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

4.4 bool의 속성 알아보기

`bool` 자료형은 부울형, 불리언 자료형이라 읽습니다. 이 자료형은 `True` 와 `False` 2가지 타입밖에 없어요.

💡 javascript에서는 `True` 를 소문자 형태인 `true` 로 표시하고 있어요! 이런 차이점에 대해서 알고 있으시면 좋답니다.

```python
#[In]

육식 = True
초식 = True
print(type(육식))
print(dir(초식))
```

Python

```
#[Out]

<class 'bool'>
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delattr__', '__
dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '__floordiv__', '__f
ormat__', '__ge__', '__getattribute__', '__getnewargs__', '__gt__', '__hash__', '__index_
_', '__init__', '__init_subclass__', '__int__', '__invert__', '__le__', '__lshift__', '__
lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow_
_', '__radd__', '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__
rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__round__', '__rpow__',
'__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__si
zeof__', '__str__', '__sub__', '__subclasshook__', '__truediv__', '__trunc__', '__xor__',
'bit_length', 'conjugate', 'denominator', 'from_bytes', 'imag', 'numerator', 'real', 'to_
bytes']
```

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

4.5 list, tuple의 속성 알아보기

`list`는 두 개 이상의 값을 저장하고 싶을 때 사용하는 자료형입니다. `str`처럼 순서가 있는 시퀀스 자료형이며, 각 원소들은 변경이 가능합니다. 리스트의 기본 형태는 아래와 같습니다.

```python
#[In]

훈장 = []
기술 = ['고기잡이', '고기팔기']
print(type(기술))
print(dir(기술))
```

Python

```
#[Out]

<class 'list'>
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__has
h__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__l
en__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr_
_', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '_
_subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']
```

Python

여기서 대괄호(`[]`)를 사용하지 않고 소괄호(`()`)를 사용하면 list가 아니라, 자료의 값이 변경 불가능한 `tuple`이 됩니다. tuple은 순서가 있고, 소괄호를 사용하며, 값을 변경할 수 없습니다. 얕은 물에서는 튜플을 상세하게 다루지 않으니 가볍게 훑고 넘어가주세요.

```python
#[In]

잡은물고기_튜플 = ('광어', '고등어', '오징어', '오징어', '광어', '광어', '고등어', '고등어', '백
상아리', '금붕어')
```

Python

앞서 말씀드린 것처럼 `list`는 순서가 있는 시퀀스형 자료형이기 때문에 아래와 같이 `str`처럼 **indexing, slicing이 가능**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

```python
#[In]

잡은물고기 = ['광어', '고등어', '오징어', '오징어', '광어', '광어', '고등어', '고등어', '백상아리', '금붕어']
```

Python

```python
#[In]

print(잡은물고기[0])
print(잡은물고기[0:3])
print(잡은물고기[0:7:2])
```

Python

```
#[Out]

광어
['광어', '고등어', '오징어']
['광어', '오징어', '광어', '고등어']
```

Python

그런데 마지막에 잡은 물고기가 금붕어죠? 잘못 표기한 것은 아래와 같이 바꿀 수 있답니다.

```python
#[In]

잡은물고기[-1] = '백상아리' # -1 인덱스는 마지막에 있는 값으로, 아래와 같은 의미를 지닙니다.
잡은물고기[9] = '백상아리'

잡은물고기
```

Python

```
#[Out]

['광어', '고등어', '오징어', '오징어', '광어', '광어', '고등어', '고등어', '백상아리', '백상아리']
```

Python

여기서 광어는 얼마나 잡았는지, 고등어는 얼마나 잡았는지 알고 싶으면 어떻게 할까요? 여기서 앞서 말씀드린 **메서드**를 사용할 수 있답니다. 아래 코드를 실행해 보세요. **메서드 정리**는 **다른 강의**해서 해 드릴거에요. 여기는 얕은물 강좌이니, 어떻게 사용하는지 가볍게 살펴봅시다.

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

```
#[In]

잡은물고기.count('백상아리')
```

Python

```
#[Out]

2
```

Python

대단하군요. 고양이가 백상아리를 잡다니요. 그럼 훈장에 '**백상아리를 잡은 고양이**'를 추가해볼게요. 이 역시 메서드를 사용합니다.

```
#[In]

훈장.append('백상아리를 잡은 고양이')
훈장
```

Python

```
#[Out]

'백상아리를 잡은 고양이'
```

Python

4.6 dict의 속성 알아보기

그런데 이렇게 물고기를 저장하는 것은 비효율 적이겠죠? 우리는 '**백상아리!**' 하면 '**2마리!**' 이렇게 바로 알려주었으면 좋겠는데요. 이것이 가능한 자료형이 Dictionary입니다.

```
# Dictionary의 구조

dic = { 'key' : 'value' }
```

Python

자, 그러면 위에 리스트로 저장되어 있던 물고기를 딕셔너리로 정리해봅시다.

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

```
#[In]

잡은물고기_딕셔너리 = {'광어' : 3, '고등어' : 2, '오징어' : 2, '백상아리' : 2}
```

Python

어떤가요? 보다 간편해졌죠? 우리는 앞으로 아래와 같이 key 값으로 value를 호출할 수 있습니다.

```
#[In]

잡은물고기_딕셔너리['광어']
```

Python

```
#[Out]

2
```

Python

값을 추가하고 싶을 때에는 아래와 같은 방법을 사용할 수 있습니다.

```
#[In]

잡은물고기_딕셔너리['고래'] = 1
잡은물고기_딕셔너리
```

Python

```
#[Out]

{'광어' : 3, '고등어' : 2, '오징어' : 2, '백상아리' : 2, '고래' : 1}
```

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

값을 지울 때에는 아래와 같은 방법을 사용합니다.

```python
#[In]

del 잡은물고기_딕셔너리['고래']
#del 키워드는 다른 자료형에서도 데이터를 지울 때 사용할 수 있습니다.

잡은물고기_딕셔너리
```

Python

```python
#[Out]

{'광어' : 3, '고등어' : 2, '오징어' : 2, '백상아리' : 2}
```

Python

Key값만 따로 알고 싶을 때와 Value값만 따로 알고 싶을 때에는 아래와 같은 방법을 사용합니다.

```python
#[In]

print(잡은물고기_딕셔너리.keys())
print(잡은물고기_딕셔너리.values())
print(잡은물고기_딕셔너리.items())
```

Python

```python
#[Out]

dict_keys(['광어', '고등어', '오징어', '백상아리'])
dict_values([3, 2, 2, 2])
dict_items([('광어', 3), ('고등어', 2), ('오징어', 2), ('백상아리', 2)])
```

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

4.7 set의 속성 알아보기

set은 집합이에요. 중복을 허락하지 않죠! 혹시 차집합, 합집합, 교집합 아시나요? 그 집합입니다!
자, 그럼 어떻게 사용할까요?

```python
#[In]

잡은물고기_집합 = set(잡은물고기)
잡은물고기_집합
```

Python

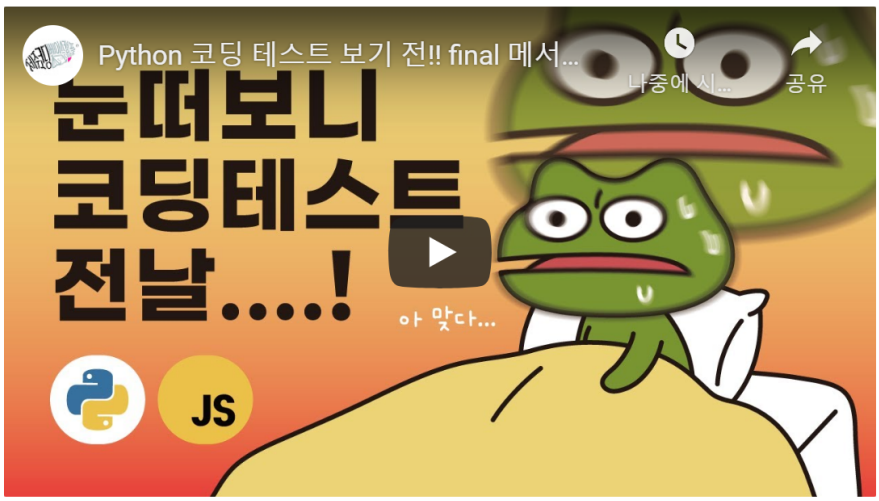
```python
#[Out]

잡은물고기 = {'광어', '고등어', '오징어', '백상아리'}
```

Python

어떤가요? 중복이 사라졌습니다! 이제 잡은 물고기의 종류를 한 번에 볼 수 있죠!

4.8 list, tuple, set, dict에 대해 더 알아보기

자료형끼리의 사칙연산과 메서드는 한 번 정리할 필요가 있어요. 더 공부하기를 원하신다면, 아래 영상을 참고하여 각 자료형 옆에 있는 메모 노트에 메모해 주세요. 하지만 기억하세요. 먼저 빠르게 훑어 간단한 웹이나 앱, 게임이나 IoT와 같은 결과물을 만들고, 나중에 좀 더 깊게 파봐도 늦지 않습니다. 너무 많은 내용을 한꺼번에 담으려 하지 마세요.



Youtube '제주코딩베이스캠프'의 영상입니다😉

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

5. 각 변수들을 형 변환 해보기

자, 형 변환에 대해 좀 더 깊게 알아봅시다.

```python
#[In]

print(type(int(3.5)))
print(int(3.5))
print(type(float(3)))
print(float(3))
print(type(str(3))
print(str(3))
```

Python

```
#[Out]

<class 'int'>
3
<class 'float'>
3.0
<class 'str'>
3
```

Python

앞서 말씀드린 것처럼 기존에 자료형에서 다른 자료형으로 바꾸는 것을 형 변환이라고 합니다. 아래처럼 `input` 함수를 이용하면 숫자나 문자를 입력받을 수 있는데요. 둘 다 `str`로 변수를 받기 때문에 주의해서 사용을 해야 합니다.

```python
#[In]

x = input('좋아하는 숫자를 입력하세요 :')
y = input('더할 숫자를 입력하세요 :')
print(x + y)
print(type(x))
print(type(y))
```

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

```
#[Out]

좋아하는 숫자를 입력하세요 : 123
더할 숫자를 입력하세요 : 123
123123
<class 'str'>
<class 'str'>
```

Python

파이썬에서의 형 변환은 아주 많은 방법들이 있지만 **내장함수(built in-functions, 이 키워드는 매우 중요하니 꼭 암기해두세요.)**를 이용해서 아주 쉽게 할 수 있습니다. 어떤식으로 쓰이는지 간단한 예시를 통해 같이 한번 살펴봅시다.

형변환

✨ int	다른 자료형을 정수형으로 변환
✨ str	다른 자료형을 문자열로 변환
✨ float	다른 자료형을 실수형으로 변환
✨ list	다른 자료형을 리스트로 변환
✨ tuple	다른 자료형을 튜플형으로 변환
✨ set	다른 자료형을 집합 자료형으로 변환
✨ dict	다른 자료형을 사전형으로 변환
+ New	

- **int로 형변환**

```
#[In]

오늘잡은_물고기_수 = '371'
print(type(오늘잡은_물고기_수))
print(type(int(오늘잡은_물고기_수)))
print(int(오늘잡은_물고기_수)
```

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

```
#[Out]

<class 'str'>
<class 'int'>
371
```

Python

- string으로 형변환

```
#[In]

오늘잡은_물고기_수 = 371
print(type(오늘잡은_물고기_수))
print(type(str(오늘잡은_물고기_수)))
print(str(오늘잡은_물고기_수))
```

Python

```
#[Out]

<class 'int'>
<class 'str'>
'371'
```

Python

- bool형으로 형변환

```
#[In]

print("bool('test') : ", bool('test!!'))
print("bool(1) : ", bool(1))
print("bool(0) : ", bool(0))
print("bool(-1) : ", bool(-1))
print("bool(' ') : ", bool(' '))
print("bool('') : ", bool(''))
print("bool(None) : ", bool(None))
```

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

```
#[Out]

bool('test') : True
bool(1) : True
bool(0) : False
bool(-1) : True
bool(' ') : True
bool('') : False
bool(None) : False
```

Python

bool()함수는 **인자값**(아규먼트, argument, parameter와는 차이가 있으니 함수에서 정리해드리도록 하겠습니다.)을 Boolean 자료형으로 형변환하게 됩니다. 부울 값은 True와 False로 나뉩니다.

부울 값은 **직접 입력된 값일 수도 있고 부울 연산에 의해 나온 결과값**일 수도 있습니다.
예를 들어 x = 10 일 때 x > 100은 False이죠. 또한 이미 Python 내에서 규정한 부울 값일 수도 있습니다. 예를 들어 0은 False, 0을 제외한 다른 숫자는 True입니다.

💡 list, tuple, dict, set은 형변환을 어떻게 할까요? 옆 노트에 정리해보세요.

---

6. 출력을 하는 여러가지 용법

`print` 내장함수(built-in functions, 빌트인펑션)를 사용하여 출력하는 방법에는 여러가지 방법이 있습니다. 물론 우리가 사용하는 `colab`이나 `jupyter notebook`은 마지막 라인에 한하여 `print`를 쓰지 않아도 출력합니다.

```
print('1. 제 이름은 ', 이름, '입니다. 제 나이는 ', 나이, '입니다')
print(f'2. 제 이름은 {이름}입니다. 제 나이는 {나이}입니다.')
print('3. 제 이름은 {}입니다. 제 나이는 {}입니다.'.format(이름, 나이))
print('4. 제 이름은 %s입니다. 제 나이는 %d입니다.'%(이름, 나이))
```

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

여기서 비교적 최근에 나온 2번 문법이 자주 사용됩니다. 4번은 잘 사용되지 않으니 참고바랍니다.

1. `,(콤마)` 로 연결하는 방법입니다. 콤마의 단위는 오브젝트입니다. `이름`, `나이` 모두 오브젝트의 한 단위이고 `'1. 제 이름은'`, `'입니다. 제 나이는'` 등의 문자열도 모두 오브젝트입니다. 앞에서는 변수라고 배웠죠? 변수에 넣을 수 있는 모든 값 또는 변수도 오브젝트입니다. 예를 들어 아래와 같이 변수나 그 값을 직접 넣는 것은 같은 값을 출력합니다.

```python
힘 = 12
print('제 힘은 ', 힘, '입니다.')
print('제 힘은 ', 12, '입니다.')
```

Python

2. python 3.6 version 이상에서는 f-string 용법을 지원합니다. 사용하는 방식은 `{}(중괄호)`에 변수 이름을 넣으시면 됩니다.
3. format을 이용한 문자열 포매팅 방식입니다. 앞서 `dir('문자열')`에서 던더함수 뒤에 있는 메서드를 확인해보시면 `format`메서드가 있습니다. `{}(중괄호)`로 변수의 자리를 비워놓고 `'문자열'.format()` 괄호 안에 넣고 싶은 오브젝트를 넣어주는 방법입니다. 역시 변수는 콤마로 구별해 넣습니다.
4. 잘 사용하지 않는 방법입니다. 포맷코드를 이용한 문자열 포매팅입니다. 넣고 싶은 오브젝트는 % 뒤에 넣습니다. 넣을 값에 맞춰서 알맞은 자료형에 대한 포맷코드를 선택해야 합니다. 아래 코드표는 참고삼아 훑고 넘어가세요.

포멧 코드의 종류

%c	문자
%s	문자열
%d	정수
%f	실수
%b	2진수
%o	8진수
%x	16진수

+ New

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