# Programming Languages

# Preliminaries

(Slides are adopted from *Concepts of Programming Languages*, R.W. Sebesta; *Modern Programming Languages: A Practical Introduction,* A.B. Webber)

國立清華大學
National Tsing Hua University

เป้าหมายของวิชา Programming Language คือการทำให้นักศึกษาเข้าใจหลักการของภาษาโปรแกรม สามารถเปรียบเทียบภาษาได้อย่างมีเหตุผล เรียนรู้แนวคิดการออกแบบภาษา พัฒนาทักษะการเขียนโปรแกรม และเรียนรู้ภาษาใหม่ได้ง่ายขึ้น รวมถึงเข้าใจการทำงานของภาษาในระดับ compiler และ runtime

# 🎯 เป้าหมายของวิชา Programming Language (Course Objectives)

## ✅ 1. ทำให้นักศึกษาเข้าใจหลักการพื้นฐานของภาษาโปรแกรม

เพื่อให้รู้ว่า *ทุกภาษา* มีโครงสร้างร่วมกัน เช่น

- syntax (ไวยากรณ์)
- semantics (ความหมายของคำสั่ง)
- type system (ระบบชนิดข้อมูล)
- control structures (โครงสร้างควบคุม)
- data abstraction (นามธรรมของข้อมูล)

> เป้าหมาย: เข้าใจ “ศาสตร์” ไม่ใช่แค่ “ภาษา”

## ✅ 2. สามารถวิเคราะห์และเปรียบเทียบภาษาโปรแกรมได้

นักศึกษาควรสามารถตอบได้ว่า

- ภาษาไหนอ่านง่ายกว่า? (readability)
- ภาษาไหนเขียนงานประเภทนี้ได้เหมาะกว่า? (writability)
- ภาษาไหนปลอดภัยกว่า? (safety + type checking)
- ภาษาไหนมีประสิทธิภาพกว่า? (efficiency)

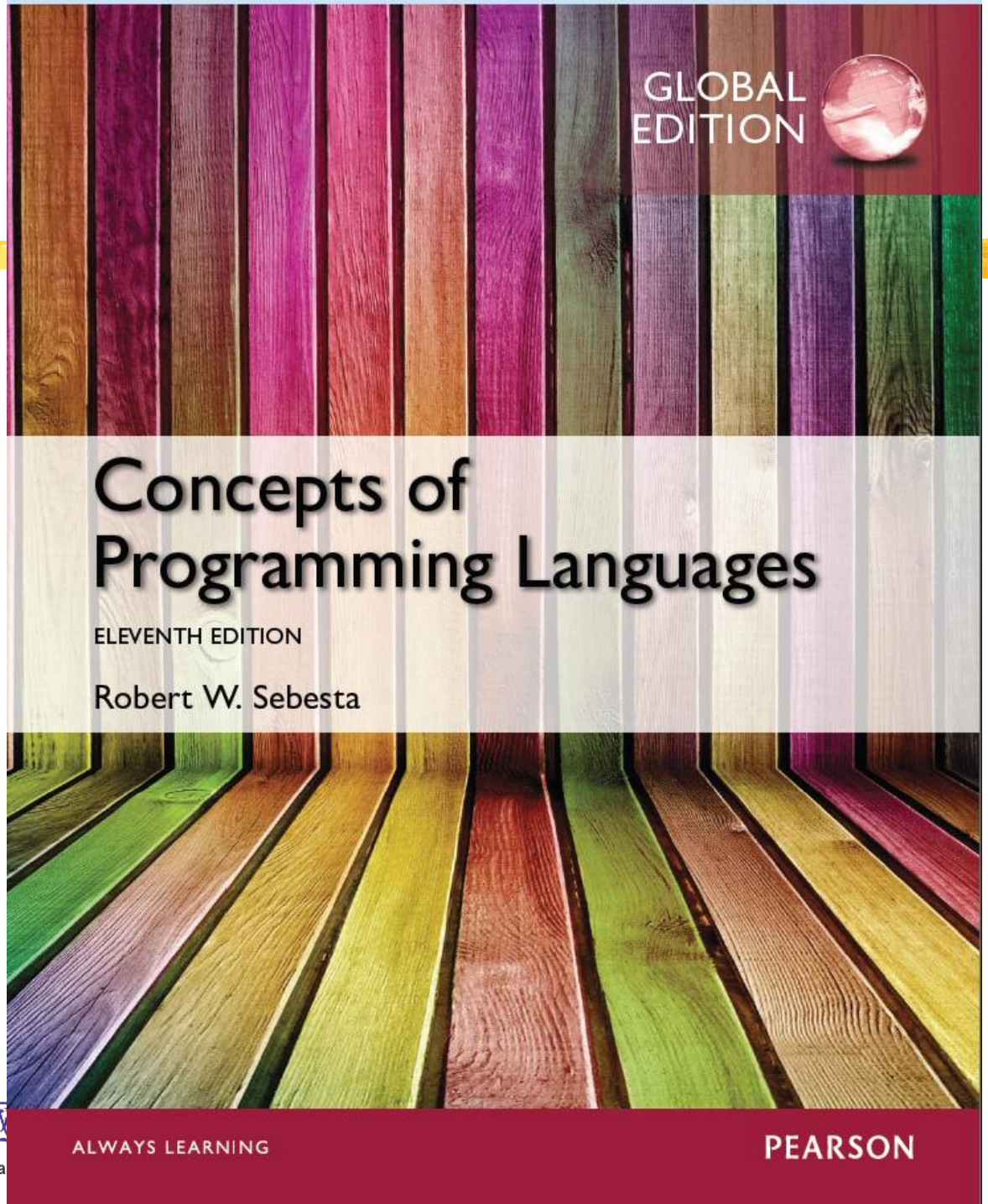
> เป้าหมาย: **เลือกภาษาให้เหมาะกับงาน ไม่ใช่เขียนตามกระแส**

## ✅ 3. เข้าใจแนวคิดการออกแบบภาษาโปรแกรม (language design principles)

เช่น

- ทำไมภาษา Python ถึงไม่มี type เดิม ๆ แบบ C?
- ทำไม Java ต้องมี garbage collection?
- ทำไม C ต้องกำหนด memory แบบ low-level?

> เป้าหมาย: **เข้าใจเหตุผลเบื้องหลังการออกแบบ**


GLOBAL
EDITION
Concepts of
Programming Languages
ELEVENTH EDITION
Robert W. Sebesta
ALWAYS LEARNING
PEARSON

```c
#include <stdio.h>


//chapter 9
double addDouble(double a, double b) {
    try { //chapter 14 (try-catch)
            a = a/b;
        }
        catch (...) {
            cout << "divide by zero"
        }
    return a + b;
}

int main() {

    //chapter 5+6
    int x=5;
    double y;

    //chapter 7
    x = y*6+1*x;

    //chapter 8
    if( x<10) {
        //chapter 9
        y = addDouble(2.5, 1.2);

    }

    //chapter 9
    return 0;
}
```

```
class Stack {
public:
    Stack();              // const
    ~Stack();             // destr

    void push(int value);
    void pop();
    int top() const;
    bool isEmpty() const;
    int size() const;

private:
    int* data;
    int capacity;
    int topIndex;
};
```

**EXAMPLE 3.1** A Grammar for a Small Language

```
<program> → begin <stmt_list> end

<stmt_list> → <stmt>
            | <stmt> ; <stmt_list>
<stmt> → <var> = <expression>
<var> → A | B | C
<expression> → <var> + <var>
             | <var> - <var>
             | <var>
```



The language described by the grammar of Example 3.1 has only one statement form: assignment. A program consists of the special word **begin**, followed by a list of statements separated by semicolons, followed by the special word **end**. An expression is either a single variable or two variables separated by either a + or - operator. The only variable names in this language are A, B, and C.

A derivation of a program in this language follows:

```
<program> => begin <stmt_list> end
          => begin <stmt> ; <stmt_list> end
          => begin <var> = <expression> ; <stmt_list> end
          => begin A = <expression> ; <stmt_list> end
          => begin A = <var> + <var> ; <stmt_list> end
          => begin A = B + <var> ; <stmt_list> end
          => begin A = B + C ; <stmt_list> end
          => begin A = B + ; <stmt> end
          => begin A = B + C ; <var> = <expression> end
          => begin A = B + C ; B = <expression> end
          => begin A = B + C ; B = <var> end
          => begin A = B + C ; B = C end
```