



Round 2

**PRESS
START**



《 Round 2 》

- 모듈
- 클래스
- 예외처리
- 파이썬 자료구조



New
Assignment



《 Round 2 》

- 모듈 《
- 클래스
- 예외처리
- 파이썬 자료구조



Let's
Go



calculator.py

```
def add() :  
def sub() :  
def mul() :  
def div() :  
def rem() :
```

```
def result_write() :  
def result_read() :
```

```
def start() :  
.  
.
```

cal_module.py

```
def add :
```

```
def sub :
```

```
def div :
```

```
def mul:
```

cal_IO.py

```
def result_write() :
```

```
def result_read() :
```

calculator.py

```
import cal_module  
import cal_IO  
import cal_start
```

```
start()  
...
```

cal_start.py

```
def start() :
```



why ?

why ?

Looks good !



《 Round 2 》

- 모듈 - complete
- 클래스 《
- 예외처리
- 파이썬 자료구조



Let's
Go



example Java class...

```
class Cal {  
    private int result;  
  
    public Cal(){  
        result = 0;  
    }  
  
    public add(int num){  
        result += num;  
        system.out.println(result);  
    }  
  
    public sub(int num){  
        result -= num;  
        system.out.println(result);  
    }  
}
```

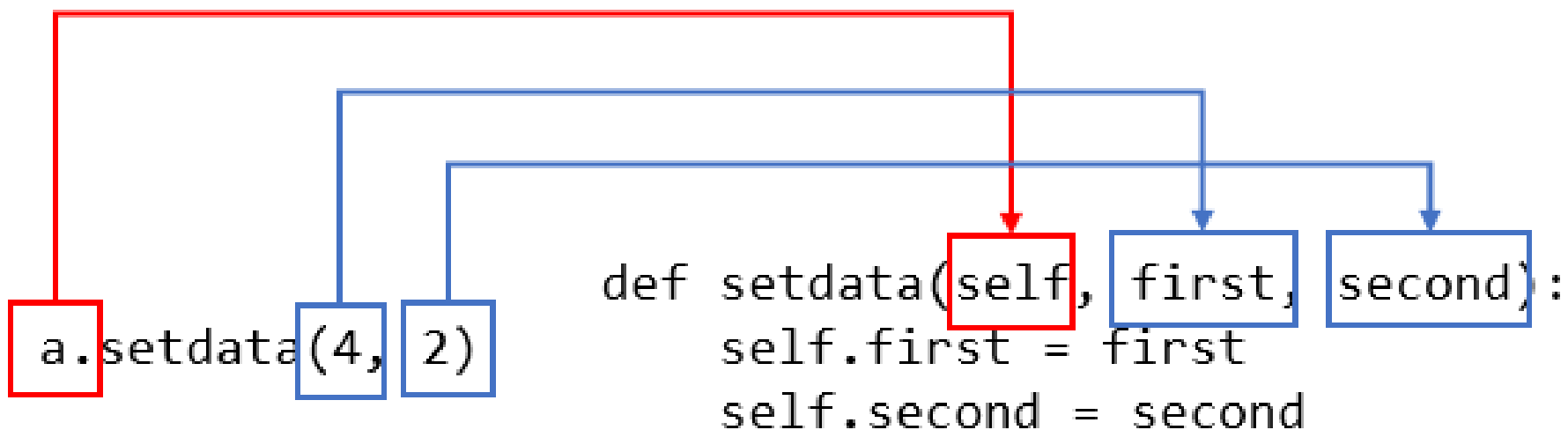
in Python class!

```
class Cal :  
    def __init__(self) :  
        self.__result = 0  
  
    def add(self, num) :  
        self.__result += num  
        print(self.__result)  
  
    def sub(self, num) :  
        self.__result -= num  
        pritrn(self.__result)  
  
...
```

in Python class!

```
class Cal :  
    def __init__(self):  
        self.__result = 0  
  
    def add(self, num) :  
        self.__result += num  
        print(self.__result)  
  
    def sub(self, num) :  
        self.__result -= num  
        pritrn(self.__result)  
  
...
```


self?



2. 클래스 내의 함수는 자신의 호출을 명시적으로 써줘야 함.
(self라는 이름을 굳이 쓸 필요는 없지만, 관례적으로 self라고 이름을 써왔다고 합니다.)

`__init__`?

== 파이썬 클래스 생성자 o o
ex) `def __init__(self)`

3. 파이썬 클래스의 생성자는 `__init__()`이다.
(얘는 이름이 원래 `__init__()`임)

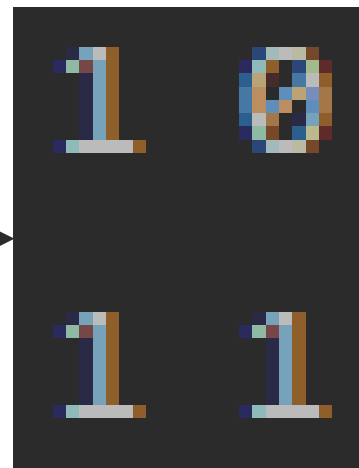
※ 클래스변수 (실수 많이 발생하는 부분)

```
# 클래스변수 result와 객체변수 results
class Cal:
    result = 0
    def __init__(self):
        self.results = 0

cal1 = Cal()
cal2 = Cal()

# 클래스변수에 1 대입
Cal.result = 1
# 클래스의 객체변수에 1대입
Cal.results = 1
# cal2 객체변수에 1대입
cal2.results = 1

print(cal1.result, cal1.results)
print(cal2.result, cal2.results)
```



1)

```
class Cal :  
    result = 0  
  
    def __init__(self):  
        self.results = 0  
  
cal1 = Cal()  
cal2 = Cal()
```



results = 0
인스턴스화(실체화)



cal1
cal1.result = 0
cal1.results = 0

cal2
cal2.result = 0
cal2.results = 0

2)

```
# 클래스변수에 1 대입  
Cal.result = 1
```



해당 클래스로 만들어진
모든 인스턴스에 공유

cal1
cal1.result = 1
cal1.results = 0

cal2
cal2.result = 1
cal2.results = 0

3)

```
# 클래스의 객체변수에 1대입  
Cal.results = 1
```

```
class Cal:  
    result = 0  
    def __init__(self):  
        self.results = 0
```

results는 생성자를 통한 초기화를 했을 때만 생성됨.
즉, 해당 코드는 무용지물

4)

```
# cal2 객체변수에 1대입  
cal2.results = 1
```

cal2
cal2.result = 1
cal2.results = 0



cal2
cal2.result = 1
cal2.results = 1

5)

```
print(cal1.result, cal1.results)  
print(cal2.result, cal2.results)
```

cal1
cal2.result = 1
cal2.results = 0

cal2
cal2.result = 1
cal2.results = 1

```
1 0  
1 1
```

《 Round 2 》

- 모듈 - complete
- 클래스 - complete
- 예외처리 《
- 파이썬 자료구조



Let's
Go



example Java...

```
static Scanner sc = new Scanner(...);
int num1 ;
int num2 ;

try {
    num1 = sc.nextInt();
    num2 = sc.nextInt();
    System.out.println(num1/num2);
} catch (ArithmeticException e) {
    System.out.println("0으로 나누지 마!")
} catch (InputMismatchException e) {
    System.out.println("int형으로 넣어라")
} finally {
    System.out.println("try문이 끝났어.")
}
```



```
try :
    num1 = int(input())
    num2 = int(input())
    print(num1/num2)

except ZeroDivisionError as e :
    print("0으로 나누지 마!")

except ValueError as e:
    print("int형으로 넣어라")

finally :
    print("try문이 끝났어.")
```

《 Round 2 》

- 모듈 - complete
- 클래스 - complete
- 예외처리 - complete
- 파이썬 자료구조 《



Let's
Go




```
# 리스트 자료형
ex_list = [1, "이", , [3]]

# 튜플 자료형
ex_tuple = (1, "이", [3])

# 딕셔너리 자료형
ex_dictionary = {1 : 1, 2 : "이", 3 : [3]}

# 집합 자료형
ex_set = set([1, "이", [3]])
```

List

```
# 리스트 자료형  
ex_list = [1, "0", , [3]]
```

요소 수정	요소 형태	특징
0	Anything	대괄호

- array(배열)과 list(리스트)의 차이

배열은 크기가 정해져 있으며, 기능이 존재하지 않는다.
리스트는 크기가 가변적이며 여러 기능이 존재한다.

배열 인덱스는 값에 대한 유일무이한 식별자.
리스트 인덱스는 몇 번째 데이터인가 정도의 의미.

배열은 요소를 삭제해도 순서가 그대로이다.
리스트는 중간 요소를 삭제하면 순서가 바뀐다.

그렇다고 한다...

- ```
딕셔너리 자료형
ex_dictionary = {1 : 1, 2 : "0|", 3 : [3]}
```

# Set

```
집합 자료형
ex_set = set([1, "이", [3]])
```

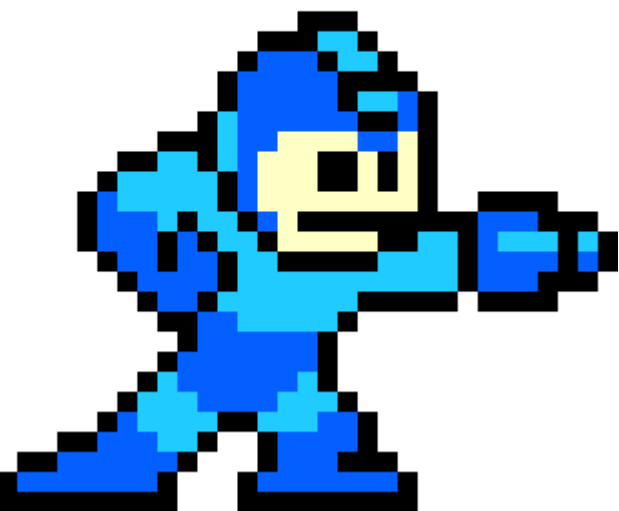
| 요소 수정 | 요소 형태            | 특징                               |
|-------|------------------|----------------------------------|
| 0     | anything but set | set()을 통해 생성<br>중복 비허용<br>순서가 없음 |

- Set은 어떤 경우에 쓴다?

자료의 중복을 제거할 때(많이 쓰지는 않음)

자료의 집합 연산을 할 때(합집합, 차집합, 교집합)  
->> 집합연산 필요 시 가장 유용하게 쓸 수 있는 자료구조

# WARNING



# WARNING

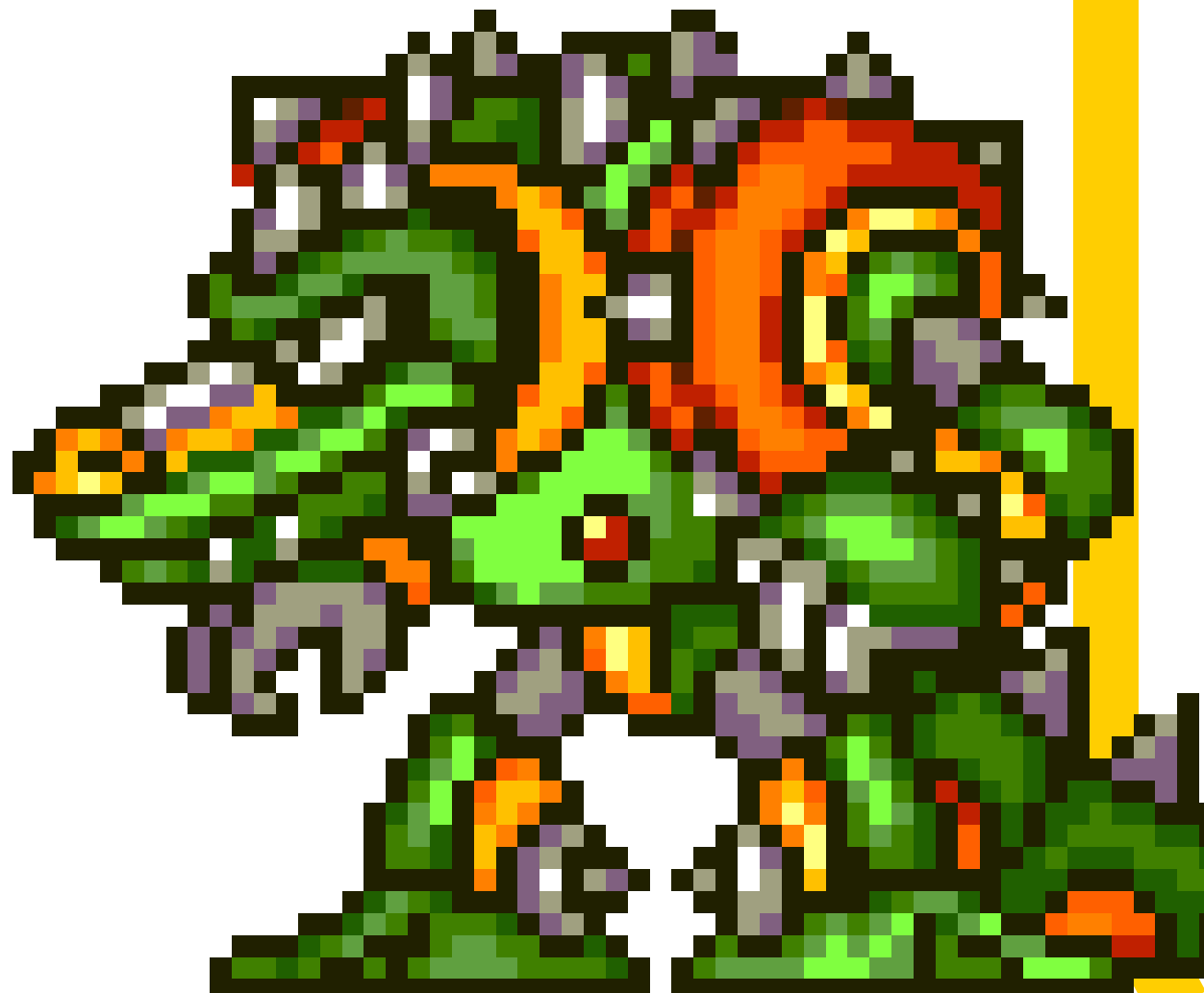
## 《 QUEST 》

- 각자 맡은 파트 발표자료 준비 -

Que/Stack/Deque/Linked-list

Graph/Tree(Binary, Heap Tree)

Sort Algorithm



# NEXT STAGE

