

초등학생을 위한 AI 캠프

에디슨칼리지 첨단방산공학전공
교육대학원 AI융합교육 전공, 주임교수
슈퍼컴퓨팅센터 센터장

조금원 교수(kwcho@kumoh.ac.kr)

네번째 날

일정

	월	화	수	목	금
09:30~10:30	인공지능이란?	나만의 웹페이지 만들기	파이썬 프로그래밍 (기초)		문제풀기
10:30~11:30	계정 만들기	동화책 만들기	외부초청강연 (주별 1명)	파이썬 프로그래밍 (응용) - 게임만들기 - 수학문제풀기	디지털윤리
11:30~12:30	인공지능으로 짹궁 만들기	슈퍼컴퓨팅센터 투어	1. 정태규 박사 (항공우주연구원) 2. 금창섭 박사 (ERTI) 3. 최선미 박사 (한의학연구원)		시상식 - 인텔코리아 - 슈퍼컴퓨팅센터 - 개근상
다음날 준비사항	사진 10장 가져오기				

파이썬 응용 - 반복문

반복이란?

- 반복(iteration)은 동일한 문장을 여러 번 반복시키는 구조
- 컴퓨터는 인간과 다르게 반복적인 작업을 실수 없이 빠르게 할 수 있다.
이것이 컴퓨터의 가장 큰 장점이다.



왜 반복이 중요한가?

- 하나의 예로 화면에 회사에 중요한 손님이 오셔서 대형 전광판에 '방문을 환영합니다!'를 5번 출력한다고 하자.



```
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
```

복사해서 붙여넣기

만약 1000번 반복해야 한다면?

```
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
...
...
print("방문을 환영합니다!")
```

{

1000번 복사해서 붙여넣기 ?



- 반복 구조를 사용하여야 한다.

```
for i in range(1000):
    print("방문을 환영합니다!")
```

}

1000번 반복시키는 구조

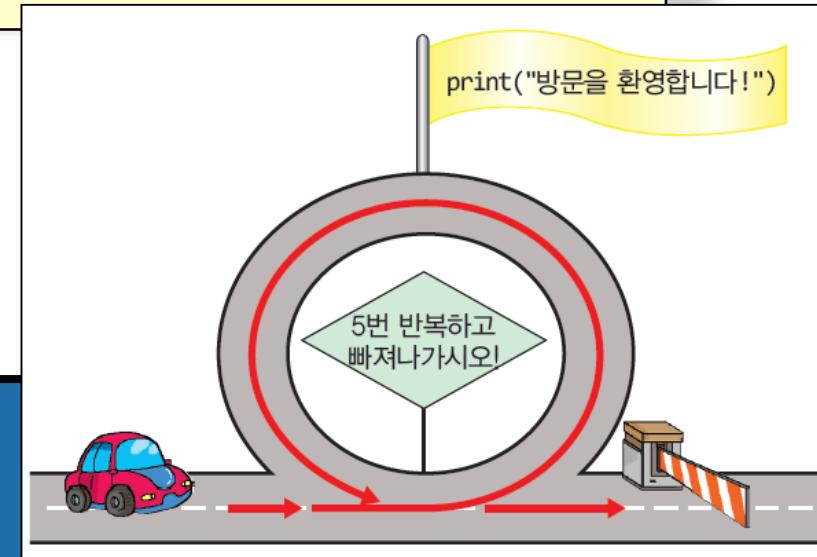
횟수 제어 반복

- 파이썬에서 횟수 제어 반복은 for 루프라고도 한다.

```
for i in [1, 2, 3, 4, 5]:      # 끝에 :이 있어야 함  
    print("방문을 환영합니다.")  # 들여쓰기하여야 함
```

i가 1부터 5까지 변경되면서 반복된다.

방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!



횟수 제어 반복의 진행 과정

```
for i in [ 1, 2, 3, 4, 5 ] :  
    print("방문을 환영합니다.")
```

```
for i in [ 1, 2, 3, 4, 5 ] :  
    print("방문을 환영합니다.")
```

```
for i in [ 1, 2, 3, 4, 5 ] :  
    print("방문을 환영합니다.")
```

```
for i in [ 1, 2, 3, 4, 5 ] :  
    print("방문을 환영합니다.")
```

```
for i in [ 1, 2, 3, 4, 5 ] :  
    print("방문을 환영합니다.")
```

i의 값을 출력해보자.

```
for i in [1, 2, 3, 4, 5]:  
    print("i=", i)
```

```
i= 1  
i= 2  
i= 3  
i= 4  
i= 5
```

숫자들의 합계

```
sum = 0
for i in [1, 2, 3, 4, 5]:
    print(f"{i}를 더하는 중")
    sum += i
print("합계는=", sum)
```

1를 더하는 중
2를 더하는 중
3를 더하는 중
4를 더하는 중
5를 더하는 중
합계는= 15

range() 함수

for 문

for 변수 in range(종료 값) :

문장

0에서 (종료 값-1)까지의 숫자를 반복한다.

반복되는 문장으로
들여쓰기 하여야 한다.

```
for i in range(5):  
    print("방문을 환영합니다!")
```

range(5) == range(0, 4, 1)

방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!

range() 함수

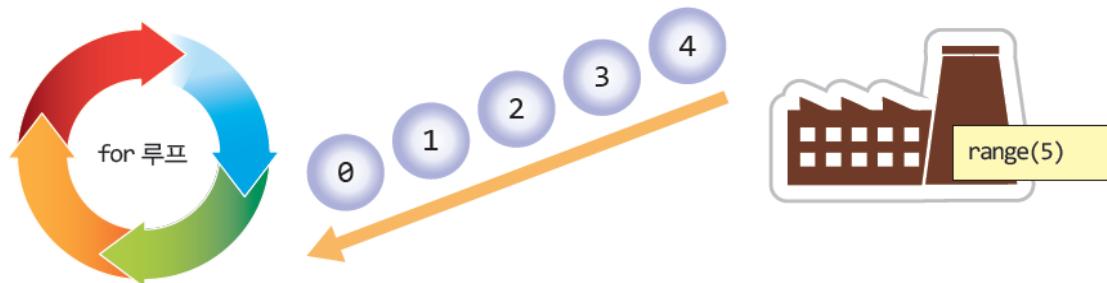
range() 함수

시작값이다.

종료 값이지만 stop은 포함되지 않는다.

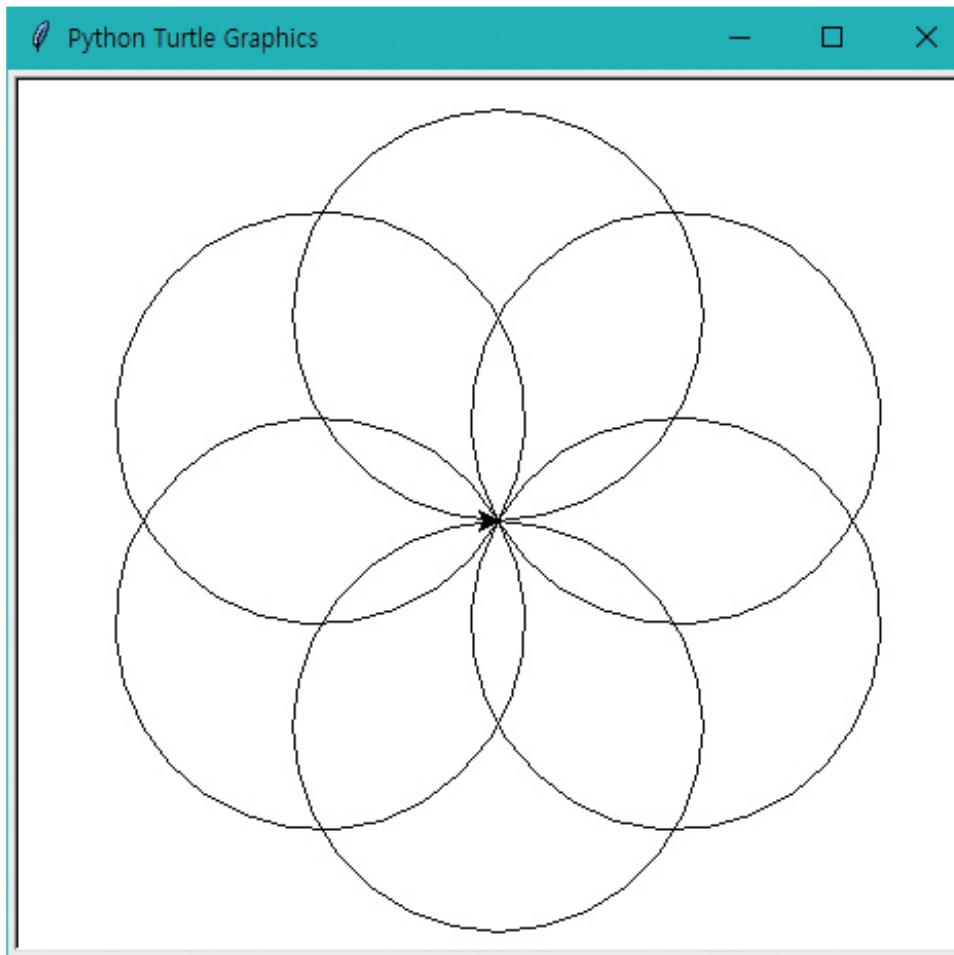
한 번에 증가되는 값이다.

range(start=0, stop, step=1)



실습 1: 6개의 원 그리기

- 6개의 원 그리기



답 1: 반복을 사용하지 않는다면

```
import turtle  
t = turtle.Turtle()
```

t.circle(100) # 반지름이 100인 원을 그린다.
t.left(60) # 60도만큼 왼쪽으로 회전시킨다.

t.circle(100)

t.left(60)

t.circle(100)

t.left(60)

t.circle(100)

t.left(60)

t.circle(100)

t.left(60)

t.circle(100)

답 2: 반복문 사용

```
import turtle  
t = turtle.Turtle()
```

6번 반복해야 하므로 range(6)을 호출하여서 6개의 숫자 [0, 1, 2, 3, 4, 5]를 생성. 각 반복에서는 반지름이 100인 원을 그리고, 거북이를 60도만큼 회전

실습 2: 구구단 출력

- 구구단 중에서 사용자가 원하는 단을 반복문을 이용하여 출력하여 보자.

원하는 단은: 9

$$9 \times 1 = 9$$

$$9 \times 2 = 18$$

$$9 \times 3 = 27$$

$$9 \times 4 = 36$$

$$9 \times 5 = 45$$

$$9 \times 6 = 54$$

$$9 \times 7 = 63$$

$$9 \times 8 = 72$$

$$9 \times 9 = 81$$

Table de multiplication									
Table de 1		Table de 2		Table de 3		Table de 4		Table de 5	
1 × 0 = 0	1 × 1 = 1	2 × 0 = 0	2 × 1 = 2	3 × 0 = 0	3 × 1 = 3	4 × 0 = 0	4 × 1 = 4	5 × 0 = 0	5 × 1 = 5
1 × 2 = 2	1 × 3 = 3	2 × 2 = 4	2 × 3 = 6	3 × 2 = 6	3 × 3 = 9	4 × 2 = 8	4 × 3 = 12	5 × 2 = 10	5 × 3 = 15
1 × 4 = 4	1 × 5 = 5	2 × 4 = 8	2 × 5 = 10	3 × 4 = 12	3 × 5 = 15	4 × 4 = 16	4 × 5 = 20	5 × 4 = 20	5 × 5 = 25
1 × 6 = 6	1 × 7 = 7	2 × 6 = 12	2 × 7 = 14	3 × 6 = 18	3 × 7 = 21	4 × 6 = 24	4 × 7 = 28	5 × 6 = 30	5 × 7 = 35
1 × 8 = 8	1 × 9 = 9	2 × 8 = 16	2 × 9 = 18	3 × 8 = 24	3 × 9 = 27	4 × 8 = 32	4 × 9 = 36	5 × 8 = 40	5 × 9 = 45
1 × 10 = 10		2 × 10 = 20		3 × 10 = 30		4 × 10 = 40		5 × 10 = 50	
Table de 6		Table de 7		Table de 8		Table de 9		Table de 10	
6 × 0 = 0	6 × 1 = 6	7 × 0 = 0	7 × 1 = 7	8 × 0 = 0	8 × 1 = 8	9 × 0 = 0	9 × 1 = 9	10 × 0 = 0	10 × 1 = 10
6 × 2 = 12	6 × 3 = 18	7 × 2 = 14	7 × 3 = 21	8 × 2 = 16	8 × 3 = 24	9 × 2 = 18	9 × 3 = 27	10 × 2 = 20	10 × 3 = 30
6 × 4 = 24	6 × 5 = 30	7 × 4 = 28	7 × 5 = 35	8 × 4 = 32	8 × 5 = 40	9 × 4 = 36	9 × 5 = 45	10 × 4 = 40	10 × 5 = 50
6 × 6 = 36	6 × 7 = 42	7 × 6 = 42	7 × 7 = 49	8 × 6 = 48	8 × 7 = 56	9 × 6 = 54	9 × 7 = 63	10 × 6 = 60	10 × 7 = 70
6 × 8 = 48	6 × 9 = 54	7 × 8 = 56	7 × 9 = 63	8 × 8 = 64	8 × 9 = 72	9 × 8 = 72	9 × 9 = 81	10 × 8 = 80	10 × 9 = 90
6 × 10 = 60		7 × 10 = 70		8 × 10 = 80		9 × 10 = 90		10 × 10 = 100	

```
dan = int(input("원하는 단은: "))
```

```
for i in range(1, 10, 1):
    print(f'{dan}*{i}={dan*i}')
```

도전문제



도전문제

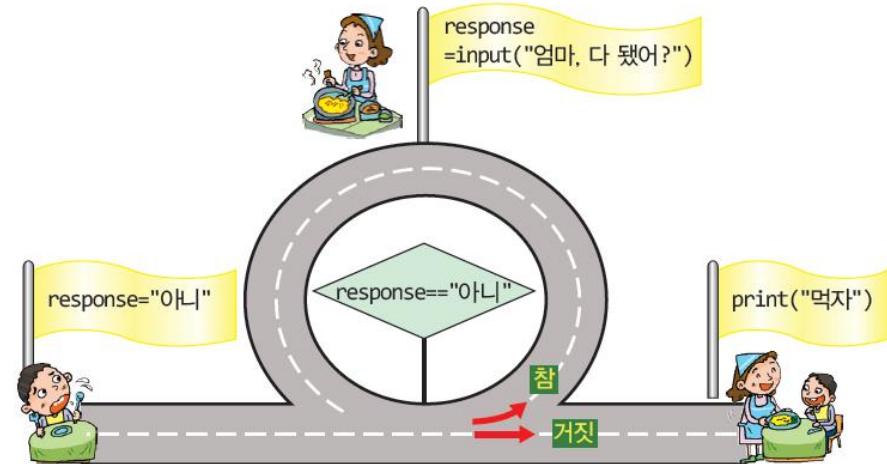
구구단의 1단부터 9단까지를 모두 출력하도록 위의 프로그램을 수정해보자.

- 1, 9까지를 사용하므로 for문과 range() 함수를 사용합니다 .
- 구구단이므로, 1에 대해서 1~9, 2에 대해서 1~9 등으로 코딩을 합니다.
이것은 for 문에 for 문을 사용합니다.
- 프린트문을 사용하여 구구단 계산값을 출력합니다.

조건 제어 반복

- 조건 제어 반복은 어떤 조건이 만족되는 동안 반복하는 구조

반복의 횟수는 모르지만,
반복의 조건은 알고 있는 경우에 사용



while 문

while 루프

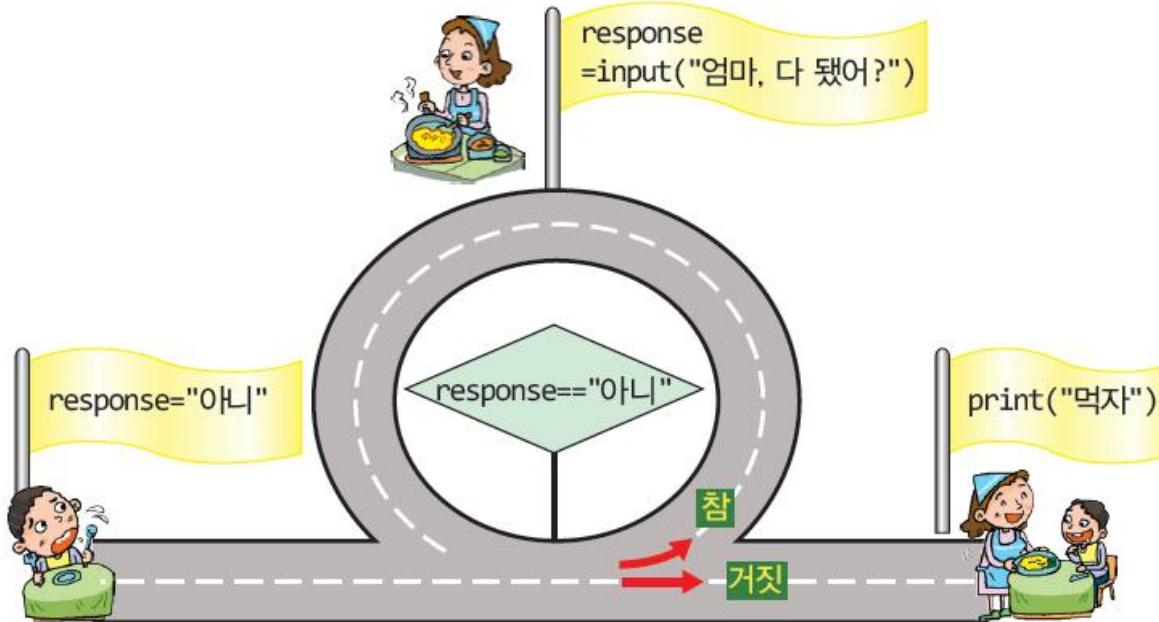
while 조건 :

반복 문장

반복을 하는 조건이다. 조건이 참이면 반복을 계속한다.

반복되는 문장이다.

while 문

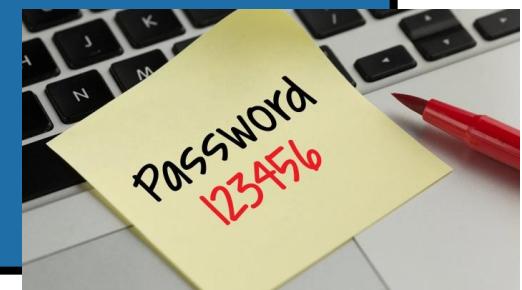


```
response = "아니"
while response == "아니":
    response = input("엄마, 다 됐어?")
    print("먹자")
```

문제 1: 로그인 프로그램

- 사용자가 암호를 입력하고 프로그램에서 암호가 맞는지를 체크한다고 하자.

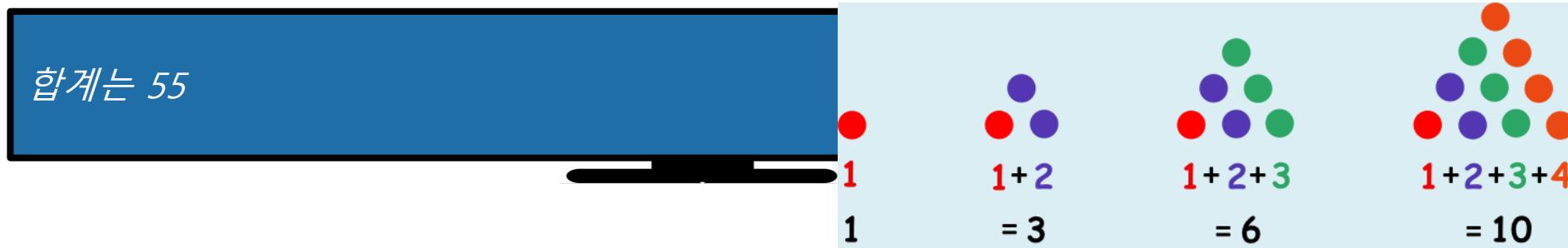
```
암호를 입력하시오: idontknow  
암호를 입력하시오: 12345678  
암호를 입력하시오: password  
암호를 입력하시오: pythonisfun  
로그인 성공
```



```
password = ""  
  
while password != "pythonisfun":  
    password = input("암호를 입력하시오: ")  
print("로그인 성공")
```

문제 2: 1부터 10까지의 합 계산

- 예를 들어서 1부터 10까지의 합을 계산하는 예제를 while 루프로 작성해 보자.



```
count = 1  
sum = 0  
while count <= 10 :  
    sum = sum + count  
    count = count + 1  
print("합계는", sum)
```

for 문으로 바꿔보기
range() 함수 사용

파이썬의 자료형

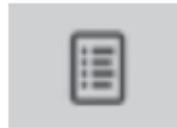
데이터 구조

리스트



```
myList = [ 10, 20, 30, 40, 50 ]
```

튜플



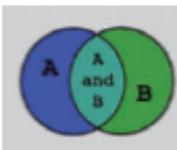
```
myTuple = ( 10, 20, 30, 40, 50 )
```

딕셔너리



```
myDict = { 1:"one", 2:"two", 3:"three" }
```

세트



```
mySet = { "one", "two", "three" }
```

리스트 선언

Syntax: 리스트

형식 리스트_이름 = [요소1, 요소2, ...]

예 temps = [28, 31, 33, 35, 27, 26, 25] 초기값을 가진 리스트를 생성한다.

리스트의 이름

대괄호를 사용하여 요소에 접근한다.

Print (e) 35

fruits = ["사과", "수박", "참외", "바나나"]

리스트를 변수에 저장한다.

리스트의 참조값을 변수에 저장한다.

대괄호로 데이터를 묶는다.

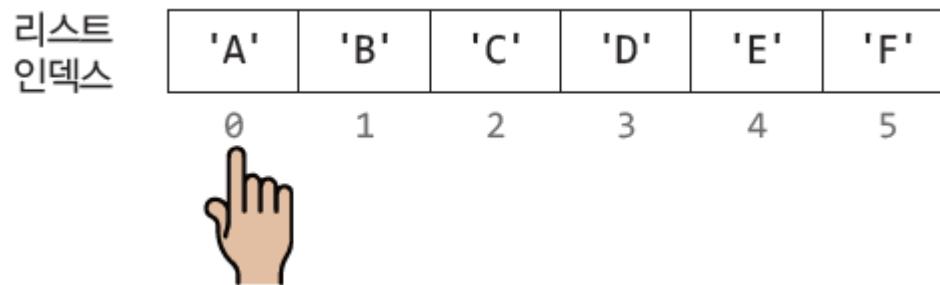
```
>>> friends = [ "게이츠", "잡스", "브린", "베이조스" ]  
>>> friends  
['게이츠', '잡스', '브린', '베이조스']
```



리스트 항목 접근하기(리스트 인덱스)

- 인덱스(index)란 리스트에서의 항목의 위치(번호)이다.
- 0부터 시작한다.

```
>>> letters = ['A', 'B', 'C', 'D', 'E', 'F']
```



```
>>> letters[0]  
A  
>>> letters[1]  
B  
>>> letters[2]  
C
```

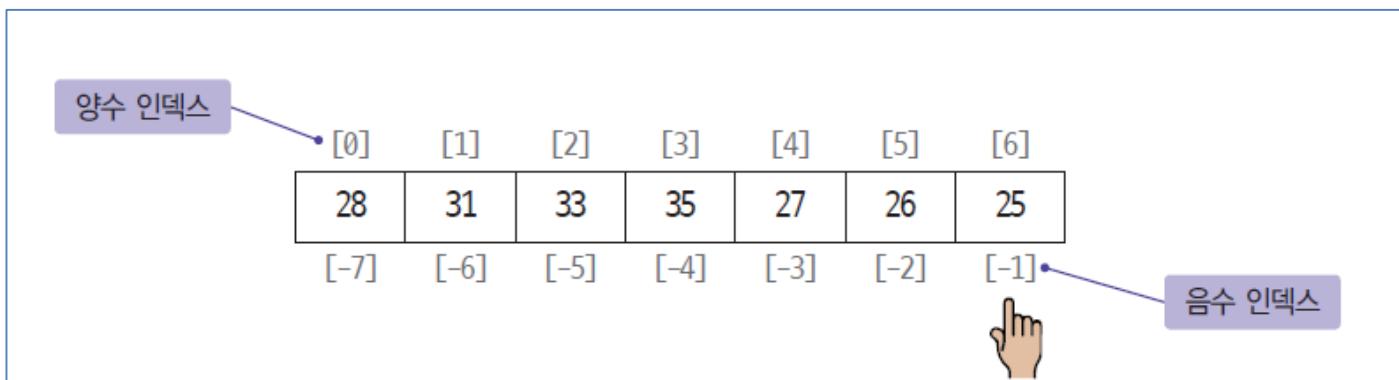
리스트의 길이

- len() 함수를 이용하면 리스트의 길이를 알 수 있다.

```
>>> numbers = [ 1, 2, 3, 4, 5, 6 ]  
>>> len(numbers)  
6
```

```
>>> s = "Hello World!"  
>>> len(s)  
12
```

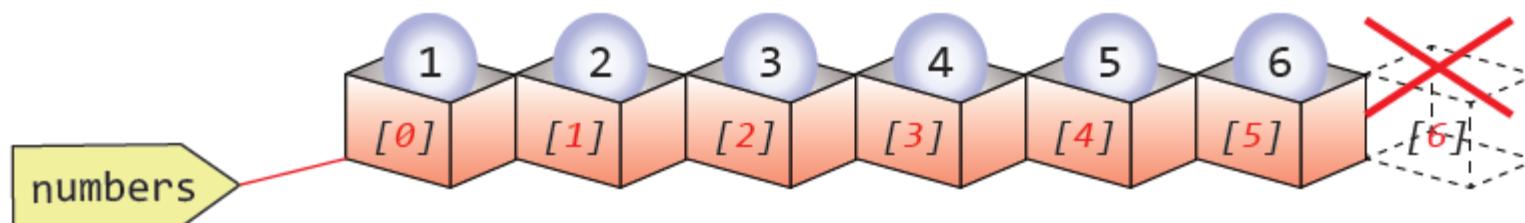
* 음수 인덱스: 리스트의 끝에서부터 매겨진다.



주의사항

- 리스트의 길이보다 더 큰 인덱스를 사용하면 오류가 발생한다

```
>>> numbers = [ 1, 2, 3, 4, 5, 6 ]  
>>> numbers[6]  
Traceback (most recent call last):  
File "<pyshell#14>", line 1, in <module>  
numbers[6]  
IndexError: list index out of range
```



in 연산자

- 어떤 값이 리스트 내부에 있는지를 확인하려면 어떻게 해야 할까?

```
>>> numbers = [ 1, 2, 3, 4, 5, 6 ]  
>>> 6 in numbers  
True  
>>> 10 in numbers  
False
```

```
>>> numbers = [ 1, 2, 3, 4, 5, 6 ]  
>>> 10 not in numbers  
True  
>>> 'e' in 'Hello'  
True  
>>> 'z' in 'Hello'  
False
```

리스트 방문 #1

- 리스트에서 가장 중요한 작업은 리스트에 저장된 데이터를 하나씩 꺼내서 어떤 처리를 하는 것이다.

```
myList = [ "우유", "사과", "두부", "소고기"]
```

```
for item in myList :  
    print(item)
```

```
우유  
사과  
두부  
소고기
```

리스트 방문 #2

- 인덱스를 사용하여 항목들을 꺼낼 수도 있다.

```
myList = [ "우유", "사과", "두부", "소고기"]
```

```
for i in range(len(myList)) :  
    print(myList[i])
```

```
우유  
사과  
두부  
소고기
```

```
temps =[28,31,33,35,27,26,25]
```

```
for i in range(len(temps)):  
    print(temps[i], end=', ')
```

```
28, 31, 33, 35, 27, 26, 25,
```

리스트 항목 교체

- 인덱스를 사용하여 항목들을 교체할 수도 있다.

```
myList = [ "우유", "사과", "두부", "소고기"]  
myList[1] = '커피'
```

```
print(myList)
```

```
['우유', '커피', '두부', '소고기']
```

리스트의 중간에 추가하기

```
myList = [ "우유", "사과", "두부", "소고기"]  
myList.insert(1, '커피')  
print(myList)
```

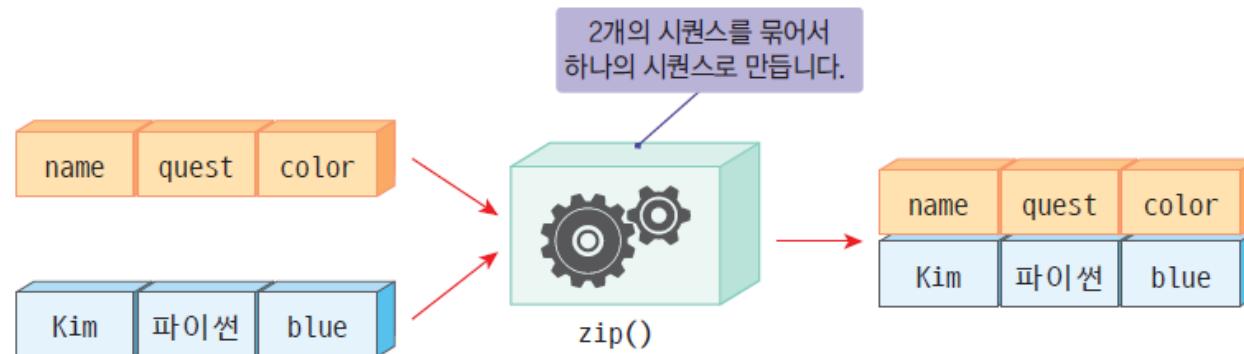
```
['우유', '커피', '사과', '두부', '소고기']
```



zip() 함수

```
questions =['name','quest','color']
answers =['Kim','파이썬','blue']
for q, a in zip(questions, answers):
    print(f"What is your {q}? It is {a}")
```

What is your name? It is Kim
What is your quest? It is 파이썬
What is your color? It is blue

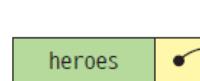


append()

```
heroes = [] # 공백 리스트를 생성한다.  
heroes.append("아이언맨") # 리스트에 "아이언맨"을 추가한다.  
heroes.append("토르") # 리스트에 "토르"를 추가한다.  
print(heroes)
```

['아이언맨', '토르']

① heroes=[]



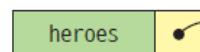
새로운 공백 리스트를 생성한다.

② heroes.append("아이언맨")



리스트에 하나의 항목을 추가한다.

③ heroes.append("토르")



리스트에 하나의 항목을 추가한다.

공백 리스트에서 추가하기

```
myList = [ ]  
myList.append("우유")  
myList.append("사과")  
myList.append("두부")  
myList.append("소고기")  
print(myList)
```

```
['우유', '사과', '두부', '소고기']
```



점의 의미

- 파이썬에서 모든 것은 객체(object)이다.
- 객체는 관련되는 변수와 함수를 묶은 것이다.
- 파이썬에서 리스트도 객체이다. 객체 안에 있는 무엇인가를 사용할 때는 객체의 이름을 쓰고 . 을 붙인 후에 함수의 이름을 적는다.



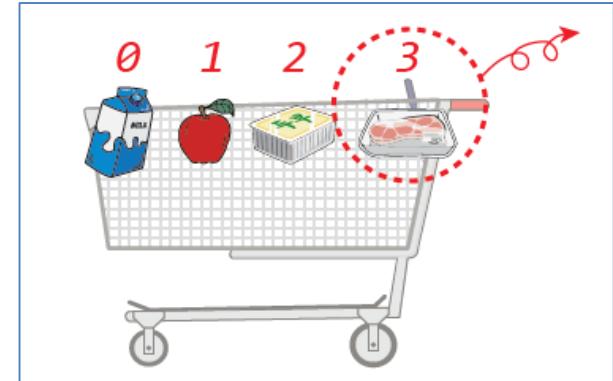
함수는 독립적으로 존재하는 코드 블록이고,
메소드는 특정 객체나 클래스에 속한 함수

항목 삭제하기

- 항목을 삭제하는 것도 가능할까? 물론이다. `remove()`와 `pop()`을 사용하여 삭제할 수 있다.
- 항목이 저장된 위치를 알고 있다면 `pop(i)`을 사용한다.
- 항목의 값만 알고 있다면 `remove(value)`를 사용한다.

```
myList = [ "우유", "사과", "두부", "소고기"]  
myList.remove("소고기")  
print(myList)
```

['우유', '사과', '두부']



항목 삭제하기

- `pop(index)`는 특정 인덱스에 있는 항목을 삭제하고 우리에게 항목을 반환한다.

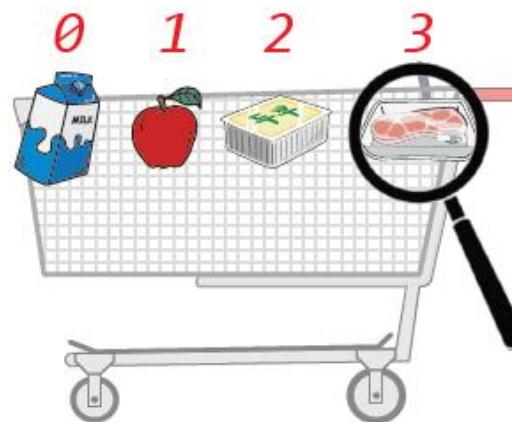
```
myList = [ "우유", "사과", "두부", "소고기"]  
item = myList.pop(0) # 0번째 항목 "우유" 삭제, item은 우유  
print(myList)
```

```
['사과', '두부', '소고기']
```

리스트 탐색하기

- 우리는 리스트에서 특정한 항목을 찾을 수 있다.

```
myList = [ "우유", "사과", "두부", "소고기"]  
if "소고기" in myList:  
    i = myList.index("소고기")      # i는 3  
    print(i)
```

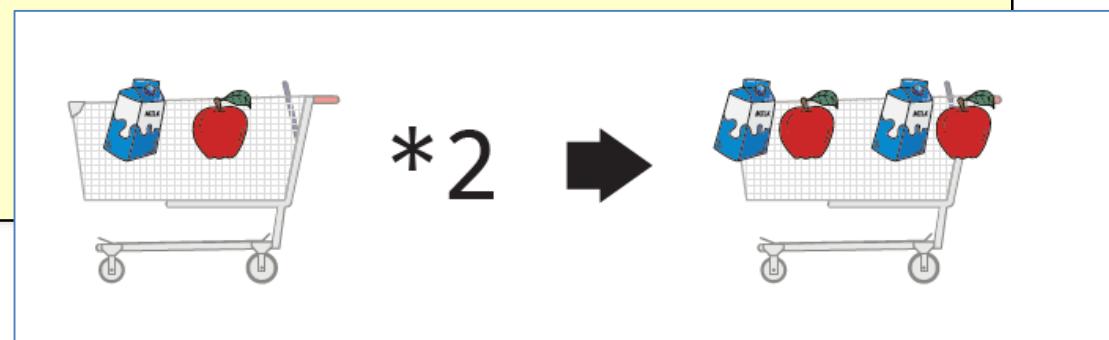


리스트 합치기

```
>>> myList = [ "우유", "사과"]  
>>> yourList = [ "두부", "소고기"]  
>>> myList+yourList  
['우유', '사과', '두부', '소고기']
```



```
>>> myList = [ "우유", "사과" ]  
>>> myList*2  
['우유', '사과', '우유', '사과']
```

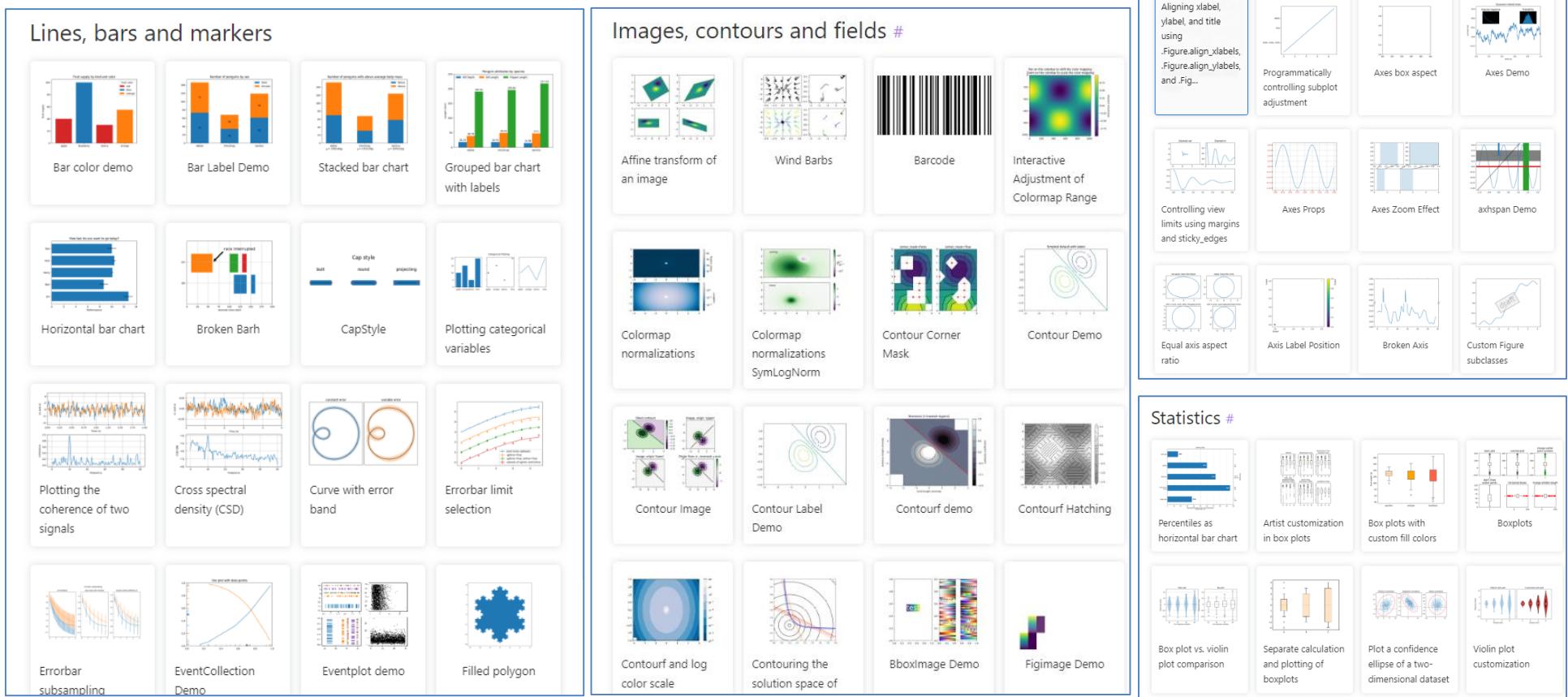


```
>>> numbers = [0]*12  
>>> numbers  
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

matplotlib로 그래프를 그려보자

- Matplotlib은 GNUpot처럼 그래프를 그리는 라이브러리이다.
- 설치 방법: "pip install matplotlib"

<https://matplotlib.org/>



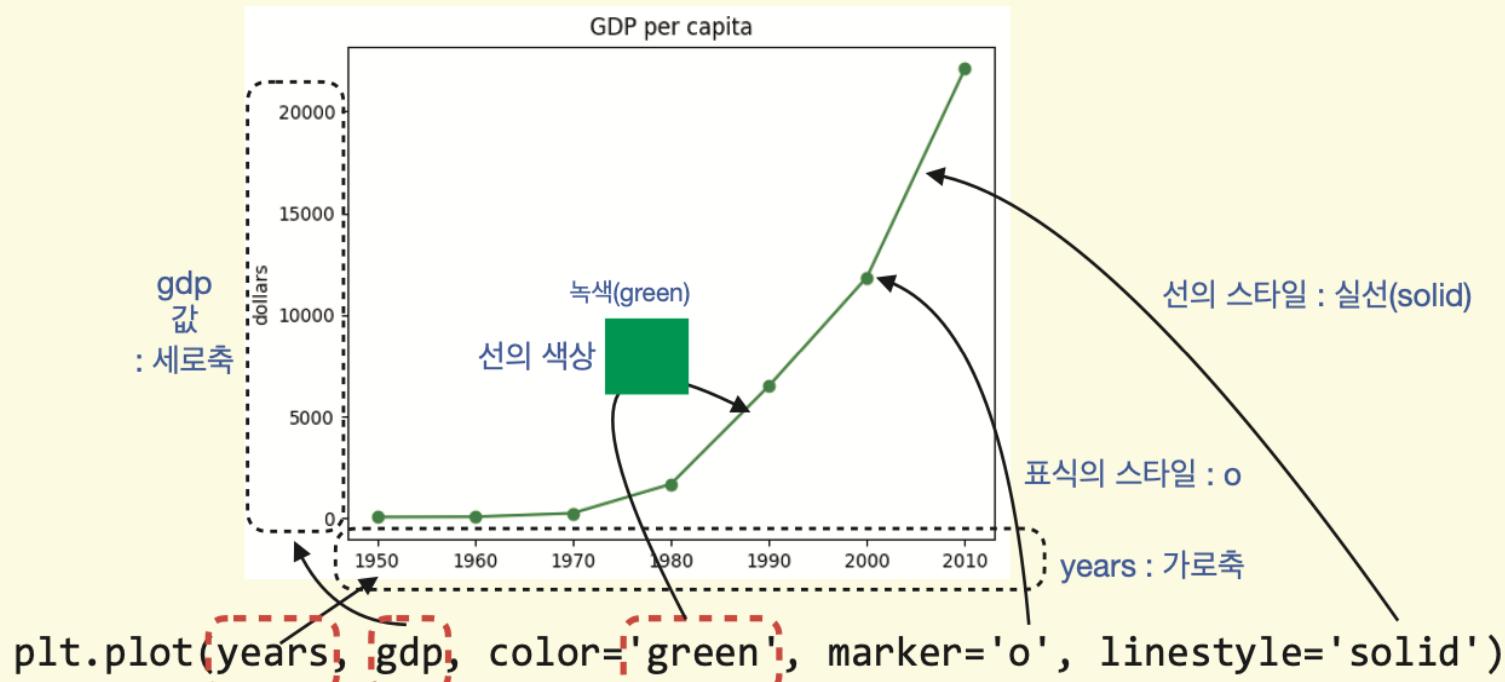
이외에도 다수 있음

matplotlib 무작정 사용해 보기

이 코드의 실행 결과는?

years	1950	1960	1970	1980	1990	2000	2010
gdp	67.0	80.0	257.0	1686.0	6505.0	11865.3	22105.3

제공되는 데이터



제공되는 데이터와 matplotlib.pyplot 모듈을 이용한 시각화 방법

각각의 인자들의 의미는 정보를 시각화하는 방법

matplotlib 무작정 사용해 보기

- matplotlib은 가장 널리 사용되는 시각화 도구이다.
- 간단한 막대 그래프, 선 그래프, 산포도를 그리는 용도로는 matplotlib가 제격이다.

```
import matplotlib.pyplot as plt

# 우리나라의 연간 1인당 국민소득을 각각 years, gdp에 저장
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [67.0, 80.0, 257.0, 1686.0, 6505, 11865.3, 22105.3]

# 선 그래프를 그린다. x축에는 years값, y축에는 gdp 값이 표시된다.
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')

# 제목을 설정한다.
plt.title("GDP per capita") # 1인당 국민소득

# y축에 레이블을 붙인다.
plt.ylabel("dollars")
plt.savefig("gdp_per_capita.png", dpi=600) # png 이미지로 저장 가능
plt.show()
```

GDP 데이터 출처: 한국은행 경제통계시스템, <https://ecos.bok.or.kr/>

matplotlib 무작정 사용해 보기

맷플롯립 코드 살펴 보기

- 첫 번째 단계는 pyplot 모듈을 불러와서 plt라는 별칭으로 지정하는 것

```
import matplotlib.pyplot as plt
```

- 연도별 GDP의 변화
 - 연도는 years 리스트에 담고, x축 데이터로 사용
 - GDP 값은 gdp라는 이름의 리스트를 만들어 데이터를 담게 하고, 이 값이 y축 데이터로 사용

```
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [67.0, 80.0, 257.0, 1686.0, 6505, 11865.3, 22105.3]
```

matplotlib 무작정 사용해 보기

matplotlib 코드 살펴 보기 (계속)

- 선형 차트를 만들려면 plt의 plot() 함수를 호출. plot()은 x축 데이터와 y축 데이터를 인수로 받음
- 선의 색, 마크의 표시방법, 선의 두께 등을 키워드 인자로 줄 수 있다.

```
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')
```

- 차트의 제목 레이블을 설정한다. 차트의 최상단에 표시된다.

```
plt.title("GDP per capita")
```

- y축의 레이블을 지정, savefig() 함수를 통해서 파일을 저장함. 해상도는 dpi dot per inch를 통해 지정
 - dpi는 1인치(약 2.54cm)당 점의 수를 의미함

```
plt.ylabel("dollars")
plt.savefig("gdp_per_capita.png", dpi=600)
plt.show()
```

matplotlib 무작정 사용해 보기

- 반드시 plt.plot() 함수를 사용하여야 화면에 차트가 나타난다

```
plt.xlabel("dollars")
plt.ylabel("gdp")
```

```
plt.plot(gdp, years, color='red',
marker='o', linestyle='solid')
```

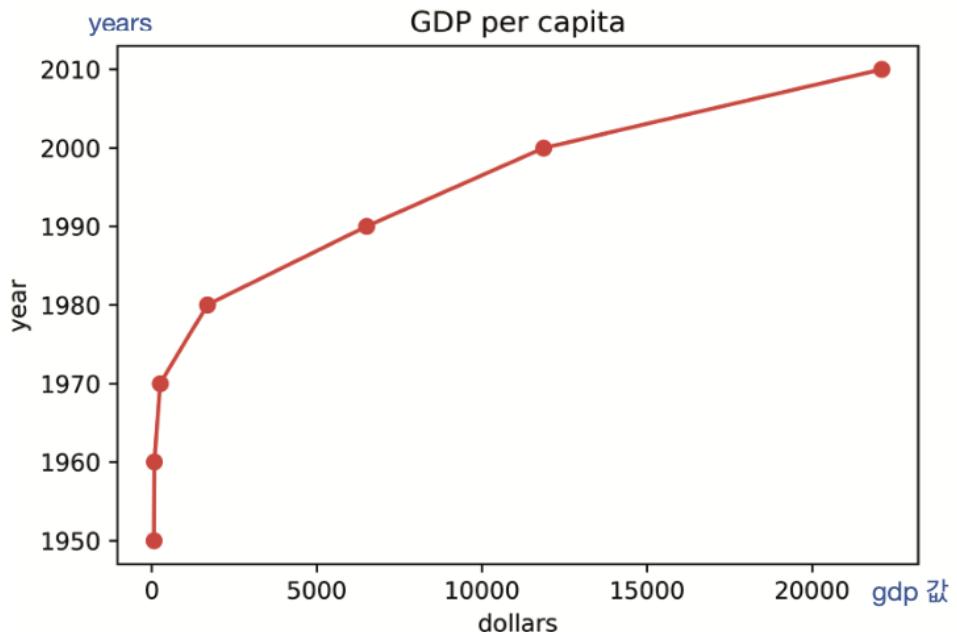


차트 장식을 도와주는 다양한 기법들

색상 코드		마커 코드		
문자	색상	문자	기호 표시	마커 모양
k	검정	.	•	점
b	파랑	o	○	원
g	초록	+	+	플러스 사인
r	빨강	x	×	
c	청록	D	◊	다이아몬드
m	자홍	v	▽	델
y	노랑	^	△	삼각형
w	하양	s	□	사각형

선 스타일 옵션과 코드

기호	스타일
-	실선
--	파선
:	점선
-.	1점 쇄선

막대형 차트 그리기

- 아래의 range() 함수는 정수의 범위를 생성하는 함수이다.
- 이 경우 years 리스트가 7 개의 항목을 가지고 있으므로 0에서 6까지의 정수 범위를 만든다.
- 이 범위가 막대형 차트의 가로축 범위가 되는 것이다.

```
plt.bar(range(len(years)), gdp)
```

- xtick() 함수를 사용하여 가로축 범위의 눈금마다 부여할 눈금값을 지정
- 아래와 같이 years 리스트의 항목들을 사용

```
plt.xticks(range(len(years)), years)
```

[실습] 막대형 차트 그리기

- plt.bar() 함수를 호출하여 화면에 막대형 차트를 그릴 수 있다.
- 앞의 1인당 국민소득을 막대 그래프로 그려보자.

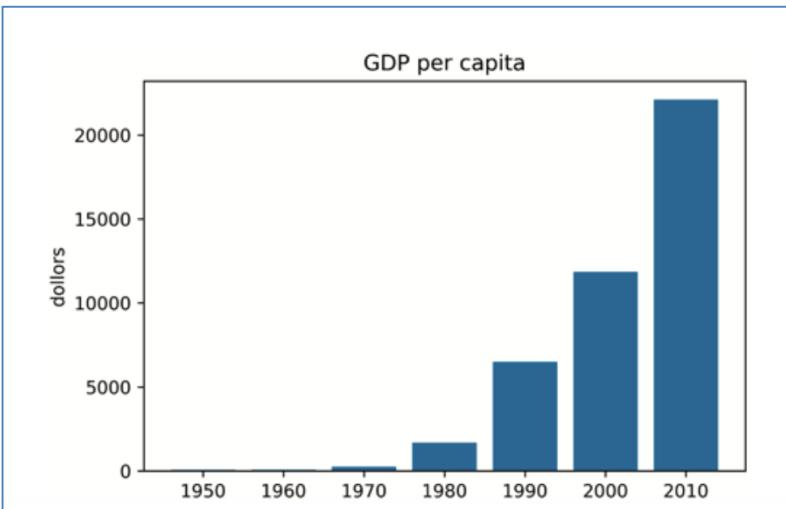
```
from matplotlib import pyplot as plt

# 1인당 국민소득
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [67.0, 80.0, 257.0, 1686.0, 6506, 11865.3, 22105.3]

plt.bar(range(len(years)), gdp) # range(len(years))는 x 축에 연도를 배치하기 위한 위치를 결정하고, gdp는 막대의 높이를 결정

plt.title("GDP per capita") # 제목을 설정
plt.ylabel("dollars")       # y축에 레이블을 붙임

# x 축에 틱을 붙인다.
plt.xticks(range(len(years)), years) # x 축에 연도 표시
plt.show()
```



[실습] 데이터를 점으로 표현하는 산포도 그래프 그리기

- 산포도 플롯 scatter plot은 개별 데이터 포인트를 그리는 차트. 산포도를 그릴 때는 각 데이터 포인트가 연결되지 않는다. 산포도 그래프를 그릴 때는 scatter() 함수를 사용함

```
import numpy as np  
from matplotlib import pyplot as plt
```

xData = np.arange(20,50)

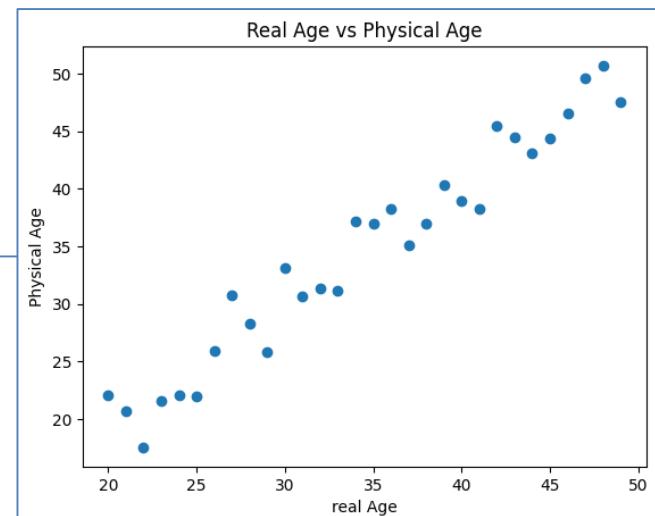
yData = xData + 2*np.random.randn(30) # xData에 randn() 함수로 잡음을 섞는다.

잡음은 정규분포로 만들어 질 것이다.

평균이 0이고 표준 편차가 1인 표준 정규 분포에서 무작위로 30개의 숫자를 생성하여 반환

```
plt.scatter(xData, yData)  
plt.title("Real Age vs Physical Age")  
plt.xlabel("real Age")  
plt.ylabel("Physical Age")
```

```
plt.savefig("age.png", dpi=600)  
plt.show()
```



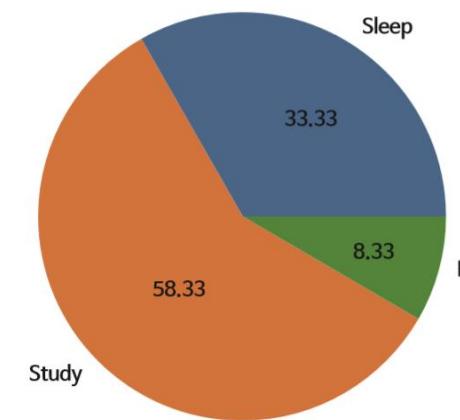
파이 차트

- 파이 차트 pie chart는 데이터의 값에 따라서 원형 비율로 나누어져 있는 차트
- 파이 차트는 비즈니스 세계에서 널리 사용(예들 들어, 상품의 시장 점유율을 표시)

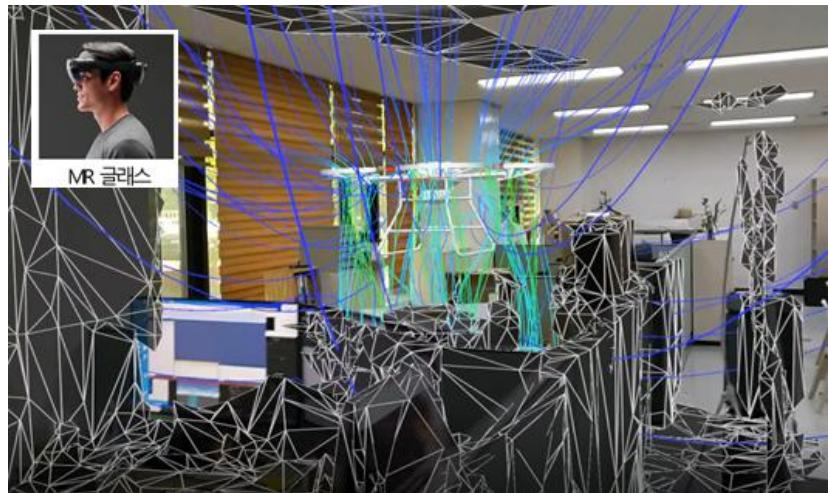
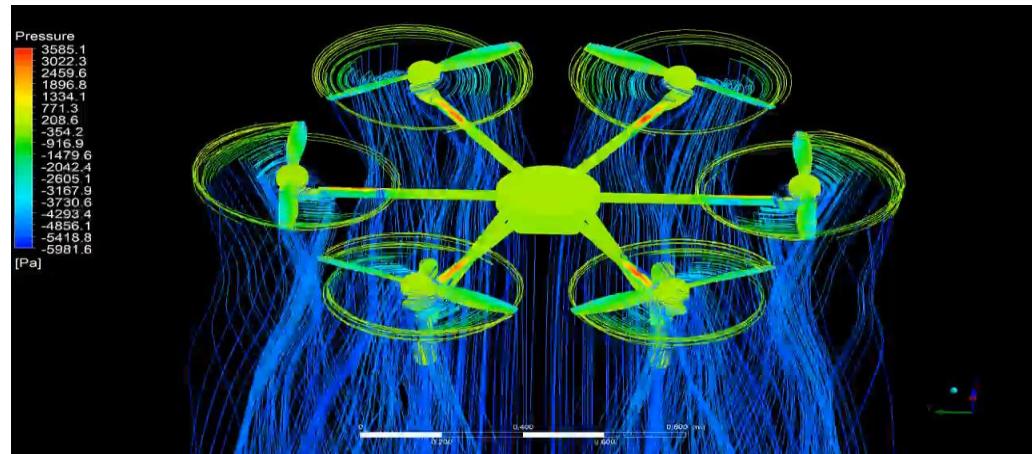
```
import matplotlib.pyplot as plt  
times = [8, 14, 2]
```

```
timelabels = ["Sleep", "Study", "Play"]
```

```
# autopct로 백분율을 표시할 때 소수점 2번째 자리까지 표시하게 한다.  
# labels 매개 변수에 timelabels 리스트를 전달한다.  
plt.pie(times, labels = timelabels, autopct = "%.2f")  
plt.show()
```



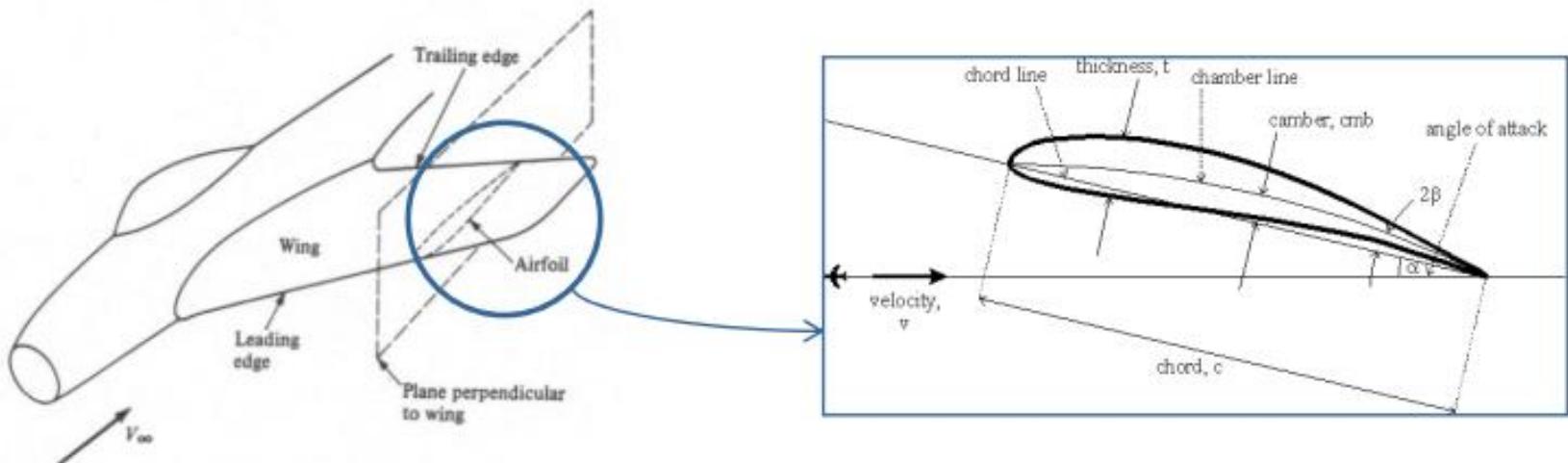
드론 시뮬레이(터)



● 에어포일 (Airfoil)

● 비행기의 날개를 수직으로 자른 단면 형상

- 항공기의 날개(wing), 보조익(aileron), 승강타(elevator), 방향타(rudder)와 같은 단면(section)을 학술적으로 정의하는 데 사용



에어포일

- 형상은 유선형
- 공기 중을 운동하면서 날개에 큰 양력과 적은 항력, 모멘트를 발생시키는 역할

비행기 날개 만들어 보기

코드에 사용된 `yt` 두께 분포 공식 설명

NACA 4자리 공기foil 두께 분포 수식은 다음과 같습니다:

$$y_t = 5t \left(0.2969 \sqrt{\frac{x}{c}} - 0.1260 \frac{x}{c} - 0.3516 \left(\frac{x}{c}\right)^2 + 0.2843 \left(\frac{x}{c}\right)^3 - 0.1015 \left(\frac{x}{c}\right)^4 \right)$$

- t : 최대 두께 비율 (여기서는 $t = 0.12 \rightarrow$ 최대 두께는 코드 길이의 12%)
- c : 코드 길이 (여기서는 $c = 1.0$)
- x : 코드 길이 방향의 위치 ($0 \leq x \leq c$)
- y_t : 해당 x -위치에서 공기foil 두께의 절반 (윗면과 아랫면을 합하면 전체 두께)

대표적인 에어포일이 NACA(National Advisory Committee for Aeronautics, 국가항공자문위원회) 0012임

예제

```
import numpy as np
import matplotlib.pyplot as plt

# Define NACA 0012 parameters
t = 0.12 # Maximum thickness ratio
c = 1.0 # Chord length

# Define x coordinates along the chord
x = np.linspace(0, c, 100)

# Thickness distribution formula for NACA 4-digit airfoils
yt = 5 * t * (0.2969 * np.sqrt(x/c) - 0.1260 * (x/c) - 0.3516 * (x/c)**2 +
               0.2843 * (x/c)**3 - 0.1015 * (x/c)**4)

# Upper and lower surfaces
xu = x
yu = yt
xl = x
yl = -yt

# Plot the airfoil shape
plt.figure(figsize=(10, 5))
plt.plot(xu, yu, 'b', label="Upper Surface")
plt.plot(xl, yl, 'r', label="Lower Surface")
plt.title("NACA 0012 Airfoil Shape")
plt.xlabel("Chord Length (x)")
plt.ylabel("Thickness (y)")
plt.axis('equal')
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.show()
```

문제

1. 예제 코드를 이해하기
2. NACA 0030의 드론 날개 만들어 보기
-> $t = 0.30$ 으로 변경

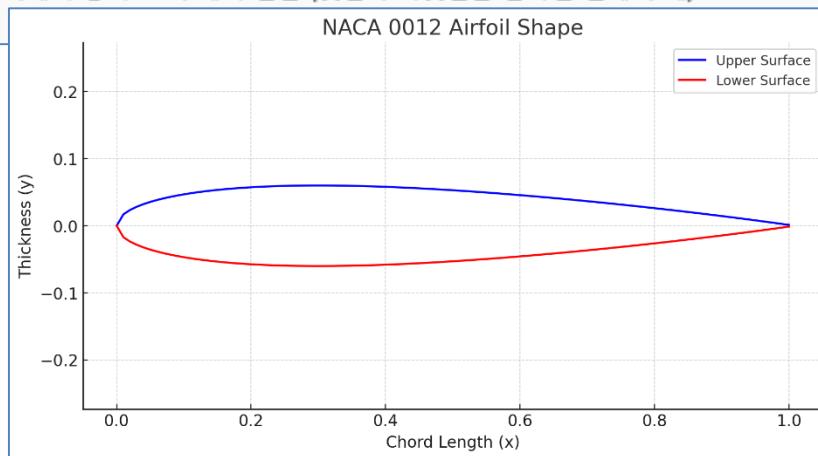
비행기 날개 만들어 보기

코드에 사용된 yt 두께 분포 공식 설명

NACA 4자리 공기foil 두께 분포 수식은 다음과 같습니다:

$$y_t = 5t \left(0.2969 \sqrt{\frac{x}{c}} - 0.1260 \frac{x}{c} - 0.3516 \left(\frac{x}{c}\right)^2 + 0.2843 \left(\frac{x}{c}\right)^3 - 0.1015 \left(\frac{x}{c}\right)^4 \right)$$

- t : 최대 두께 비율 (여기서는 $t = 0.12 \rightarrow$ 최대 두께는 코드 길이의 12%)
- c : 코드 길이 (여기서는 $c = 1.0$)
- x : 코드 길이 방향의 위치 ($0 \leq x \leq c$)
- y_t : 해당 x -위치에서 공기foil 두께의 절반 (윗면과 아랫면을 합하면 전체 두께)



예제

```
import numpy as np
import matplotlib.pyplot as plt

# Define NACA 0012 parameters
t = 0.12 # Maximum thickness ratio
c = 1.0 # Chord length

# Define x coordinates along the chord
x = np.linspace(0, c, 100)

# Thickness distribution formula for NACA 4-digit airfoils
yt = 5 * t * (0.2969 * np.sqrt(x/c) - 0.1260 * (x/c) - 0.3516 * (x/c)**2 +
               0.2843 * (x/c)**3 - 0.1015 * (x/c)**4)

# Upper and lower surfaces
xu = x
yu = yt
xl = x
yl = -yt

# Plot the airfoil shape
plt.figure(figsize=(10, 5))
plt.plot(xu, yu, 'b', label="Upper Surface")
plt.plot(xl, yl, 'r', label="Lower Surface")
plt.title("NACA 0012 Airfoil Shape")
plt.xlabel("Chord Length (x)")
plt.ylabel("Thickness (y)")
plt.axis('equal')
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.show()
```

문제

1. 예제 코드를 이해하기
2. NACA 0030의 드론 날개 만들어 보기
-> $t = 0.30$ 으로 변경

죽는 날까지 하늘을 우러러
한 점 부끄럼이 없기를,
잎새에 이는 바람에도
나는 괴로워했다.

별을 노래하는 마음으로
모든 죽어가는 것을 사랑해야지,
그리고 나에게 주어진 길을
걸어가야겠다.

오늘 밤에도 별이 바람에 스치운다.