06-1 클래스

- 클래스란?
  - ✓ 객체를 정의해 놓은 것
  - ✓ 객체를 생성하는데 사용한다
  - ✓ 반복하여 사용하는 변수 & 함수를 미리 정해 놓은 틀

    ex) 회사에서 문서작업 시 공용으로 쓰는 양식, 설계도
  - ✓ 클래스가 설계도라면 객체는 설계도를 통해 만들어진 제품

06-1 클래스

- 클래스가 왜 필요한가?
  - ✓ 고객사의 회계를 처리해주는 프로그램이 있다.
  - ✓ A사, B사, C사 3개의 고객사를 담당하며
  - ✓ 하나의 프로그램으로 돌려야 한다면...

06-1 클래스

- 클래스가 왜 필요한가?

  ✓ 이를 해결하기 위해 클래스가 등장.

  ✓ 흔히들 클래스를 붕어빵 틀이라 비유하고

  ✓ 클래스는 틀, 인스턴스(객체)는 붕어빵이라 비유한다.

06-1 클래스

✓ 아래 그림처럼 붕어빵은 공통된 모양은 갖지만

✓ 팥이 들어 갈수도, 슈크림이 들어 갈수도 있다.

06-1 클래스

- 클래스의 기본구조

  ✓ class 클래스명 : <<일반적으로 대문자로 시작

  변수명 = ? << 변수 선언

  def 함수명 (매개변수) : << 클래스 안의 함수는 메소드

  실행문

06-1 클래스

✓ class Calculator :

```
def __init__(self) :                << 생성자 선언
    self.result = 0
def numSum(self, a) :               << 함수 선언
    self.result += a
    return self.result              << 결과값 반환
```

06-1 클래스

- __init__ 이란 무엇인가?
  - ✓ 생성자
  - ✓ 클래스가 인스턴스(객체)화 될 때 가장 먼저 실행되는 함수

06-1 클래스

- self란 무엇인가?
  - ✓ self란 붕어빵 그 자신, 즉 인스턴스를 의미

- self를 왜 쓰는가?
  - ✓ 어떠한 객체의 것인지 인지하기 위해
  - ✓ Heap 메모리에 저장해 두기 위해
  - ✓ 즉, 변수가 휘발되지 않도록 하기 위해

06-1 클래스

▪ 클래스 연습1

- ✓ 아래와 같이 인스턴스를 선언해 add, sub, mul, div를 실행하면 결과값을 리턴하는 사칙연산 클래스를 만들어보자.
- ✓ class1 = Calculator()
- ✓ class1.setdata(10, 20)
- ✓ result = class1.add()
- ✓ print(result)

06-1 클래스

- 클래스 연습2
  - ✓ Calculator라는 클래스, def __init__(self)함수를 통해 result값을 0으로 초기화 하고, result값에 누적해서 더하는 sum(self, startNum)함수 만들어라.
  - ✓ sumA라는 Calculator의 인스턴스를 만들고, ~까지 더할 숫자 하나를 입력받아 해당 객체의 result에 1~10까지 더하는 while문을 실행하라.
  - ✓ 그리고 결과값을 출력하라.
  - ✓ 단, sum함수는 리턴값이 없는 함수이다.

06-1 클래스

- 클래스 연습3
  - ✓ Ractangle 클래스를 구현하라
  - ✓ 생성자 : width, height 변수 초기화
  - ✓ area 메소드 : 가로 * 세로
  - ✓ circum 메소드 : (가로 + 세로) * 2

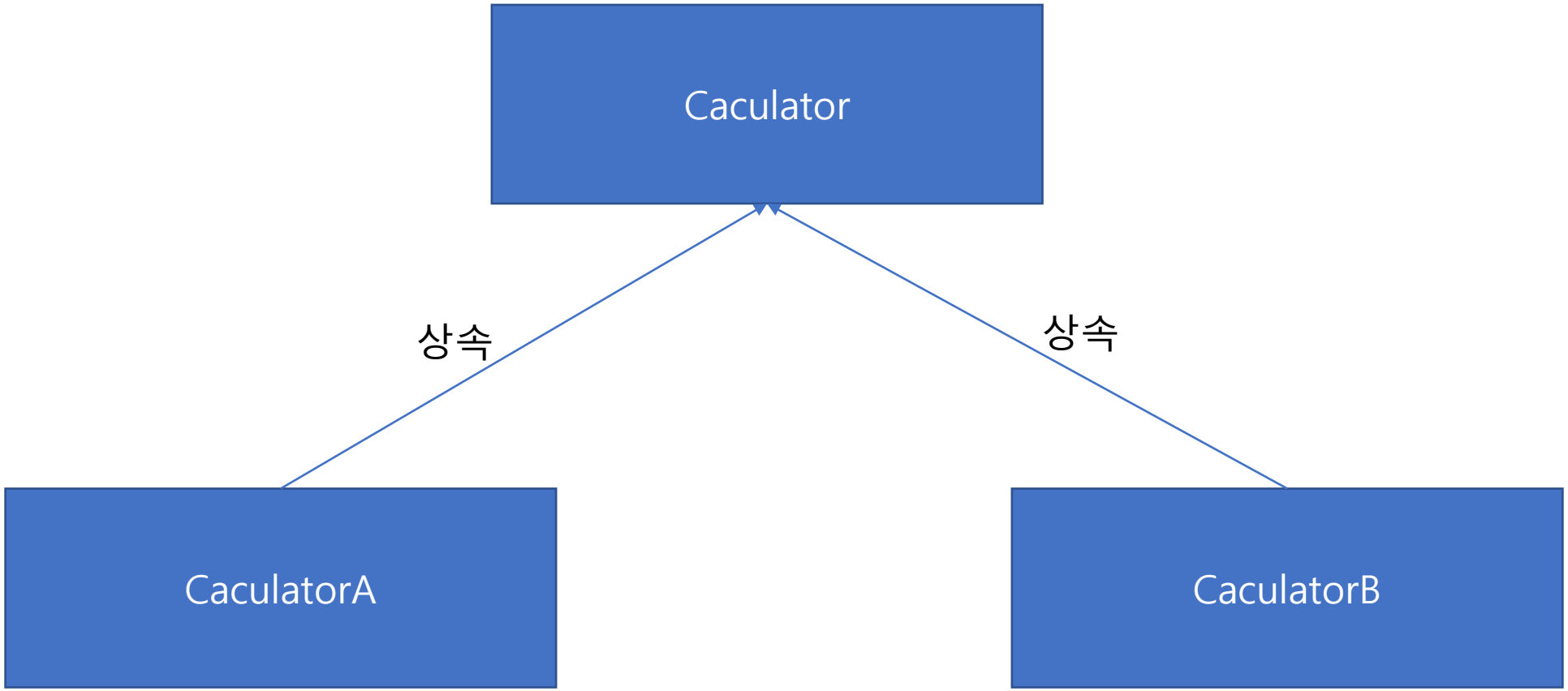
  [결과화면]
  사각형의 넓이와 둘레를 계산하라.
  가로 : 3
  가로 : 4
  ==========
  넓이 12
  둘레 : 14
  ==========

06-1 클래스

- 클래스의 상속

06-1 클래스

- 클래스의 상속



06-1 클래스

- 클래스의 상속이란?
  - ✓ 상속 : 물려받다 라는 의미.
  - ✓ 어떠한 새로운 클래스 생성 시 기존 클래스의 기능을 물려 받는 것.
  - ✓ 물려받는 클래스(부모)의 속성(메소드, 변수 등)을 모두 받는 것.
  - ✓ 추가적으로 필요한 함수를 자식 클래스에 생성 할 수 있다.
  - ✓ 단, 자식 클래스에서 추가한 것이므로 영향도는 자식 클래스에게만 있다.

06-1 클래스

- 클래스의 상속 방법
  - ✓ class Calculator :
  - ✓ class CalculatorA(Calculator) :

06-1 클래스

- 메소드 오버라이딩(overriding)
  - ✓ 부모 클래스를 상속 받은 후 부모 클래스의 메소드를 수정하려는 행위.
  - ✓ 같은 함수명을 선언하여 덮어 씌우는 개념.
  - ✓ 이 역시 영향도는 자식 클래스에서만 있다.

06-1 클래스

- 클래스 상속, 오버라이딩 실습
  - ✓ 현재 부모 클래스, 상속 클래스A가 있다.
  - ✓ 상속 클래스B를 만들어 div 함수를 수정하라
  - ✓ 분모가 0인 경우 return 0

06-1 클래스

- super()의 역할
  - ✓ 상속 과정에서 부모 함수의 내용을 살리며 생성자를 추가하고 싶을 때
  - ✓ super().__init__()

06-2 모듈

- 모듈이란?
  - ✓ 지금까지 실습과정에서 만들었던 모든 py.
  - ✓ 함수, 변수, 클래스 등을 정의해 놓고 가져다 쓸 수 있게 만들어 놓은 파일

06-2 모듈

- 모듈 이용 기본 구조
  - ✓ import 모듈 이름
  - ✓ from 모듈 이름 import 모듈 함수

06-2 모듈

- 모듈 사용 실습

[동일 depth에서의 사용]

✓ 함수를 사용하기

✓ 클래스를 사용하기

[다른 depth에서의 사용]

✓ 함수를 사용하기

✓ 클래스를 사용하기

06-2 모듈

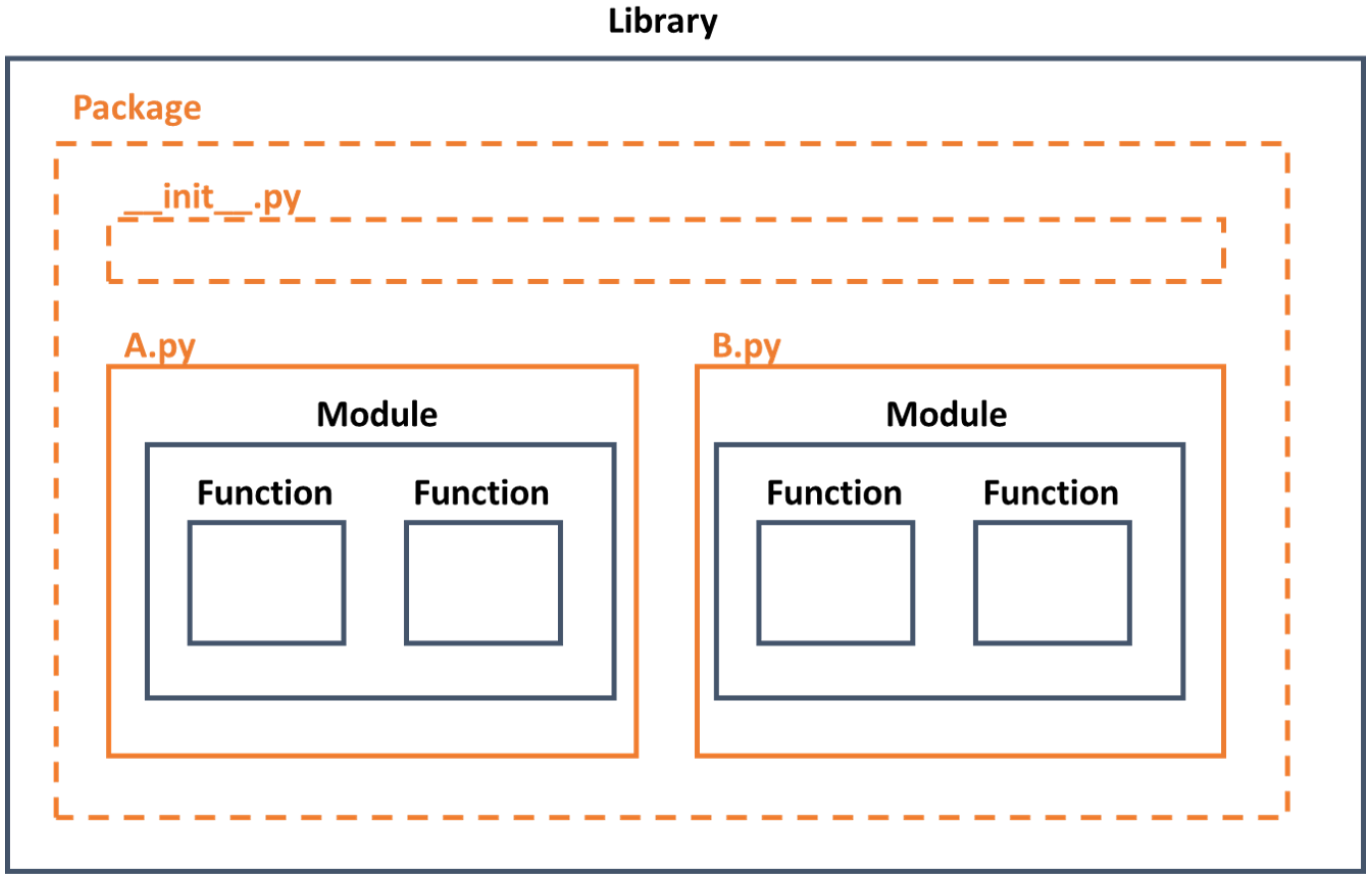
- __name__의 의미
  - ✓ if __name__ == '__main__':
  - ✓ __name__ : 부르는 모듈의 이름
  - ✓ __main__ : 실행의 주체가 되는 프로그램의 이름

06-3 패키지

- 패키지란?
  - ✓ 모듈을 여러 개 모아 놓은 개념
  - ✓ 라이브러리

06-3 패키지

- 용어(포함관계)



06-3 패키지

- 패키지 실습
  - ✓ 1. bank 라는 디렉토리를 선언 한다.
  - ✓ 2. 내부에 deposit, profile, withdraw 폴더를 생성하고 각 폴더 내에 같은 이름의 모듈을 생성한다.
  - ✓ 3. 모든 모듈내에 print문으로 구성된 함수를 만들고 서로의 관계를 이해한다

06-4 예외처리

- 예외처리의 목적

  ✓ 기본적으로 오류가 발생하면 프로그램이 종료 되도록 설계되어 있으나,

  프로그램이 오류에 의해 강제종료 되지 않도록 별도 지정을 해놓는 행위

  ✓ 정수를 0으로 나눈다.

  ✓ 리스트의 범위에 벗어난 값을 읽어온다.

06-4 예외처리

- 예외처리의 기본 구조

  ✓ try :

  오류가 발생할 수 있는 구문

  except :

  오류 발생시 실행문

06-4 예외처리

▪ 예외처리 실습

✓ x = [10, 30, 25.2, 'num', 14, 51] 리스트가 있다.

✓ For문을 이용하여 해당 값을 제곱하고 그 결과를 프린트 문을 이용하여 출력하는 프로그램을 만들어라.

✓ 단, 에러가 나는 부분은 예외처리를 통하여 해결하라.

[결과화면 예시]

```
프로그램 시작!
=======10=======
대상 = 10, 제곱값 = 100
=======30=======
대상 = 30, 제곱값 = 900
=======25.2=======
대상 = 25.2, 제곱값 = 635.04
=======num=======
unsupported operand type(s) for ** or pow(): 'str' and 'int', num는 숫자가 아닙니다.
=======14=======
대상 = 14, 제곱값 = 196
=======51=======
대상 = 51, 제곱값 = 2601
프로그램 종료!
```

06-4 예외처리

- 예외처리의 심화 구조

  ✓ try :

  오류가 발생할 수 있는 구문

  except Exception as e :

  오류 발생

  else :

  오류 발생하지 않음

  finally :

  default 무조건 실행

06-4 예외처리

- 주로, 데이터베이스 CRUD, 타 시스템과의 통신 등 에서 사용.
- 예상치못한 상황이 발생 가능한 프로그램에는 반드시 예외처리.

06-4 예외처리

- 예외처리 강제 에러발생
  - ✓ 오버라이딩 강제

    >> 반드시 상속을 받은 후 오버라이딩 하여 사용하라
  - ✓ Java의 인터페이스와 유사한 개념

06 클래스 최종 실습

```
고용형태 선택(정규직<P>, 비정규직<T>) : t
이름 : 메가
작업시간 : 20
시급 : 300
==============================
이름 : 메가
고용형태 : 비정규직
급여 : 6000
```

```
고용형태 선택(정규직<P>, 비정규직<T>) : k
지원하지 않는 고용형태입니다. 입력을 다시하세요!
고용형태 선택(정규직<P>, 비정규직<T>) : p
이름 : 메가
기본급 : ㄱ
숫자가 아닌것 같은데요?
기본급 : 100
보너스 : 200
==============================
이름 : 메가
고용형태 : 정규직
급여 : 300
```

- 클래스 최종 실습

  - ✓ **employee_parent.py** 모듈을 생성하고 **Employee** 클래스를 정의하라. 해당 클래스는 이름**(name),** 고용타입 **(emptype)**을 매개변수로 받아 초기화 하며 **print_profile**이라는 함수로 이름과 고용형태를 그대로 출력해준다.
  - ✓ employee.py 모듈을 따로 만들고 **Employee클래스**를 상속받아 Permanent, Temporary라는 자식 클래스를 생성하라. Permanent클래스는 월급(**money**), 보너스(**bouns**) 변수가 추가되며 생성자 함수에서 이를 합산하여 pay라는 변수에 할당한다. **Temporary** 클래스는 시간(**time**), 시급(**t_money**)이라는 변수가 추가되며 생성자 함수에서 이를 곱하여 **pay**라는 변수에 할당 한다. 단, 이때 기존 변수들은 상속받아 해결하라.
  - ✓ **emptype** 의 값을 **p** 또는 **P**로 받으면 **Permanent**클래스를 사용하여 인스턴스를 생성하고 **t** 또는 **T**를 받으면 Temporary클래스를 사용해 인스턴스를 생성한다. 다른값이 들어오면 올바른 값이 들어올때 까지 다시 값을 받는다.