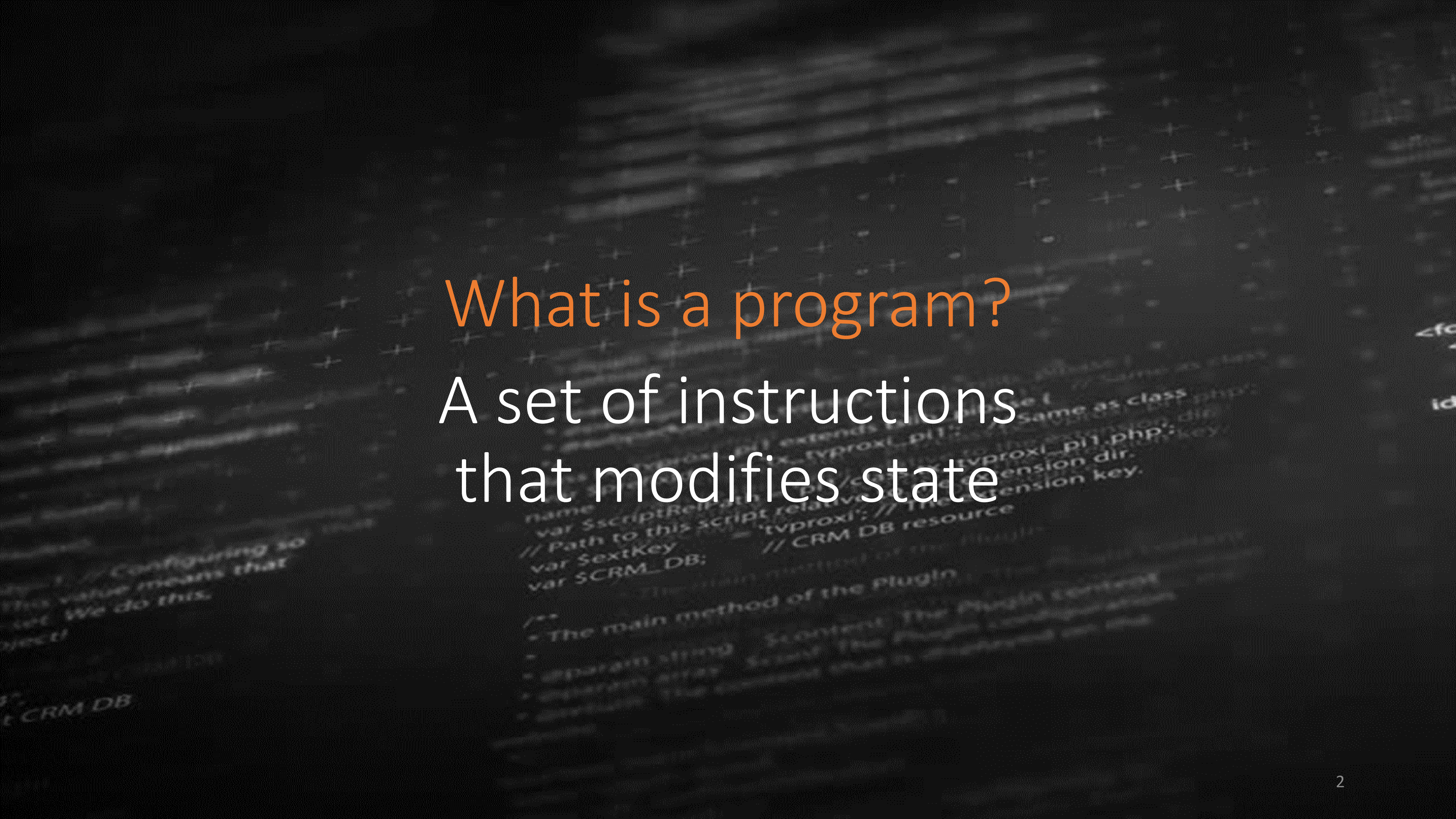
Lecture 1

Basics of Programming & C

TIC1001 Introduction to Computing and Programming I



What is a program?

A set of instructions that modifies state

Like a cooking recipe

What is the state?

- The plate? The food?
- A recipe is a set of instructions that manipulate the state

Starting State

- various raw ingredients

Ending State

- coherent tasty dish

Many ways to write a recipe

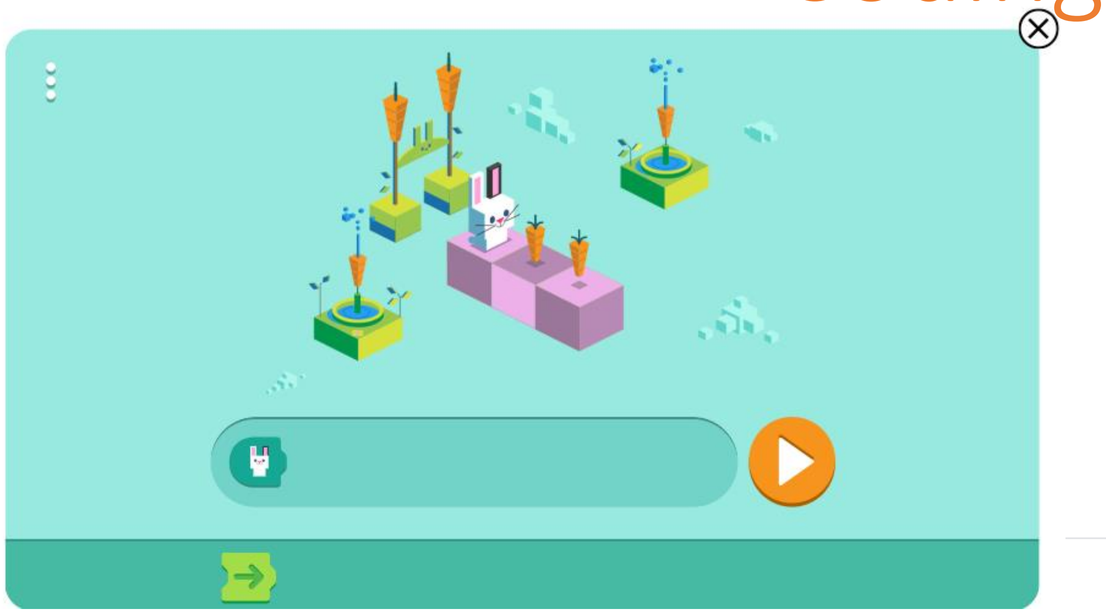
Declarative: describe what you want

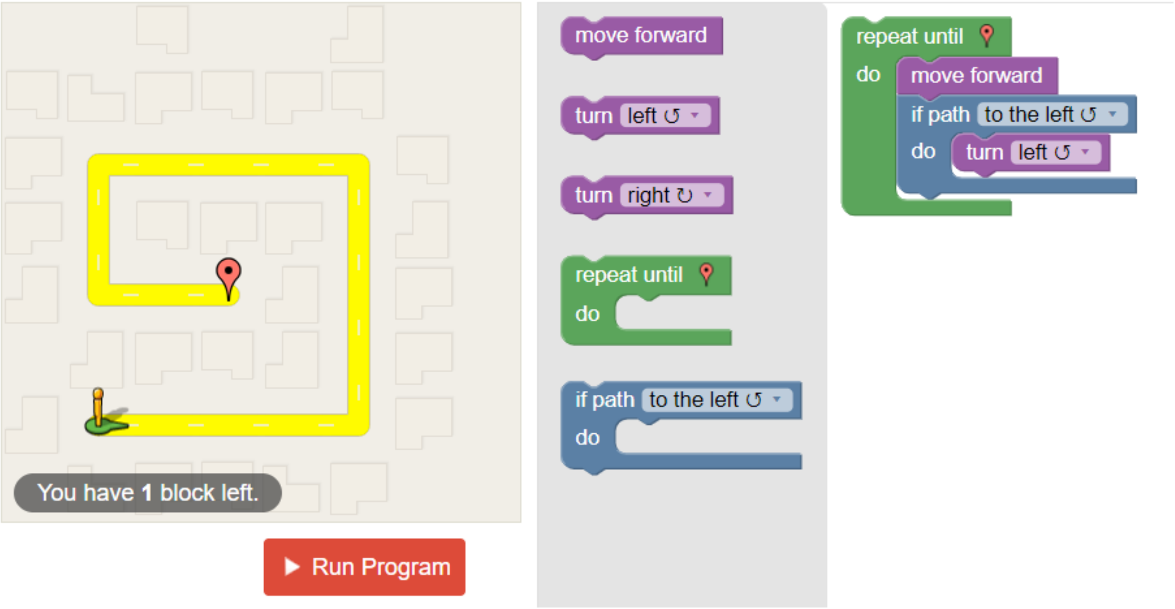
- Whisk egg white until firm
- Fold into mixture until combined

Imperative: state how it is done

1. Suspend a whisk in the egg white
2. Whirl whisk as fast as possible for 10 secs
3. Lift whisk from egg white
4. Check for peaks
5. Repeat from step 5 if no peaks

Coding Games





```
repeat until
do  move forward
    if path to the left ↺
    do  turn left ↺
```

What is the state?

- – Position of character
- – Carrots/Destination

What are the instructions?

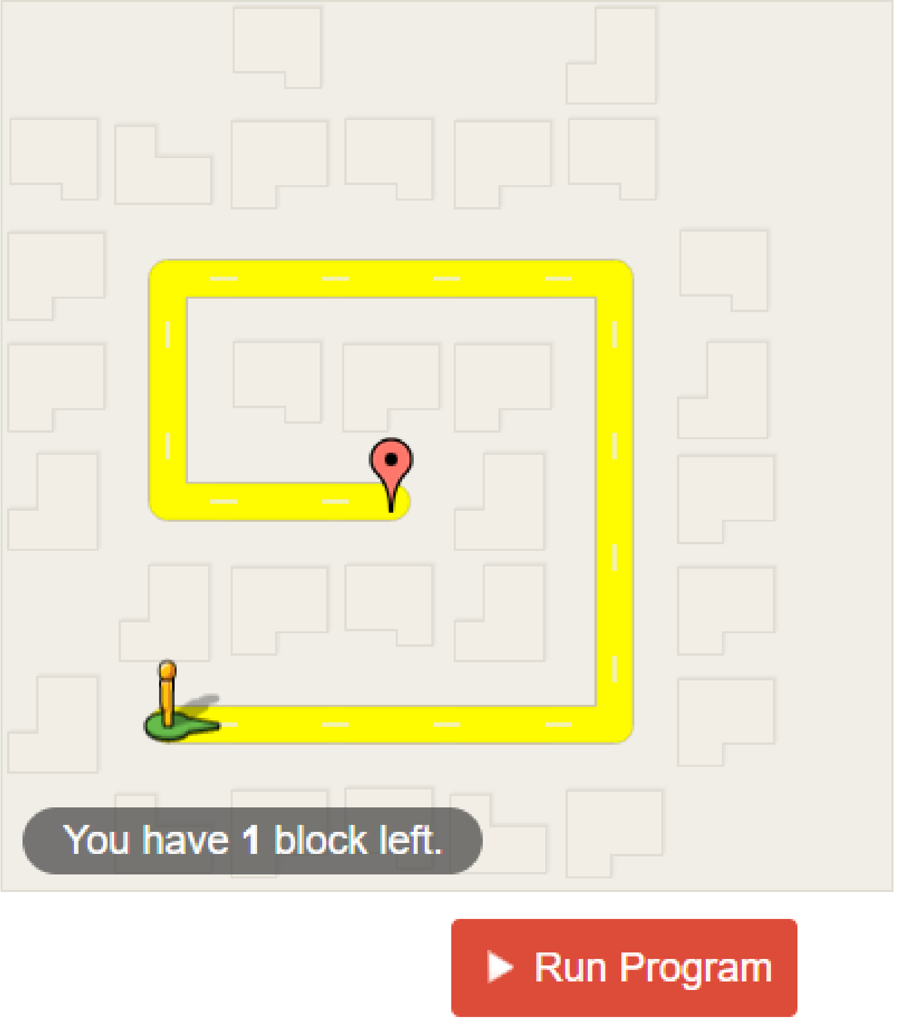
- – Move/Turn
- – Repeat
- – Check

Computers are dumb
(at least currently)

They require precise instructions

They require primitive instructions

Precise instructions are needed

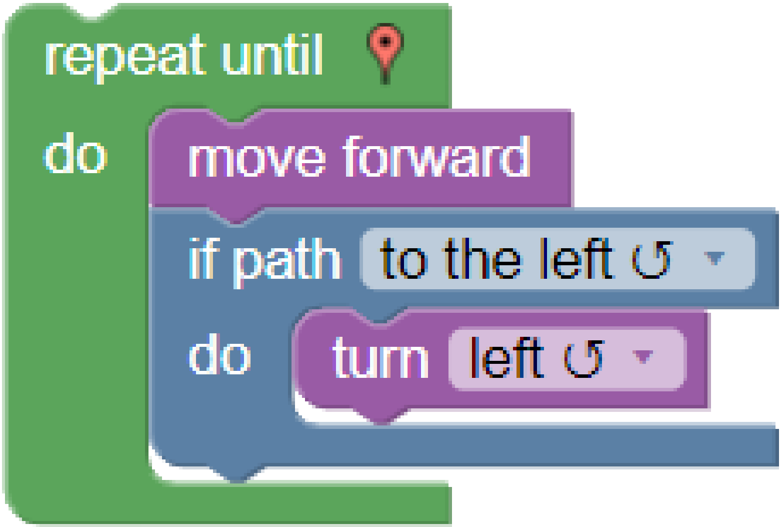


You have 1 block left.

Imprecise

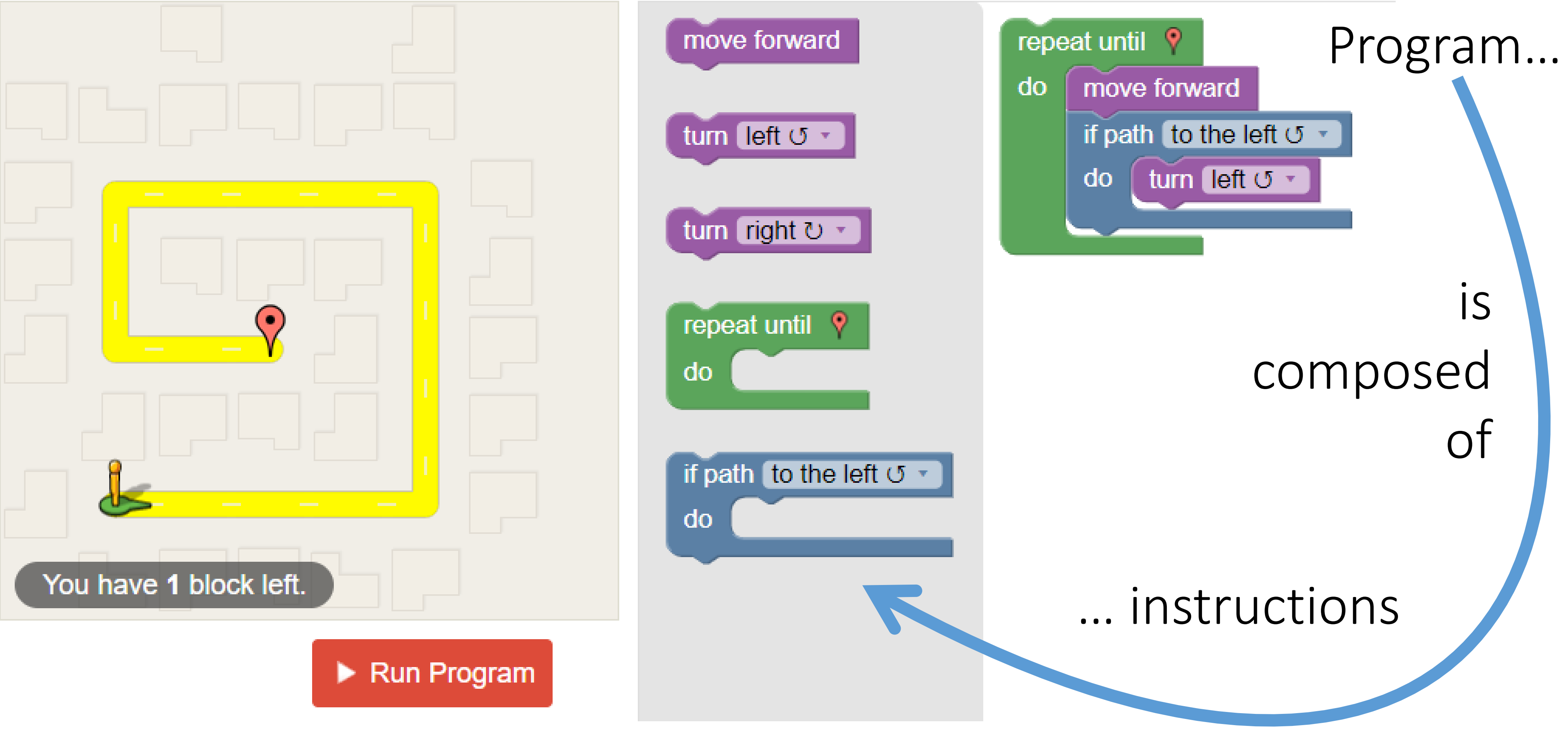
– Follow the path until destination

Precise



```
repeat until 📍
do  move forward
    if path to the left ↺
    do  turn left ↺
```

Primitives



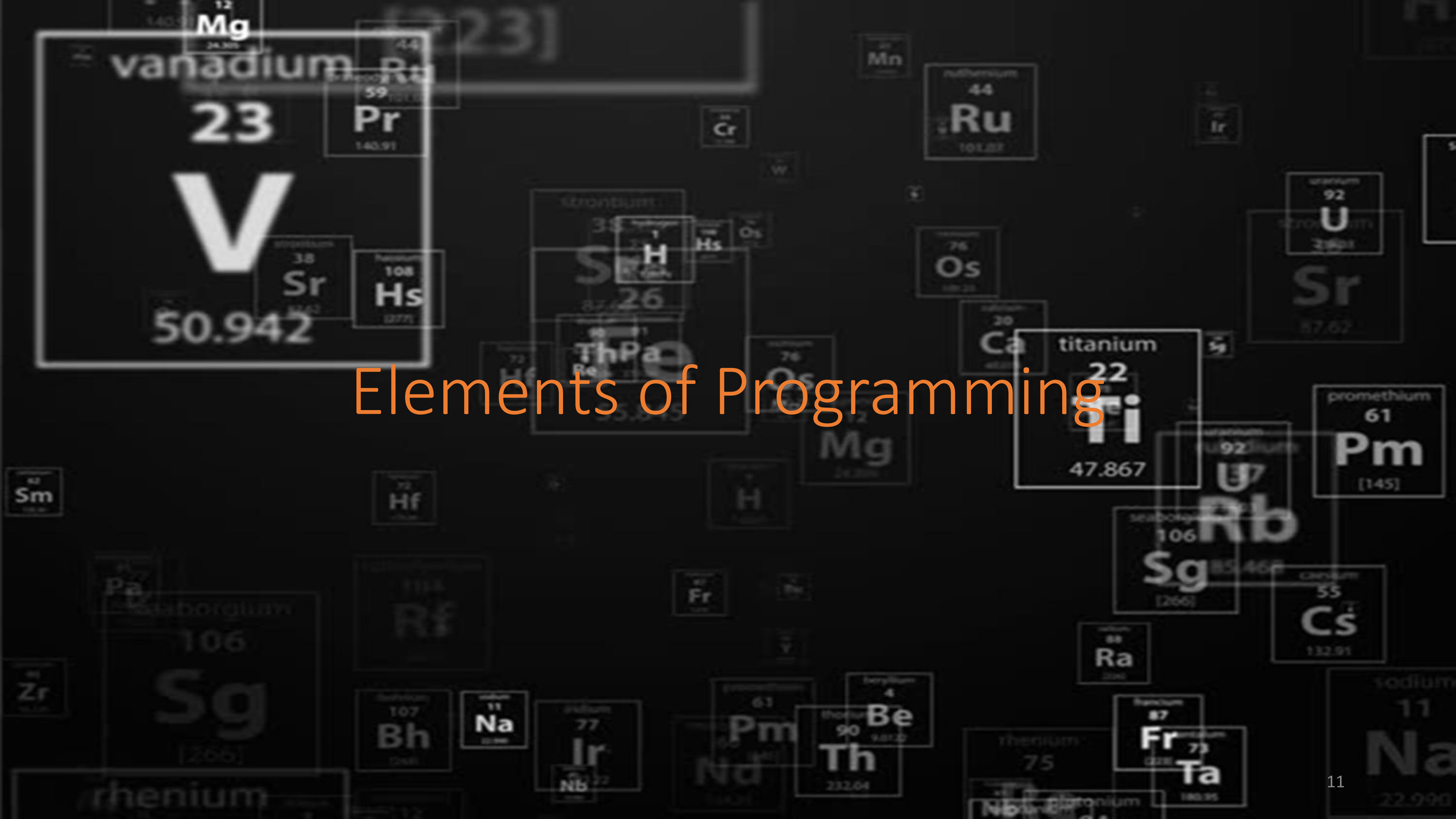
You have 1 block left.

```
repeat until
do
    move forward
    if path to the left ↺
    do
        turn left ↺
```

Program...

is composed of

... instructions

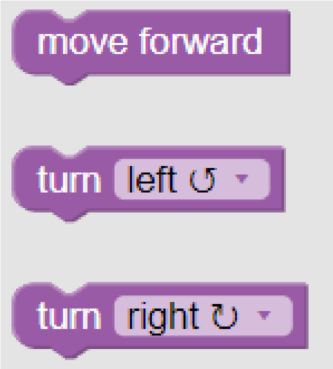


Elements of Programming

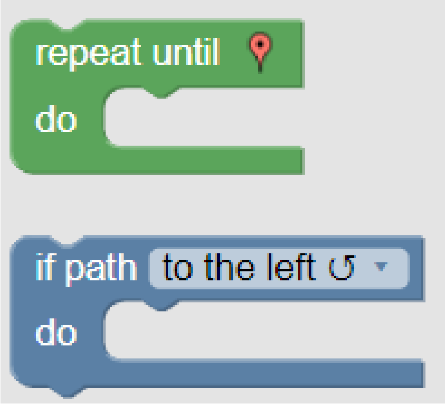
Elements of Programming

1. Abstraction of the state
   - `x, y, position`

2. Means of mutating state
   - `y = x + 1`



3. Controlling flow with logic
   - while x > 1, do this…
   - if x > 1, do this…



A C/C++ program?

What is the state?

- The memory space of the computer
- Contains data, i.e. numbers
- Accessed by variables

The instructions manipulate the state

- Modifying the variables
- Written in C language

Computers are programmed using a language

What is a language?

What is a language?

A method of communication

- A means to codify the elements of programming

A language consist of

- **Vocabulary**: symbols or words
- **Grammar**: rules how these symbols are used
- **Syntax**: how the symbols are placed
- **Semantics**: meaning communicated by the symbols

Language of C/C++

Variables

Symbols that map to some part of memory (state)

- kinda like in math

Contain only alphanumeric characters or _

- `A-Z a-z 0-9 _`
- Cannot start with a number

Case sensitive

- `my_variable ≠ My_Variable`

Cannot be a reserved keyword

- e.g. `if for int`

Types

A variable refers to some data in memory

- Type refers to how the data is interpreted

Example:

- Files stored in your hard disk are just data
- Different types: .png, .mp3, .doc, etc.
- Determine how the file should be interpreted

Primitive Types in C

A variable must be declared as a specific type

`short`
`int`
`long`
	integers, e.g. 42

`float` `double`	“floating point”, decimal, e.g. 42.525
`char`	a character, enclosed in single quotes, e.g. **`'c'`**

All these types are just numbers

Size of types

Why different types for numbers?

- Specify the size of the type, i.e. range of numbers

Integer types

- `char` ≤ `short` ≤ `int` ≤ `long` ≤ `long long`

Decimal types

- `double` is more precise than `float`

Character type?

- an integer from 0 to 256, interpreted using ASCII

More details when you learn about Computer Organisation

Declaring Variables

Before using variables, they need to be declared

- by specifying the name
- and the type

Syntax:

- `<type> <var>;`
- `<type> <var1>, <var2>, …;`
- `<type> <var> = <value>;`

Declaring variables

Examples:

```
int age;
int height, weight;
double area;
double pi = 3.1415927;
```

Declaring variables

Once declared, a variable cannot be re-declared (within the same scope)

- Scope? Discuss few weeks later

Example:

```
int age;
age = 15;
double age;
```

Error!

Initializing variables

Variables should be initialized with a value before use

- Otherwise, it will contain some random value
- Why? Again you will learn about in Computer Organisation

```
int age;
cout << age;
```

Contains random value

Use meaningful variables

Avoid random letters

- e.g. `abc`, `foo`, `xxx`

Use meaningful words

- e.g. `height`, `weight`

Separate words with _

- e.g. `radius_of_circle`, `height_in_inch`

Style is to use lowercase for variables

What's this ;

Semicolon

Denotes the end of a statement

- In English, we use the full stop.
- In C/C++, we use the semicolon;

Whitespace matters not

In C/C++, whitespace is ignored

- extra spaces
- tabs
- newlines

Example:

```
int birth_year=2001;
int age = 2017 -
          birth_year;
```

Assignment

Sets a variable to the value of an expression

Example:

```
int age;
age = 15;
age = 10 + 15;
age = age + 1;
```

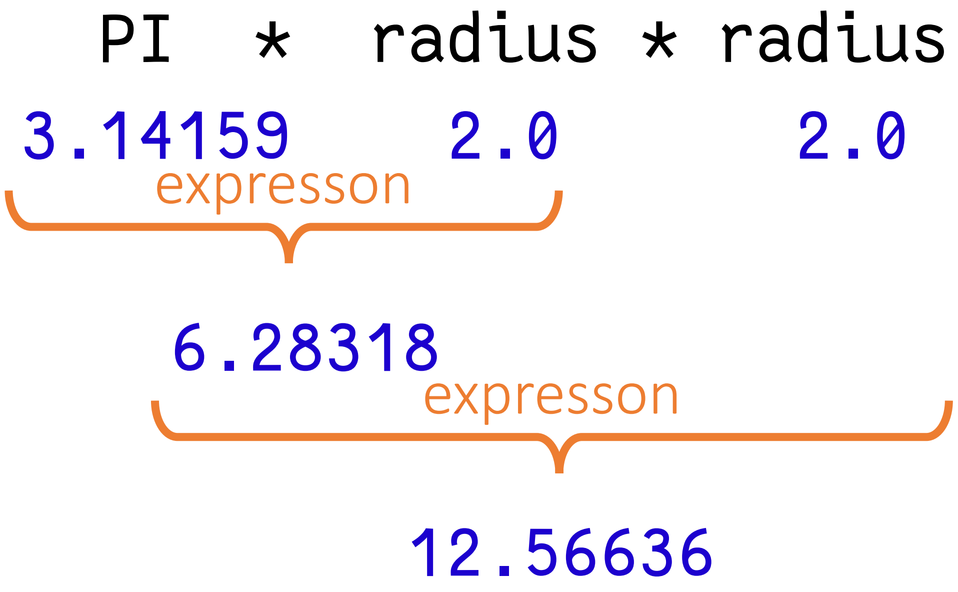
Expression (Arithmetic)

Involves an arithmetic operation

`+ - * / %`

between two other expressions

Example:



```
   PI   *   radius * radius
3.14159     2.0       2.0
     expresson
     6.28318
              expresson
          12.56636
```

Arithmetic Operators

+	addition
-	subtraction
*	multiplication
/	division
%	modulo (remainder)
++	increment
--	decrement

Type Conversion

Converting from one type to another

Two kinds

- Implicit type conversion
- Explicit type conversion

Implicit Type Conversion

All expressions are typed

For *op* ∈ { `+,-,*,/` }

- i op j → `int` if both i and j are `int`
- i op j → `double` if i or j is `double`

`%` can only be used on integers

Exercise:	`22+7` →	`29`	`22/7.0` →	`3.142857`
	`22.0-7.0` →	`29.0`	`22/7` →	`3`
	`22.0*7` →	`154.0`	`22%7` →	`1`

Implicit Type Conversion

What happens when $v = \frac{4}{3}\pi r^3$ is written as

```
v = 4/3 * 3.142 * r * r * r;
```

Implicit Promotion

- Small to big
- `55 + 1.75`

Implicit Demotion

- Big to small
- `int money = 23.16;`
- Lost of precision or unpredictiable results

Explicit Type Conversion

Casting operator

- Syntax: `(type) operand`

Example:

- `(int) 3.14`

Type Conversion

Examples

```
1.0 * 22/7       → 22.0/7          → 3.142
(double)22/7     → 22.0/7          → 3.142
1.0 * (22/7)     → 1.0 * 3         → 3.0
(double)(22/7)   →(double)3        → 3.0
(int)(22.0/7)    →(int)3.142…      → 3
```

Typed Assignment

Expressions are evaluated before assignment

Example:

```
int miles = 3;
double kms;
miles = miles * 1.609;
kms = miles;
```

What are the values stored in miles and kms?

So far, we have seen

Elements of C language

- Variables
- Types
- Assignment Statements
- Arithmetic Expressions

Allows us to manipulate the program state

What is missing?

- Starting/Initial state (Input)
- Ending/Goal state (output)

Organization of C/C++ programs

C/C++ programs are made up of functions

- a C/C++ file can contain several functions
- statements must belong to a function

Think of functions as mini programs

- Each function maintains its own state

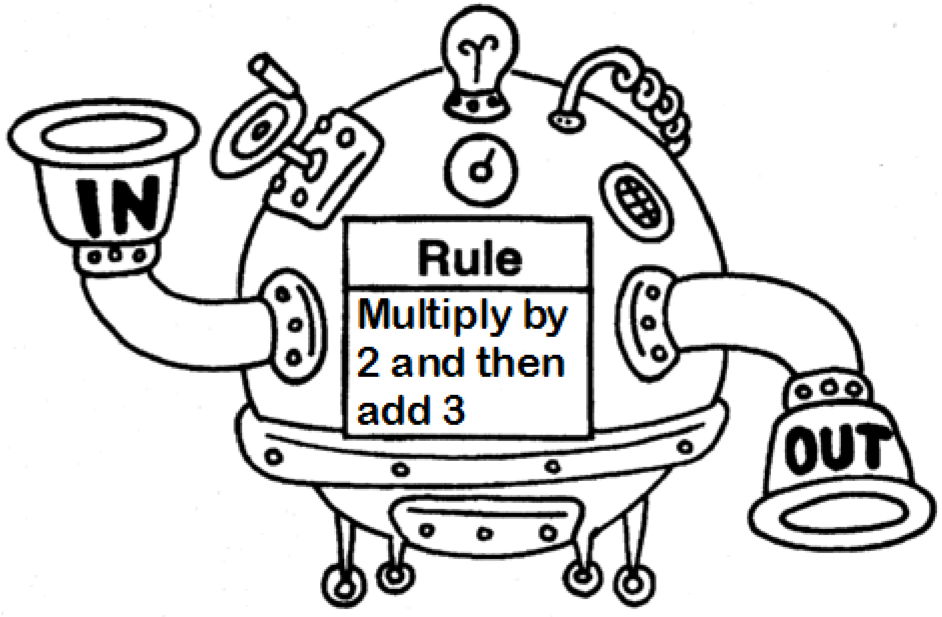
What is a Function?

A construct that

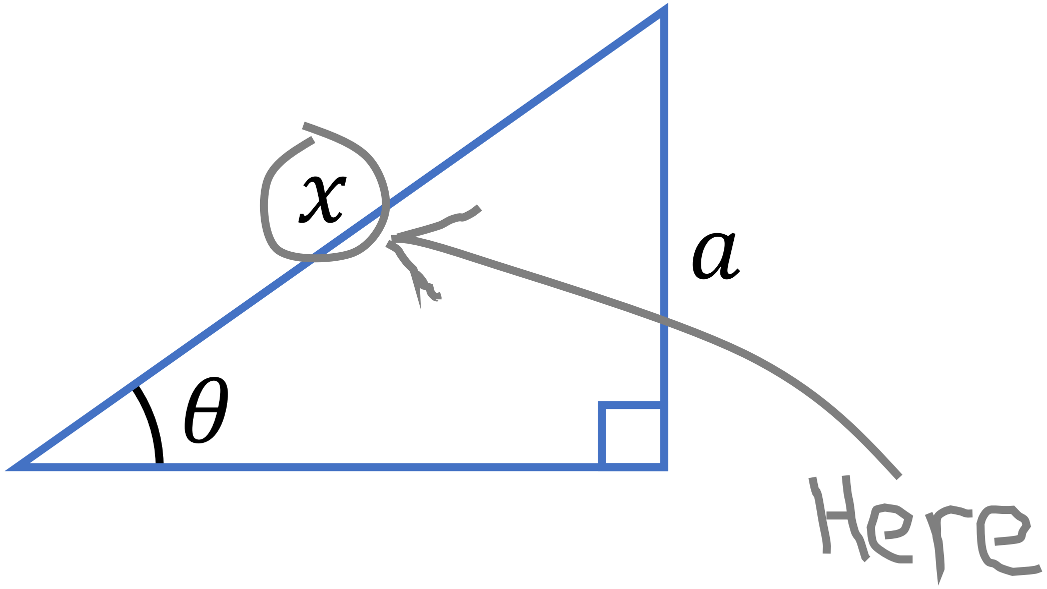
- takes in some inputs
- performs some rules/instructions
- produce an output

In other words,

- a mini program



Functions are nothing new



Find x?

$$\sin(\theta) = \frac{a}{x}$$

Function

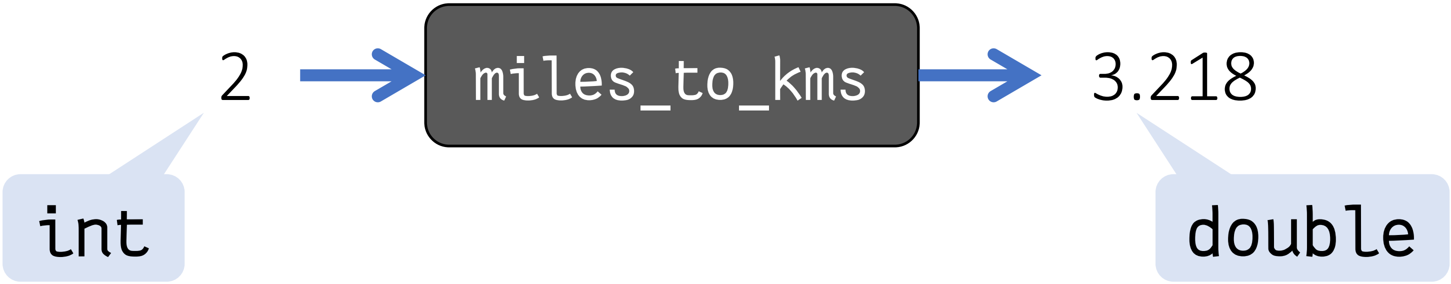
Input

Output

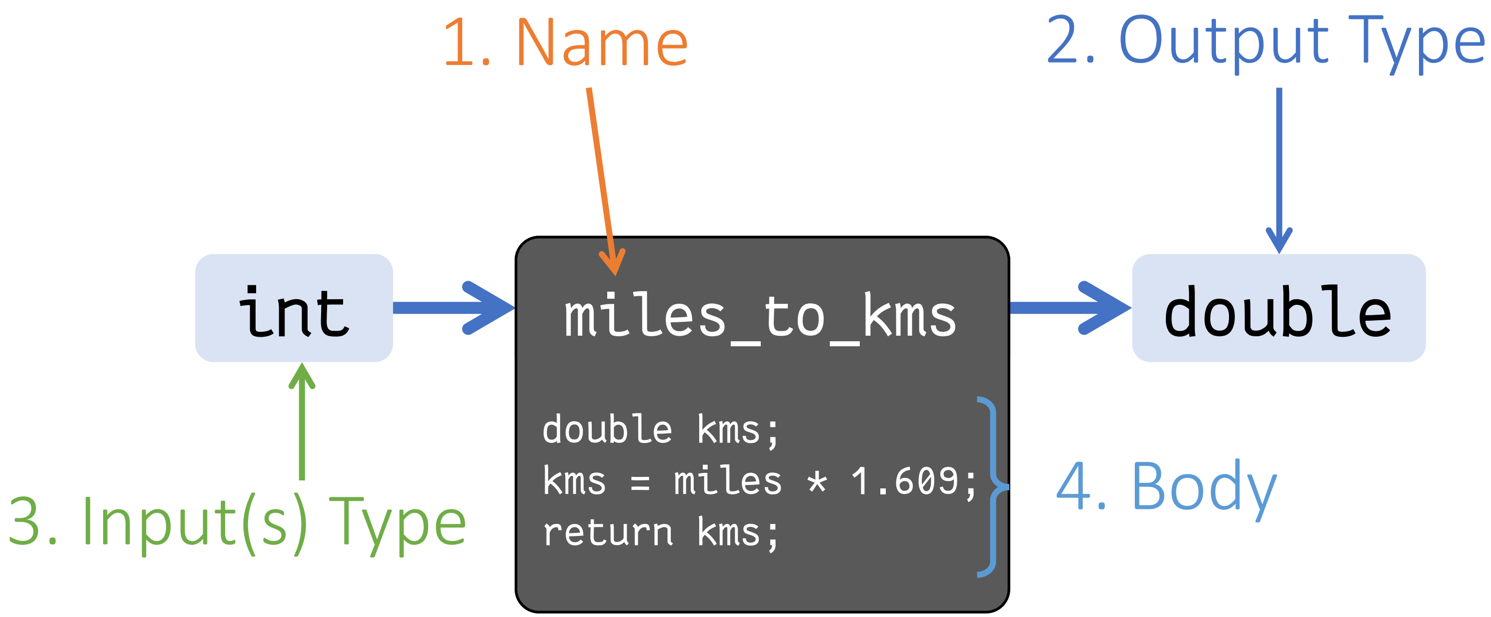
Miles to km

Suppose we have a function miles_to_kms.

- What do you think it does?
- What is the input?
- What is the output?
- What is the type?



Components of a Function



1. Name

2. Output Type

int

miles_to_kms

double

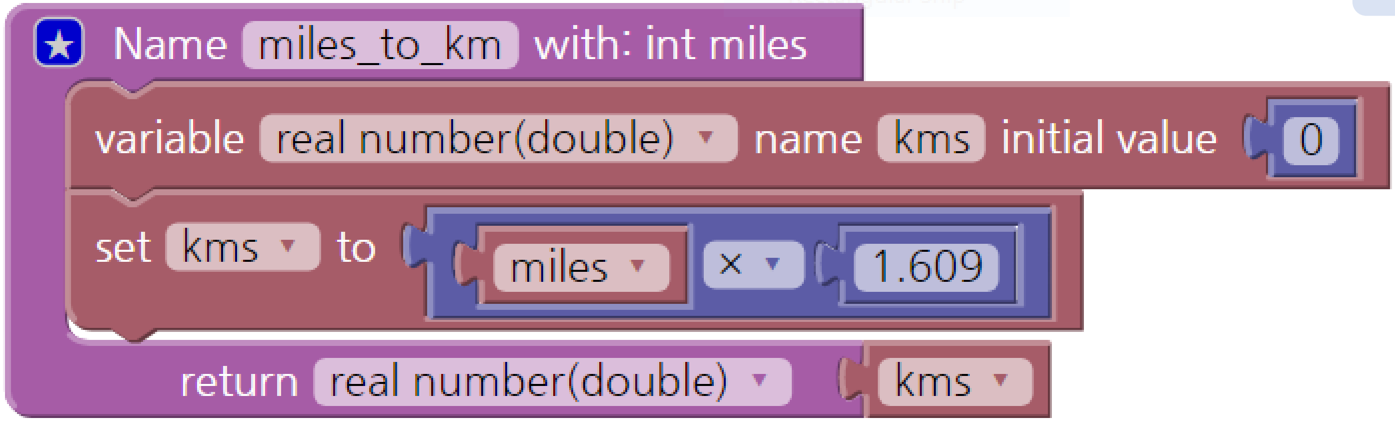
```
double kms;
kms = miles * 1.609;
return kms;
```

3. Input(s) Type

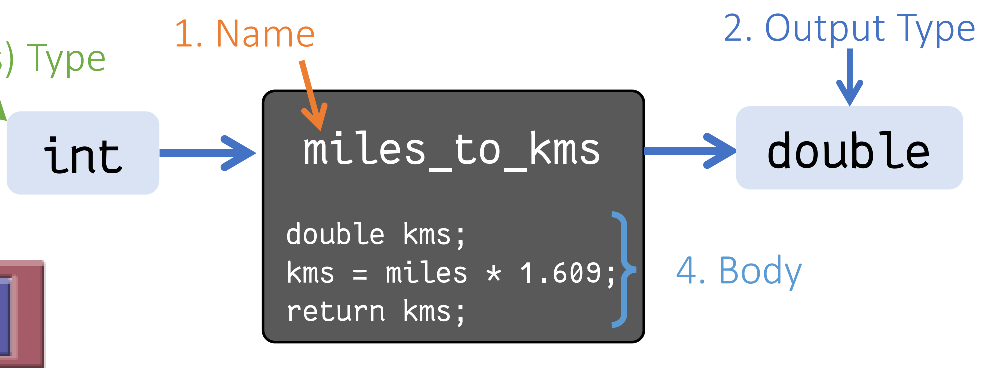
4. Body

Cake Graphical Editor

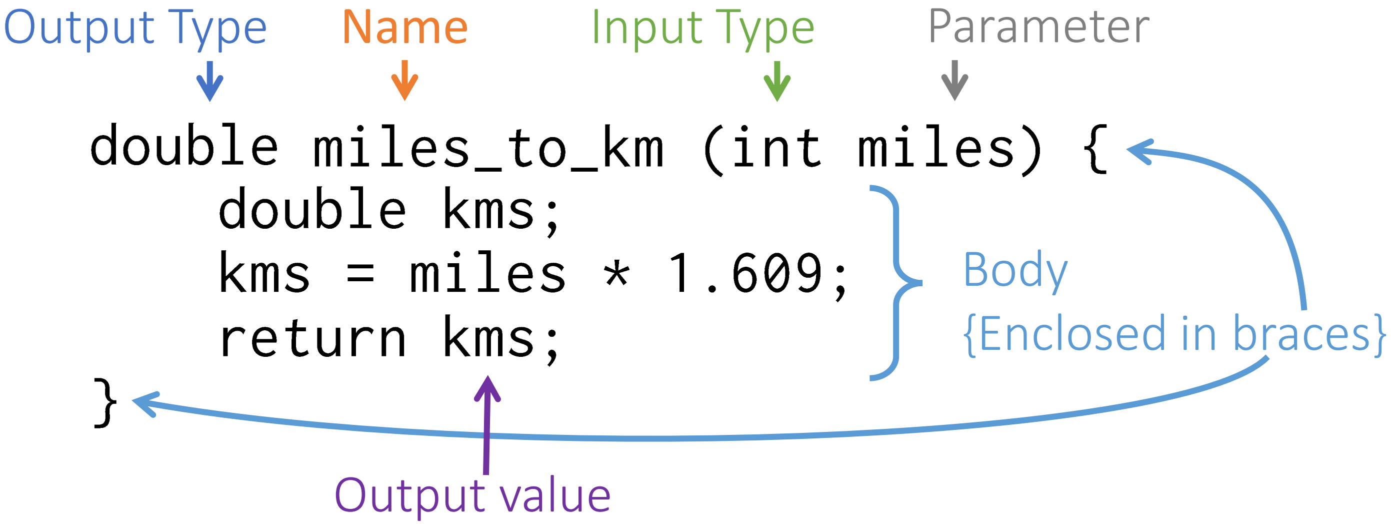
In C Syntax



```
★ Name miles_to_km with: int miles
    variable real number(double) ▾ name kms initial value 0
    set kms ▾ to miles ▾ × ▾ 1.609
    return real number(double) ▾ kms ▾
```



```
double kms;
kms = miles * 1.609;
return kms;
```



```
double miles_to_km (int miles) {
    double kms;
    kms = miles * 1.609;
    return kms;
}
```

Trace the Function

```
double miles_to_km(int miles) {
    double kms;
    kms = miles * 1.609;
    return kms;
}
```

Suppose the input is 3.

```
miles: 3
  kms: 30915
```

Trace the Function

```
double miles_to_km(int miles) {
    double kms;
    kms = miles * 1.609;
    return kms;
}
```

Suppose the input is 3.

```
miles: 3
  kms: 4.827
```

Trace the Function

```
double miles_to_km(int miles) {
    double kms;
    kms = miles * 1.609;
    return kms;
}
```

Suppose the input is 3.

```
miles: 3
  kms: 4.827
```

The output of the function is 4.827

Another Example

```
int midpoint(int a, int b) {
    return (a + b) / 2;
}
```

What will be the output if the function is called with `midpoint(5, 10)`?

```
    a = 5
    b = 10
```

Recall

In C/C++, statements must belong to a function

The `main` function

The entry point of a program

- Program cannot run without a main function

The syntax:

```
int main(void) {

    /* body of function */

    return 0;
}
```

The `main` function

```
int main(void) {

    /* body of function */

    return 0;
}
```

What are the inputs?

What is the output?

The `main` function

For now, you do not need to bother with this function.

We will provide a template for your assignments.

Only copy and paste the relevant function into Coursemology for submission.

Example Template

```c
#include <stdio.h>

// Edit your answers here
double f_to_c() {

}

double c_to_f() {

}
```

Copy and paste this part into Coursemology

```c
// main function for your testing. Do not copy into Coursemology
int main(void) {
    double out;
    out = f_to_c(72);    // edit the input to test
    printf("Your function output is: %f\n", out);

    out = c_to_f(27);    // edit the input to test
    printf("Your function output is: %f\n", out);
}
```

Comments

- Text that is not part of the program
  - for human eyes only
- Block comments
  - Delimited by `/* ... */`
- Single-line comments
  - All text after `//`

Indentation

```
double miles_to_km(int miles) {
    double kms;
    kms = miles * 1.609;
    return kms;
}
```

Notice the body is indented

- makes code easier to read
- +1 level for each block of code `{ ... }`
- typically 4 spaces (or just press tab in your editor)

Compiling Code

- The CPU does not actually understand C
- Only understands it own machine code
  - Intel i386
  - Amd64
  - PowerPC
  - ARM
- The compiling process
  - translate C language to machine code

Edit-Compile-Run

Editing

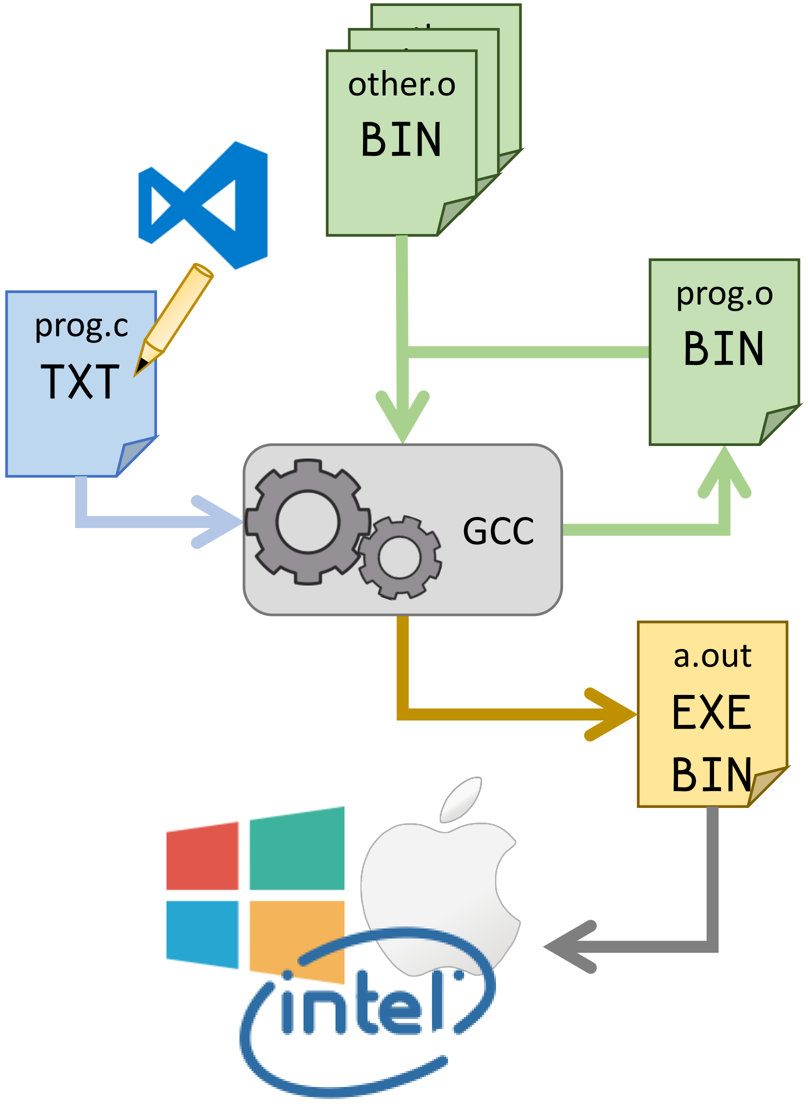
- Write code with a text editor

Compile (+ Linking)

- Translates C code to machine code
- gcc compiler generates a.exe or a.out

Run

- loads and executes the machine code



So far so good?

Let's try an experiment

```c
int main(void) {
    int age;
    printf("Input your age:");
    scanf("%d", &age);

    if (age >= 18) {
        printf("You can vote");
    } else {
        printf("You are not eligible for voting");
    }
    return 0;
}
```

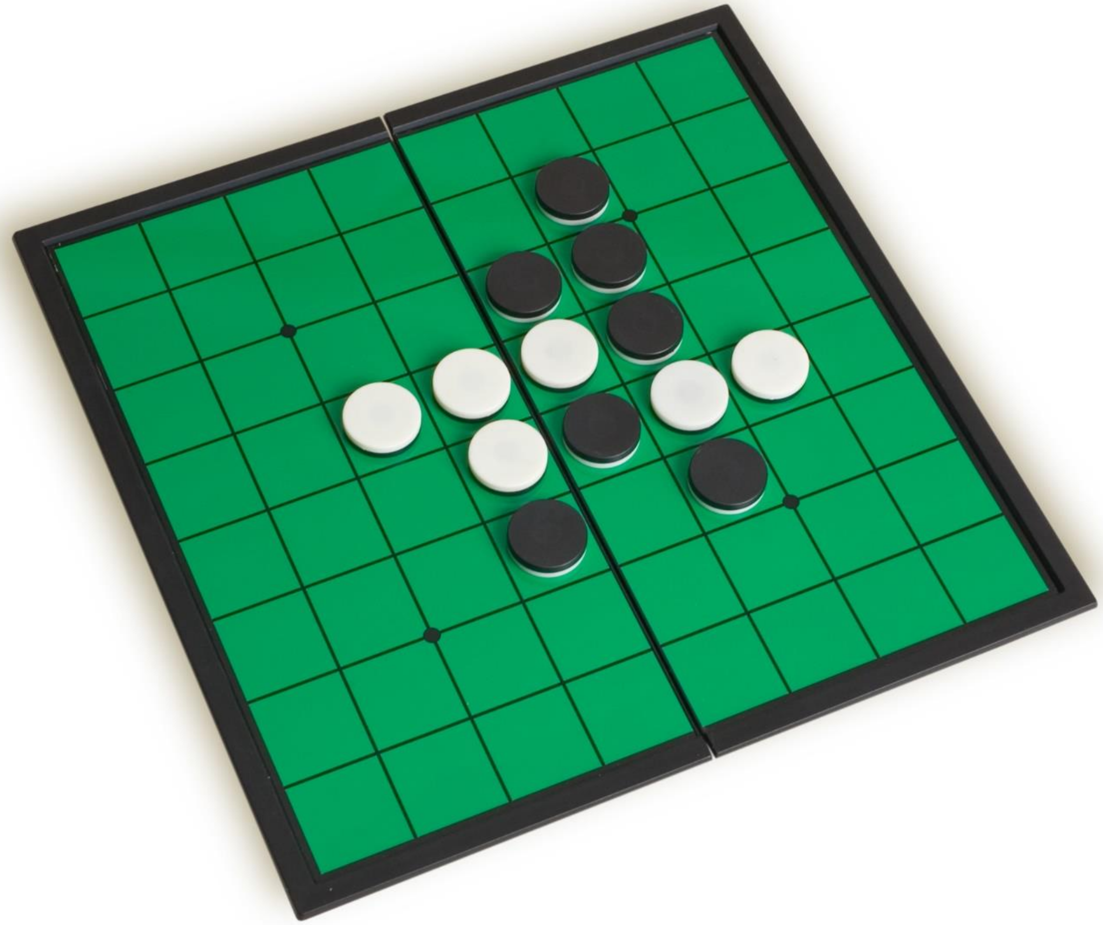
Can you guess what this program does?

Seems easy yah?

Why do people find programming hard?

An analogy

> A minute to learn,
> But a lifetime to master



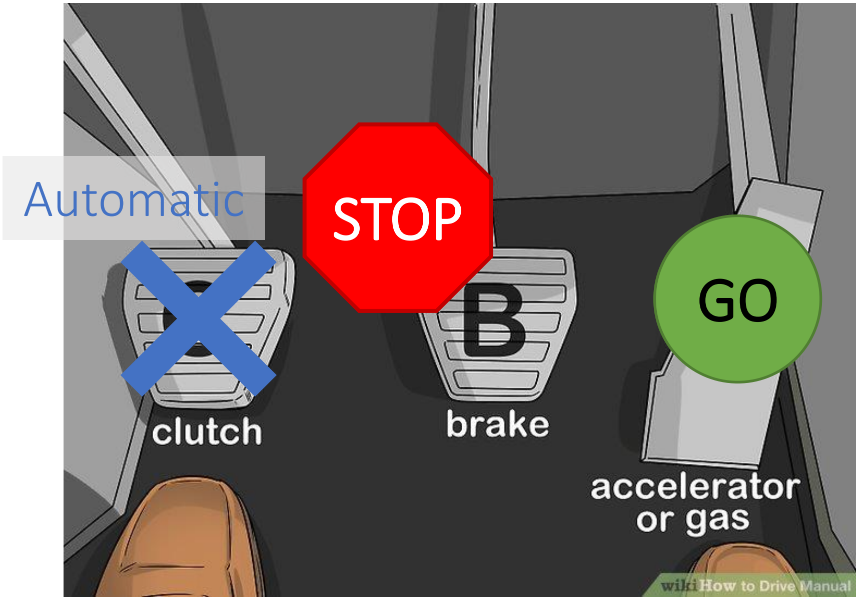
An analogy

Driving a car is easy

- Why need expensive lessons?

Just learn from

- Manuals
- Textbook
- YouTube



Programming = Problem Solving

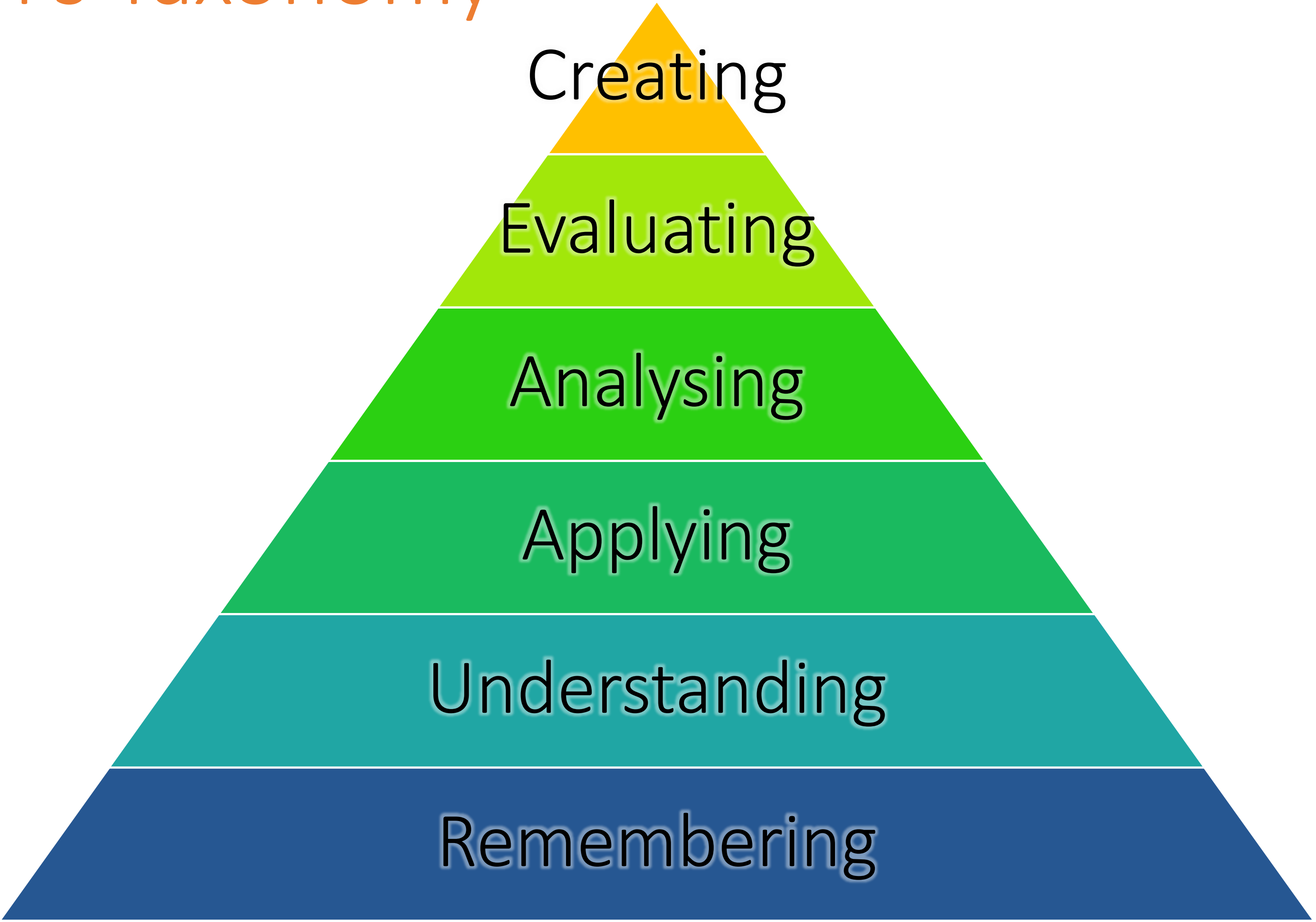
Writing code is a creating process

It requires a certain

- way of thinking
- approach to problem solving
- creative process
- domain knowledge

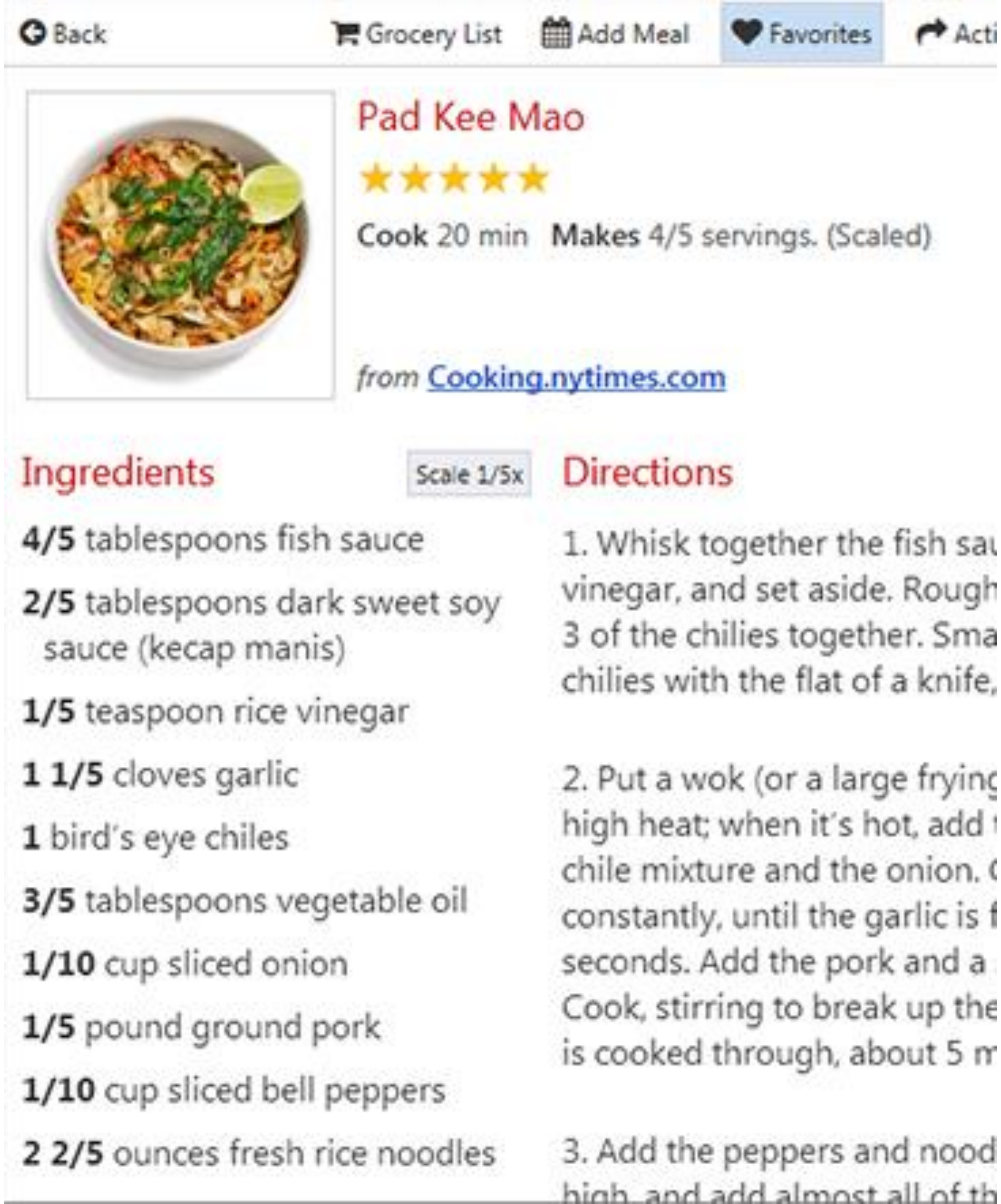
Cannot be taught, only trained

Bloom’s Taxonomy



MasterChef

Can you prepare a tasy dish?



Pad Kee Mao

★★★★★

Cook 20 min **Makes** 4/5 servings. (Scaled)

from Cooking.nytimes.com

Ingredients

4/5 tablespoons fish sauce

2/5 tablespoons dark sweet soy sauce (kecap manis)

1/5 teaspoon rice vinegar

1 1/5 cloves garlic

1 bird's eye chiles

3/5 tablespoons vegetable oil

1/10 cup sliced onion

1/5 pound ground pork

1/10 cup sliced bell peppers

2 2/5 ounces fresh rice noodles

Directions

1. Whisk together the fish sau
vinegar, and set aside. Rough
3 of the chilies together. Sma
chilies with the flat of a knife,

2. Put a wok (or a large frying
high heat; when it's hot, add
chile mixture and the onion.
constantly, until the garlic is
seconds. Add the pork and a
Cook, stirring to break up the
is cooked through, about 5 m

3. Add the peppers and nood
high, and add almost all of th

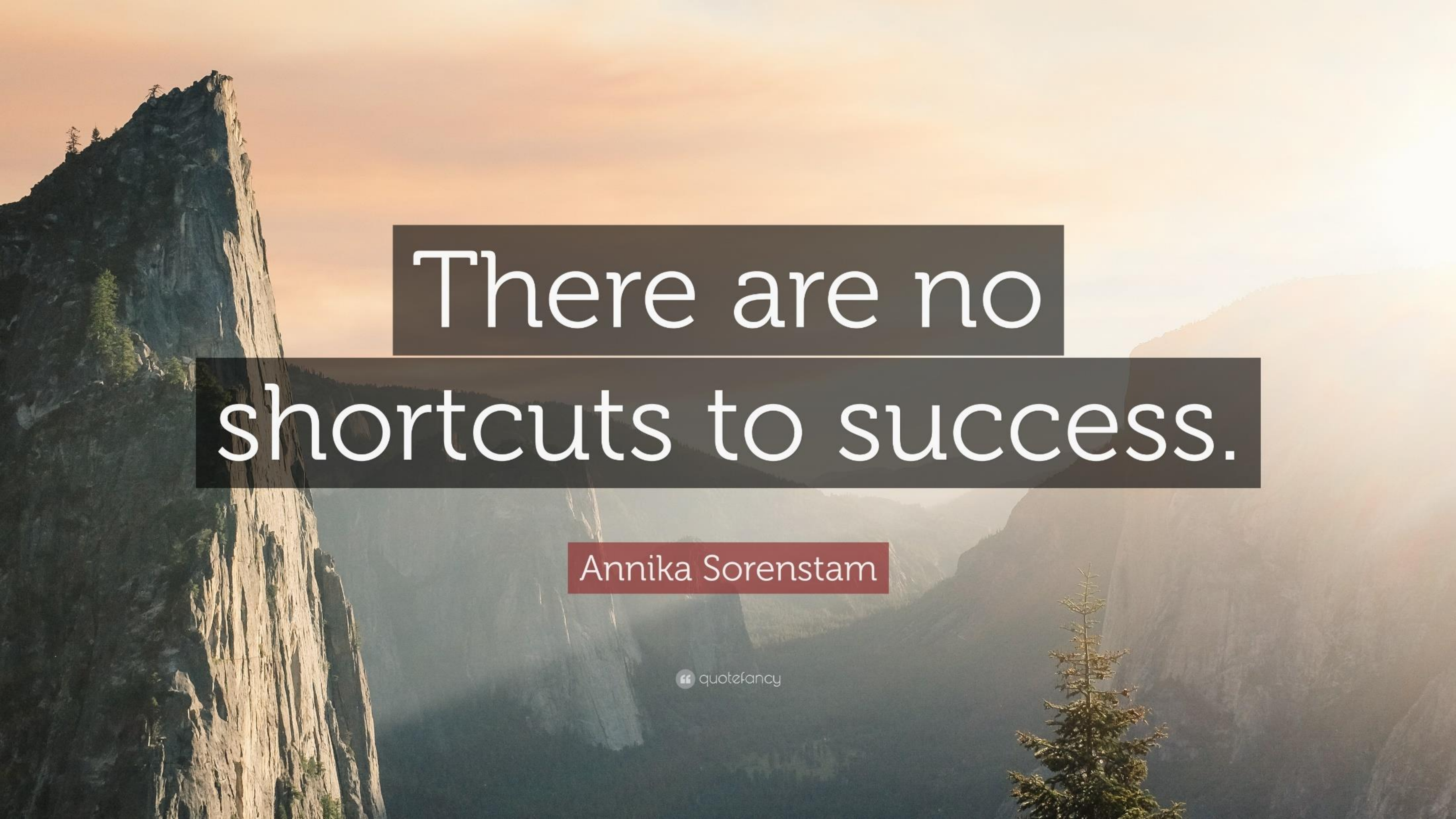
Simply following a recipe

No recipe available

How to become a chef?

Watch cooking shows?

Study recipes?



There are no shortcuts to success.

Annika Sorenstam

PrAcTicE
PRACTICE
practice

The problem is time

Expected hours for each 4MC module?

- 10 hours per week

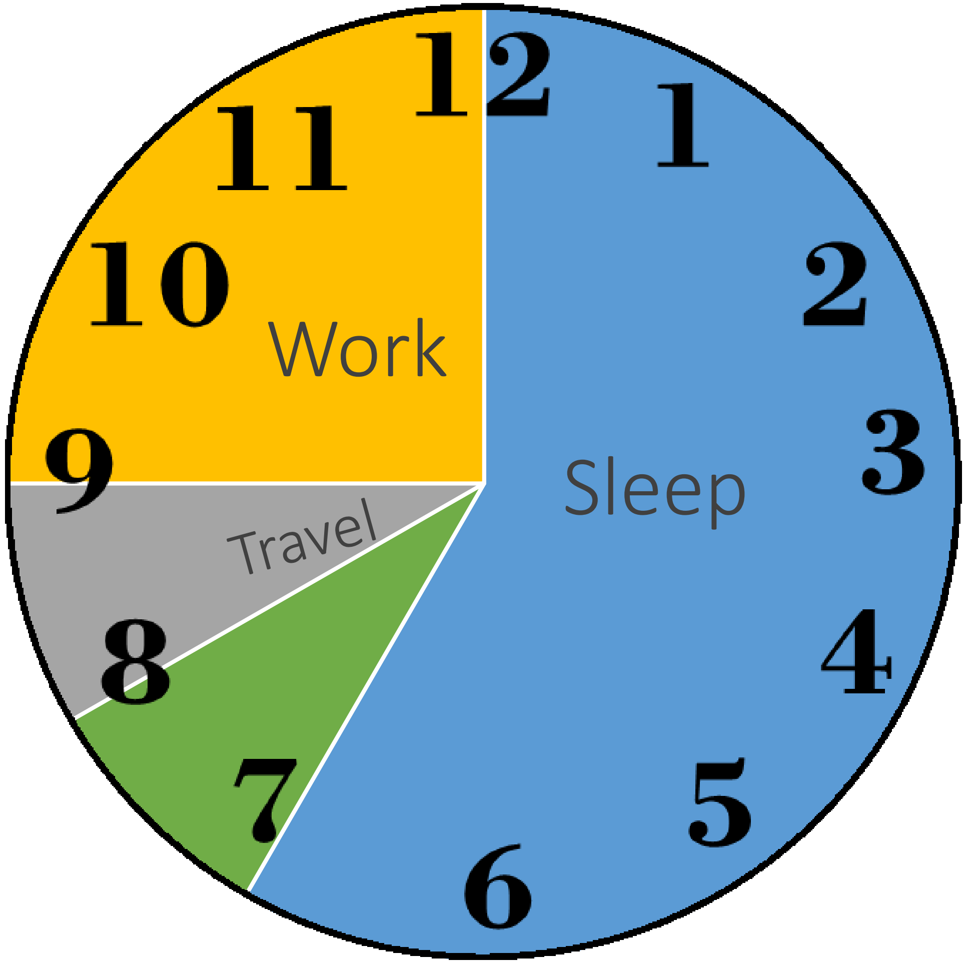
How classes are you taking?

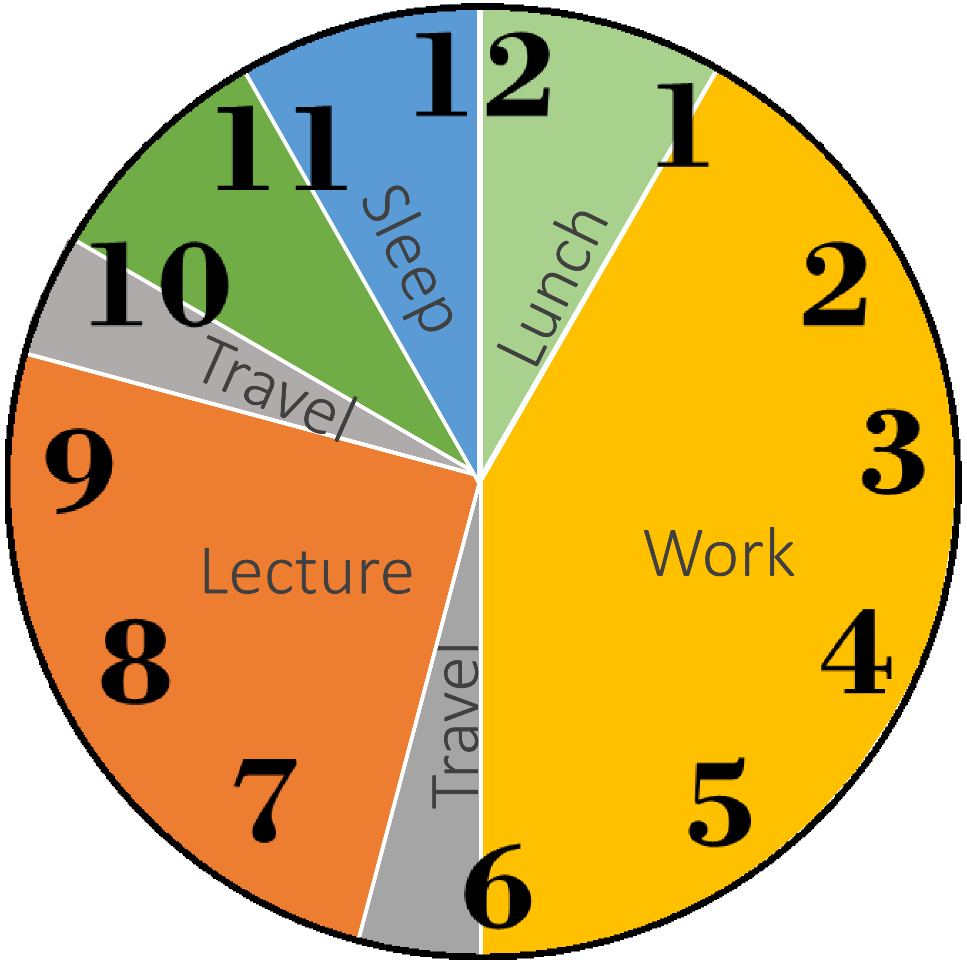
- 3 classes
- 30 hours studying per week

How many hours are you working?

- 40 hours per week?
- 30 + 40 = 70 hours per week

Daily schedule



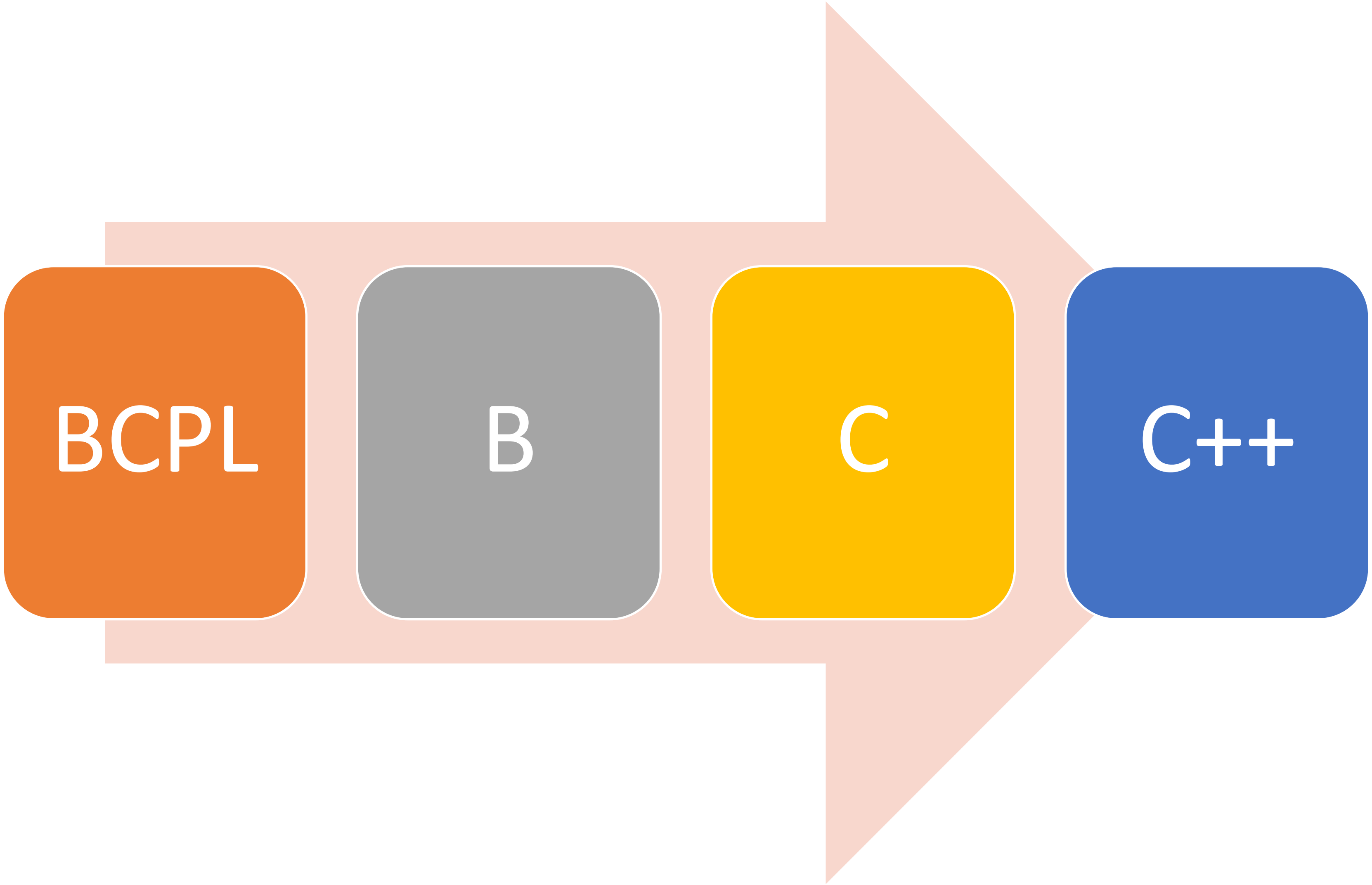


Study Hacks

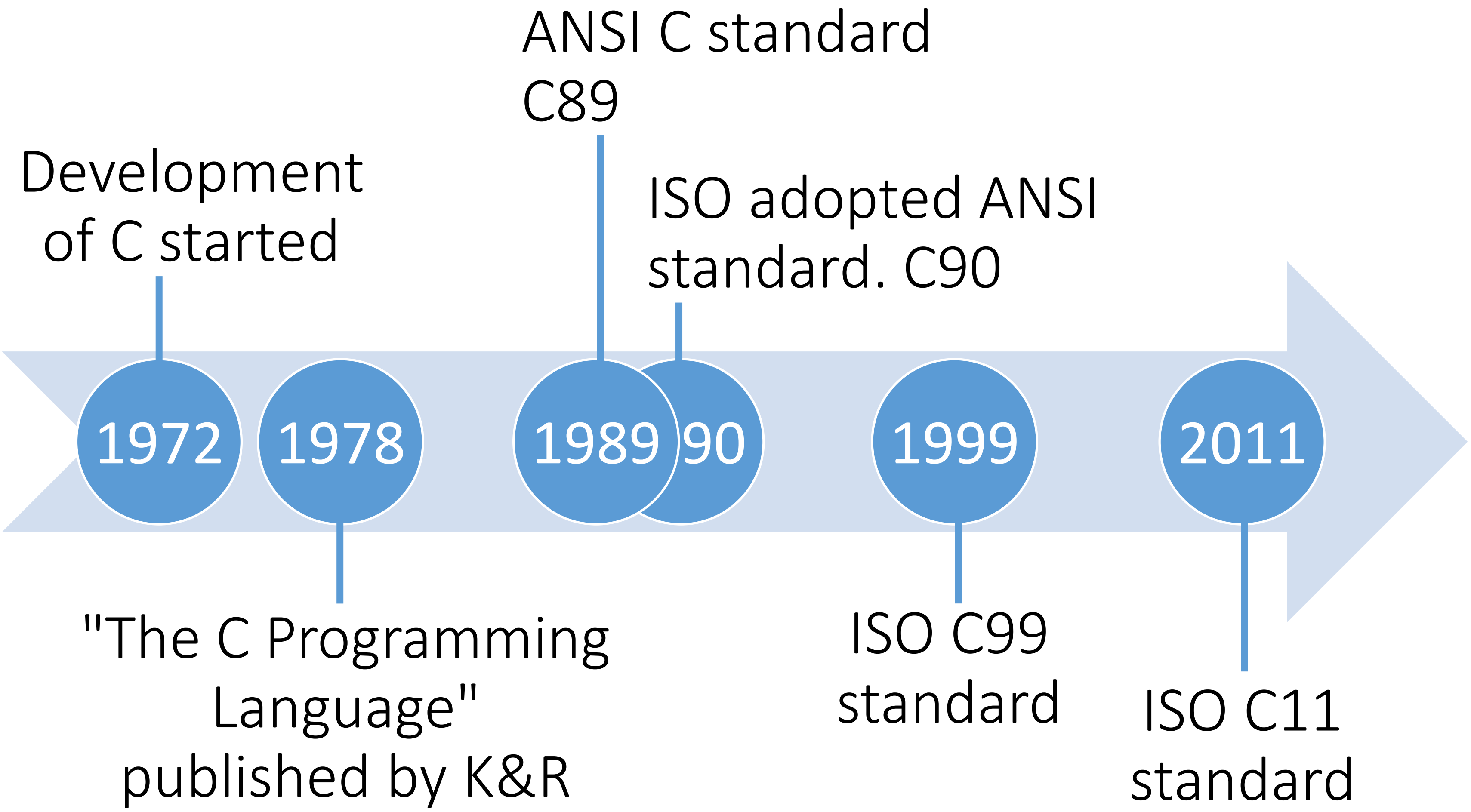
1. Study in short bursts
    - 30 – 50 mins
2. Mental Spacing
    - Spread out learning
3. Teach others (or pretend to)
    - More effective than studying to pass a test
4. Take notes by hand
    - Better retention
5. Sleep and Nap
    - Brain needs rest

History of C

The Genesis of C



Evaluation of C



1972 – Development of C started

1978 – "The C Programming Language" published by K&R

1989 – ANSI C standard C89

90 – ISO adopted ANSI standard. C90

1999 – ISO C99 standard

2011 – ISO C11 standard

Different Standards

The standard players

Compilers

- Turbo C
- Borland C
- GCC (Gnu Complier Collection)
- Clang/LLVM
- MSVC (Microsoft Visual C++)

Each has its favourite default

C89/90

C99

C11

GCC

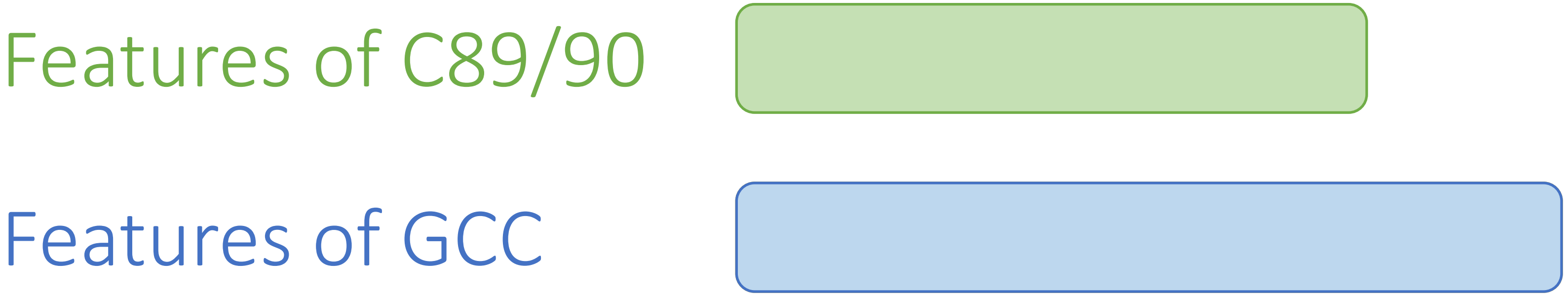
Features of C89/90

GCC

Features of C89/90

Features of GCC

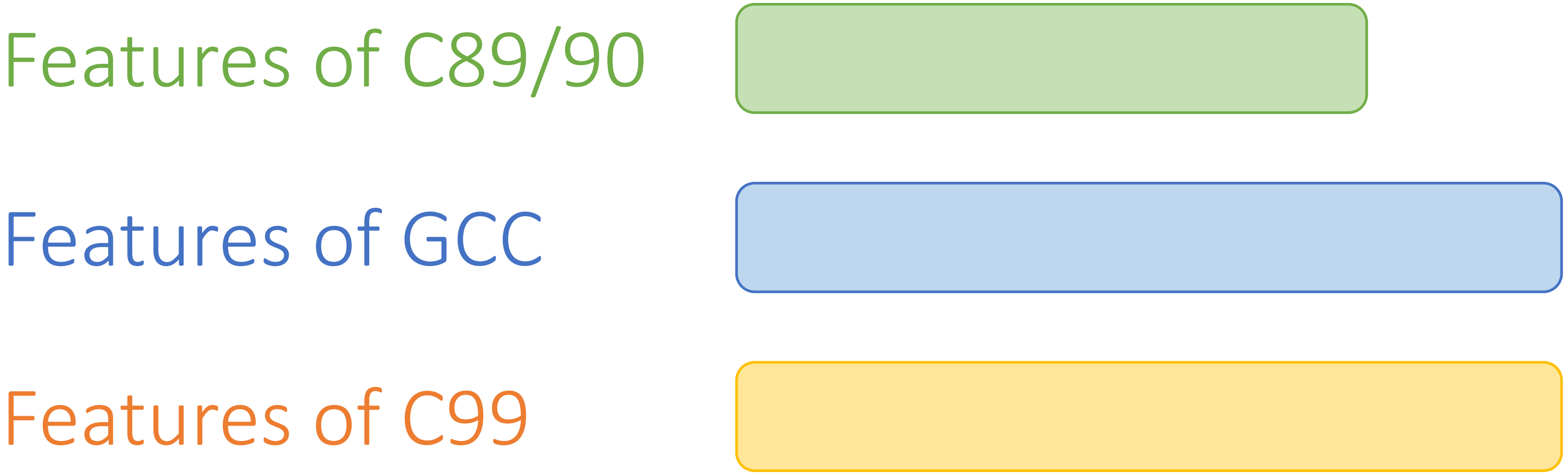
GCC



Features of C89/90

Features of GCC

GCC

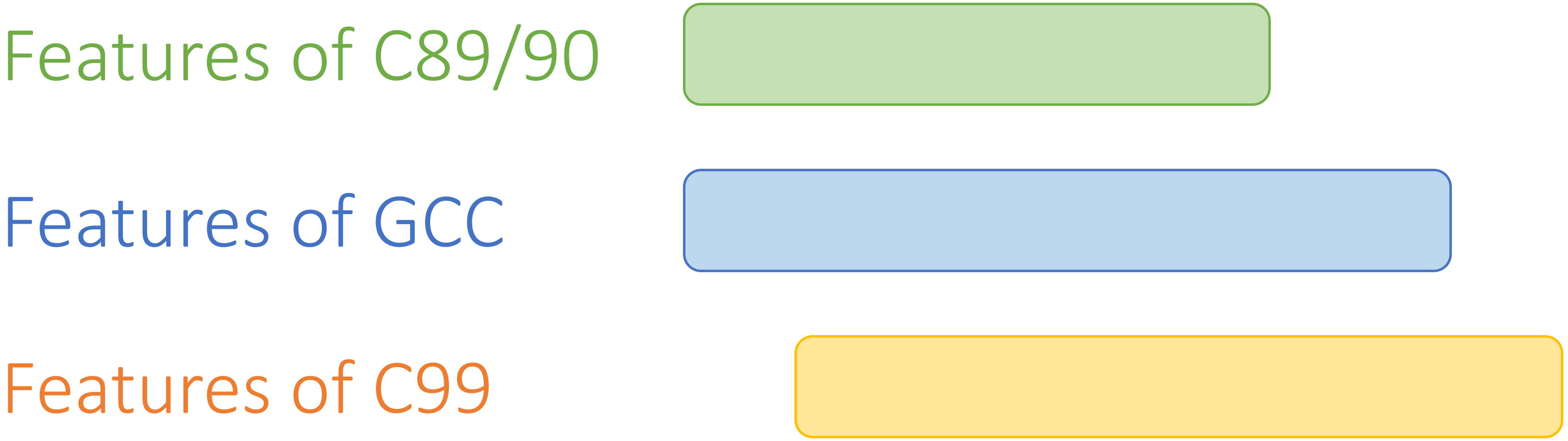


Features of C89/90

Features of GCC

Features of C99

GCC



Features of C89/90

Features of GCC

Features of C99

For our class

We will use GCC with C11 standard

- Mainly for convenience

MinGW compiler on Windows

- Installed in the labs
- For practical exam

On your own

- use whatever compiler you wish
- just note the quirks (unlikely to affect us)

For this week

Problem Set 1

- Due in 2 weeks

Lecture Training

- Bonus cut-off on Monday

Extra Training

- Will be released soon

Next week…

First lab session

- This week?
- Sat 17 Aug
- Installation issues on your laptop
- Simple training exercises

You may bring your laptop to use

- but you might want to familiarize yourself with the lab PC for practical exam.

Questions?

See you next week.