Object-oriended programming is a programming paradigm based on object

An object consists of attributes(data) and methods(operations on data)

With this paradigm, a computer program is made out of objects that interact with each other

OOP is

Putting objects at the center of program development, a part of recipe

Not regarding them only as a container for data and operations

<-> procedural programming : logic at the center, functions or sub-routines

OOP is

Define the objects to deal with in a program and how these are related to each other

Flesh them out with logic to make the program work

OOP’s Principle

* Encapsulation (information hiding) : Packing data(attributed) and operations(method) on data in a container called an object such that data are transparent to users
* Abstraction : Suppressing details for conceptual clarity
* Inheritance : Inheriting the data (attributes) and operations on data(methods) from the ancestors of a class

Abstraction achieved through encapsulation

캡슐화 ; 객체지향프로그래밍에서 중요한 특징 중 하나로 연관된 데이터와 함수를 논리적으로 묶어 놓은 것이며, 데이터를 보호하기 위해 다른 객체의 접근을 제한하는 접근 제한 수식자의 기능을 제공

Class – cake recipe, Object – cake, Attributes(instance variable) – ingredients, Method – instruments

main에서가 아니라 class 내에서 변수 정의 – Class variable

class 내에서 수정들이 이루어져도 절대 main은 건들어지지 않는다. 과제 때 해봐서 체감 올 듯

Inheritance(상속) : 새로운 클래스가 기존의 클래스의 자료와 연산을 이용할 수 있게 하는 기능이다. 상속을 통해서 기존의 클래스를 상속받은 하위 클래스를 이용해 프로그램의 요구에 맞추어 클래스를 수정할 수 있고 클래스 간의 종속 관계를 형성함으로서 객체를 조직화할 수 있다.

type(hubo)

: <class ‘cs1robots.Robot’ >

An object has its type and value. The type of an object determines the kinds of operations applied to it.

The attributes of an object are its data elements. Their values represent the state of the object (Class variable는 공유)

The methods of an object are obtained from a class or its ancestors through instantiation

Simple GUI – Flow based programming

Wait()

Event = canvas.wait()

getDescription()– 어떤 Event?

getTrigger() – 어디로부터 유래?

getKey() – 어떤 키를 눌렀는가

getMouseLocation() – 마우스 위치는?

