




Google 아이디만 있다면 사용할 수 있는 Google Colab!
여러가지 불편하게 설치하는 작업 없이 여러분의 빠른 위니브 월드 진입을 도와줄 것입니다.



1. Google Colab 간단한 사용법!


- 1. google에 로그인을 해주시고, google에서 google colab이라고 검색을 합니다.





google colab


X





 전체

 뉴스

 동영상

 이미지

 도서

 더보기

설정

도구

검색결과 약 3,330,000개 (0.35초)

colab.research.google.com

이 페이지 번역하기

Google Colab

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your ...

Introduction to Colab and Python · Tensorflow with GPU · Overview of Colaboratory

이 페이지를 여러 번 방문했습니다. 최근 방문 날짜: 20. 6. 16

colab.sandbox.google.com

notebooks

Colaboratory란? - Google Colab

Google 드라이브 계정에서 스프레드시트를 비롯한 데이터를 Colab 메모장으로 가져오거나 GitHub 등의 여러 다른 소스에서 데이터를 가져올 수 있습니다. Colab을 ...

zzsza.github.io

data

2018/08/30

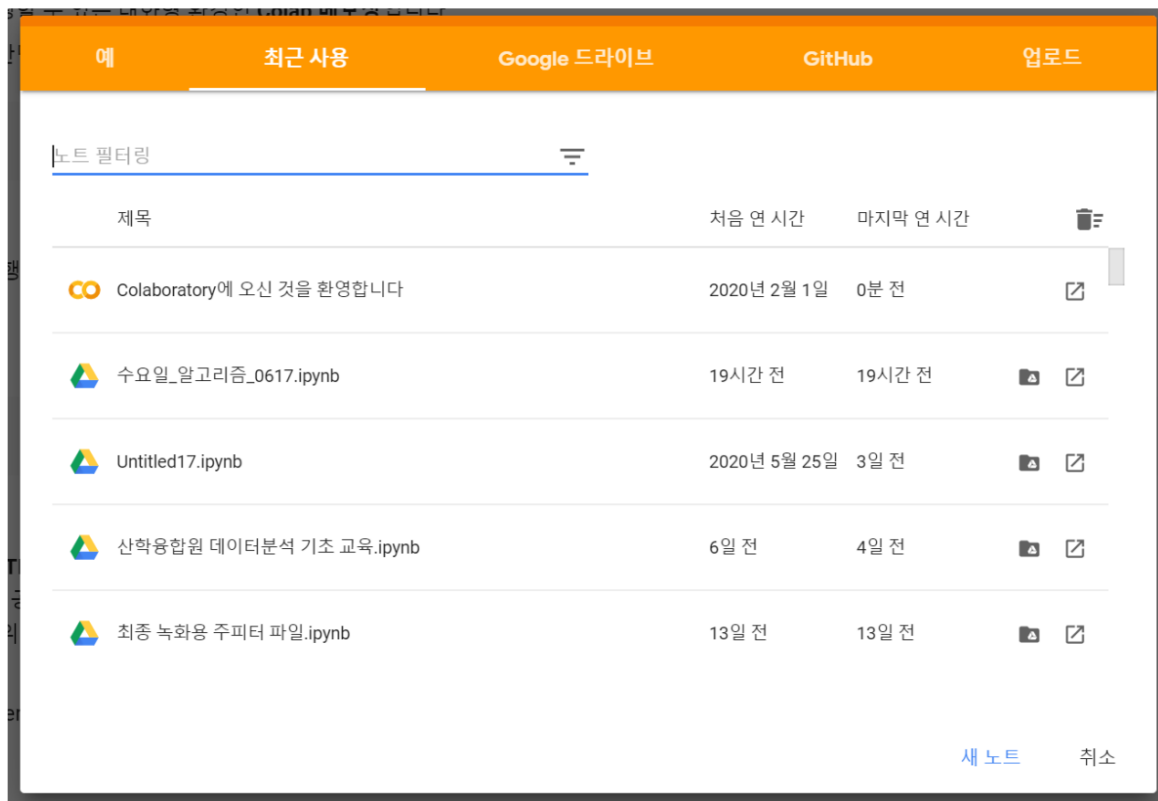
google-colab

Google Colab 사용하기 · 어쩐지 오늘은

2018. 8. 30. - Google의 Colab 사용법에 대해 정리한 글입니다 이 글은 계속 업데이트 될 예정입니다! 목차 UI 상단 설정 구글 드라이브와 Colab 연동 구글 ...

구글 드라이브와 Colab 연동 · 구글 드라이브와 로컬 연동 · Github 코드를 Colab에서 ...

2. 다음과 같이 문서를 설정하는 창이 뜨는데요. 여기서 **새노트를 클릭**해주세요. 기존에 사용하신 파일이 있다면 업로드 해서 편집도 가능합니다. 그리고 **모든 파일은 자동으로 google drive에 저장**됩니다!



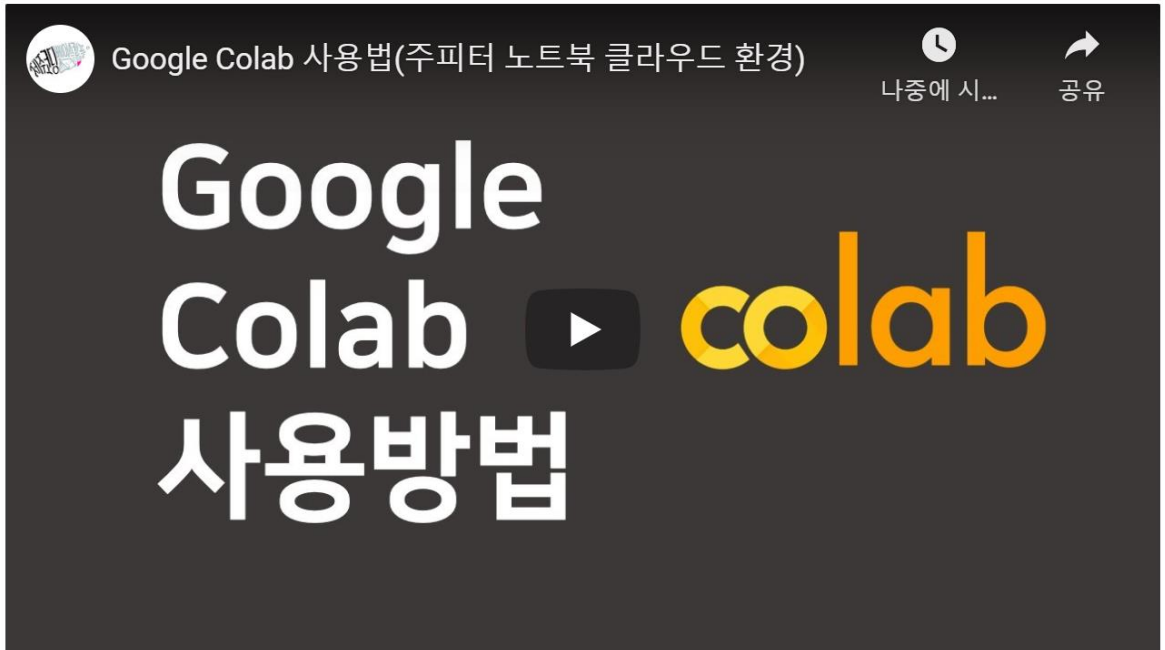
3. 아래와 같이 새 노트가 만들어집니다. 메뉴를 설명해 드리도록 하겠습니다!

- 맨 위 제목을 설정하는 창이 있습니다. 누르시면 파일의 제목을 수정할 수 있어요. 오른쪽에 보시면 누군가와 파일을 공유할 수도 있습니다.
- 파일, 수정, 보기, 삽입, 런타임, 도구, 도움말 등의 상태창이 떠요. 이건 지금 단계에서 필요한 것 아니 오른쪽에 위로 화살표가 그려진 것을 눌러 접어두세요. (상세 기능 설명은 아래 동영상 참고하세요. 그러나, 초반에 너무 상세한 내용은 오히려 즐거운 코딩을 방해합니다.)
- 그 아래 재생 버튼을 누르면 코드를 실행할 수 있어요! 단축키로는 **Ctrl + Enter** 입니다. 이 단축키는 많이 사용하니 알고 있으셔야 합니다!
- 그 아래 코드를 삽입할 수도 있어요. 일반 텍스트를 삽입하여 설명을 추가할 수도 있죠. 일반 텍스트는 **마크다운** 을 활용하고 있어요. 코드를 실행시키고 바로 코드 셀을 추가하는 명령어는 **Alt + Enter** 입니다. 역시 많이 쓰는 단축키니 꼭 기억해두세요.



2. 상세한 사용법!

1. colab : 처음에 말씀드렸듯이, 초반에 접하는 상세한 내용은 즐거운 코딩을 방해합니다. 가능하다면 이 챕터를 건너뛰시고, 어느정도 적응이 된 후 다시오세요. 😊



2. jupyter notebook : 로컬에서 jupyter notebook을 설치하시면 조금 더 쾌적한 환경에서 개발이 가능합니다. 그렇지만, 기억하세요. 구글 colab은 무려 Ram 할당량이 12G 입니다!





1편 라이캣의 탄생

1. 캣의 일상
2. 캣의 정보 변경
3. 각 변수들의 타입을 알아보기
4. 각 변수들의 속성을 알아보기
 - 4.1 int의 속성 알아보기
 - 4.2 float의 속성 알아보기
 - 4.3 str의 속성 알아보기
 - 4.4 bool의 속성 알아보기
 - 4.5 list, tuple의 속성 알아보기
 - 4.6 dict의 속성 알아보기
 - 4.7 set의 속성 알아보기
 - 4.8 list, tuple, set, dict에 대해 더 알아 보기
5. 각 변수들을 형변환 해보기
6. 출력을 하는 여러가지 용법

1. 캣의 일상

한편, 유니브 월드 외곽에서는 생선가게를 운영하는 평민 '캣'이 살고 있었다.



생선 가게를 운영하는 평범한 소년 '캣'이 있습니다. 이 소년은 아픈 어머니를 대신하여 새벽이면 험한 바다에 나가 고기를 잡고, 오후가 되면 생선을 파는 생활을 반복하였어요.



캣에게는 비밀이 한가지 있었어요. 부모님께서 신신당부하여 남들에게 말하지 않았지만, 자신이 원할 때 자신의 능력치를 볼수 있는 신기하고 특별한 능력이 있었어요.

"나에겐 **고기잡기**와 **고기팔기 스킬**이 있다냥, 매일매일 훈련해서 이 두개의 스킬 만큼은 최고의 스킬로 만들어보자냥!"

캣은 묵묵히, 매일 실력을 갈고닦았습니다.

```
#[In]

# 라이캣의 개인정보 입력하기

이름 = '캣'
설명 = '위니브 월드 외각에 살고 있는 생선가게 주인 캣(cat)'
나이 = 10
오늘_잡은_물고기 = '10'
키 = '45cm'
몸무게 = 1.2
육식 = True
초식 = True
돈 = 1000
훈장 = []
기술 = ['고기잡이', '고기팔기']

'''
라이캣의 개인정보입니다.
성장함에 따라 해당 값이 변합니다.

주의) 마음대로 성장시키지 마십시오.
'''
```

Python ▾

이름, 나이, 오늘_잡은_물고기 등을 변수라고 합니다. 변수는 변할 수 있는 수입니다. 또, 맨 위에 있는 # 으로 되어 있는 부분과 ''' 로 감싸여져 있는 부분은 주석이라고 해요. Code의 양이 많거나 복잡하여 짧은 시간 내 이해하기 힘들 때 이해를 돕기 위해 주석이 필요합니다. 이 부분은 실행되지 않습니다. 코드를 잠시 보류하는 용도로도 사용하죠.

이렇게 하고 **Alt + Enter** 를 실행해 보세요. 아래 셀이 추가되었죠? 따로 메시지는 뜨지 않았을 거예요. 무언가 실행시키지 않았기 때문입니다. 실행은 그 셀만 실행하는 **Ctrl + Enter** 와 실행하고 셀 아래 셀을 추가하는 **Alt + Enter** 두가지가 있습니다.

💡 주석으로 변경을 원하는 코드에서 **Ctrl + /** 를 누르면 주석처리 됩니다. 여러줄도 가능해요.

2. 캣의 정보 변경

```
#[In]

# 라이캣의 개인정보 변경하기

나이 = 11
이름, 나이
```

Python ▾

이렇게 하고 **Alt + Enter** 를 실행해 보세요. 나이에 11이 출력되었죠? 이처럼 변수는 변경할 수 있습니다. 그리고, 위에 있는 변수를 가져와 사용할 수도 있어요.

```
#[In]

나이 = 나이 + 1
나이
```

Python ▾

연산은 나중에 할 예정이지만, 이렇게 덧셈을 할 수도 있습니다. 그럼 아래와 같이 덧셈을 해볼게요.

```
#[In]

# 오늘_잡은_물고기 = '10'
오늘_잡은_물고기 = 오늘_잡은_물고기 + 1
```

Python ▾

이렇게 연산을 실행하게 되면 Error 문구가 뜨게 됩니다. 그 이유는 오늘 잡은 물고기는 문자열이기 때문인데요. 이처럼 파이썬과 같은 **고급프로그래밍 언어**에서는 그 형태를 구분하여 걸맞는 연산이 가능하도록 장치를 해두었습니다.

💡 Python이 무엇이고, 무엇을 할 수 있는지는 아래 영상을 참고해주세요. 고급 프로그래밍 언어가 어떤 뜻인지도 설명하고 있습니다. (책으로 보시는 분들은 YouTube에서 제주코딩베이스캠프를 검색해주세요.)

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





가능하다면, 1편인 Front-end도 시청해주세요. 😊

조금 더 알려드리자면, 변수에는 몇 가지 규칙이 있어요.

1. 변수 중간에 띄어쓰기를 하지 않습니다. 띄어쓰기를 명시하고 싶으실 때에는 언더스코어(`_`)를 사용하세요!
2. 첫 글자는 숫자나 특수문자를 쓰지 않습니다. 물론 언더스코어(`_`)제외입니다.
3. 첫 글자를 대문자로 쓰지 않습니다.(Class가 대문자로 쓰기 때문이지만, 대문자로 써도 실행됩니다.)
4. 예약어를 사용하지 않습니다. 예를 들어 뒤에 `print` 구문이 나오는데요. 이러한 함수명이나 구문은 변수명으로 사용하지 않습니다.
5. 변수명은 대소문자를 가립니다! `Apple` 과 `apple` 은 다른 변수가 됩니다.

```
#[In]

# 오늘_잡은_물고기 = '10'
오늘_잡은_물고기 = int(오늘_잡은_물고기) + 1
오늘_잡은_물고기
```

Python ▾

자, 위와 같이 변경하고 `Alt + Enter` 를 눌러 실행시켜보세요! 연산이 되었죠? 그 이유는, `int` 라는 형변환 함수가 문자열을 숫자로 바꾸어 주었기 때문입니다. 형변환은 이번 챕터 5장에서 자세히 살펴볼 것입니다.

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. 각 변수들의 타입을 알아보기

이번에는 위에서 코드를 복사해와서 아래와 같이 작성 후 실행해보세요.

```
#[In]

print(type(이름)) # '켓'
print(type(나이)) # 현재 12
print(type(오늘_잡은_물고기)) #현재 11
print(type(몸무게))
print(type(육식)) #True
print(type(기술)) #['고기잡이', '고기팔기']
```

Python ▾



`print`를 써도 되고 쓰지 않아도 되는 것인가요? 네, 주피터 노트북에서는 `print`를 쓰지 않아도 마지막에 있는 변수는 출력을 합니다.(Python 기본 에디터 사용시 Error!) 그러나 여러개를 출력할 때에는 꼭 `print`를 명시해주셔야 합니다.

4. 각 변수들의 속성을 알아보기

위에서 실행시킨 결과값은 아래 주석과 같이 나왔을거예요! 물론 더 많은 자료형이 있지만, 이 수업은 기초 수업이기 때문에 간단한 내용들만 살펴볼 것입니다.

```
#[In]

print(type(이름)) # class 'str' - 문자열
print(type(나이)) # class 'int' - 정수
print(type(오늘_잡은_물고기)) # class 'int' - 정수
print(type(몸무게)) # class 'float'은 실수입니다.
print(type(육식)) # class 'bool' - 참거짓
print(type(기술)) # class 'list' - 배열
```

Python ▾

각 변수들에는 속성이 있습니다. 예를 들어 스트링의 경우에는 덧셈을 하면 이어 붙인다든지 하는 속성이요.

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]

x = '10'
y = 10
print(x + x)
print(y + y)
```

Python ▾

위와 같이 실행을 시켰을 경우 `'10' + '10'` 은 `'1010'` 이 되지만, `10 + 10` 은 `20` 이 됩니다. 이렇게 각 자료형마다 특징들이 있고, 이번 챕터에서는 이 특징들을 살펴볼 생각이예요. 그러나 걱정마세요. 너무 깊게 다루지는 않을 것입니다.



Q : 깊게 다루지 않고 데이터 분석, 서비스 개발, IoT, 게임 개발 등을 할 수 있나요?

A : 아닙니다! 하지만, 초급자 분들은 문법에 매몰 되시면 안되요. 일단 한 두 서클을 돌아보시면, 자연스레 이해되는 문법들이 있고, 또 그렇게 한 번 성공한 결과물을 갖게되면 자신감도 생기거든요! 그러니 초급자 분들은 한 서클(결과물을 만드는 과정)을 도시는 것에 초점을 맞춰서 공부해주세요!

4.1 int의 속성 알아보기

int는 정수입니다! 정수가 가진 속성에 대해 알아보도록 하겠습니다. 우선 아래와 같이 실행시킨 다음에 말씀드리도록 하겠습니다.

```
#[In]

나이 = 10
print(type(나이))
print(dir(나이))
```

Python ▾

그럼 아래와 같이 무시무시한 길이의 코드가 나타납니다. 겁내지 마세요. 다 알 필요도 없을 뿐더러, 우리는 아주 얇고 넓게, 실용적인 부분만 배울 것이기 때문입니다.

일단 첫줄 먼저 설명해드릴게요. `<class 'int'>`, class는 맨 뒤에 챕터에서 다루게 되고, int는 우리가 앞에서 공부한 것처럼 **정수형**을 나타냅니다. 이처럼 `type(변수)`를 해보시면 **원하는 변수의 타입을 알아낼 수 있어요.**

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 'int'>
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '__floordiv__', '__format__', '__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__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv__', '__trunc__', '__xor__', 'bit_length', 'conjugate', 'denominator', 'from_bytes', 'imag', 'numerator', 'real', 'to_bytes']
```

Python ▾

여러분, 아래와 같이 정수와 실수를 더하면 얼마가 나올까요? 네, **20.1** 입니다. 그런데 다른 언어들은 이렇게 융통성이 있지 않아요. 만약 이 연산이 다른 언어였다면(JS는 더 융통성이 있으니 제외입니다.) 아래와 같이 말했을 거예요.

#[In]

#나이는 10이고, 몸무게는 10.1 입니다.

나이 + 몸무게

Python ▾

실수로 더할꺼니? 그럼 실수로 바꿔줘야지. 정수로 더할꺼니? 그럼 뒤에 수에서 0.1을 포기해.

그런데 파이썬은 놀랍게도(!?), 이 연산이 가능합니다. 위 처럼 실행을 시켰다면 **20.1**의 출력 결과가 나올게요. 그럼 아래와 같은 코드는 어떨까요?

#[In]

10 + '10'

Python ▾

이것을 실행시키면 또 아래와 같은 무시무시한 에러가 나옵니다. 앞으로 자주 만나게 될 에러명이기 때문에 한번 읽어보죠.

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]

...
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Python ▾

타입에러인데 정수형(int)과 문자열(str)의 덧셈을 허락하지 않는다고 합니다. 자, 이렇게 덧셈에 대해 어떻게 할 것인지에 대해 정의되어 있는 부분이, `'__add__'`입니다! 이부분은 class 챕터에서 더 배우게 되실거예요. 곱하기는 `'__mul__'`, 비교는 `'__eq__'`합니다.

아, 언더바(_)는 하나가 아니라 두개예요! 그래서 던더함수라고 부르기도 하죠. 매직메서드라고 부르는 것이 통상적입니다.

그렇다면, 언더바가 없는 것들의 정체는 무엇일까요? 아래와 같은 코드를 실행해볼게요.

```
#[In]

나이 = 10
print(나이.bit_length())
print(bin(나이))

나이_90년후 = 100
print(나이_90년후.bit_length())
print(bin(나이_90년후))
```

Python ▾

이것들은 `.`을 찍어 사용할 수 있어요. `bit_length()`만 사용해보았습니다. 이 **메서드**(지금은 점을 찍어서 사용할 수 있는 코드라고만 이해를 해주세요.)는 각 숫자를 bit로 변환한 자리숫자를 출력해준답니다! bit는 0과 1로 이루어진 세계입니다. **2진법**을 사용하죠.

자, 너무 많은 내용을 했어요. 모두 이해하실 필요 없습니다. 여기서 중요한 핵심은 `.`을 찍어 활용할 수 있는 코드를 `dir`로 볼 수 있다는 사실만 알고 넘어가도록 하겠습니다!

4.2 float의 속성 알아보기

float형은 실수입니다! 실수가 있으니 허수도 있을까요? 네, 물론입니다. 하지만, 우리 수업에서 그렇게 어려운 수학적인 내용은 다루지 않아요. 아래와 같이 실행해본 후 이 챕터는 넘어갑시다.

```
#[In]

# 몸무게는 10.1kg 입니다!
print(type(몸무게))
print(dir(몸무게))
```

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



4.3 str의 속성 알아보기

str 자료형은 스트링이라 읽습니다. 이 자료형은 작은따옴표(' ')나 큰따옴표(" ") 또는 삼중따옴표(""" """)로 둘러싸서 나타냅니다. 삼중따옴표를 사용할 경우에는 줄단위의 문자열을 나타낼 수 있습니다. 이러한 문자열은 **시퀀스 자료형**에 해당됩니다.

Sequence, 순서가 있는 자료형이라는 뜻으로 각 요소들은 해당하는 **Index 값을 갖습니다**. 파이썬에서 **가장 많이 사용되는 형태의 자료형**이며, 시퀀스 자료형이 가지는 특성을 통하여 인덱싱, 슬라이싱, 반복 등 여러 연산으로 그 활용 범위가 넓습니다. 파이썬에서는 **문자열, 리스트** 등이 시퀀스 자료형에 해당됩니다.

```
#[In]

# 설명 = '위니브 월드 외각에 살고 있는 생선가게 주인 캣(cat)'
# 기술 = ['고기잡이', '고기팔기']

print(설명[0])
print(설명 [1])
print(설명 [2])
print(기술 [0])
print(기술 [1])
print(기술 [0][0])
print(기술 [0][1])
```

Python ▾

```
#[Out]

위
니
브
고기잡이
고기팔기
고
기
```

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 **설명[0]** 안에 들어가는 0이라는 숫자를 index라고 부릅니다! 그리고 이렇게 index를 활용하여 특정 값을 호출하는 방식을 indexing이라고 하죠!

또, 이 index를 활용하여 문자열에서 원하는 부분만을 추출해낼 수 있습니다.

```
#[In]
```

```
설명 = '위니브 월드 외각에 살고 있는 생선가게 주인 캣(cat)'
print(설명[0:6])
```

Python ▾



인덱스가 start 인 지점에서 end 미만인 지점까지 추출합니다. 여기서 start 를 생략했을 경우 문자열의 시작 지점, end 를 생략했을 경우 문자열의 끝지점이 설정됩니다.

```
#[In]
```

캣의_생년월일 = "2220.02.22"

```
생년 = birth[:4]
```

```
info = birth[5:7]
```

```
mom = birth[8:]
```

```
print("태어난 연도 : ", 생년)
```

```
print("태어난 월 : ", 월)
```

```
print("태어난 일 : ", 일)
```

Python ▾

#[Out]

태어난 연도 : 2220

태어난 지 : 02

태어난 일 : 22

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



`변수명[start:stop:step]` 와 같은 형식으로 사용도 가능합니다. 이 경우 순회가능한 객체에 `start index` 부터 `stop index` 바로 전만큼의 범위에서 k만큼 건너뛰게 됩니다.

```
#[In]
```

```
숫자 = '123456789101112'  
print(숫자[::-1])  
print(숫자[1:7:2])
```

Python ▾

```
#[Out]
```

```
'2111101987654321'  
'246'
```