# CSIT121
# Object-Oriented Design and Programming

Dr. Fenghui Ren

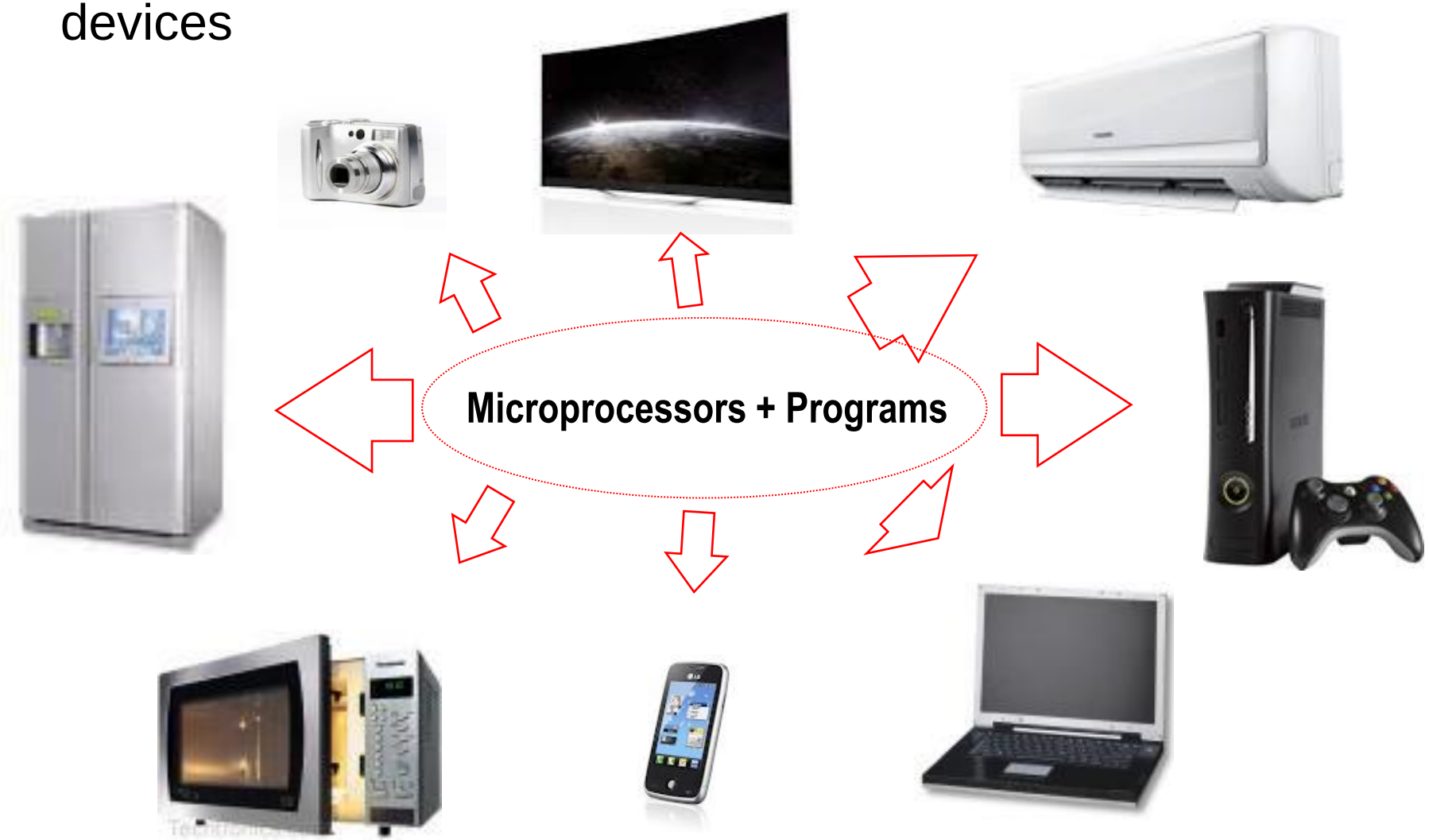
School of Computing and Information Technology
University of Wollongong

# Lecture 1 outline

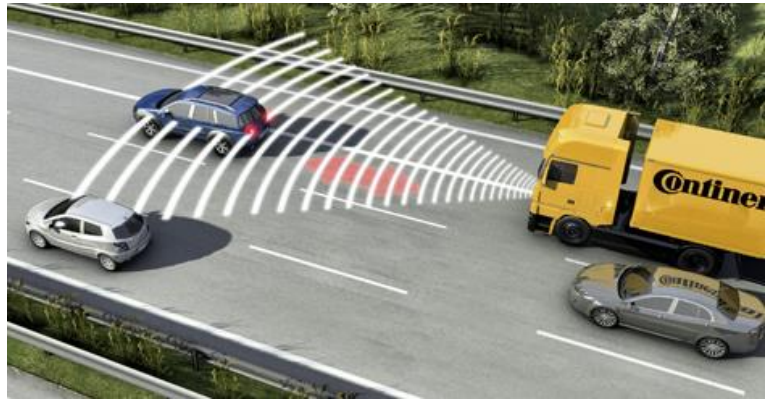
- Computer programs
- Programming languages
- What object-oriented means
- Object-oriented analysis (OOA)
- Object-oriented design (OOD)
- Object-oriented programming (OOP)
- Python IDLE
- Your first object-oriented program in Python

# Digital World

In modern life, we are surrounded by digital electronic devices



# Digital World



Automatic parking
Smart cruise control

Self driving cars

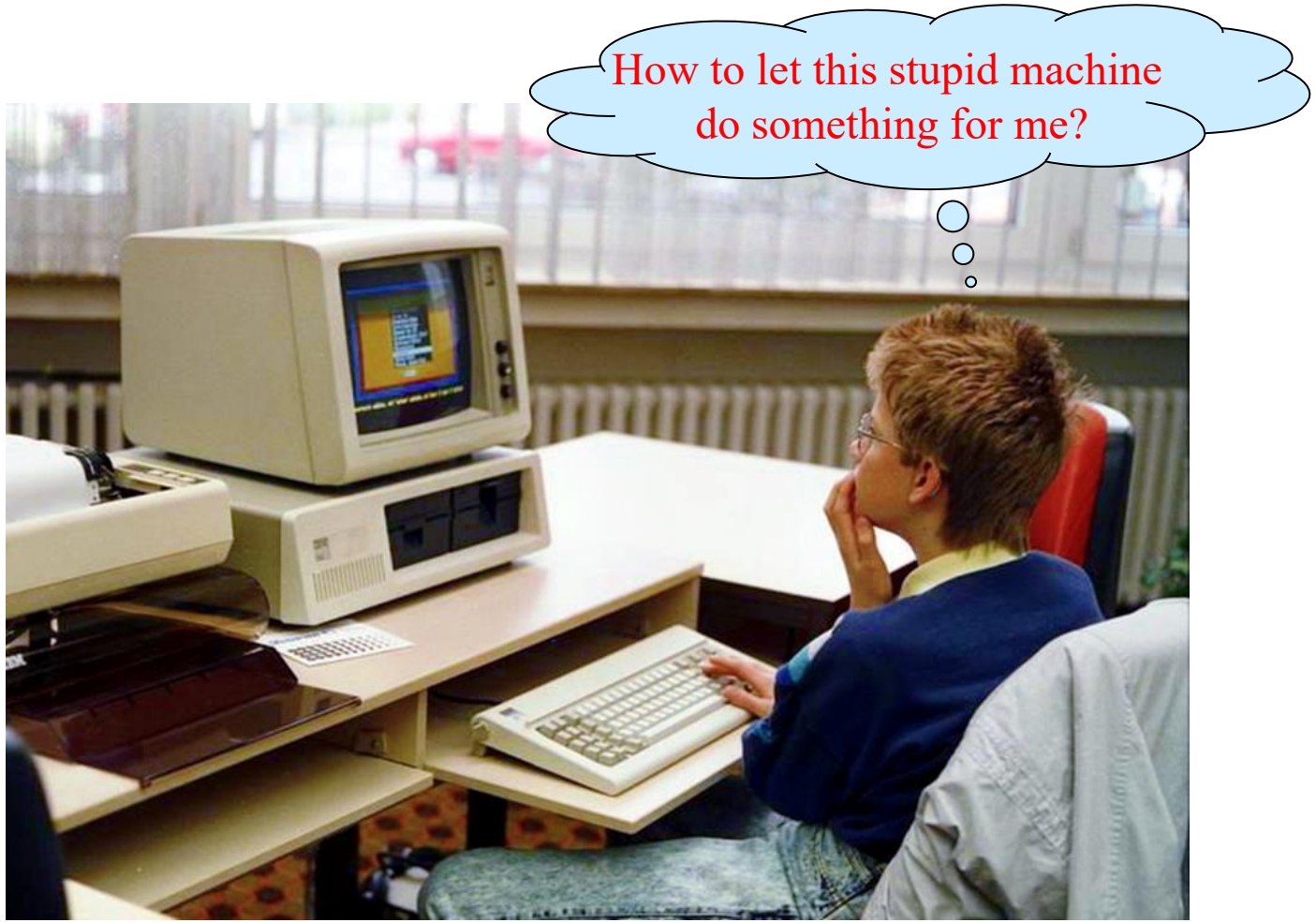
Automated assembly lines

Home robots

Military robots

Computers are incredibly *fast*, *accurate* and *stupid*.
Human beings are incredibly *slow*, *inaccurate* and *brilliant*.
**Together they are powerful beyond imagination**.

*Albert Einstein?… Leo Cherne?... Stuart Walesh?*



# Personal Computer




1) Scanner
2) CPU (Microprocessor)
3) Memory (RAM)
4) Expansion cards (graphics cards, etc.)
5) Power supply
6) Optical disc drive
7) Storage (Hard disk or SSD)
8) Motherboard
9) Speakers
10) Monitor
11) System software
12) Application software
13) Keyboard
14) Mouse
15) External hard disk
16) Printer

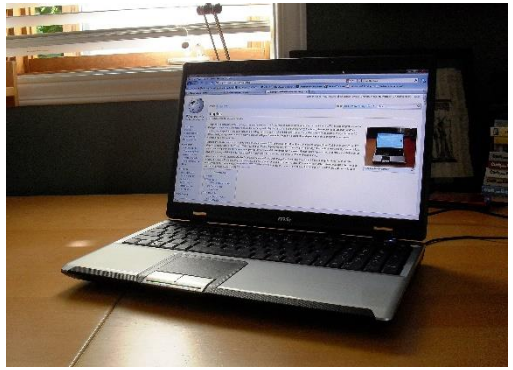
One of the most important components is missing here

# Let a computer do something!

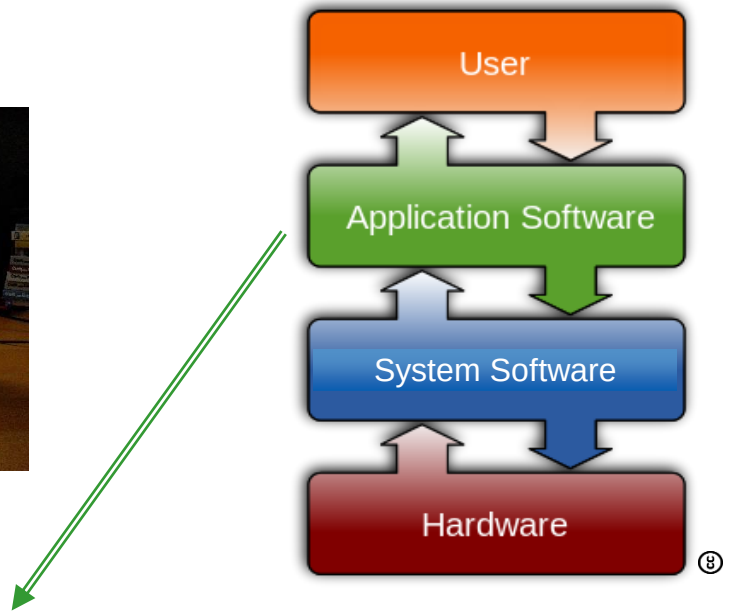
Computers don't do anything without someone telling them what to do

- How to let a computer to do something?
  - Instruct it
- How to instruct a computer to do something?
  - Use a language to instruct it
- How to use a language to instruct a computer to do something?
  - Write a sequence of instructions in a language – a program
- How to write a program?
  - Learn programming – **this subject !**

# How computers work



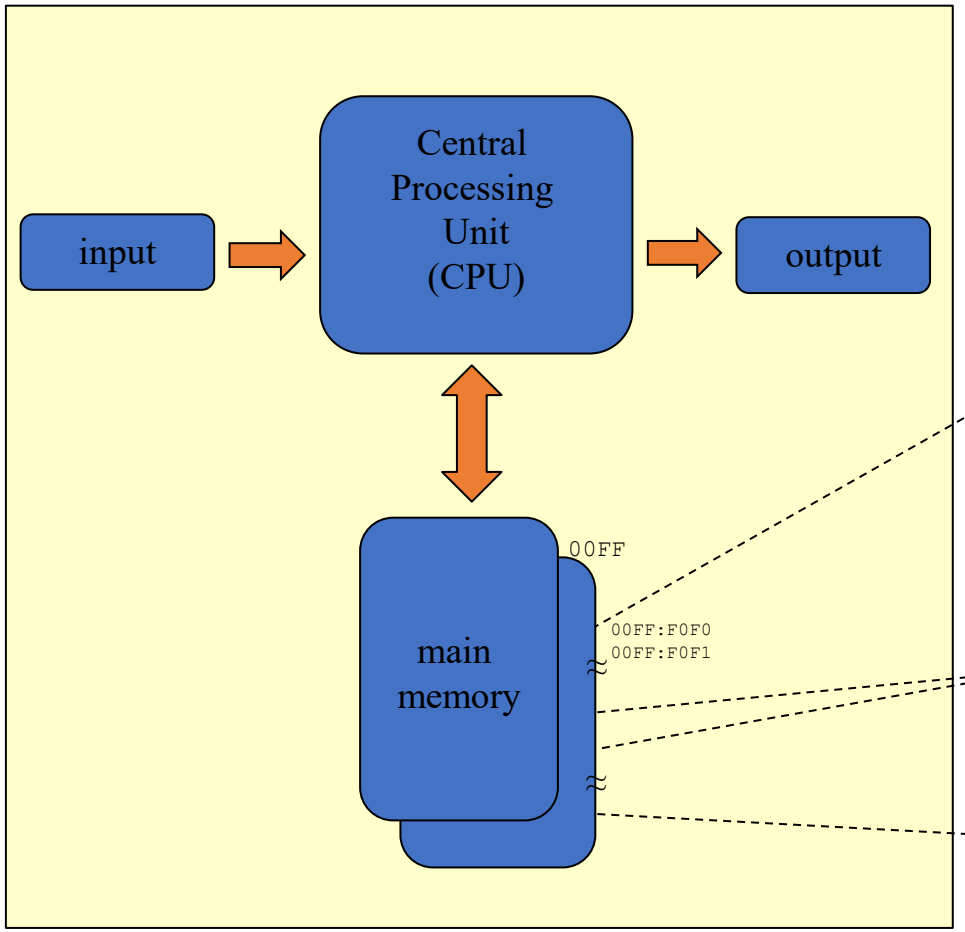





The focus of CSIT121

```
/**
 * The HelloWorldApp class implements an
 *   application that displays "Hello object world!"
 *   to the standard output.
 */
class HelloWorldApp:
  def greet(self):
    // Display "Hello world!"
    print("Hello object world!")
greeter = HellowWorldAPP()
greeter.greet()
```

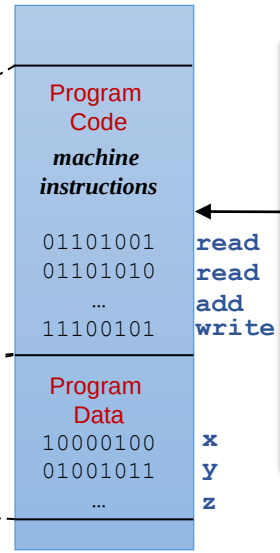
# How computer programs work




x86 instructions

| | |
|---|---|
| ADD | add |
| SUB | substract |
| MUL | multiply |
| CMP | compare |
| INC | increment |
| MOV | move data |
| … | |

program in memory




Task:

$z = x + y$

CPU instructions

```
Read location x
Read location y
Add
Write to location z
```

*The instruction binaries are not real and for illustration only*

# Evolution of Programming Languages

Machine languages

1950

```
10001101
00111100
00101101
11000101
```

Symbolic languages

1950

```
mov al,40
add al,dl
cmp cl,dl
call RESET
```

High-level languages

1960

```
weight = mass * 9.8;
if(weight > MAX_WEIGHT)
  status = sendWarning();
```

- 1989 – C
- 1991 – Python
- 1995 – Java
- 1998 – C++

# Task: “Hello, World!”

- How do people say hello to the world?

Hello
G’day
你好
Bonjour
مرحبا
こんにちは
여보세요

- How to instruct your computer to say hello to the world ?

# “Hello World” programs

```
.model small
.stack 100h

.data
msg db 'Hello world!$'

.code
start:
    mov ah, 09h
    lea dx, msg
    int 21h
    mov ax, 4C00h
    int 21h
end start
```

Intel x86 instruction set for 16-bit DOS

```
section .data
str:     db 'Hello world!', 0Ah
str_len: equ $ - str
section .text
global _start
_start:
 mov eax, 4
 mov ebx, 1
 mov ecx, str
 mov edx, str_len
 int 80h
 mov eax, 1
 mov ebx, 0
 int 80h
```

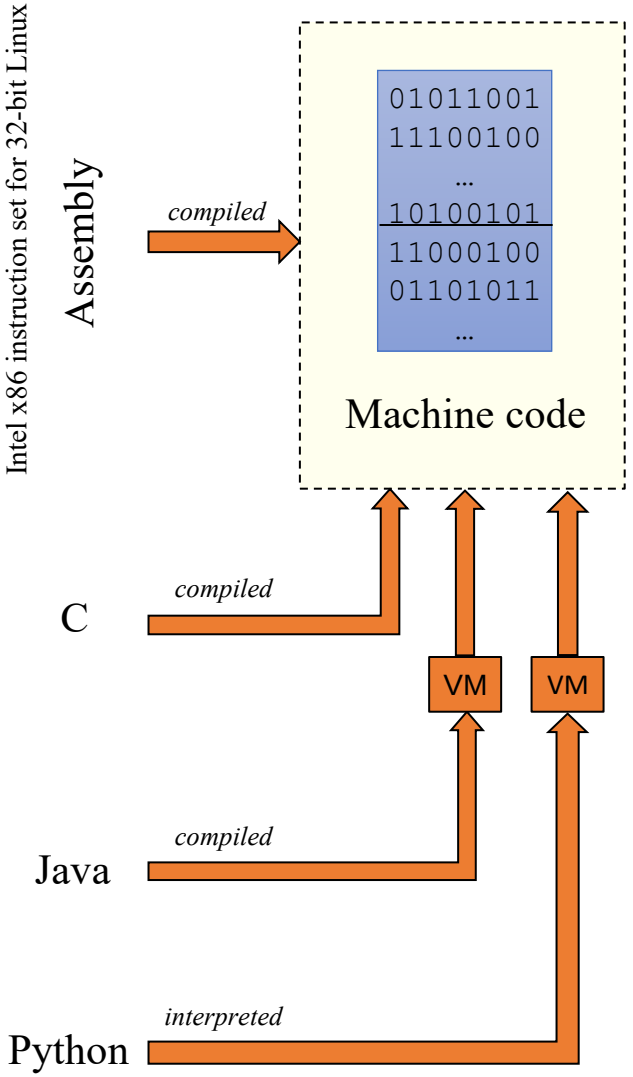
Intel x86 instruction set for 32-bit Linux

Levels of programming languages: *Low* → *High*

```
#include <stdio.h>
int main() {
    printf("Hello World!");
}
```

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

```
print "Hello World!"
```



# Programming languages

*Anything that can be done using one language can be done using any language.*
*Some language may be easier for certain things*

- Compiled languages
  - Compiled to machine code
    - Architecture-dependant, high performance
    - **`Assembly, C, C++`**
  - Compiled to bytecode
    - Architecture-neutral (running in a virtual machine)
    - **`Java`**
- Interpreted (scripting) languages
  - Programming languages without explicit compilation, interpreted at run-time
    - **`JavaScript, PHP, Perl, shell`**
    - **`Python`** is used without compilation, but is compiled to bytecode on-the-fly and running in a virtual machine
  - Simple and less lines of code, less access to computer native resources, slower execution

*The differences are becoming fewer*

# Programming

- Programming is a problem-solving activity
  - A program tells the computer how to solve a specific problem
  - A problem can be broken down into a set of sub-problems
    - There are many ways how a problem can be subdivided into sub-problems
    - The subdivision affects the program implementation
- Programming is not very difficult, but time-consuming
  - The most challenging part about programming is to match subdivision of the problem with the program design methodology
  - Computers cannot guess what problem you are solving. They simply follow your program instructions even though they may be wrong
  - A lot of time will be spent to figure out why the computer does not do what you expect it to do - debugging

Remember? *Brilliant Humans beings are incredibly slow and inaccurate*

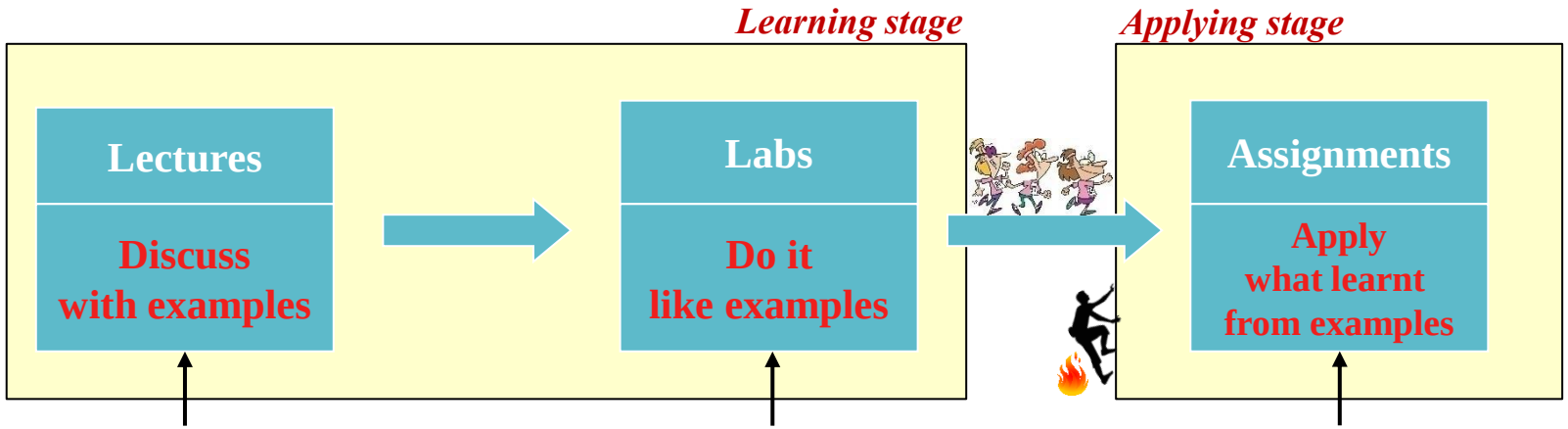
# The subject

- Objective
  - Learn the Object-oriented view of problem analysis and solving
- Learning Outcomes
  1. Effectively design and implement object-oriented programs in an integrated development environment;
  2. Demonstrate an understanding and appreciation of the concepts of a well-structured solution and good coding style within an object-oriented programming environment;
  3. Create correct and maintainable object-oriented programs using an object-oriented programming language;
  4. Apply the principles of reuse in software design and implementation

This subject is not just about Python, but you will learn programming in Python

# Learning programming

- Learn by ***doing***
  - ***If you want to learn programming, you must "do" programming***
  - ***Spend sufficient time doing it***




- You should listen to what the lecturer says about what on the presentation slides;
- Slides are not for you to read - books are

- You must do exercises if you want to learn programming.
- If you do not spend sufficient time to actually write actual code, you are not learning programming.

- When you are ready, you will find assignments not very difficult to complete

# Objects

- Physical objects: a tangible thing that we can sense, feel, and manipulate

- Software objects: models of something (class) that can do certain things (behaviors) and have certain things (data) done to them

- A software object is a collection of data and associated behaviors.

# Object-oriented analysis, design & programming

- Object-oriented means functionally directed toward modeling objects.

- Terms:
  - object-oriented analysis (OOA)
  - object-oriented design (OOD)
  - object-oriented programming (OOP)

# Object-oriented analysis, design & programming

- object-oriented analysis (OOA): looking at a problem, system or task and identifying the objects and interactions between those objects. **WHAT**

- object-oriented design (OOD): converting requirements into an implementation specification. Name the objects, define the behaviors, and specify object interaction. **HOW**

- object-oriented programming (OOP): converting the design into a working program that does exactly what the customer wants

# Object-oriented analysis

- Features of objects: names, attributes, behaviors, relationships, etc
- Objects are differentiated based by the values/instances of these features (**Data & Functions**)
- If objects share the same **structure** of features. We define the object's structure as a **class**.
- Classes are blueprints for creating objects. They descript the general attributes and behaviors of objects.
- But classes do not contain Data and can't be used directly.
- Objects are the instances of classes. Objects contain data and can be used.
- OOA needs to consider the features of objects belonging to the same class

# Object-oriented analysis

- An object-oriented program may contain multiple objects, which are instances of multiple classes.
- OOA needs to consider the structure and relationships of multiple classes in a program
- How to organise these classes?
  - OOD patterns
- How to describe the hieratical relationships of classes within a class family?
  - Single inheritance
  - Multiple inheritances
  - Abstract class
  - Polymorphism
- How to describe the horizontal relationships between classes not in the same family?
  - Association
  - Multiplicity
  - Aggregation
  - Composition

# Object-oriented analysis

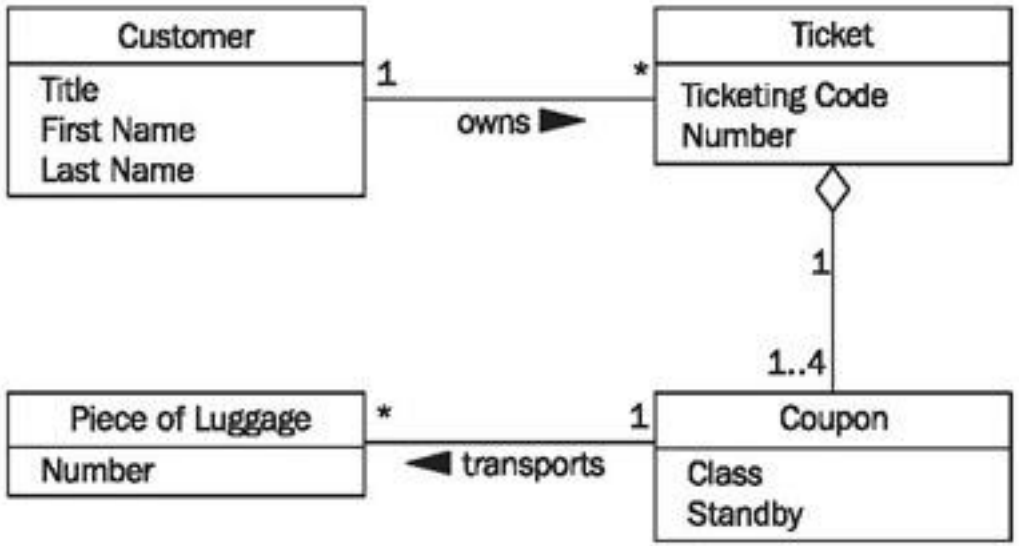
A **user** of eBay might need to do

- *Review* his/her purchase/sell **historical records**
- *Post* a new **advertisement**
- *Browse, compare and order* **items**
- …

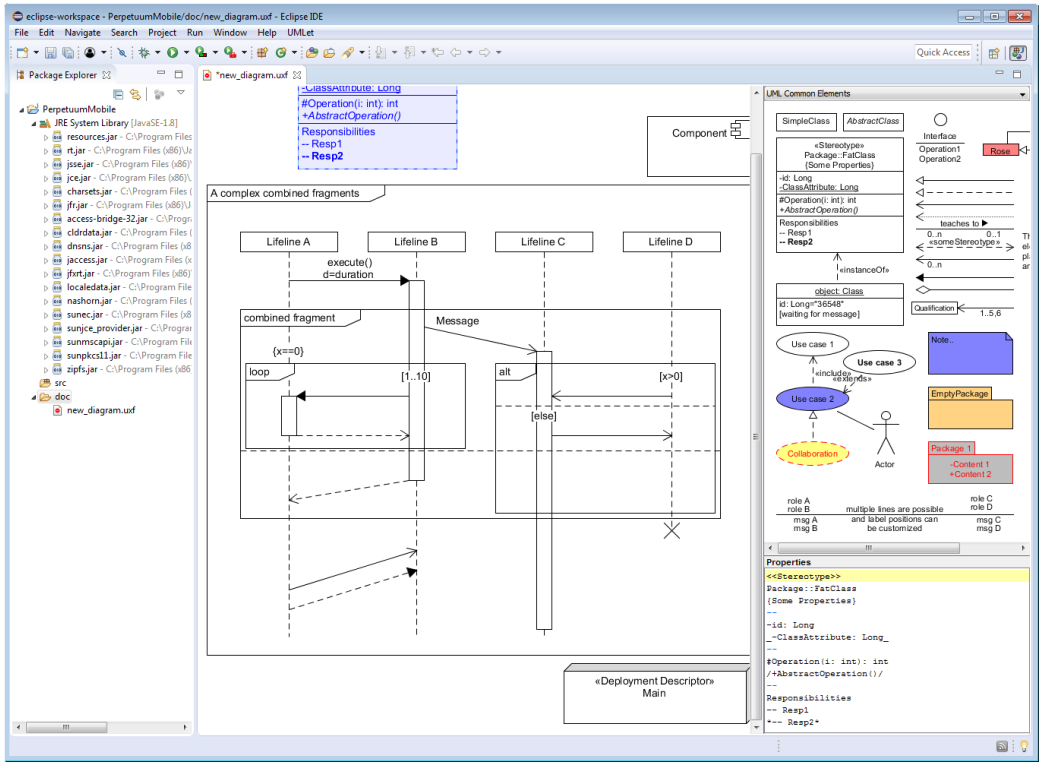
# Object-oriented design

- Formally specify the structures and relationships of classes
- Unified Modeling Language (UML)
  - Class diagram
  - Sequence diagram
  - UMLet (https://www.umlet.com/)
- The iterative design and development model will be used
  - Initialising class design using the class diagram
  - Evaluating the class design in scenarios/tasks using the sequence diagram
  - Implementing the design and testing the program
  - Updating the class design
  - Repeat the above steps till a satisfactory evaluation results

# Object-oriented design




UML class diagram



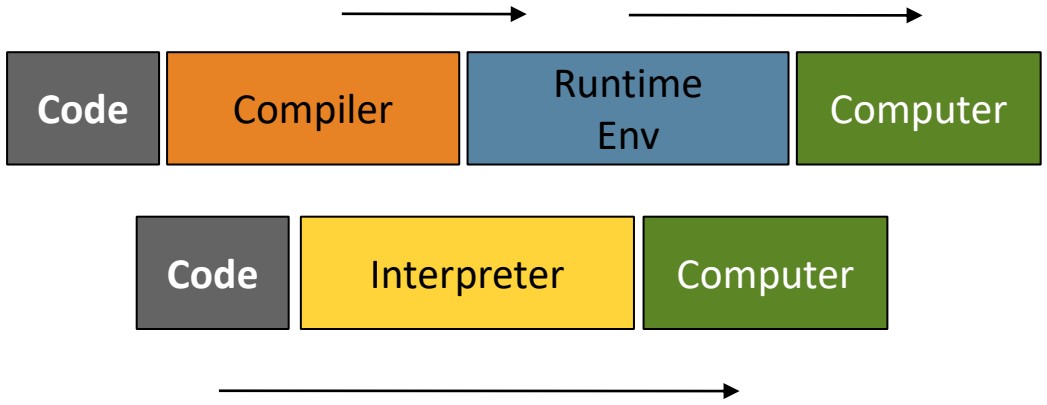

UML sequence diagram

# Object-oriented programing

- Using a particular object-oriented programming language to implement the design
  - Python
  - Java
  - C++
- Following the language syntax and specification
- Compile and execute the program
- Test the program
  - Unit test
  - Integration test
- This may involve the modification of existing designs and re-evaluation.

# Python

- interpreted
- targeted towards short to medium sized projects
- useful as a scripting language




Guido van Rossum

A Dutch programmer, the creator of Python

# Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an **object-oriented way** or a functional way.

# Python

Windows

1) Download Python from python.org.
2) Run 'python' using the run command.

-or-

Run Idle from the Start Menu.

Mac OSX

1) Python is already installed.
2) Open a terminal and run python or run Idle from finder.

Linux

1) Chances are you already have Python installed. To check run python from the terminal.
2) If not, install python through your distribution's package system.

# Python

Unlike in Java, in Python whitespace (think tabs and spaces) matters.  Instead of using braces ({}) to designate code blocks, Python uses the indentation. This was done to promote readable code. In Java you may or may not indent and it will work. In Python you **must** indent.

**Hello.java**

```
1  public class Hello {
2          public static void main(String[] args){
3                  System.out.println();
4          }
5  }
```

**hello.py**

```
1  def hello():
2          print "Hello \"world\"!"
3
4  def does_nothing():
5          pass
6
7  hello()
```

# Python

While commenting is flexible it is good practice to comment code with **#** comments before code in question.

Comment methods with a doc string (""") on the first line in a method that way it will be used in **help**().

**comments.py**

```
1  # This is a file full of comments and code.
2
3  def hello():
4      """print 'Hello world'
5      this comment can be multi-line"""
6      print "Hello world" # this is in-line
7  hello()
```

# Python

Python can read the function comments and make them available when you need them. To access these comments call help(<function>).

**comments.py**

```
1  # This is a file full of comments and code.
2
3  def hello():
4      """prints 'Hello world'
5      this comment can be multi-line"""
6      print "Hello world" # this is in-line
7  hello()
```

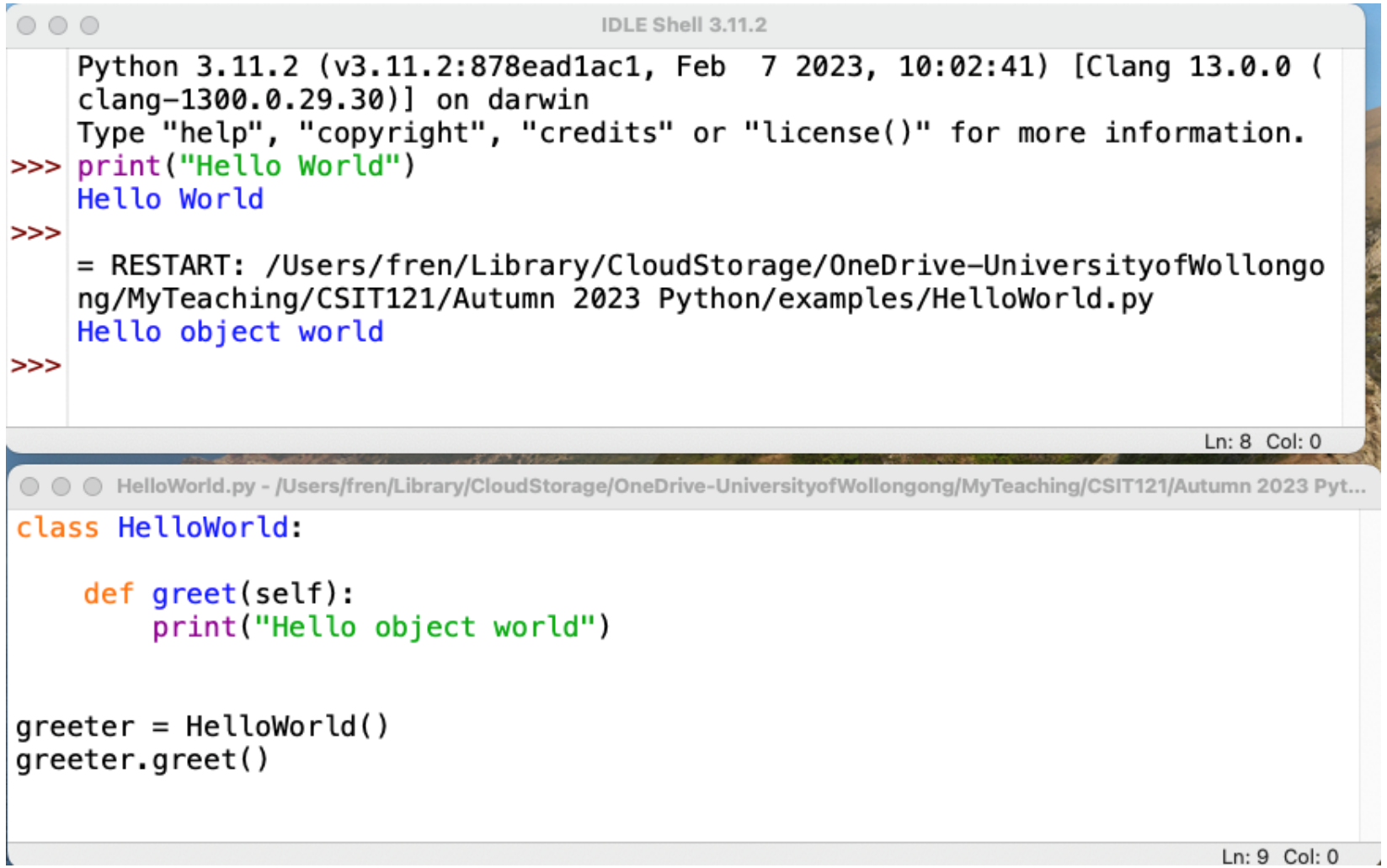
```
>>> hello()
Hello world
>>> help(hello)
Help on function hello in module __main__:

hello()
    prints 'Hello world'
    this comment can be multi-line
```

# Python – IDLE

- IDLE is an Integrated Development Environment for Python
- Multi-window text editor with syntax highlighting, auto-completion, smart indent and other.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persistent breakpoints, and call stack visibility

# Python IDE – IDLE

- https://www.python.org



# Suggested reading

Python 3 Object-Oriented Programming

- Preface
- Chapter 1: Object-Oriented Design

Python

- https://www.python.org/