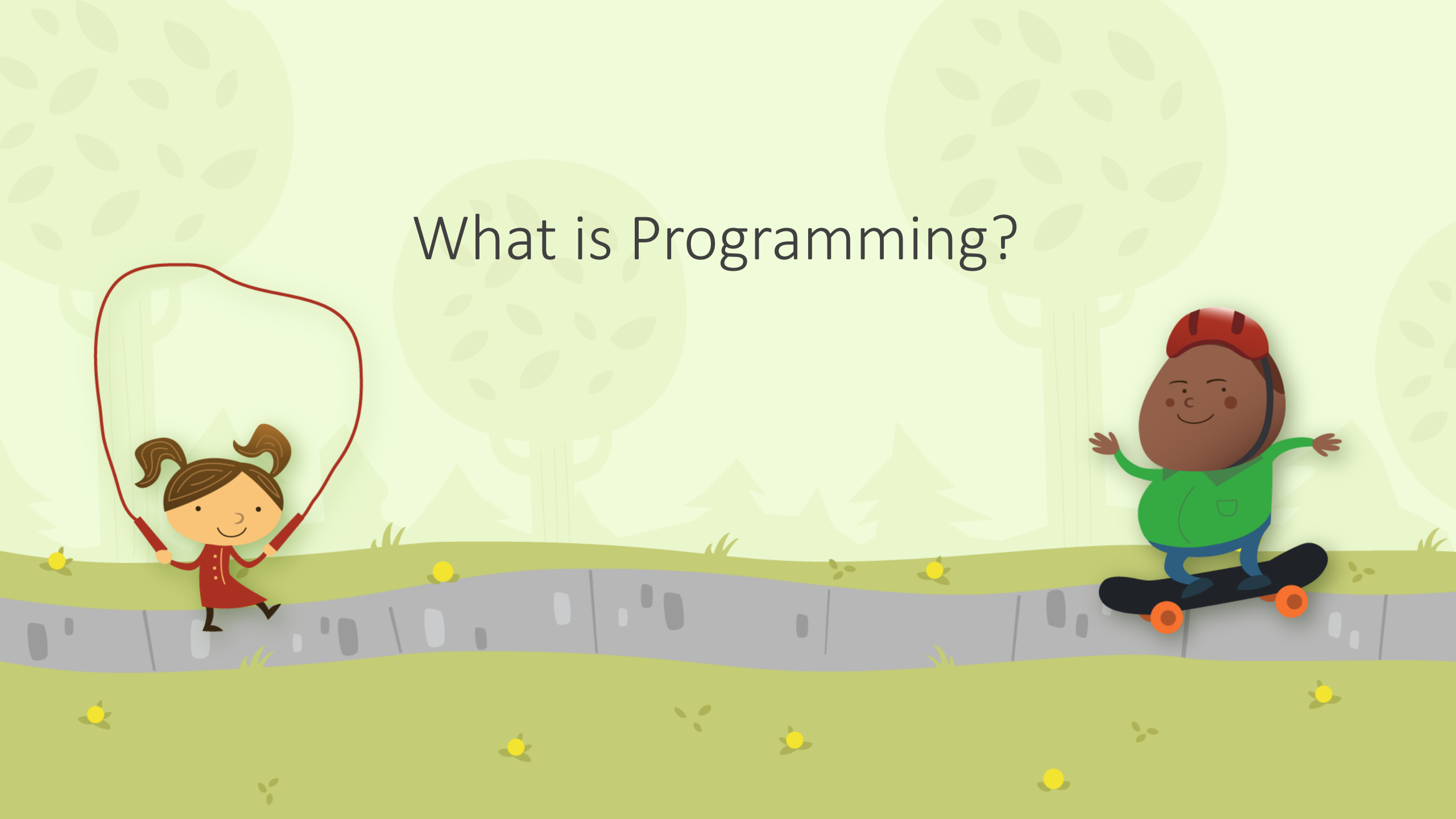
## Introduction

Victor Lee

Department of Electrical and Electronic Engineering



# What is Programming?



# What is Programming?

Problem: how to clean hair with a bottle of shampoo

Read the label on a shampoo bottle

1. Wet hair
2. Pour shampoo to hand
3. Apply shampoo to hair
4. Work into a lather
5. Rinse thoroughly
6. Repeat step 2 to 5 until clean



Programming is about writing a computer program to solve a problem.

Problem solved BUT how?



# Why Programming Skill is Essential to Everyone?

- It trains your **problem-solving skill** which means your ability to
  - Understand and formulate the problem (requirements and scope)
  - Design and develop a solution to the problem
  - Express the solution clearly and accurately
  - Evaluate tradeoffs among alternatives
  - Discover more creative solution
- It makes you more ready to the Information Age



# Why Programming Skill is Essential to Everyone?

- You will be patient
- You will be more attentive to details

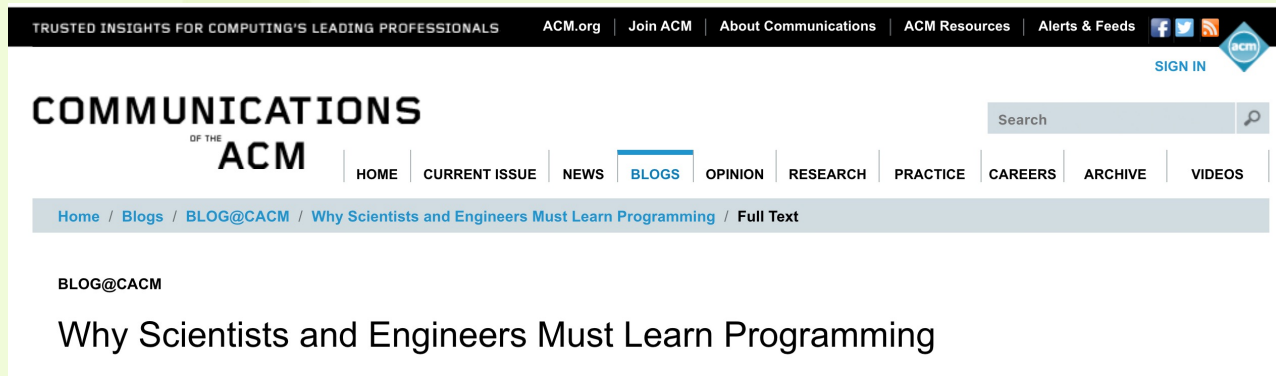
Do you need more good reasons to convince yourself to learning programming?



# Why Programming Skill is Essential to Engineering Students?

- Improve working efficiency (task automation)
- Discover creative solutions
- Communicate effectively (with programmers)

<https://cacm.acm.org/blogs/blog-cacm/166115-why-scientists-and-engineers-must-learn-programming/fulltext>



# Once again, what is Programming?

- Programming is a process to define the sequence of instructions (i.e., a computer program) to be executed on the computing device, one after another

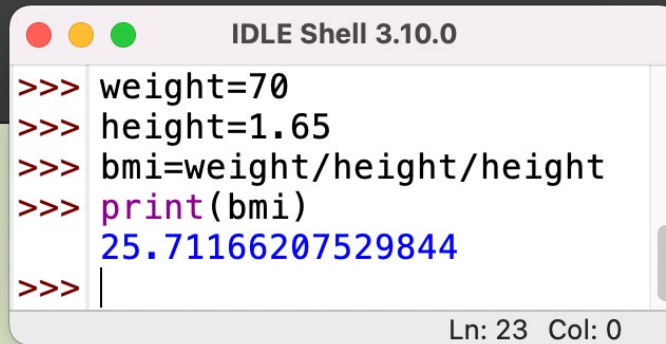
Let's try a toy problem:  
How can I tell whether I  
am too fat?





# A Sample Computer Program

```
weight=70 #weight in kg  
height=1.65 #height in m  
bmi=weight/height/height  
print(bmi)
```



```
IDLE Shell 3.10.0  
>>> weight=70  
>>> height=1.65  
>>> bmi=weight/height/height  
>>> print(bmi)  
25.71166207529844  
>>> |  
Ln: 23 Col: 0
```





# What is a Computer Program?

- A sequence of instructions executing on a **computer / computing device**
- Written in **programming language** (the language computers understand)
- A solution to a problem
- An artwork



What is a Computer?



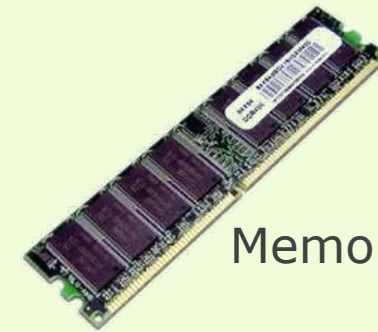
# Computing Device

## Secondary Storage



1. To run a program stored on the disk.

2. The program is loaded from the disk to the memory.



Memory

3. Instructions are read from memory to the CPU for execution, one at a time.



CPU (Central Processing Unit)



Motherboard

4. Results are stored back to memory or displayed through the output device.



Input device



Output device



# Some Difficulties

- Computer only follows instructions. It won't solve problems by itself.
- Programmers need to:
  1. develop an appropriate solution (logic)
  2. express the solution in programming language (implementation)
  3. find and correct bugs (programming errors) (debugging)
  4. validate the logic and implementation (testing)



# Requirements

- Correct syntax
- Correct logic
- Efficient
- Robust: running properly under various constraints
- Scalability
- Maintainability
- Portability: platform independent

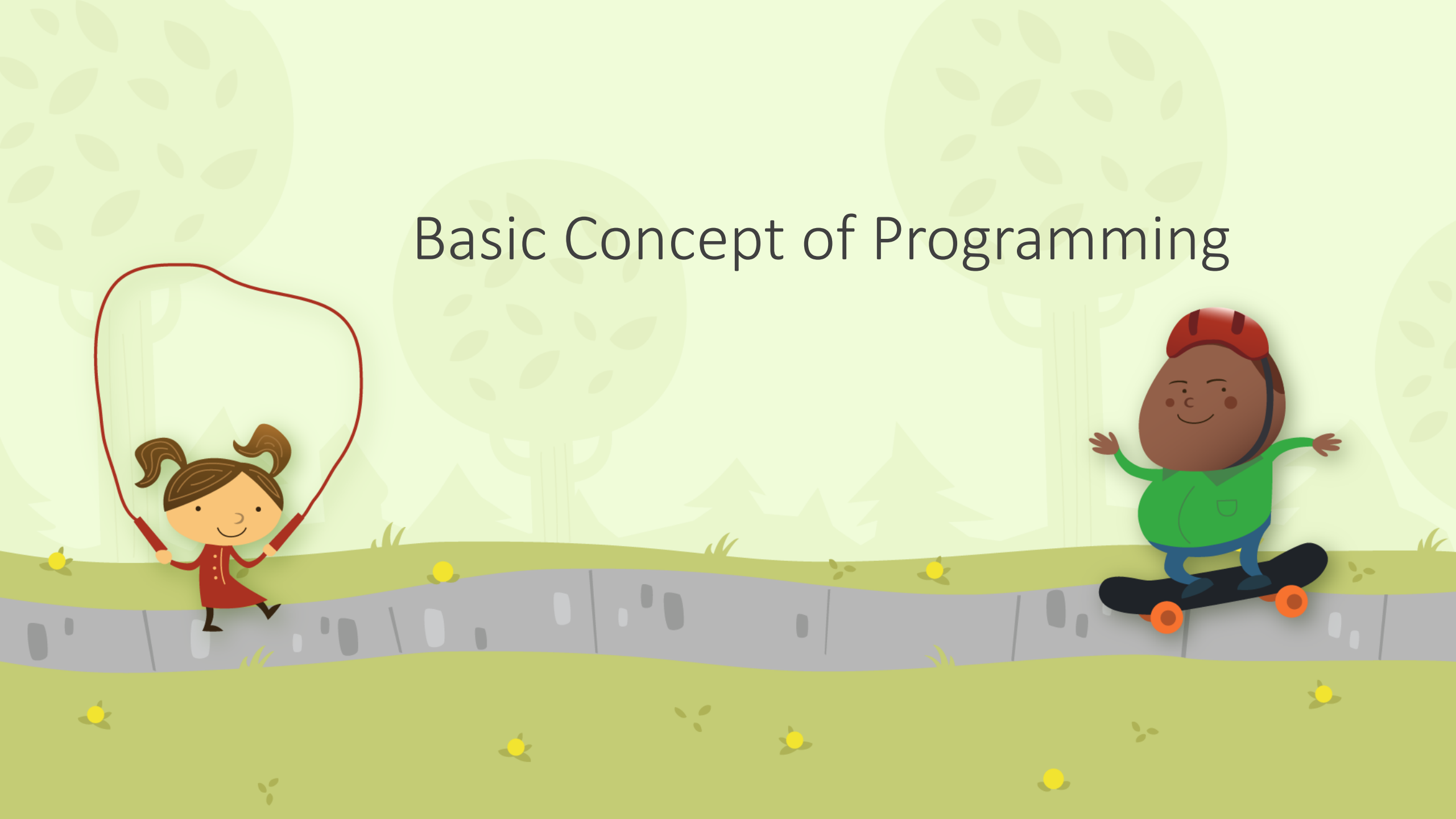




# Do Things Right vs Do the Right Things?



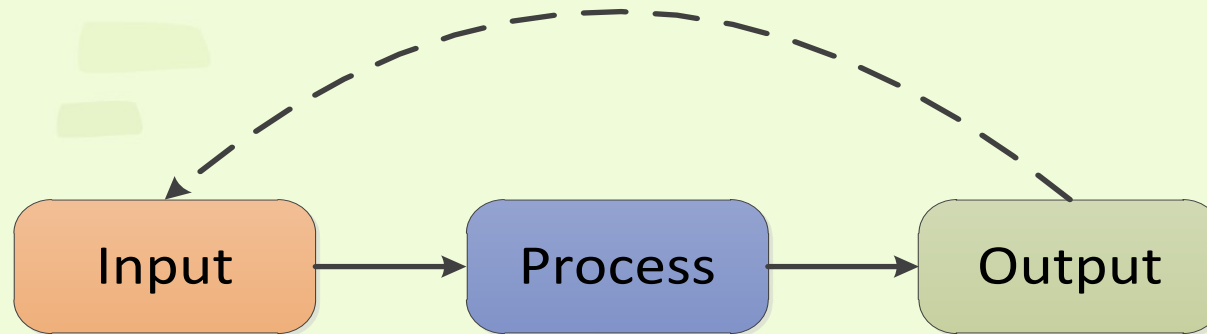
# Basic Concept of Programming





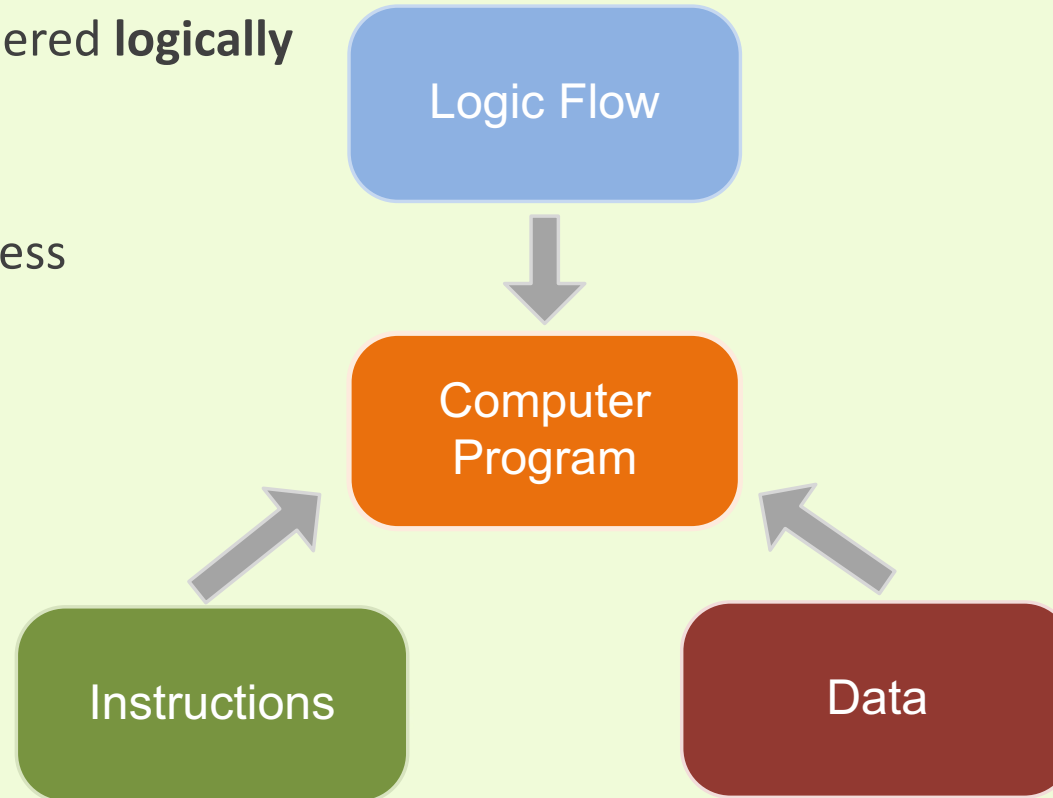
# Computer Program (External View)

- Basic elements of a computer program
  - Input: get data from keyboard, file, etc.
  - Process: process the data such as perform mathematical operations
  - Output: display result on the screen, save it in a file, etc.



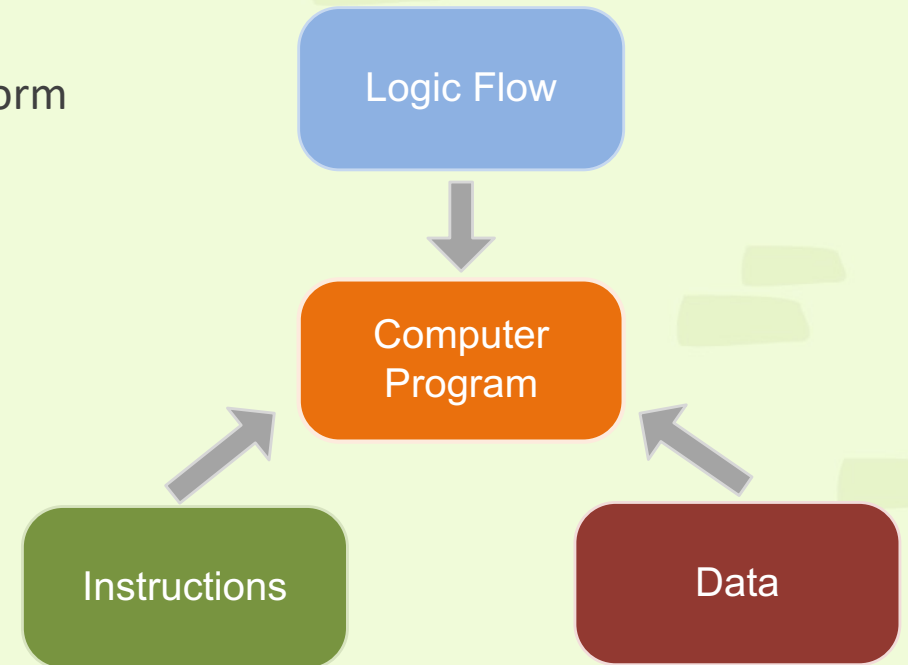
# Computer Program (Internal View)

- A list of **instructions** ordered **logically**
- Usually involve **data** access



# Computer Program

- Instructions
  - A set of predefined actions that a computer can perform
  - e.g., addition, subtraction, read, write
- Logic Flow
  - Arrangement of instructions
  - e.g., Calculate BMI
    1. Read weight from keyboard
    2. Read height from keyboard
    3. Compute BMI = weight / (height \* height)
    4. Write BMI to screen
- Variable (data)
  - A space for storing value temporarily for processing
- Constant (data)
  - A value that will not be changed throughout the processing



# Difficult...

- Computer has its own language.....Machine Language (low-level language)

1		00000000	00000100	0000000000000000
2	01011110	00001100	11000010	0000000000000010
3		11101111	00010110	00000000000000101
4		11101111	10011110	00000000000001011
5	11111000	10101101	11011111	00000000000010010
6		01100010	11011111	00000000000010101
7	11101111	00000010	11111011	00000000000010111
8	11110100	10101101	11011111	00000000000011110
9	00000011	10100010	11011111	00000000000100001
10	11101111	00000010	11111011	00000000000100100
11	01111110	11110100	10101101	
12	11111000	10101110	11000101	00000000000101011
13	00000110	10100010	11111011	00000000000110001
14	11101111	00000010	11111011	00000000000110100
15		01010000	11010100	00000000000111011
16			00000100	00000000000111101



# Symbolic Language

- Uses symbols, or mnemonics, to represent the various machine language instructions
- Translated into machine language by an **assembler**

```
1      entry    main, ^m<r2>
2      subl2    #12, sp
3      jsb      C$MAIN_ARGS
4      movab     $CHAR_STRING_CON
5
6      pushal   -8(fp)
7      pushal   (r2)
8      calls    #2, SCANF
9      pushal   -12(fp)
10     pushal   3(r2)
11     calls    #2, SCANF
12     mull3    -8(fp), -12(fp), -
13     pusha    6(r2)
14     calls    #2, PRINTF
15     clr1     r0
16     ret
```



# High-level Language

- Resembles human language
- Translated into machine language by a **compiler** or an **interpreter**

```
weight=70 #weight in kg  
height=1.65 #height in m  
bmi=weight/height/height  
print(bmi)
```



# Programming Languages

- To write a program for a computer, we must use a **computer language**



## Machine Language

Language directly understood by computers

## Symbolic Language

English-like abbreviations representing elementary computer operations

## High-level Language

Close to human language.

Example:  $a = a + b$

[add values of  $a$  and  $b$ , and store the result in  $a$ , replacing the previous value]



*binary code*

*assembly language*

*C, C++, Java, Basic*







# Programming Languages

- Programming languages usually differ in two aspects
  - Language syntax: rules that govern the structure of a program
  - Standard libraries/SDKs (software development kits) /functions



# “Hello World” in C++

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World!" << endl;
}
```



# “Hello World” in Python

```
print ("Hello World!")
```



# Building a C++ Program

- **Writing** source code in C++.
  - e.g., hello.cpp
- **Compilation**
  - Checks the **grammatical rules** (syntax).
  - Source code is converted to **object code** in machine language (e.g., hello.obj).
- **Linking**
  - Combines object code and libraries to create an **executable** (e.g., hello.exe).
  - Library: common functions (input, output, math, etc.)

```
#include  
<iostream>  
using namespace  
std;  
int main(){  
    int x;  
    int y;  
    int result;  
    cin >> x;  
    cin >> y;  
    result = x * y;  
    cout << result;  
    return 0;  
}
```

Source

Compiler

```
00101010 10110101  
10101011 10101111  
11110000 10101110  
10110101 10000001  
10101011 10101111  
11110000 10101010  
10101011 10111111  
10110101 10001001
```

Object

Linker

```
00101010 10110101  
10101011 10101111  
11110000 10101110  
10110101 10000001  
10101011 10101111  
11110000 10101010  
10101011 10111111  
10110101 10001001  
11011101 00100111  
11010110 10100110  
11011001 01010111
```

Executable



# Building a Python Program

- An **interpreter** is a program that reads and executes program code.

```
weight=70 #weight in kg  
height=1.65 #height in m  
bmi=weight/height/height  
print(bmi)
```

Source

```
weight=70  
#weight in kg  
height=1.65  
#height in m  
bmi=weight/height/height  
print(bmi)
```

Interpreter

Executable





# Where is Your Program in a Computer?

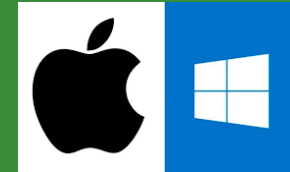
Application Program

C++ Program

Python  
Program

Interpreter

Operating System



Hardware





# Python vs C++

#1

## Advantages :

- Easy to learn
- Easy to access libraries
- Scientific community sharing (open source, many libraries)

## Disadvantages:

- Slow
- Interpreted (dependencies)
- Not typed (errors at runtime)

## Advantages :

- Execution speed
- Pre-Compiled (exe on machine)
- Typed (well defined)
- Modern professional libraries

## Disadvantages:

- Learning curve
- Harder to access libraries (less sharing than in Python)



# Who Use Python?



Quora

Google



Instagram



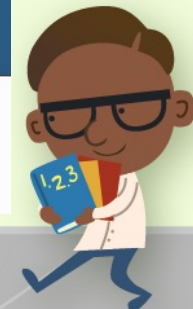
# Programming Environment

Python 3



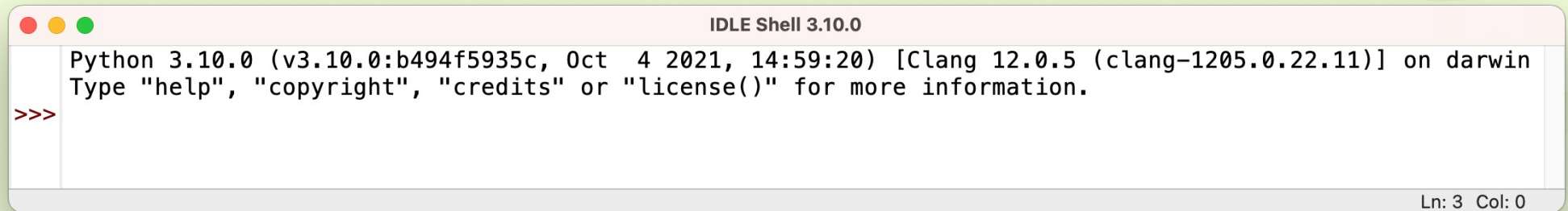
# Installing Python 3

- Go to Python home page: <https://www.python.org>
- Download and install the version that matches your OS



# Interactive Mode vs. Script Mode

- Interactive mode
  - Programmer directly interacts with the interpreter
  - Type a line of code at the prompt and hit Enter, the interpreter displays the results



```
IDLE Shell 3.10.0
Python 3.10.0 (v3.10.0:b494f5935c, Oct 4 2021, 14:59:20) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Ln: 3 Col: 0

- Script mode
  - Python code is stored in a file called a script with file extension `.py`
  - Run the interpreter to execute the script





# The First Program

Program comments,  
carry no action

No indentation

print is a function,  
to be discussed

```
#ENGG1330 Computer Programming I
```

```
print('Hello world') #display a string on the screen
```

The quotation marks ( ' ' or " " )  
mark the beginning and end of  
the string

Semicolon is optional



# Comments

- Start with "#" symbol
- Can be placed in separated line or at the end of a line
- Document non-obvious meaning of the code, especially the rationale behind

```
print('.....')    #print six dots to the screen
```

```
print('.....')    #print "work in progress" sign
```





# Function - print

- Print a value on the screen

- `print(1330)`
- `print('ENGG1330')`
- `print('GPA')`
- `print(3.8)`

## Data types:

- Integer (int) , e.g., 1330
- String (str) , e.g., 'ENGG1330' , 'GPA'
- Floating-point number (float) , e.g., 3.8

The data type can be checked using the function – `type`.

```

IDLE Shell 3.10.0
>>> type(1330)
<class 'int'>
>>> type('ENGG1330')
<class 'str'>
>>> type(3.8)
<class 'float'>
>>>
Ln: 37 Col: 13

```



# Function - print

- Print an expression (a combination of values and operators) on the screen
  - `print(13+30)`
  - `print(13-3)`
  - `print(13*3)`
  - `print(13/3)`
  - `print(13**3)`
  - `print('ENGG1330' + 'Computer Programming I')`

## Arithmetic operators:

- `+` : addition
- `-` : subtraction
- `*` : multiplication
- `/` : division
- `**` : exponentiation

This + operator performs string concatenation, joining strings end-to-end



# Function - print

- If the expression to print contains both string and number, use the `str()` function to convert the number to a string.
  - `print('GPA: '+str(3.8))`
- More examples:

```
print('2 + 3 = '+str(2+3))  
print(2+3)
```

```
2 + 3 = 5  
5
```



## The Second Program

```
#in addition to Hello, get a user input (string)  
#print a word Hello with input string  
  
print('Hello '+input())
```

```
>> John  
Hello John
```



# Function - input

- Read a string from keyboard
- Input string can be converted to integer or float by
  - `int(input())`
  - `float(input())`



## The Third Program

```
print('How old are you?')  
  
print('Next year you will ' \  
      'be ' + str(int(input())+1)+' years old! ')
```

If your code is too long, this is the way to break it into two lines

How old are you?

20

Next year you will be 21 years old!





# Summary

- Why programming skill is essential to everyone (and you)?
  - Programming skill helps one improve problem solving skill
  - Learning programming helps engineer works better and faster
- Basic concept of programming
- Programming is a process of writing, compiling, testing and debugging a computer program
- Simple programs

