CSE 1061 **Introduction to Computing**
Lecture 1

Fall 2015

Department of Computing
The School of EE & Computing

Adama Science & Technology University

OUTLINE

Goals of the course

What is computation ?

Computational thinking

About Python

2D robot control

Reading assignment:

Chapter 1 of the textbook

Learning programming with robots
(You may download the PDF file on Hisnet site)

GOALS OF THE COURSE

ASTU

Two-level goals

- Building up **a basis on ICT (Information and Communications Technology)**
- **Computational thinking and programming**

(but not learning a programming language **Python**)

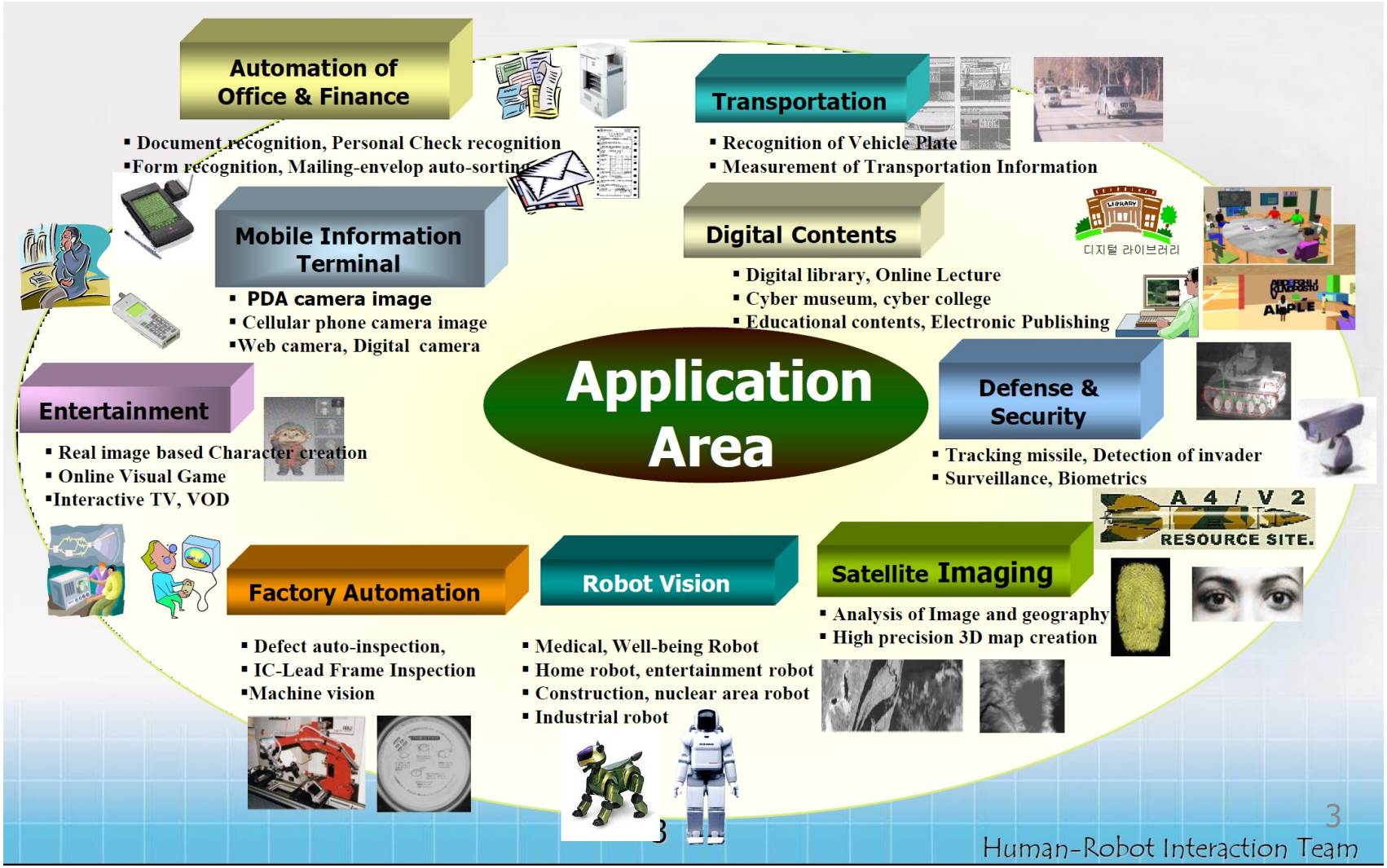
Think like a computer scientist for problem solving !

Add Example of ICT technology scenes !!!

WHAT IS COMPUTATION ?

ASTU

Computing Application 1: Computer Vision Technology



WHAT IS COMPUTATION ?

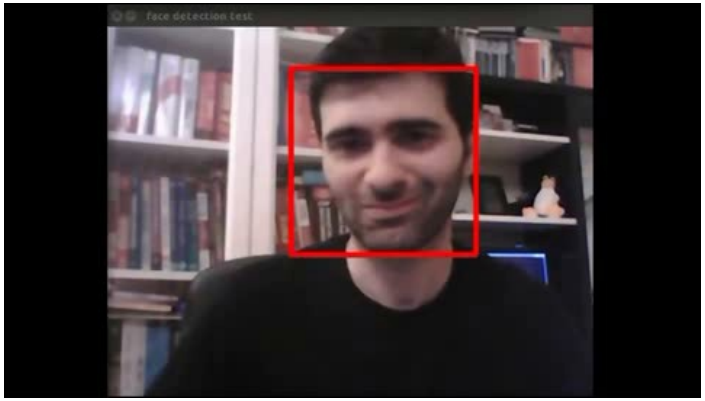
ASTU

Computing Application 2: Human Recognition Demo

WHAT IS COMPUTATION ?

ASTU

Computing Application 3: Face Recognition & Detection Demo



1) Detecting faces using OpenC

2) Detecting faces using OpenCV & OpenCL

from movie “Matrix”

WHAT IS COMPUTATION ?

ASTU

Problem solving with computer

- **Finding the facts** that a solution satisfies
- **Designing an algorithm(recipe) to find the solution**
- **Mapping the algorithm to a program**
- Understanding abilities and limitations

"Algorithm" is at the heart! -

- Efficient good method to solve the problem !

- 1+ 2+ 3+ 4 + 5+ 6 + 7 + 8 + 9 +10 = ?

Method 1: 1+ 2+ 3+ 4 + 5+ 6 + 7 + 8 + 9 + 10 = 55

Method 2: n(n+1)/2 = 10*(10+1)/2 = 10 * 11/2 = 110/2 =55

- It saves computing time..

9 times of computing vesus 3 times of computing

Knowledge

Declarative	Imperative
Statement of facts	Recipes for deducing information "How to" knowledge
$\sqrt{x}$is $\pm$y such that y^2 is x.	Start with guess G. If $G^2 \approx x$, stop and return $\pm G$. Otherwise, G← (G + x/G)/2. Repeat.

Heron of Alexandria(10-70 AD)
Ancient Babylonians

$\sqrt{x}$ is $\pm y$ such that y^2 is x.

Start with guess G.
If $G^2 \approx x$, stop and return $\pm G$.
Otherwise, $G \leftarrow (G + x/G)/2$.
Repeat.

Problem is "finding y value"

----->G2 = x**

19.99878

Guess G,

ex) x= 20, G=4

round	G		G**2	≈	x	status
1	4		4**2=16		20	fail
2	4.5	=(4+20/4)/2	4.5**2=20.25		20	fail
3	4.47222	=(4.5+20/4.5)/2	4.472**2= 19.9988		20	TRUE
	G is	4.472	**return with G = 4.472**			

Fixed program computers

ASTU

- **Atanasoff and Berry(1941**): **a linear equation solver**
  - The **Atanasoff–Berry computer** (**ABC**) was the first automatic electronic digital computer,
  - an early electronic digital computing device

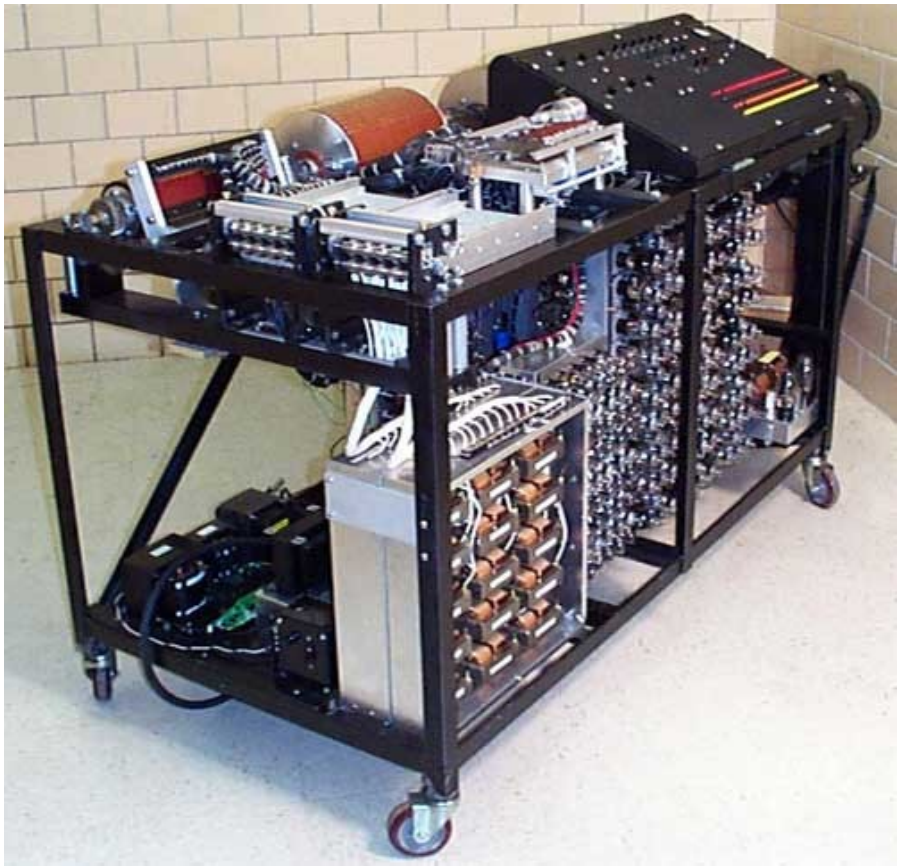
- **Alan Turing: bombe machine**

- **Calculators**

Fixed program computers

ASTU

Atanasoff and Berry(1941): **a linear equation solver**



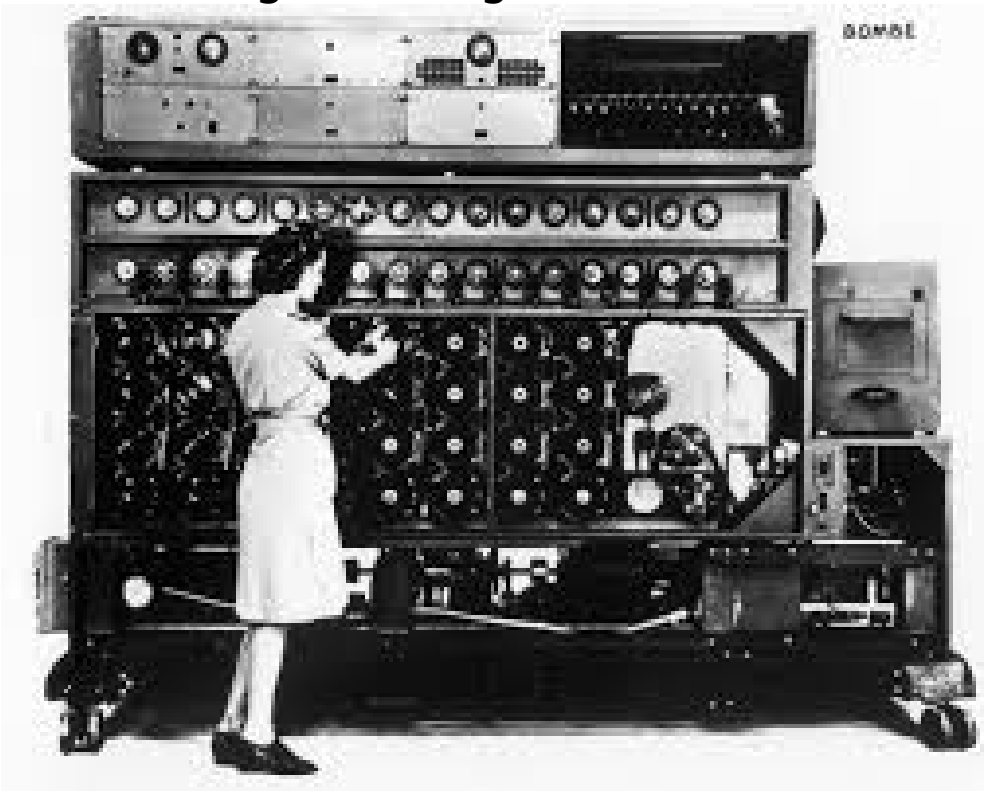
- The **Atanasoff–Berry computer** (**ABC**) was the first automatic electronic digital computer,
- An early electronic digital computing device

Fixed program computers

ASTU

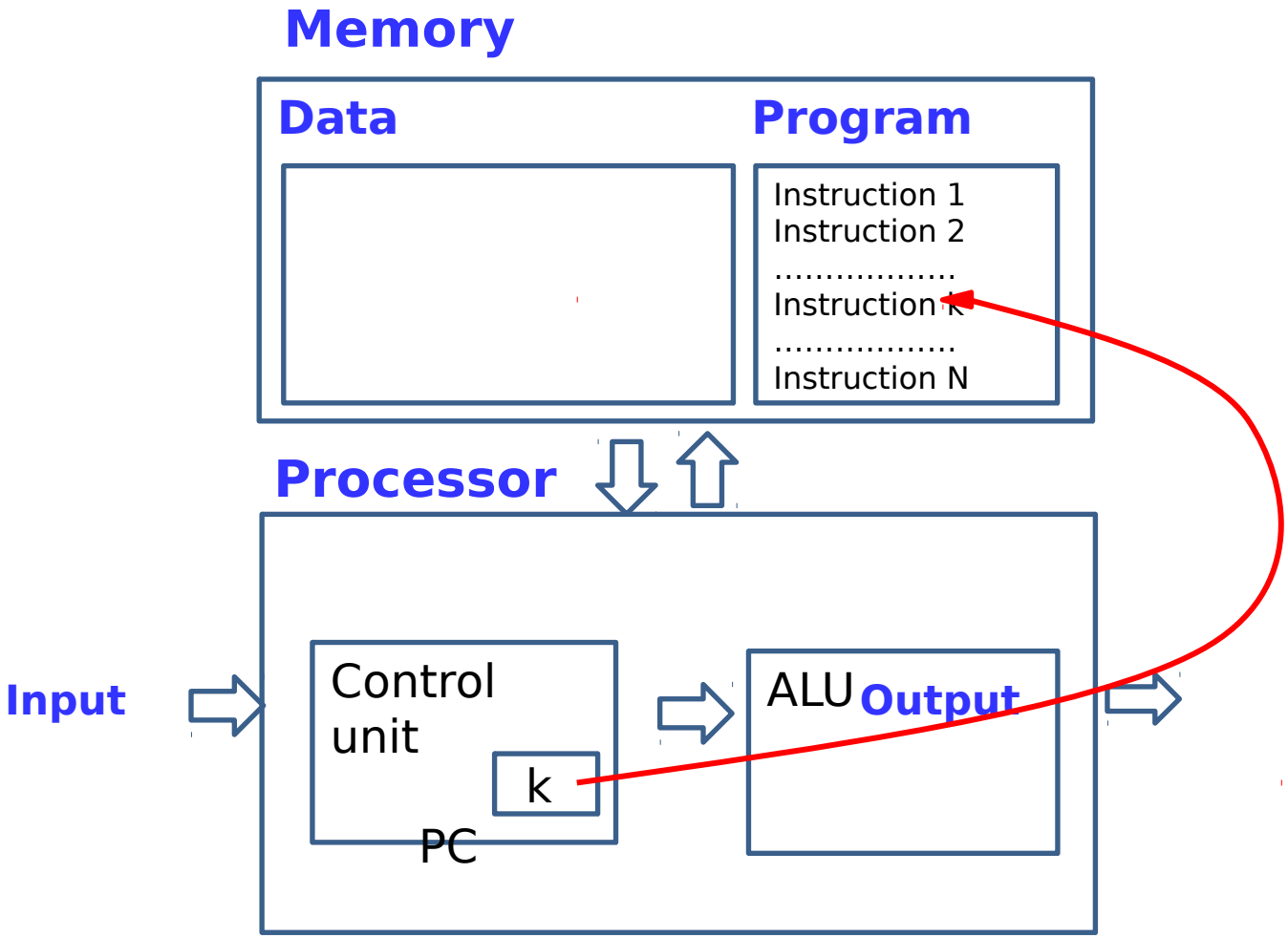
Alan Turing: **bombe machine**

- The **bombe** was an electromechanical device used by British cryptologists to help decipher German Enigma-machine-encrypted se cret messages during World War II



Stored program computers

(* Human brain cells: + 100 bilion neurons*)



Computation

Computation is **solving** a problem with a **program**.

A **program** is a **realization** of an **algorithm**(recipe)
on a **computer**.

An **algorithm** is a **sequence** of **instructions** to do a task.

imperative knowledge

(for humans)

An **algorithm** should be **refined** enough to be **easily translated** into a **programming language.**

(for computers)

COMPUTATIONAL THINKING

How to design an algorithm : **top-down design**

How to convert it to a program: **coding** and **debuggi ng**

What to do with **computers** ?

Top-down design

Decomposing a problem into smaller sub-problems

Decompose each of the smaller sub-problems **recursively** until every sub-problem is simple enough to map to a few instructions in a program language

Multi-level abstraction
Divide and conquer

Coding and debugging

Coding is "a process of fighting with bugs (errors)."

- **Syntax error**: Python cannot understand your program, and refuses to execute it.
- **Runtime error**: At runtime, your program suddenly terminates with an error message.
- **Semantic error:** Your program runs without error messages, but does not do what it is supposed to do.

Why making **such bugs (errors) ?**

- Well, … , that is the **difference** between **humans** and **computers**.

What to do with computers?

According to **Turing-Church Thesis (in 1936),**
modern computers are essentially equivalent to
a **stored program computer.**

What kind of problems can we solve with a stored program machine ?

Decidable problems

Tractable problems : good algorithms

Intractable problems: no good algorithms

e.g., travelling salesman's problem

approximate algorithms

Undecidable problems: no algorithms ever found

e.g. halting problem

ABOUT PYTHON

ASTU

Low level Lang. vs **High level Lang.**

General vs Targeted

Compiled vs **Interpreted**

Python is relative **young** but one of the most **popul ar** programming languages

Open software

ABOUT PYTHON

ASTU

- **Guido van Rossum,** Python's principal author
- Employed by Google during 2005 ~ 2012, where developing the Python language.
- 2001 Award for the Advancement of Free Software
- Open Source
- multi-paradigm programming language:
  - object-oriented programming and
  - structured programming

- Since 2003, Python has consistently ranked in the top 10 most popular programming languages
- As of September 2015, it is in the fifth position.
- It was ranked **as Programming Language of the Year** for the year 2007 and 2010

Why Python ?

A programming language easy to learn and very powerful

- Used in **many universities** for introductory courses
- Main language used for **web programming** at **Google**
- Widely used in **scientific computation**, e.g., at **NASA**
- Large portions of **games** written in Python (**Civilization IV)**

Once you learnt programming in one language,

it is relatively easy to learn another language, such as C++ or Java.

Characteristics of Python

Instruction set

Arithmetic and logical operations
- +, -, *, /, and **
- and, or, not

Assignment

Conditionals

Iterations

Input/output

} for defining expressions

No pointers

No declarations

Why programming ?

Every scientist and **engineer must know** some programming.

It is part of basic education, like calculus, linear algebra, introductory physics an d chemistry, or English.

Alan Perlis 1961

After half a century later, we should change it as follows:

Every student in a **university should learn** some programming.

It is part of basic education, like calculus, linear algebra, introductory physics an d chemistry, or English.

2D ROBOT CONTROL

ASTU

A small grid-like 2D world

Basic actions

- move (): moving one grid forward
- turn_left (): turning left by 90°
- pick_beeper(): pick ing up beepers
- drop_beeper(): putting down beepers

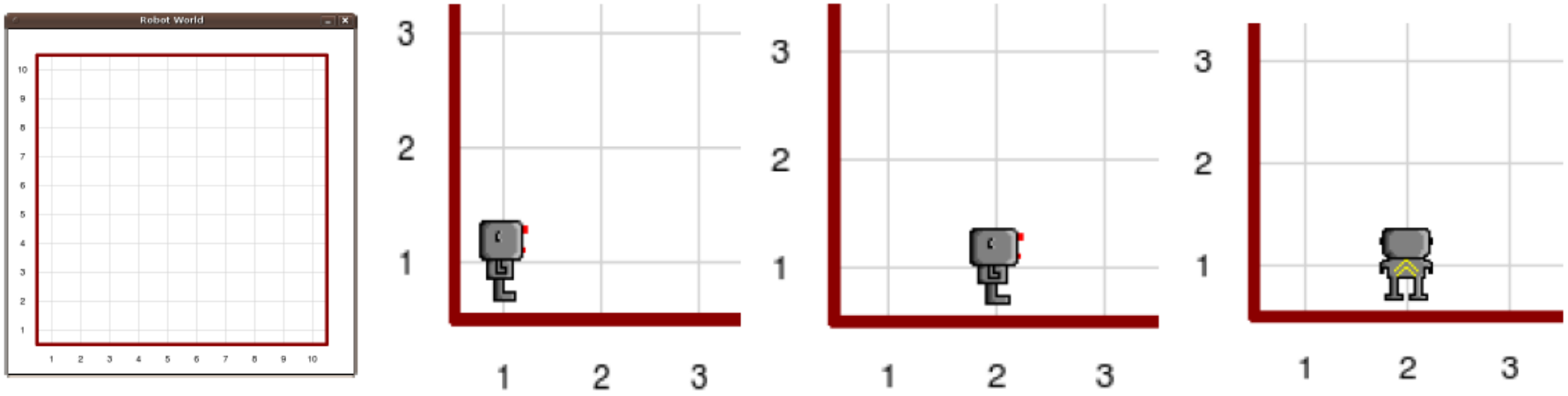
Our own instructions: functions

Comments

Interactive mode

Python programs (scripts)

Interative mode




```
>>>from cs1robots import *
>>>create_world()
                >>>hubo = Robot()
                              >>>hubo.move()
                                            >>>hubo.left_turn()
```

Script mode

```
from cs1robots import *
create_world()
hubo = Robot()
hubo.move()
hubo.turn_left()
```

Functions

A **function definition** specifies the **name** of a function and the **sequence of statements** that are executed when the function is called.

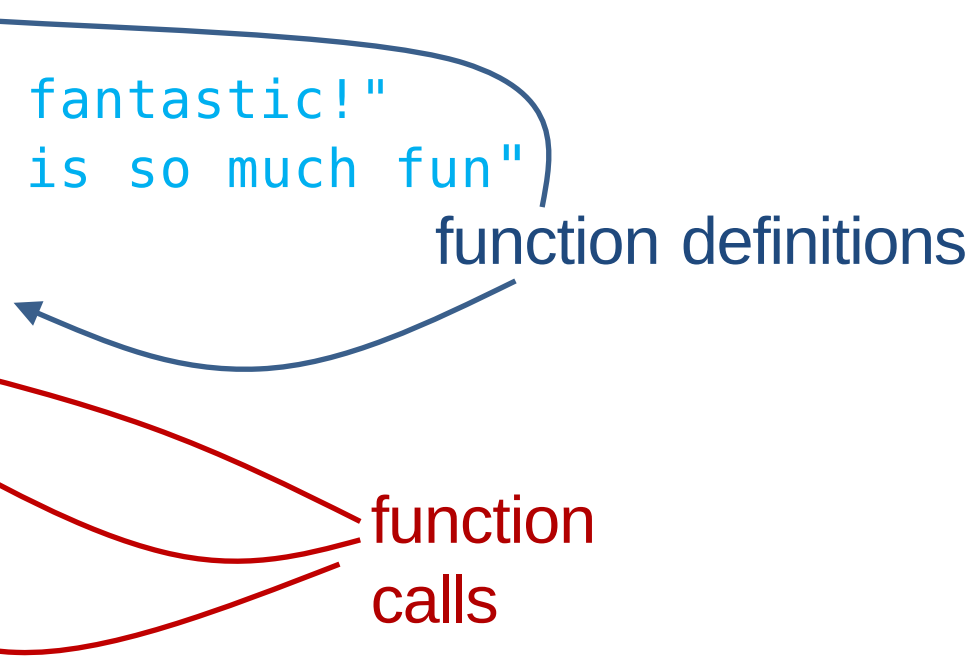
```
def print_message():
  print "CCE20003 is fantastic!"
  print "Programming is fun!"
```

You can call a function inside another function:

```
def repeat_message():

  print_message()

  print_message()
```

Flow of execution

```
def print_message():
  print "CCE20003 is fantastic!"
  print "Programming is so much fun"

def
  repeat_message():
   print_message()
   print_message()
repeat_message(
)
print ‘Done’
```



Execution begins at the first statement. Statements are executed
one by one, top to bottom.

Function definitions do not change the flow of execution but only define a function.

Function calls are like detours in the flow of execution.

Comments

```
# create a robot with one beeper
hubo = Robot(beepers = 1)

# move one step forward
hubo.move()

# turn left 90 degrees
hubo.turn_left()
```

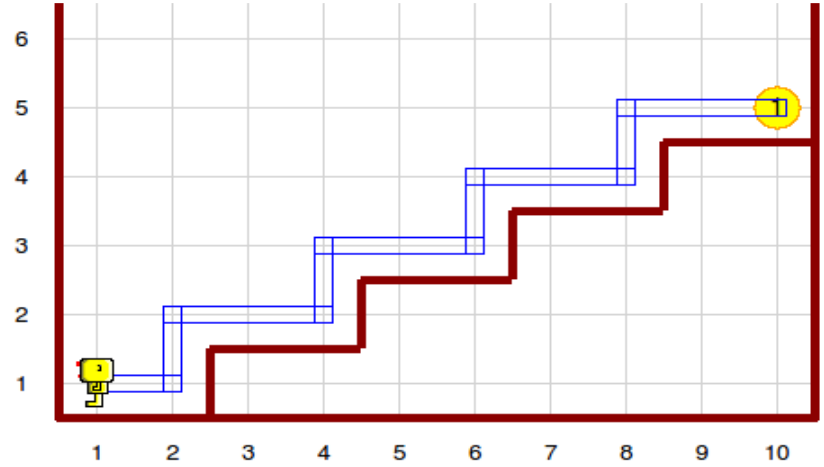
Turning right

Define a function!

```
def turn_right():
    hubo.turn_left()
    hubo.turn_left()
    hubo.turn_left()
```

Newspaper delivery

Hubo should climb the stairs to the front door, drop a newspaper there, and return to his starting point.



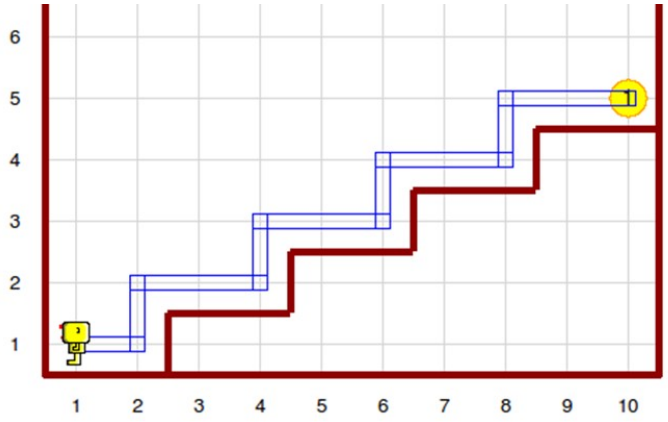
Algorithm(pseudo code):

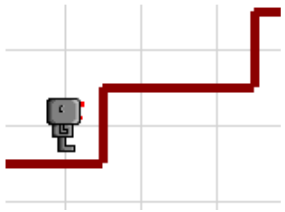
Climb up four stairs
Drop the newspaper
Turn around
Climb down four stairs

Python version:

```
climb_up_four_stairs()
hubo.drop_beeper()
turn_around()
climb_down_four_stairs()
```

Climbing up stairs





```
def climb_up_four_stairs():
    climb_up_one_stair()
    climb_up_one_stair()
    climb_up_one_stair()
def climb_up_one_stair()
    climb_up_one_stair():
    hubo.turn_left()
    hubo.move()
    turn_right()
    hubo.move()
    hubo.move()
def turn_around():
    hubo.turn_left()
    hubo.turn_left()
```

Iteration: for-loops

We should **avoid writing** the same code **repeatedly.**
A for-loop allows us to write it more elegantly:

```
def climb_up_four_stairs():
  climb_up_one_stair()
  climb_up_one_stair()
  climb_up_one_stair()
  climb_up_one_stair()

def climb_up_four_stairs():
  for i in range(4):
    climb_up_one_stair()
```

To repeat the same instruction 4 times:

```
for i in range(4):
    print "CCE20003 is fantastic!"
```

for-loop

Don’t forget the indentation!

What is the difference between the following two programs?

```
for i in range(4):
    print "CCE20003 is great!"
    print "I love programming!"
```

```
for i in range(4):
    print "CCE20003 is great!"
print "I love programming!"
```