[STAY] SNS Project3

아이디 일치 시에만 수정, 삭제 가능하도록 처리

```
if (<%=sid %> == feeds[i].id) {
  str +="<button type='button' class='btn btn-white' id='btn'><svg xmlns='http://www.w3.org/2000/svg' width='16' height='16' fill='cur
}
```

Python

function

JAVA

함수X, 클래스 내부의 메소드만이 존재, 클래스 밖이라는 개념이 없다

Python

함수란?

- 입력 값에 따라 다른 실행 결과를 내보낸다
- 함수를 중요하게 많이 사용한다
  - 같은 내용의 반복을 줄이기 위해 함수를 만들어서 필요마다 호출
  - 프로그램의 흐름도 잘 파악할 수 있고 오류가 어디서 발생하는지 쉽게 찾을 수 있다

```
#정의
def add(a,b):#a,b 매개변수
  return a+b

#호출
print(add(1,2))#1,2 인수
print(add(2,6))#2,6 인수
```

매개변수

- 함수의 입력으로 전달 된 값을 받는 변수

인수

- 함수를 호출할 때 전달하는 입력 값

함수의 4가지 형태

1. input O, output O

```
def add(a,b):
  return a+b

print(add(1,2))
```

2. input X, output O

```
def say():
  return 'Hello World'

print(say())
```

3. input O, output X(return X)

print 기능이 있는 함수일 뿐 output(return)은 없다

```
def yesinput(name):
  print(name,'님 어서오세요')

yesinput('심현민')
```

4. input X, output X(return X)

print 기능이 있는 함수일 뿐 input, output(return)은 없다

```
def noinnoout():
  print('환영합니다.')

noinnoout()
```

매개변수의 개수를 알 수 없을 때

가변 매개변수 args

```
def add(*args):
  result = 0
  for i in args:
    result += i
  print(result)

add(1,2)
add(1,2,3)
```

함수의 return 값은 하나

```
#오류
def addmul(a,b):
  return a+b
  return a*b

print(addmul(2,3))

#자동으로 튜플 1개로 return한 것
def addmul2(a,b):
  return a+b, a*b

addmul2(2,3)
```

```
5
(5, 6)
```

매개변수에 초기값 미리 설정

gender=True 로 미리 설정

```
def aboutme(name, age, gender=True):
  print(f'나의 이름은 {name}입니다.')
  print(f'나의 나이는 {age}입니다.')
  if gender:
    print(f'나의 성별은 여성입니다.')
  else:
    print(f'나의 성별은 남성입니다.')

aboutme('심현민', 26)
```

함수 안에서의 변수 효력 범위

오류1 함수 밖 선언

```
a = 1
def vartest(a):
  a += 1

vartest(a)
print(a)
```

```
1
```

오류2 선언x

```
#오류2
def vartest(b):
  b += 1

vartest(3)
print(b)
```

```
---------------------------------
-
NameError
last)
<ipython-input-20-eee2226d46b1> in
      3
      4 vartest(3)
----> 5 print(b)

NameError: name 'b' is not defined
```

성공

```
def vartest(c):
  c += 1
  return c

vartest(3)
```

4

Class

OOP Object Oriented Programming 객체 지향 프로그래밍

class (철학, 방법)	object (실체화)
붕어빵 틀	붕어빵
쿠키 틀	쿠키
게임 속 적 틀	게임 속 적

맛과 색은 다르더라도 붕어빵, 쿠키, 적인 것은 동일

Object

인스턴스 VS 객체

- 클래스로 만든 객체를 인스턴스라고 한다
- 차이
  - a는 쿠키 클래스의 인스턴스이다. a는 객체이다
  - *인스턴스는 클래스와의 관계를 설명할 때 사용되고 객체는 그 자체를 설명할 때 사용된다*

```
# 클래스 정의
class Cal:
  pass

# 클래스로 인스턴스 생성
a = Cal()
```

> *동일한 클래스로 만든 객체들은 각기 다른 특성을 가지게 된다*

```python
class Cal:
  def __init__(self):#생성자
    self.result = 0

  def add(self, num):
    self.result += num
    return self.result

cal1 = Cal()
cal2 = Cal()
cal3 = Cal()

print(cal1.add(2))
print(cal1.add(1))
print(cal1.add(3))
print()
print(cal2.add(5))
print(cal2.add(4))
print(cal2.add(1))
print()
print(cal3.add(2))
```

```
2
3
6

5
9
10

2
```

Self

```
class BigCal:
  def setdata(self, first, second):# self는 호출한 객체의 이름이 온다
    self.first = first
    self.second = second

  def add(self):
    result = self.first + self.second
    return result

  def sub(self):
    result = self.first - self.second
    return result

  def mul(self):
    result = self.first * self.second
    return result

  def div(self):
    result = self.first / self.second
    return result

cal1 = BigCal()
cal2 = BigCal()

cal1.setdata(2,9) # self:cal1 / first:2 / second:3
cal2.setdata(5,1) # self:cal2 / first:5 / second:1
```

```
print(cal1.first)
print(cal1.second)
print(cal2.first)
print()
print(cal1.add())
print(cal2.add())
print()
print(cal1.sub())
print(cal2.sub())
print()
print(cal1.mul())
print(cal2.mul())
print()
print(cal1.div())
print(cal2.div())
```

```
2
9
5

11
6

-7
4

18
5

0.2222222222222222
5.0
```

> *하나의 클래스로 생성된 여러개의 객체들은 각각 다른 성질을 유지한다*

상속 Inheritance