Imperative Paradigm

<u>Sequence Statement (assignment)</u>

c.f) 객체지향

장점 : 코드 작성 용이 / 단점 ?

Functional Paradigm

<u>function composition (expression)</u>

c.f) pure function (수학 함수 (Input, output), w/o side effects)

mutable

Immutable

c.f) 상수 개념, encapsulation

loop → Less or W/0 loop

1 Divide And Conquer (분할정복) / 알고리즘

Recursive

c.f) tail recursion을 미지원 (언어자체)

2 한꺼번에 동시에 전체 값 변화

High Order Function

map. filter (itertools, functools)

closure/currying, decorator

3 여러 개 중에 하나의 값 변화

Iterator : iter()

Generator : yield, expression

4 Syntactic sugar

comprehension

c.f) Haskel에서 차용

First Class Function

#1 No side effect

```
# Example of Function w/ Site Effect

def sayhi(name)
     print("Hi there, my name is %s" % name)

def write_to_file(file, text):
   with open(file, 'w+') as f:
      f.write(text)

def sort_nums(nums):
   nums.sort() # inplace sort
   return nums
```

pure function

#2 Same input, same output

```
# Example of Function that Violates this

counter = 0
def foo(x) :
   global counter
   counter += 1
   return counter + x
```

expression over statement

```
# Statement
def abs1(val):
    if val < 0:
        return -val
    else:
        return val

# Expression
def abs2(val):
    return -val if val < 0 else val
```

recursion over iteration

```
# Iteration
def sum1(nums):
    total = 0
    for num in nums:
    total += num return total

# Recursion
def sum2(nums):
# Not very efficient in Python but you got the idea
    return 0 if len(nums) == 0 else nums[0] + sum2(nums[1:])
```