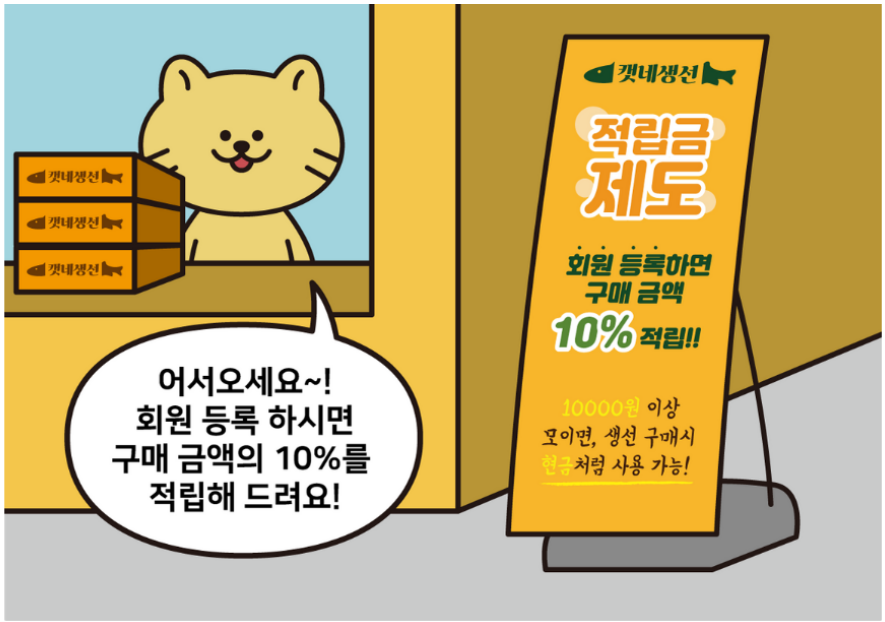
캣의 고객관리

PYTHON BOOTCAMP



캣은 효율적으로 고객들을 관리하기 위해 고객들을 회원으로 등록하고 구매 금액의 10%를 적립해주는 "**적립금 제도**"를 시작하였습니다.

단, 적립금은 **회원인 경우에만 적립**할 수 있으며 **10,000노드 이상이 모이면 생선 구매 시 현금처럼 사용**할 수 있습니다.

---

1. **회원으로 등록하겠습니까?**

   손님 'A캣'과 'D캣'은 캣의 생선 회사에 회원인지 확인하고 싶습니다. 만약 회원이 아닐 경우에는 회원으로 등록하고 싶다고 합니다.

   아래 코드를 작성하여 'A캣'과 'D캣'이 회원인지 확인하고 회원으로 등록해 주세요.

```python
#[In]

회원명 = input('회원명 입력하세요 : ')
회원 = ['A캣', 'B캣', 'C캣']

def 회원등록():
    '정답을 입력해주세요.'

if '정답을 입력해주세요.':
    '정답을 입력해주세요.'
else:
    회원등록()
```

Python

```
#[Out]

회원명 입력하세요 : A캣
이미 등록된 회원입니다.

---

회원명 입력하세요 : D캣
['A캣', 'B캣', 'C캣', 'D캣']
회원으로 등록되었습니다.
```

Python

2. **회원에게 적립금을 지급하라!**

   손님 '애옹'이와 '냐옹'이는 생선을 각각 15,000노드, 5,000노드씩 구입하였습니다. 이때 '애옹'이와 '냐옹'이가 회원인 경우를 체크합니다. 회원인 경우 적립받게 되는 금액을 출력하고, 회원이 아닌 경우에는 아직 회원이 아닙니다를 출력해주세요.

```
#[In]

구매가격 = input('가격을 입력하세요 :')
회원명= input('회원명을 입력하세요 :')

회원 = ['씨-캣', '자바캣', '파이캣', '썬캣', '애옹']
```

Python

```
#[Out]

회원명 입력하세요 : 애옹
회원입니다.
15000노드 중 1500노드가 적립됩니다.

---

회원명 입력하세요 : 냐옹
회원이 아닙니다. 회원가입을 해주시기 바랍니다.
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

3. **적립금을 사용하고 싶어요!**

   손님 '파이캣'과 '썬캣'은 생선을 각각 10,000노드씩 구매하였습니다. 두 손님은 적립금을 사용할 수 있는지 알고 싶습니다.

   사용할 수 있다면 적립금 5,000노드를 사용합니다. 이때 구매 가격과 남은 적립금은 얼마인지 출력하고 남은 적립금은 Dictionary에 반영해 주세요.

   그렇지 않을 경우에는 현재 적립금은 얼마인지 출력해주세요.

```python
#[In]

구매가격 = input('가격을 입력하세요 :')
회원명 = input('회원명을 입력하세요 :')

#회원 = {회원명:적립금}
회원 = {'씨-캣': 5000, '자바캣': 3500, '파이캣': 15000, '썬캣': 7000}
```

Python

```
#[Out]

In 가격을 입력하세요 : 10000
In 회원명을 입력하세요 : 파이캣
Out 적립금을 사용할 수 있습니다. 현재 적립금액은 15000노드 입니다.
In 사용할 적립금 노드를 입력하세요 : 5000
Out 5000노드를 사용하였습니다. 남은 적립금은 10000노드입니다. 결제금액은 5000원 입니다.

---

가격을 입력하세요 : 10000
회원명을 입력하세요 :썬캣
현재 적립금이 7000노드이므로 아직 적립금을 사용할 수 없습니다. 결제금액은 10000노드이고, 적립 포인트는 1000노드, 합산 포인트는 8000 노드입니다.
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

4. **적립금 이벤트를 진행합니다!(심화, google에서 Factory 함수라고 검색해보세요.)**

   캣은 매달 월과 같은 요일에 적립금 이벤트를 진행하기로 했습니다.
   예를들어, 2월 2일에는 적립금의 2배를, 3월 3일에는 적립금의 3배를 적립해줍니다.

   만약 손님이 2월 2일에 5000원의 생선을 구매하고, 3월 3일에 15000원을 구매 했다면 각각 얼마의 적립금을 받을 수 있을까요? 중첩 함수를 사용하여 풀어보세요!

```python
def 배수(n):
    def 적립(value):
        '정답을 입력하세요.'
    return 적립

Feb = 배수(2)
Mar = 배수(3)

print('2월 적립금 이벤트 :', '정답을 입력하세요.')
print('3월 적립금 이벤트 :', '정답을 입력하세요.')
```

Python

Project name :

DATE .

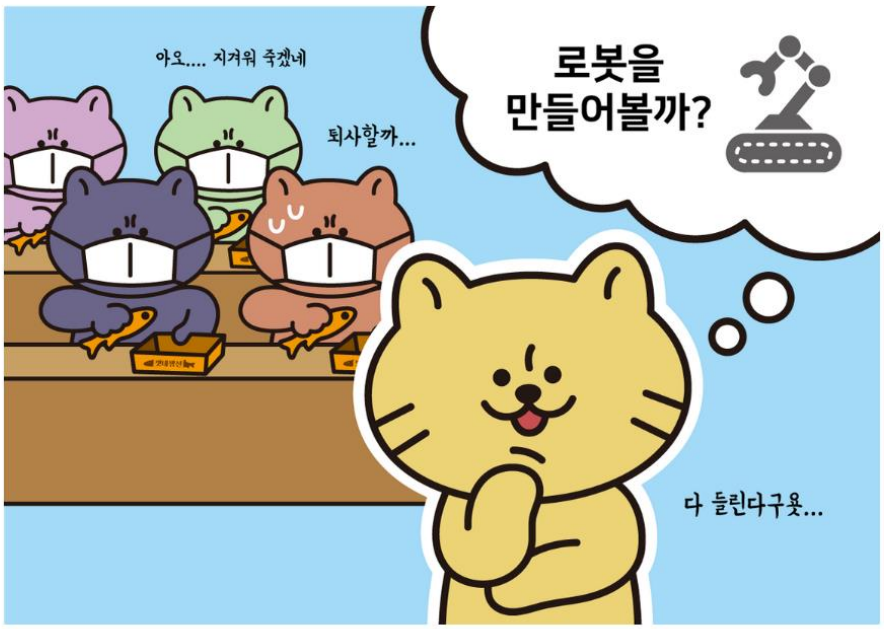
비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

5편 라이캣의 EXIT

1. 캣의 로봇(for)
    1.1 for문의 기본 구조
    1.2 for, else
    1.3 지능형 리스트(list comprehension)의 for
    1.4 다중인자 리스트 순회
    1.5 enumerate
    1.6 계산하는 로봇
2. 캣의 로봇(while)
    2.1 무한반복 while, break
    2.2 while, else
3. break
    3.1 break 문
4. continue, pass
5. else
6. 중첩 반복문

1. 캣의 로봇(for)



캣의 생선회사는 어느덧 위니브 월드 전체에서 가장 빠르게 성장하는 생선회사가 되었습니다. 대부분의 스타트업이 그렇듯이 성장세에 일만하다 보니 높은 연봉에도 직원들의 불만이 늘어가고 있었습니다.

"자율성과 창의적 생각이 보장되고, 개인이 성장하면서, 각자 하는 일에 대한 명료한 목적과 동기부여를 줄 수는 없을까냥?"

캣은 어느새 대표다운 생각을 하고 있었어요. 매주 아침 하는 회의에서 캣은 말했습니다.

"생산성을 극대화 하면서도, 직원들의 휴식시간을 늘려줄 수 있는 로봇을 만들어보는 것은 어떨까냥?"

하지만 이미 불만이 가득차 있는 직원들은 냉담한 반응이었습니다.

"누가 만드냥!? 대표가 만드냥!?"

캣은 더이상의 회의가 무의미하다는 생각에 회의를 종료했습니다. 그리고, 정말 혼자 만들기 시작했어요.

"밤을 새서라도 다음주까지 만들겠다냥!"

- **라이캣의 상태창!**

```python
#[In]

이름 = '캣'
설명 = '위니브 월드에서 가장 급성장하는 생선회사 대표 캣'
나이 = 19
오늘_잡은_물고기 = '100000'
직원수 = 4
키 = '45.9cm'
몸무게 = 1.6
잡식 = True
돈 = 1100000
훈장 = ['백상아리를 잡은 고양이', '성실한 납세자', '해골섬 낚시꾼', '청년 고용 착한 기업', '회사를 설립한 자']
기술 = ['고기잡이', '고기팔기', '낚시_Lv5', '통발_Lv5', '큰그물_Lv5']
```

Python

캣은 가용한 모든 자원, 알고 있는 모든 지식을 투입하기 시작했어요. 아래 코드를 봅시다.

```python
#[In]

for x in (1,2,3,4,5):
  print(f'고등어 포장 {x}번째 입니다.')
```

Python

우선 직원들이 가장 힘들어하는 고등어 포장 로봇을 만들기 시작했어요. `for` 라는 강력한 프로그래밍 문법을 사용하였습니다.

위 코드를 실행시키면 아래와 같이 출력됩니다.

```python
#[Out]

고등어 포장 1번째 입니다.
고등어 포장 2번째 입니다.
고등어 포장 3번째 입니다.
고등어 포장 4번째 입니다.
고등어 포장 5번째 입니다.
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

위 코드처럼 **정해진 횟수를 반복하는 것을 for문**이라고 합니다. 코드를 조금 더 살펴보죠. `x`라는 변수를 선언하여 튜플 `(1,2,3,4,5)`의 길이만큼 for문 아래에 있는 문장을 반복하게 됩니다.

또, 변수 x 는 튜플 요소의 값을 하나씩 가지게 되죠. 조금 어렵죠? 더 자세한 설명은 다음 챕터에서 하도록 하겠습니다.

1.1 for문의 기본 구조

위에서 배운 for문의 기본 구조를 더 자세하게 정리하고 넘어가도록 하겠습니다. for문은 순서열을 순회하며 순서열의 끝에 도달하면 반복을 멈추게 됩니다. 또한 객체를 처음부터 끝까지 하나씩 **추출하며 순회**하기 때문에 그 사용법이 쉬워 가장 많이 사용되는 반복문 입니다.

다음 챕터에서 배울 while은 비교할 변수를 먼저 선언 해주어야 하기 때문에, 비교적 for를 더 많이 사용합니다. 하지만 각자의 용법이 있어, 어느 문법이 좋다고는 할 수 없습니다!

```python
#[In]

for x in (1,2,3):
  print(f'{x}번째 로봇을 뚝딱뚝딱')
```

Python

위 코드를 실행시키면 아래와 같이 출력됩니다.

```
#[Out]

1번째 로봇을 뚝딱뚝딱
2번째 로봇을 뚝딱뚝딱
3번째 로봇을 뚝딱뚝딱
```

Python

위의 코드를 실행하면 변수 x에는 튜플(1, 2, 3)의 요소가 순서대로 출력됩니다. 위 예제에서 튜플의 길이는 3이므로 for문은 반복을 3번 수행합니다. 여기서 x는 다른 언어처럼 초기화 하지 않아도 작동합니다.

```python
#for 문의 구조
for (변수명) in (순회 가능한 객체) :
    수행할 문장1
    수행할 문장2
```

Python

for 문의 범위로 사용되는 것은 시퀀스 자료형 자료 또는 반복 가능한 자료형이어야 합니다.

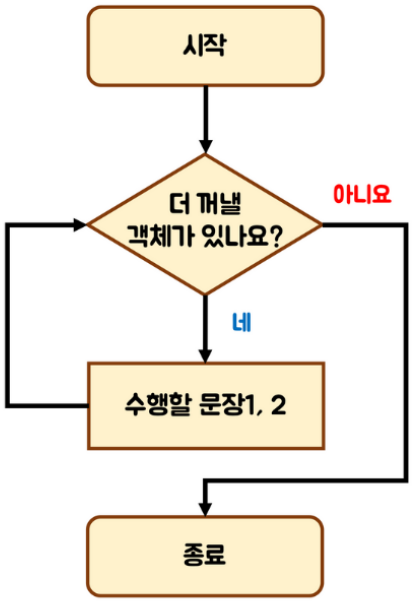
Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

- 반복문 순서도(Flow Chart) - for 문



1. String(문자열)을 범위로 지정한 예시

```python
#[In]

왕국 = '위니브 월드'
for a in 왕국 :
    print(a)
```

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

```
#[Out]

위
니
브

월
드
```

Python

2. List(리스트)를 범위로 지정한 예시

```
#[In]

기술 = ['고기잡기', '고기팔기', '낚시_Lv1', '통발_Lv1', '큰그물_Lv1']
for a in 기술:
    print(a)
```

Python

```
#[Out]

고기잡기
고기팔기
낚시_Lv1
통발_Lv1
큰그물_Lv1
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

3. Dictionary(사전)을 범위로 지정한 예시

```python
#[In]

캣의_상태창 = {
    '이름': '캣',
    '설명':'위니브 월드에서 가장 급성장하는 생선회사 대표 캣',
    '나이':'19',
    '키': '45.9cm',
    '몸무게': 1.6
}

for a in 캣의_상태창:
    print(a)
```

Python

```
#[Out]

이름
설명
나이
키
몸무게
```

Python

위의 예제에서 Dictionary(사전)형 자료의 경우에는 key(키)만을 가져오게 됩니다.

key(키)에 해당하는 value(값) 또한 가져오고 싶다면 아래와 같이 튜플 언패킹을 사용할 수 있습니다.

```python
#[In]

for key,value in 캣의_상태창.items():
    print("{0} : {1}".format(key, value))
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

위 코드를 실행시키면 아래와 같이 출력됩니다.

```
#[Out]

이름 : 캣
설명 : 위니브 월드에서 가장 급성장하는 생선회사 대표 캣
나이 : 19
키 : 45.9cm
몸무게 : 1.6
```

Python

for문에서 가장 많이 사용되는 range에 대해 알아봅시다. range는 특정 범위를 생성하기 위해 사용할 수 있습니다.

> 💡 range는 2.x에서는 선언하는 즉시 list가 되었으나 3.x으로 넘어오면서 range는 list로 변환해 주어야 list가 됩니다. 이는 2.x에서 썼었던 xrange와 같은 함수입니다.

range 함수는 연속하는 수열을 만듭니다. `range(시작_값, 종료_값, 연속하는_두_수의_차)`형식으로 선언되며 아래 예제를 보며 자세히 설명해 드리도록 하겠습니다.

> 💡 range(start, stop, step)으로 주요 표현합니다. 어디서 많이 보았던 형식이죠? 바로 슬라이싱에서 보았던 형식입니다.

시작 값 : 0, 종료 값 : 5, 연속하는 두 수의 차 : 1

```
#[In]

for a in range(0, 5, 1):
    print(a)
```

Python

```
#[Out]

0
1
2
3
4
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

시작값 0과 멈춤값 5 사이의 연속하는 두 수의 차 1을 가지고 있는 요소들이 생성되었습니다. 생성된 range의 마지막 요소가 멈춤값보다 한 단계 작다는 사실에 주의하세요.

시작 값 : 0, 종료 값 : 10, 연속하는 두 수의 차 : 2

```python
#[In]

for a in range(0, 10, 2):
    print(a)
```

Python

```python
#[Out]

0
2
4
6
8
```

Python

시작 값 0과 종료 값 10 사이의 연속하는 두 수의 차 2를 가지고 있는 요소들이 생성되었습니다.

생성된 range의 마지막 요소가 종료 값보다 한 단계 작다는 사실에 주의하세요. range()함수의 마지막 매개변수인 연속하는 두 수의 차는 생략할 수 있습니다. 생략할 시 연속하는 두수의 차는 1입니다.

시작 값 : 0, 종료 값 : 5, 연속하는 두수의 차 : 생략

```python
#[In]

for a in range(0, 5):
    print(a)
```

Python

위 코드를 실행시키면 아래와 같이 출력됩니다.

```python
#[Out]

0
1
2
3
4
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

마지막 매개변수인 연속하는 두 수의 차를 생략하고 range()함수를 호출하였습니다. 이 경우 range() 함수의 마지막 매개변수인 연속하는 두 수의 차는 1이 됩니다.

시작 값 : 생략, 종료 값 : 5, 연속하는 두 수의 차 : 생략

```
#[In]

for a in range(5):
    print(a)
```

Python

```
#[Out]

0
1
2
3
4
```

Python

또한 range() 함수는 종료 값만을 입력하여 호출할 수 있습니다. 종료 값만 입력했을 시에는 시작값은 0이 되며 두 수의 차는 1이 됩니다.

시작 값 : 10, 종료 값 : 5, 연속하는 두 수의 차 : -1

```
#[In]

for a in range(10, 5, -1):
    print(a)
```

Python

```
#[Out]

10
9
8
7
6
```

Python

연속하는 두 수의 차를 마이너스로 둘 수도 있습니다. 이 경우 시작 값과 종료 값 보다 크게 주셔야 한다는 점도 기억해주세요.

좀 더 알아볼까요? 얕은물에서 할 내용은 아니니 가볍게 읽고 넘어가주세요. for 문에 사용될 수 있는 시퀀스형 자료는 문자열, 리스트, 튜플이 있습니다. 이 자료형은 메서드로 __iter__와 __next__를 가지고 있으며 이 두개의 메서드로 순회가 가능하게 합니다.

이는 아래와 같이 iter()함수와 next()함수로도 실행을 할 수 있습니다.

```python
#[In]

listx= [100,200,300,400]
strx= 'abcd'
listxlter = iter(listx)
strxlter= iter(strx)
print(next(listxlter),next(listxlter),next(listxlter),next(listxlter))
print(next(strxlter),next(strxlter),next(strxlter),next(strxlter))
```

Python

위 결과값은 아래와 같습니다.

```
#[Out]

100 200 300 400
a b c d
```

Python

1.2 for, else

```python
#[In]

for i in range(100):
    print(f'{i} 물고기를 잡았습니다.')
    if i == 5:
        print('만선입니다. 물고기를 다 잡았습니다.')
        break
else:
    print('아직 여유가 좀 있지만, 물고기가 더 없는 것 같으니 이정도로 만족하고 돌아갑시다.')
print('수고하셨습니다.')
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

위 코드를 실행시키면 아래와 같이 출력됩니다.

```
#[Out]

0마리의 물고기를 잡았습니다.
1마리의 물고기를 잡았습니다.
2마리의 물고기를 잡았습니다.
3마리의 물고기를 잡았습니다.
4마리의 물고기를 잡았습니다.
5마리의 물고기를 잡았습니다.
만선입니다. 물고기를 다 잡았습니다.
```

Python

```
#[In]

for i in range(5):
    print(f'{i} 물고기를 잡았습니다.')
    if i == 5:
        print('만선입니다. 물고기를 다 잡았습니다.')
        break
else:
    print('아직 여유가 좀 있지만, 물고기가 더 없는 것 같으니 이정도로 만족하고 돌아갑시다.')
print('수고하셨습니다.')
```

Python

위 코드를 실행시키면 아래와 같이 출력됩니다.위 코드를 실행시키면 아래와 같이 출력됩니다.

```
#[Out]

0마리의 물고기를 잡았습니다.
1마리의 물고기를 잡았습니다.
2마리의 물고기를 잡았습니다.
3마리의 물고기를 잡았습니다.
4마리의 물고기를 잡았습니다.
아직 여유가 좀 있지만, 물고기가 더 없는 것 같으니 이정도로 만족하고 돌아갑시다.
수고하셨습니다.
```

Python

if문 뿐만 아니라 for에서도 else를 사용할 수 있습니다. else는 루프가 정상 종료되었을 때, 처음부터 자료형이 비어있었을 때 실행됩니다. break문을 만나면 else 문을 실행하지 않고 빠져나옵니다.

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

1.3 지능형 리스트(list comprehension)

얕은물에서는 가볍게 이런 것이 있다는 것만 알고 넘어가도록 하겠습니다. 가장 많이 사용되는 list 생성 기법입니다. 'list comprehension'은 한국어로 표현할 때 리스트 표현식, 지능형 리스트로 번역이 되곤 합니다.

```python
#[In]

#1
x= [i for i in range(1, 10)]
print(x)

#2
y=['{}X{}={}'.format(i, j, i*j) for i in range(2, 10) for j in range(1, 10)]
print(y)

#3
def sumthingFunction(i):
    if i % 100 ==0:
        return i
    else:
        return 0
기존리스트= [100,200,300,101,202,303]
새로운리스트= [sumthingFunction(i) for i in 기존리스트]
print(sum(새로운리스트))
```

Python

위 코드를 실행시키면 아래와 같이 출력됩니다.

```python
#[Out]

#1
[1,2,3,4,5,6,7,8,9]

#2
구구단 리스트 출력

#3
600
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO