



# 데이터 관리를 위한 파이썬 PYTHON 기초 과정



Make !t Now

## 1일차 : Python 기초 (1)

- Python 특징
- 기초 자료형(1)
- 제어문

## 2일차 : Python 기초 (2)

- 함수 (입출력)
- 파일 읽고 쓰기
- 클래스

## 3일차 : 데이터 분석 툴

- Numpy
- Pandas
- matplotlib

## 4일차 : 데이터 분석 실습

- 전국 민간 아파트 분양가 분석

## 1. 파이썬 특징, 파이썬 설치

---

- 파이썬 설치하기
- 파이썬 소개
- 파이썬 개발환경 설치 (IDLE)



파이썬(Python)은 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 **인터프리터 언어**이다. 귀도는 파이썬이라는 이름을 자신이 좋아하는 코미디 쇼인 "몬티 파이썬의 날아다니는 서커스(Monty Python's Flying Circus)"에서 따왔다고 한다.

파이썬은 문법이 쉬워 빠르게 배울 수 있다

파이썬은 무료이지만 강력하다

파이썬은 간결하다

파이썬은 개발 속도가 빠르다



"Life is too short, You need python." (인생은 너무 짧으니 파이썬이 필요해.)

# 파이썬 설치

The image shows the Python 3.8.3 (32-bit) Setup window. The background is the Python.org website, which includes a navigation bar with links to Python, PSF, Docs, PyPI, Jobs, and Community. The main content area has tabs for About, Downloads, Documentation, Community, and Success. The Downloads tab is active, showing a list of download links for various operating systems. The Setup window is titled "Python 3.8.3 (32-bit) Setup" and contains the following text:

**Install Python 3.8.3 (32-bit)**  
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ **Install Now**  
C:\Users\kkong\AppData\Local\Programs\Python\Python38-32  
Includes IDLE, pip and documentation  
Creates shortcuts and file associations

→ **Customize installation**  
Choose location and features

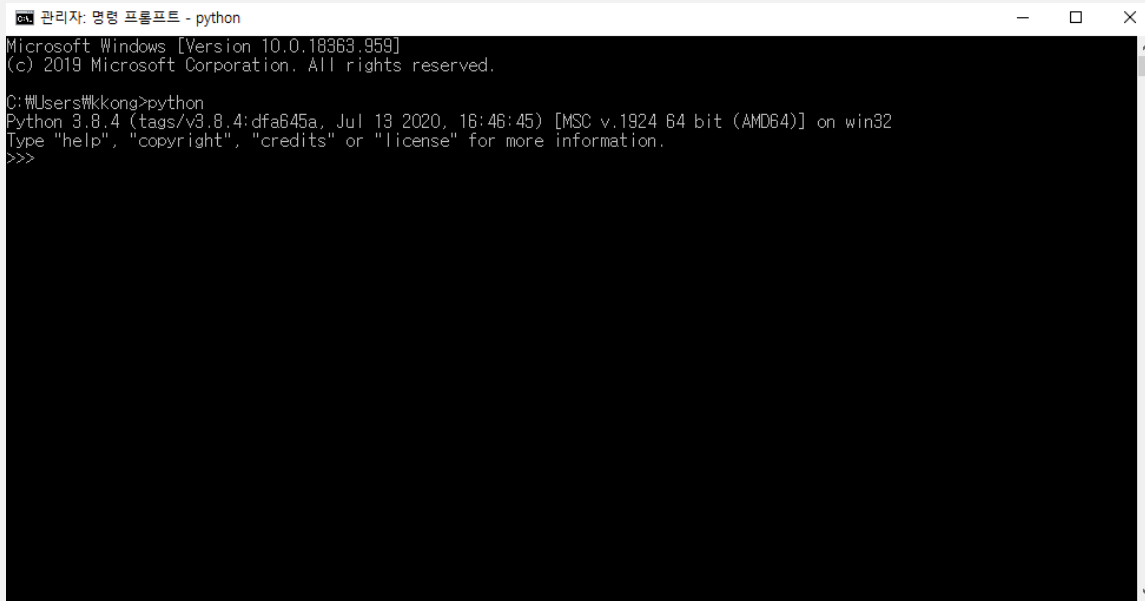
☒ Install launcher for all users (recommended)  
☒ **Add Python 3.8 to PATH**

Cancel

The background website also features a "Get Started" section with a power icon, a "Download" section with a download icon, and a "Docs" section with a document icon. The "Get Started" section includes a link to "Start with our Beginner's Guide". The "Download" section mentions "Latest: Python 3.8.3". The "Docs" section provides a link to "docs.python.org".

# 통합 개발 환경 (IDE) 설치

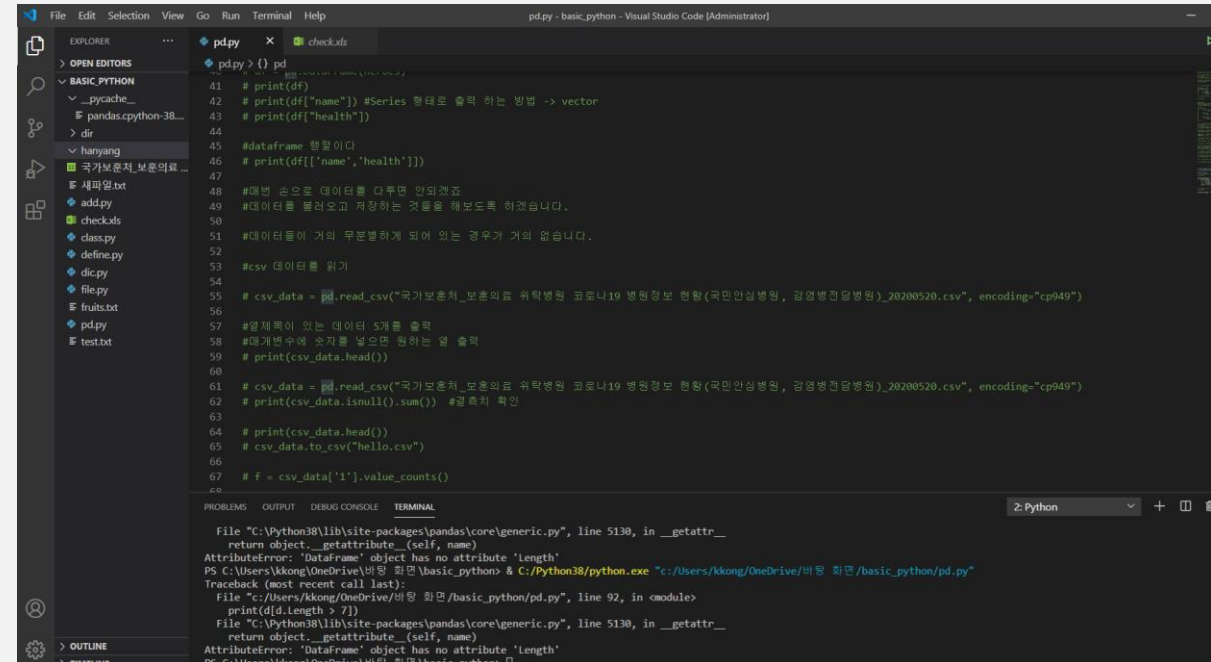
## 파이썬 IDE (Integrated Development Environment, 통합개발환경툴)



```
관리자: 명령 프롬프트 - python
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\kkong>python
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:46:45) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Shell



```
pd.py - basic_python - Visual Studio Code [Administrator]

EXPLORER
> OPEN EDITORS
  BASIC_PYTHON
    __pycache__
    pandas.cpython-38...
    > dir
  hanyang
    국가보훈처 보훈이력...
    새파일.txt
    add.py
    checks
    class.py
    define.py
    dicpy
    file.py
    fruits.txt
    pd.py
    test.txt

pd.py
41 # print(df)
42 # print(df[["name"]]) #Series 형태로 출력 하는 방법 -> vector
43 # print(df[["health"]])
44
45 #dataframe 형식이다
46 # print(df[["name", "health"]])
47
48 #대변 손으로 데이터를 다루면 안되겠지
49 #데이터를 불러오고 저장하는 것들을 해보도록 하겠습니다.
50
51 #데이터들이 거의 무분별하게 되어 있는 경우가 거의 없습니다.
52
53 #csv 데이터를 읽기
54
55 # csv_data = pd.read_csv("국가보훈처_보훈이력_위탁병원_코로나19 병원정보 현황(국민안심병원, 감염병전담병원)_20200520.csv", encoding="cp949")
56
57 #알짜책이 있는 데이터 5개를 출력
58 #대개 변수에 숫자를 넣으면 원하는 것 출력
59 # print(csv_data.head())
60
61 # csv_data = pd.read_csv("국가보훈처_보훈이력_위탁병원_코로나19 병원정보 현황(국민안심병원, 감염병전담병원)_20200520.csv", encoding="cp949")
62 # print(csv_data.isnull().sum()) #공백치 확인
63
64 # print(csv_data.head())
65 # csv_data.to_csv("hello.csv")
66
67 # f = csv_data['1'].value_counts()
68

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
File "C:\Python38\lib\site-packages\pandas\core\generic.py", line 5130, in __getattr__
    return object.__getattr__(self, name)
AttributeError: 'DataFrame' object has no attribute 'length'
PS C:\Users\kkong\OneDrive\바탕 화면\basic_python> & C:\Python38\python.exe "c:/Users/kkong/OneDrive/바탕 화면/basic_python/pd.py"
Traceback (most recent call last):
  File "c:/Users/kkong/OneDrive/바탕 화면/basic_python/pd.py", line 92, in <module>
    print(df.length > 7)
  File "C:\Python38\lib\site-packages\pandas\core\generic.py", line 5130, in __getattr__
    return object.__getattr__(self, name)
AttributeError: 'DataFrame' object has no attribute 'length'
PS C:\Users\kkong\OneDrive\바탕 화면\basic_python>
```

Code Editor

# Python Code Editor



Visual Studio code



# Visual Studio Code (VSCode)

Google

Visual studio code

전체 이미지 동영상 쇼핑 도서 더보기

설정 도구

검색결과 약 625,000,000개 (0.51초)

code.visualstudio.com ▾ 이 페이지 번역하기

## Visual Studio Code - Code Editing. Redefined

**Visual Studio Code** is a code editor redefined and optimized for building web and cloud applications. **Visual Studio Code** is free and ...

이 페이지를 3번 방문했습니다. 최근 방문 날짜: 20. 6. 24

### Download Visual Studio Code

Visual Studio Code is free and available on your favorite ...

### Docs

Intro Videos - Windows - GCC on Windows - Basic Editing - Java

### Updates

May 2020 - April 2020 - November 2019 - HexEditor - ...

### License

This license a Studio Code p

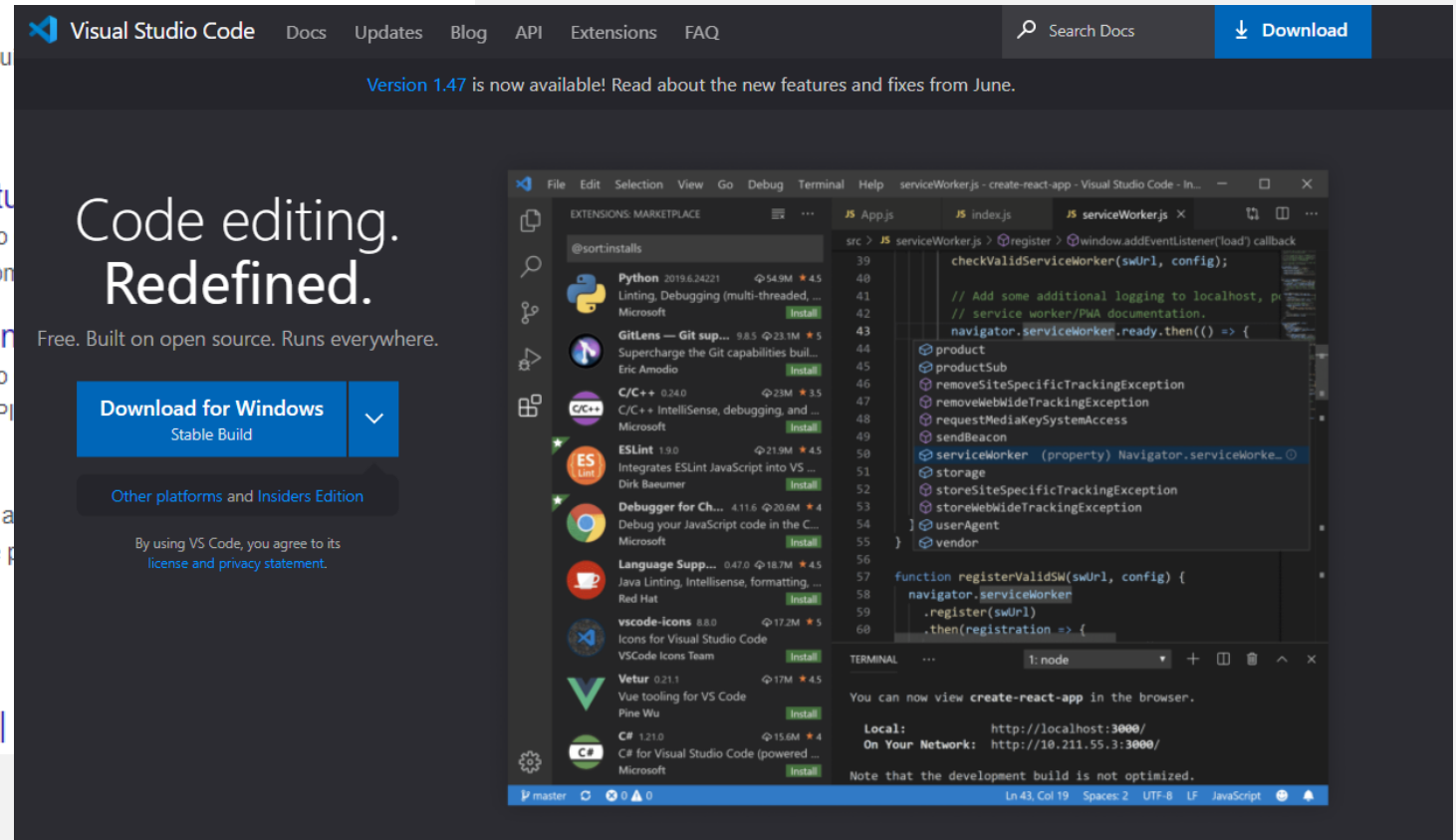
### Extension

Visual Studio extension API

visualstudio.com 검색결과 더보기 »

www.44bits.io > keyword > visual-studio-code

## 비주얼 스튜디오 코드(Visual Studio Code)란?





## 2. 파이썬 기초 자료형

---

- 숫자형
- 문자열 자료형
- 리스트, 튜플 자료형
- 딕셔너리 자료형
- 불(Bool) 자료형
- 연습문제

# 파이썬 기본 문법 : 숫자형

## 정수형

```
>>> a = 123
>>> a = -178
>>> a = 0
```

## 실수형

```
>>> a = 1.2
>>> a = -3.45
```

## 사칙연산

```
>>> a = 3
>>> b = 4
>>> a + b 7
>>> a * b 12
>>> a / b 0.75
```

## 제곱 연산

```
>>> a = 3
>>> b = 4
>>> a ** b 81
```

## 나머지 연산

```
>>> 7 % 3
1
>>> 3 % 7
3
```

## 몫

```
>>> 7 // 3
2
>>> 3 // 7
0
```

## 파이썬 기본 문법 : 문자열

- 문자는 `'''` 또는 `''` 안에 문자를 작성
- 다음 줄로 쓰는 것은 `\n` 을 활용하여 작성
- 문자열끼리 + 연산이 가능함
- 문자열 \* (숫자) 로 연산이 가능
- 문자열 길이를 찾을 수 있음.

```
>>> food = "apple"
```

```
>>> food  
"apple"
```

```
>>> multiline = "Life is too short\nYou need python"
```

```
>>> head = "Python"
```

```
>>> tail = " is fun!"
```

```
>>> head + tail 'Python is fun!'
```

```
>>> a = "python"
```

```
>>> a * 2  
'pythonpython'
```

```
>>> a = "Life is too short"
```

```
>>> len(a) 17
```

## 파이썬 기본 문법 : 문자열 인덱싱

### 문자변수[숫자]

변수[ '시작' : '끝' : '간격' ] \* 단 빈칸은 모두를 나타낸다

```
>>> a = "Life is too short, You need Python"
>>> a[0] 'L' >>> a[12] 's' >>> a[-1] 'n'
```

```
>>> a[-2] 'o'
>>> a[-5] 'y'
```

```
>>> a = "Life is too short, You need Python"
>>> b = a[0] + a[1] + a[2] + a[3]
>>> b 'Life'
```

```
>>> a = "Life is too short, You need Python"
>>> a[0:4] 'Life'
```

```
>>> a[19:] 'You need Python'
```

```
>>> a[:17] 'Life is too short'
```

```
>>> a[:] 'Life is too short, You need Python'
```

# 파이썬 기본 문법 : 문자열 포매팅

## 문자열 포맷 코드

### 숫자 출력

```
a = 10
b = 20
print('사과를 %d개 %d번 먹었습니다.' % (a,b))
```

### 문자 출력

```
a = 10
b = '엄청 많이'
print('사과를 %d개 %s 먹었습니다.' % (a,b))
```

## f문자열 포매팅 (파이썬 3.6버전 이상에서만 사용)

```
a = 10.5
b = '엄청 많이'
```

```
print(f'{a}개의 사과를 {b} 먹었습니다.')
```

## 파이썬 기본 문법 : 문자열 관련 함수

### 문자개수 세기

```
>>> a = "hobby"  
>>> a.count('b') 2
```

### 위치 알려주기 (find)

```
>>> a = "Python is the best choice"  
>>> a.find('b')  
14  
>>> a.find('k')  
-1
```

### 위치 알려주기2 (index)

```
>>> a = "Life is too short"  
>>> a.index('t') 8  
>>> a.index('k')  
Traceback (most recent call last): File  
"<stdin>", line 1, in <module> ValueError:  
substring not found
```

소문자 -> 대문자(upper)

대문자 -> 소문자 (lower)

왼쪽 공백 지우기(lstrip)

오른쪽 공백 지우기(rstrip)

양쪽 공백 지우기(strip)

문자열 바꾸기 (replace)

문자열 나누기(split)

## 파이썬 기본 문법 : 리스트

### 리스트의 생성

리스트는 배열과 같음, 리스트는 숫자, 정수, 문자 ,배열 모두 올 수 있음

```
>>> a = []  
>>> a = list()  
>>> b = [1, 2, 3]  
>>> c = ['Life', 'is', 'too', 'short']  
>>> d = [1, 2, 'Life', 'is']  
>>> e = [1, 2, ['Life', 'is']]  
  
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

# 파이썬 기본 문법 : 리스트

## 리스트의 슬라이싱

리스트 내부 값을 추출할 때 사용하는 방법

```
>>> a[0]
1
>>> a[-1]
['a', 'b', 'c']
>>> a[3]
['a', 'b', 'c']
>>> a[-1][0]
'a'

>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
[1, 2]
```

## 리스트 값 수정

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a [1, 2, 4]
```

## 리스트 값 삭제

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a [1, 3]
```



## 파이썬 기본 문법 : 리스트

### 리스트에 요소 추가(append)

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a [1, 2, 3, 4]

>>> a.append([5,6])
>>> a [1, 2, 3, 4, [5, 6]]
```

### 리스트에 요소 추가(insert)

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4)
>>> a [4, 1, 2, 3]
```

### 위치 반환(index)

```
>>> a = [1,2,3]
>>> a.index(3)
2
>>> a.index(1) 0
```

### 리스트 정렬 sort

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a [1, 2, 3, 4]
```

### 리스트 뒤집기

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a ['b', 'c', 'a']
```

## 파이썬 기본 문법 : 리스트

### 리스트에 값 빼기(pop)

가장 마지막에 입력된 값을 빼내는 함수 pop()

```
>>> a = [1,2,3]
>>> a.pop()
3
>>> a [1, 2]
```

```
>>> a = [1,2,3]
>>> a.pop(1)
2
>>> a [1, 3]
```

# 파이썬 기본 문법 : 딕셔너리

## 딕셔너리 구조

{Key1:Value1, Key2:Value2, Key3:Value3, ...}

```
>>> a = {1: 'hi'}
```

```
>>> a = { 'a': [1,2,3]}
```

```
>>> a = {1: 'a'}
```

```
>>> a[2] = 'b'
```

```
>>> a {1: 'a', 2: 'b'}
```

```
>>> del a[1]
```

```
>>> a {2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

## 딕셔너리

```
>>> grade = {'pey': 10, 'julliet': 99}
```

```
>>> grade['pey']
```

```
10
```

```
>>> grade['julliet']
```

```
99
```

```
>>> dic = {'name': 'pey', 'phone': '0119993323',  
'birth': '1118'}
```

```
>>> dic['name']
```

```
'pey'
```

```
>>> dic['phone']
```

```
'0119993323'
```

```
>>> dic['birth']
```

```
'1118'
```

```
>>> a = {'name': 'pey', 'phone': '0119993323',  
'birth': '1118'}
```

```
>>> 'name' in a True
```

```
>>> 'email' in a False
```

## 파이썬 기본 문법 : 딕셔너리

### 딕셔너리 keys 살펴보기

```
>>> a = {'name': 'pey', 'phone': '0119993323',  
'birth': '1118'}  
>>> a.keys() dict_keys(['name', 'phone',  
'birth'])
```

### 딕셔너리 삭제

```
>>> a.clear()  
>>> a  
{}
```

### 딕셔너리 values 살펴보기

```
>>> a.values() dict_values(['pey',  
'0119993323', '1118'])
```

## 파이썬 기본 문법 : 불 자료형(True or False)

Bool 자료형이란 True(참) False(거짓)을 나타내는 자료형  
True / False 2가지만 존재

첫번째 문자는 반드시 대문자로 사용

Bool 경우에는 값이 있으면 True, 값이 없으면 False  
0이면 False, 1 이면 True

## 연습 문제 1

홍길동 씨의 주민등록번호는 “881120-1068234”

홍길동 씨의 주민등록번호를 연월일(YYMMDD) 부

분과 그 뒤의 숫자 부분으로 나누어 출력

(문자열 슬라이싱)

성별을 나타내는 숫자를 출력

(문자열 인덱싱 사용)

### 3. 제어문

---

- if문
- while 문
- for문
- 연습문제

### 3. 제어문

---

#### - if문 : 조건문

어떠한 조건이 충족 되었을 때에만 작동하는 제어문

*if 조건문:*

수행할 문장1

수행할 문장2

...

*else:*

수행할 문장1

수행할 문장2

...

들여쓰기!



# 파이썬 기본 문법 : 조건문

## If 조건문

```
>>> money = True
>>> if money:
...     print("택시를 타고 가라")
... else:
...     print("걸어 가라")

... 택시를 타고 가라
```

## elif 조건문

```
money = 2000

if money >= 3000:
    print('택시를 타고 가라')
elif money >= 1500:
    print('자전거를 빌리세요')
else:
    print('걸어서 가세요')
```

## 파이썬 기본 문법 : while문

### while 문

조건에 충족하지 않을 때 까지 반복

while 조건문:  
    수행하는 문장

```
treeHit = 0
```

```
while treeHit < 10:  
    treeHit += 1  
    print('나무를 %d번 찍었습니다.' %treeHit)  
    if treeHit == 10:  
        print("나무가 넘어갑니다.")
```

```
count = 0
```

```
while True:  
    count += 1  
    print(count)  
    if count == 10:  
        break  
    print('빠져나갑니다.')
```

```
while True:  
    count += 1  
    if count % 2 == 1: #홀수라면  
        continue  
    print(count)  
    if count == 10:  
        break
```

## 파이썬 기본 문법 : for문

### For문

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list:
...     print(i)
... one two three
```

### For문 활용

```
# marks1.py
marks = [90, 25, 67, 45, 80]
number = 0
for mark in marks:
    number = number + 1
    if mark >= 60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```

## 2. 파이썬 기초 자료형

---

### 연습문제 1

while문을 사용해 1부터 1000까지의 자연수 중 3의 배수의 합을 구해 보자.

## 2. 파이썬 기초 자료형

---

### 딕셔너리 자료형 연습문제

리스트에 10개 이상의 과일을 적고 (중복허용)  
이를 분류하는 코드를 작성하시오.

예) fruits = ['사과', '딸기', '파인애플' ... '사과' ]

Hint 1

딕셔너리 추가 a[(추가할 키)] = Value

Hint 1

딕셔너리 내에 Key가 있는지 확인 'key' in 딕셔너리

## 4. 프로그램의 입력과 출력

---

- 함수
- 사용자 입력과 출력
- 파일 읽고 쓰기
- 연습문제

## 4. 프로그램의 입력과 출력

---

- 함수 수학에서 사용하는 함수와 매우 유사하다

입력 값  
(input)

$$X = 2$$



$$f(x) = 2x + 3$$

결과 값  
(output)

7

단! 프로그래밍에서는 입력 값과 결과 값이 없을 수도 있음 (필요할 때만 사용)

## 4. 프로그램의 입력과 출력

---

### - 함수

함수를 사용하는 이유?

- 1) 매번 반복적인 작업들을 하는 것을 하나로 만들어 계속 호출하여 사용
- 2) 프로그램을 함수화하면 일목요연하게 정리된 하기 위해  
공장에서 각 공정들을 모듈화하여 동작하는 것과 비슷한 원리



## 4. 프로그램의 입력과 출력

---

### - 함수 정의 하기

```
def 함수명 (매개변수) :  
    수행할 문장  
    ...  
    수행할 문장  
    return 결과 값
```

print() / len() / count() ...

## 4. 프로그램의 입력과 출력

---

### - 함수

함수 예제 1) 매개변수 a,b를 더하는 함수를 만들어 봅시다.

```
def add(a,b):  
    return a+b
```

```
# add(2,3)
```

```
c = add(2,3)
```

```
print(c)
```

## 4. 프로그램의 입력과 출력

---

### - 함수

함수 예제 2) 입력 값과 출력 값이 없는 함수?

입력 값과 결과 값 모두 없는 함수

```
def say():  
    print('안녕하세요')
```

```
c = say()
```

```
print(c)
```

입력 값은 없지만 결과 값이 있는 함수

```
def say():  
    print('안녕하세요')  
    return 'HI'
```

```
c = say()
```

```
print(c)
```

## 4. 프로그램의 입력과 출력

---

### - 함수

인자의 갯수가 정해져 있지 않을 때 방법

```
def add(*args):  
    r = 0  
    for i in args:  
        r = r + i  
    return r
```

```
c = add(2,1,2,3,4,1,2,3,4)  
print(c)
```

## 4. 프로그램의 입력과 출력

---

인자가 여러 가지인데 반드시 하나는 구분하고 싶을 때

```
def calculator(cal, *args):
```

```
    if cal == "add":
```

```
        r = 0
```

```
        for i in args:
```

```
            r = r+i
```

```
    elif cal == "mul":
```

```
        r = 1
```

```
        for i in args:
```

```
            r = r * i
```

```
    return r
```

```
c = calculator('mul', 1,2,3,4,5,6,7,8,9)
```

```
print(c)
```

## 4. 프로그램의 입력과 출력

---

입력으로 들어오는 모든 수의 평균 값을 계산해 주는 함수를 만들어보세요

(단, 입력으로 들어오는 수의 개수는 정해져 있지 않다.)

※ 평균 값을 구할 때 len 함수를 사용

## 4. 프로그램의 입력과 출력

---

결과 값은 반드시 하나!

결과 값은 반드시 하나!

```
def add_mul(a,b):  
    return a+b, a*b
```

```
print(add_mul(2,3))
```

해당 값의 결과는??

한 개의 튜플 형태의 한 덩어리로 return 되는 것을 확인 할 수 있음

## 4. 프로그램의 입력과 출력

---

### - 사용자 입력과 출력

사용자 입출력 : 사용자가 입력한 값을 변수에 담아야 하는 경우

```
a = input()
```

```
print(a)
```

```
a = input('a 의 값을 넣어주세요 ')
```

```
print(a)
```



## 4. 프로그램의 입력과 출력

---

### - 사용자 입력과 출력

#### 사용자 입출력

```
a = input('더하고자 하는 값들을 넣어주세요')
```

```
a = a.split(',')  
s = 0
```

```
for i in a:  
    s = s + int(i)
```

```
print(s)
```

## 4. 프로그램의 입력과 출력

---

### - 파일 읽고 쓰기

**open()** 함수를 이용해서 파일을 읽고, 쓸 수 있음  
읽고, 쓰고, 수정 하고를 open함수를 이용

읽어올 때에는 'r' (Read)

새로 쓸 때에는 'w' (Write)

마지막에 추가하는 것은 'a' (Add)

## 4. 프로그램의 입력과 출력

---

### - 파일 읽고 쓰기

#### 파일 생성 / 데이터 쓰기

```
f = open('test.txt', 'w')  
f.close()
```

#### 파일에 데이터 작성

```
f = open('test2.txt', 'w', encoding='UTF-8')  
f.write('test 파일입니다.')
```

```
for i in range(1,101):  
    f.write("%d 번째 줄입니다\n" %i)  
f.close()
```

#### 데이터 읽어오기 readline()

```
f = open('test.txt', 'r', encoding='UTF-8')  
print(f.readline()) #한줄만 읽어 오는 함수  
f.close()
```

#### 데이터를 모두 읽어 오기

```
f = open('test.txt', 'r', encoding='UTF-8' )  
while True:  
    line = f.readline()  
    if not line:  
        break  
    print(line)  
f.close()
```

## 4. 프로그램의 입력과 출력

---

### - 파일 읽고 쓰기

#### readlines() 함수를 사용

```
f = open('test.txt','r',encoding='UTF-8')
# print(f.readlines())
lines = f.readlines()
for i in lines:
    print(i)
f.close()
```

#### 문서 전부 읽어 오기

```
f = open('test.txt','r',encoding='utf-8')
print(f.read())
f.close()
```

## 4. 프로그램의 입력과 출력

---

### - 파일 읽고 쓰기

#### Add (값 추가 하기 )

```
f = open('test.txt', 'a', encoding='utf-8')
for i in range(101,201):
    f.write("%d 번째 줄입니다\n" %i)
f.close()
```

## 4. 프로그램의 입력과 출력

---

### - 실습 예제

numbers.txt 파일 생성 후,

무작위 숫자 10개를 작성 후 저장

10

21

35

102

...

해당 파일을 읽어서 모두 더한 후 결과 값을 print로 출력해보세요

Read한 값은 str파일 int()로 변환 후에 프로그래밍 작성

## 5. 파이썬의 활용

---

- 클래스
- 모듈
- 패키지
- 예외처리
- 내장 함수
- 라이브러리
- 연습문제

## 5. 파이썬의 활용

---

### 클래스

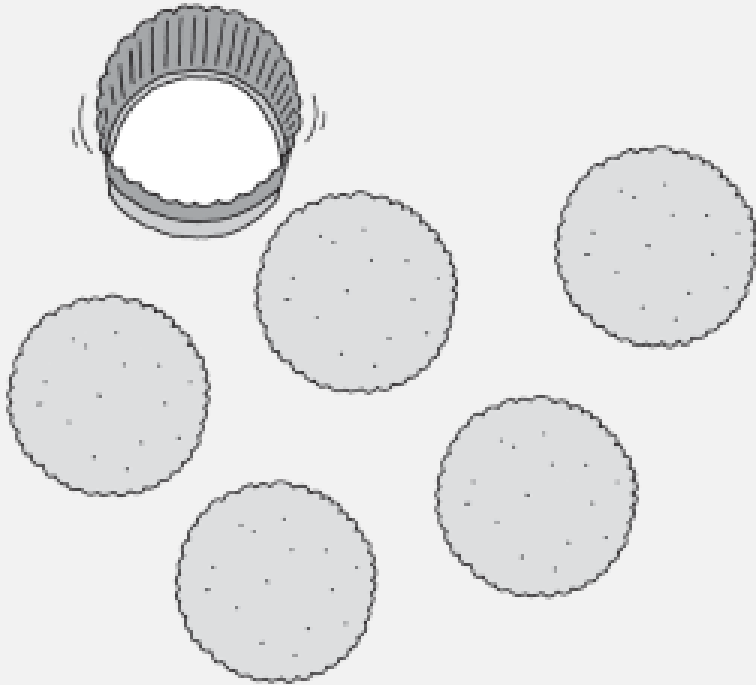
- 1) 클래스는 함수와 변수를 모아놓은 것
- 2) C언어에는 클래스가 없음.
- 3) 프로그램의 재 사용성을 위해 클래스를 사용  
(객체지향언어에서 사용)



## 5. 파이썬의 활용

### 클래스

과자 틀 = 클래스



만들어진 과자 = 객체 (Object)

[객체와 인스턴스??]

클래스로 만든 객체를 인스턴스라고 함.

`a = Cookie()` 이렇게 만든 `a`는 객체이다.  
그리고 `a` 객체는 `Cookie`의 인스턴스이다.

`a`는 객체

`a`는 `Cookie`의 인스턴스

이다.

## 5. 파이썬의 활용

---

### 클래스 왜 사용 할까?

입력한 값들을 더하는 함수를  
계속해서 만들어야 한다면??

```
add1()  
add2()  
add3()
```

```
result = 0  
result2 = 0
```

```
#값을 더하는 함수를 만들었습니다.  
def add(num):  
    global result  
    result = result + num  
    return result
```

```
#더하기 함수를 추가했을 때  
def add2(num):  
    global result2  
    result2 = result2 + num  
    return result2
```

```
print(add(3))  
print(add(4))
```

```
print(add2(5))  
print(add2(10))
```

## 5. 파이썬의 활용

---

### 클래스를 활용하여 간단히 작성

```
class Calculator:
    #추후 설명
    def __init__(self):
        self.result = 0

    def add(self, num):
        self.result += num
        return self.result

a = Calculator()
b = Calculator()
print(a.add(3))
print(a.add(7))
print(b.add(5))
print(b.add(10))
```

## 5. 파이썬의 활용

---

### 사칙연산 클래스 만들기

값을 지정해두고, 더하기, 나누기, 곱하기, 빼기 모두 가능한 클래스  
해당 클래스의 값은 100, 200 두 가지를 지정  
더해 300 빼 -100 곱해 200000 ...

```
# class FourCal:
#     def setdata(self,first,second):
#         self.first = first
#         self.second = second

# a = FourCal()
# a.setdata(100,200)
# print(a.first) #self.first
# print(a.second) #self.second
```

#더하기 기능 추가 만들어 보자

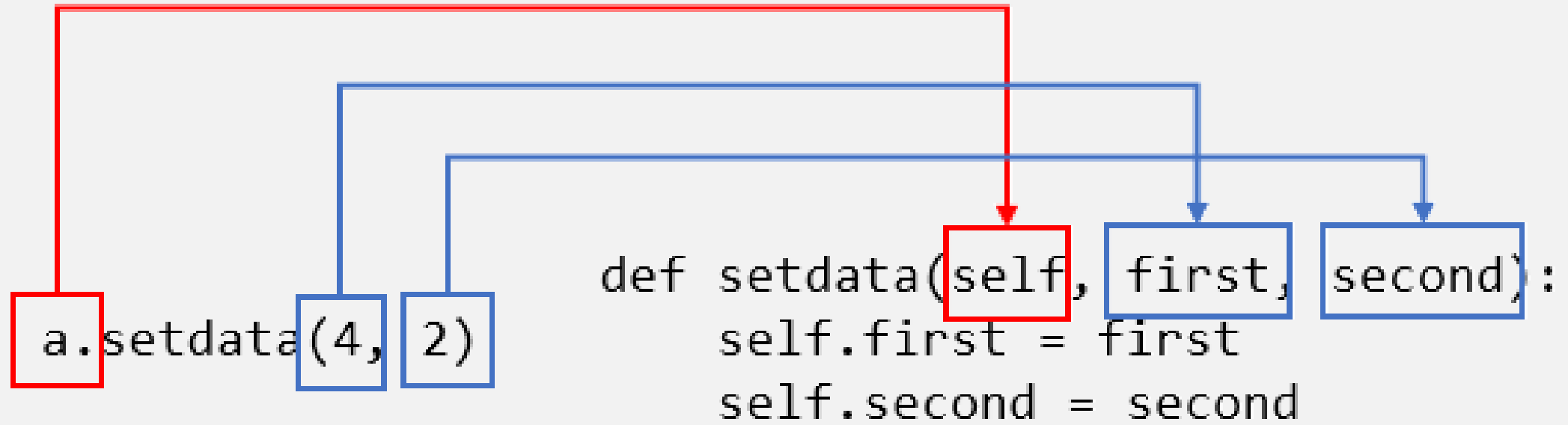
```
# class FourCal:
#     def setdata(self,first,second):
#         self.first = first
#         self.second = second

#     def add(self):
#         result = self.first + self.second
#         return result
```

## 5. 파이썬의 활용

---

### 클래스



## 5. 파이썬의 활용

---

클래스 생성자? ( Constructor )  
\_\_init\_\_ (언더스코어)

클래스가 생성됨과 동시에 실행되어 지는 함수  
\_\_ 언더스코어 2개와 \_\_init\_\_(self, 매개변수들 )로 정의

클래스를 선언 할 때

a = 클래스명(매개변수들)로 선언

## 5. 파이썬의 활용

---

클래스 상속? ( Extends )  
기존에 만들어진 클래스의 기능들을 그대로 가져오는 것

생성 방법

```
class 클래스이름 ( 상속받을 클래스명 ):  
    pass
```

라이브러리를 사용하거나, 클래스 수정이 불가능 한 경우 등등.. 매우 유용하게 사용

```
# 상속(Extends)  
class MoreFourCal(FourCal):  
    def pow(self):  
        result = self.first ** self.second  
        return result
```

## 5. 파이썬의 활용

---

### 매서드 오버라이딩? (Overriding)

쉽게 말해서 덮어쓰기 기능, 상속받은 클래스의 함수를 바꾸고 싶을 때 사용하는 방식

```
class MoreFourCal(FourCal):  
    def pow(self):  
        result = self.first ** self.second  
        return result  
  
    def div(self):  
        if self.first == 0 or self.second == 0:  
            return 0  
        else:  
            return self.first / self.second
```



## 5. 파이썬의 활용

---

- 모듈 : 함수나 변수 또는 클래스를 모아 놓은 파일

### 사용 방법

**import 모듈 이름**

※ import는 현재 디렉터리에 있는 파일이나 파이썬 라이브러리가 저장된 디렉터리에 있는 모듈만 불러올 수 있다. 파이썬 라이브러리는 파이썬을 설치할 때 자동으로 설치되는 파이썬 모듈을 말한다.

## 5. 파이썬의 활용

---

### - 예외 처리

프로그램을 만들다 보면 수없이 많은 오류를 만나게 됨. 하지만 파이썬은 try, except를 사용해서 예외적으로 오류를 처리할 수 있어서, 발생 가능한 오류처리를 하는 것이 필요함.

### - 예외 처리

```
try:
    ... (실행할 코드)
except 예외 as 변수:
    ... (실행할 코드)
```

\* 모든 에러에 대해서는 Exception as ...로

### - 예외 처리 후 작업

```
try:
    ... (실행할 코드)
except 예외 as 변수:
    ... (실행할 코드)
finally:
    (실행할 코드)
```

## 7. 데이터 분석 라이브러리 사용

---

- Numpy (배열 만들기)
- Pandas (데이터 시리즈와 프레임 다루기)
- 데이터 시각화 (Matplotlib)
- 연습 예제

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy (배열 만들기)

외부 패키지 설치하기

› pip install numpy

```
\basic_python> pip install numpy
```

외부 패키지 설치 후 사용하기

import numpy as np

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy ??

numpy 배열은 파이썬 리스트보다 더욱 간편하고 빠르다.

Python 연산과학 분야에서 상당히 유용하게 사용되어지고 있음  
(고성능 과학 계산용 패키지)

Matrix와 Vector와 같은 Array연산의 사실상의 표준

고성능 다차원 matrix 연산을 위한 여러가지 함수들을 제공

참고문서

<https://numpy.org/doc/1.19/numpy-ref.pdf>

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy 추천 문서

<https://cs231n.github.io/python-numpy-tutorial/#numpy>

<https://numpy.org/doc/stable/user/quickstart.html>

### 참고문서

<https://numpy.org/doc/1.19/numpy-ref.pdf>

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy

matrix연산, row연산, col연산을 깔끔하게 지원

list 연산과 numpy 연산과의 차이점

```
list_a = [1,2,3]
list_b = [4,5,6]

lsum = list_a + list_b
print(lsum)
```

```
import numpy as np

np_list_a = np.array([1,2,3])
np_list_b = np.array([4,5,6])

nsum = np_list_a + np_list_b

print(nsum)
```

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy Array 생성

matrix연산, row연산, col연산을 깔끔하게 지원

```
import numpy as np

ndArray_A = np.array([1,2,3])
ndArray_B = np.array([[1,1,1],[2,2,2]], dtype=float)

print(ndArray_A)
print(ndArray_B)

print(ndArray_A.shape)
print(ndArray_B.shape)
```



## 7. 데이터 분석 라이브러리 사용

---

### - Numpy Array 생성

matrix연산, row연산, col연산을 깔끔하게 지원

matrix 생성을 할 때, 일정 값으로 채울 때

```
zeros = np.zeros([4,2] , dtype = float)
print(zeros)
```

```
ones = np.ones([4,2], dtype=float)
print(ones)
```

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy Array 생성

matrix연산, row연산, col연산을 깔끔하게 지원

단위 행렬 생성하기

```
eye = np.eye(3)
print(eye)
```

```
eye = np.eye(3,2, k=1)
print(eye)
```

## 7. 데이터 분석 라이브러리 사용

---

### - 행렬 전환과 형태 변형하기

#### 1차원 배열을 변형해보기

```
arr = np.array(range(1,9))  
print(arr)  
print(arr.shape)  
  
arr2 = arr.reshape(2,4)  
arr2 = arr.reshape(-1,4)  
  
print(arr2)  
  
# print(arr2.T)
```

## 7. 데이터 분석 라이브러리 사용

---

### - flatten

다차원 array를 1차원 array로 변환

```
# flatten  
  
arr = np.array([[1,2,3],[4,5,6],[7,8,9]])  
print(arr.flatten())  
|
```

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy 인덱싱과 슬라이싱

#### 인덱싱

```
a = np.array([[1,2,3],[4.5,5,6]], dtype=int)
a[1,0] = 5
print(a[1,0])
```

List와는 달리 이차원 배열에서 [0,0]과 같은 표기법을 제공함

Matrix일 경우 앞은 row 뒤는 column을 의미함

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy 인덱싱과 슬라이싱

#### 슬라이싱

```
list_A = [[1,2,5,8],  
          [1,2,5,8],  
          [1,2,5,8],  
          [1,2,5,8]]  
  
a = np.array(list_A, dtype=int)  
print(a)  
  
print(a[0:2,1:3])  
print(a[1,:])
```

## 7. 데이터 분석 라이브러리 사용

---

### - Numpy 인덱싱과 자르기

데이터클리닝 (data cleaning)을 위해, 일정 조건에 부합하는 데이터를 처리하는 방법

```
indexing = a > 3
print(indexing)

a[indexing] = 0
print(a)
```

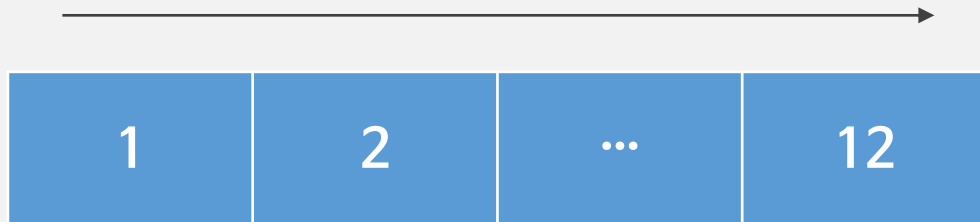
## 7. 데이터 분석 라이브러리 사용

---

### – Numpy operation Function

#### 1차원 Vector Sum

axis = 0



```
a = np.arange(1,13)
print(a)
print(a.sum())
```



## 7. 데이터 분석 라이브러리 사용

### – Numpy operation Function

axis : operation Function을 할 때 기준이 되는 축

axis = 1

1	2	3	4
5	6	7	8
9	10	11	12

axis = 0

```
a = np.arange(1,13).reshape(3,4)
print(a)
```

```
print(a.sum(axis = 0))
print(a.sum(axis = 1))
```

## 7. 데이터 분석 라이브러리 사용

---

### - Pandas

numpy는 수치형 데이터를 행과 열, 인덱스와 같은 데이터 속성과 엮지는 못한다.  
pandas로 이러한 어려움을 해결

외부 패키지 설치하기

› pip install pandas

```
> pip install pandas
```

외부 패키지 설치 후 사용하기

import pandas as pd

## 7. 데이터 분석 라이브러리 사용

---

### - Pandas

python에서 Data를 matrix단위로 처리할 때 많이 사용

row나 col씩 접근이 가능하다! = 데이터 처리속도가 빠름

Vector 형태의 **Series**

Matrix 형태의 **DataFrame**이 존재

[https://pandas.pydata.org/docs/user\\_guide/10min.html](https://pandas.pydata.org/docs/user_guide/10min.html)

## 7. Pandas 데이터 분석하기

---

### - Pandas

Series 만들기 (1차원 Vector)

DataFrame 만들기 (Matrix)

```
a = pd.Series([1,2,3,4,5])  
b = pd.DataFrame([1,2,3,4,5])
```

## 7. Pandas 데이터 분석하기

---

### - Pandas

#### DataFrame 만들기

DataFrame은 Matrix 형태로 생성할 때 사용

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame({  
    "a" : [4,5,6],  
    "b" : [7,8,9],  
    "c" : [10,11,12]  
}, index=[1,2,3])
```

## 7. Pandas 데이터 분석하기

---

### - Pandas

DataFrame 열(column) 데이터 인덱싱 (데이터 가져오기)

Series 는 인덱싱 숫자로 가져오고, DataFrame은 열 이름으로 가져옴

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
a = pd.DataFrame(li, index=[1,2,3])  
print(a['a'])  
print(a[['a']])  
print(a[['a', 'b']])
```

df["컬럼명"] => Series    // df.컬럼명으로도 가져올 수 있음

## 7. Pandas 데이터 분석하기

---

### - Pandas

DataFrame 열(column) 데이터 인덱싱 수정 및 추가

수정

`df["컬럼명"] = 값` ( or `df.컬럼명 = 값` )

추가

`df["신규 컬럼명"] = 값`

```
df['c'] = [2,3,4] #수정 할 때 |
df['d'] = [5,3,4]
print(df)
```

## 7. Pandas 데이터 분석하기

---

### - Pandas

DataFrame 열(column) 데이터 인덱싱 수정 및 추가

Series 데이터를 활용하여 기존 Matrix에 추가 하는 방법

```
val = pd.Series([5,6], index=(2,3))  
df['add'] = val  
print(df)
```

기존 데이터를 활용하여 열 데이터 추가

```
df['filter'] = df['b'] > 5  
print(df)
```



## 7. Pandas 데이터 분석하기

---

### - Pandas

DataFrame 데이터를 가져오는 함수

Series 는 인덱싱 숫자로 가져오고, DataFrame은 열 이름으로 가져옴

```
print(df.head())  
print(df.tail())  
print(df.sample(n=3))
```

Matrix의 윗 부분 5개 행을 가져올 때

Matrix의 맨 아래 5개 행을 가져올 때

Sample 랜덤으로 값을 가져올 때 사용

## 7. Pandas 데이터 분석하기

### - Pandas

#### DataFrame 행(row)데이터 인덱싱 하는 방법

df[0:2] 추출 하는 방법은 비슷하지만, 혼돈이 될 수 있으므로, 비추천

행 데이터를 가져올 때에는 loc, iloc 함수를 사용

loc() 행, 열 이름으로 가져오고 / iloc() - numpy 인덱싱 방식

	a	b	c	d
1	1	4	2	5
2	2	5	3	3
3	3	6	4	4

기존 데이터 프레임

```
print(df)
print()
print(df.loc[1:2])
print(df.loc[1:2, 'a': 'c'])
df.loc[4] = [5,6,7,8,8,False]
```

행 데이터 가져오기

## 7. Pandas 데이터 분석하기

---

### - Pandas

#### DataFrame 불리언 인덱싱

[]에 들어가는 조건을 기준으로 데이터 프레임 안의 각 열을 순회하며 검사

df에 'a' 열에 있는 컬럼 숫자가 2보다 큰 row값을 출력

```
print(df[df['a'] > 2])
```

2개 이상의 조건일 때 값을 가져오는 것

```
print(df[ (df['a'] > 2) & (df['a'] < 6) ])
```

## 7. Pandas 데이터 분석하기

---

### - Pandas

#### 엑셀 파일 읽어오기

공공데이터 포털을 활용해서 데이터 읽어오기

```
df = pd.read_csv('./파일명.csv', encoding = 'cp949')
```

```
df = pd.read_csv('서울특별시 동대문구_동대문구 코로나 행정동별 확진자 수_20200713.csv', encoding='cp949')
print(df.info())
print(df.shape)
print(df.tail())
print(df.describe())

print(df['확진자 수'].sum())
```



## 7. 데이터 분석 라이브러리 사용

---

– matplotlib

데이터 시각화 도구

외부 패키지 설치하기

› `pip install matplotlib`

```
PS C:\Users\kkong\OneDrive\바탕 화면\basic_python> pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.3.1-cp38-cp38-win_amd64.whl (8.5 MB)
    |#####| 8.5 MB 6.8 MB/s
Collecting certifi>=2020.06.20
  Downloading certifi-2020.6.20-py2.py3-none-any.whl (156 kB)
    |#####| 156 kB 3.3 MB/s
Collecting cycler>=0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Requirement already satisfied: python-dateutil>=2.1 in c:\python38\lib\site-packages (from matplotlib) (2.8.1)
Collecting pillow<6.3.0
```

외부 패키지 설치 후 사용하기

`import matplotlib.pyplot as plt`

## 7. 데이터 분석 라이브러리 사용

---

– matplotlib

matplotlib이란??

matplotlib은 python에서 가장 많이 사용되는  
데이터 시각화 (Data Visualization ) 라이브러리 패키지

Line Plot, Bar chart, Pie chart, Histogram, Box Plot등 다양한  
차트와 스타일을 지원함

참조 문서      <https://matplotlib.org>

## 7. 데이터 분석 라이브러리 사용

### - matplotlib

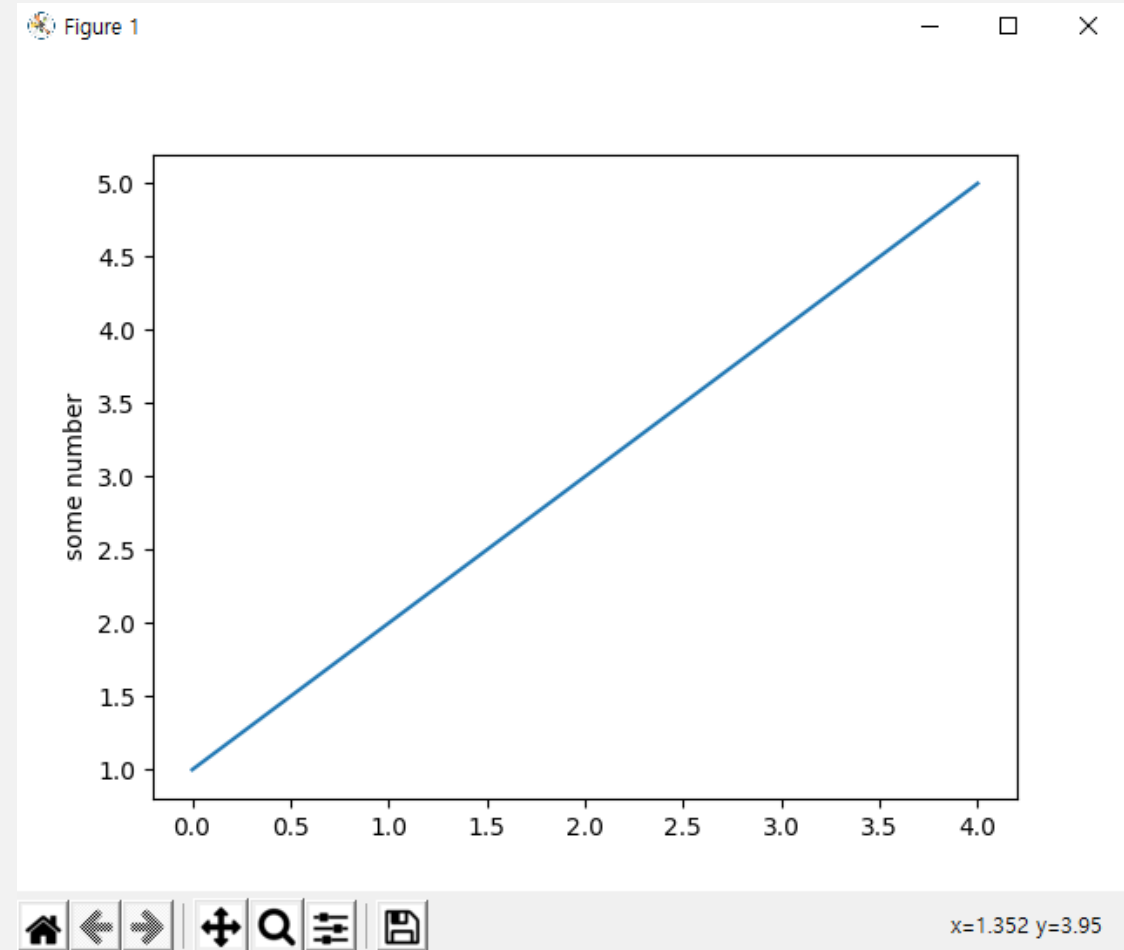
데이터 시각화는 보고서의 필수 아이템 !!

pyplot 객체를 활용해서 데이터를 표시  
pyplot에 데이터를 쌓은 다음 show를 통해 flush

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4,5])
plt.ylabel('some number')
plt.show()
```

```
plt.plot([1,2,3,4,5],[1,3,5,7,9]) # x, y
plt.ylabel('some number')
plt.show()
```



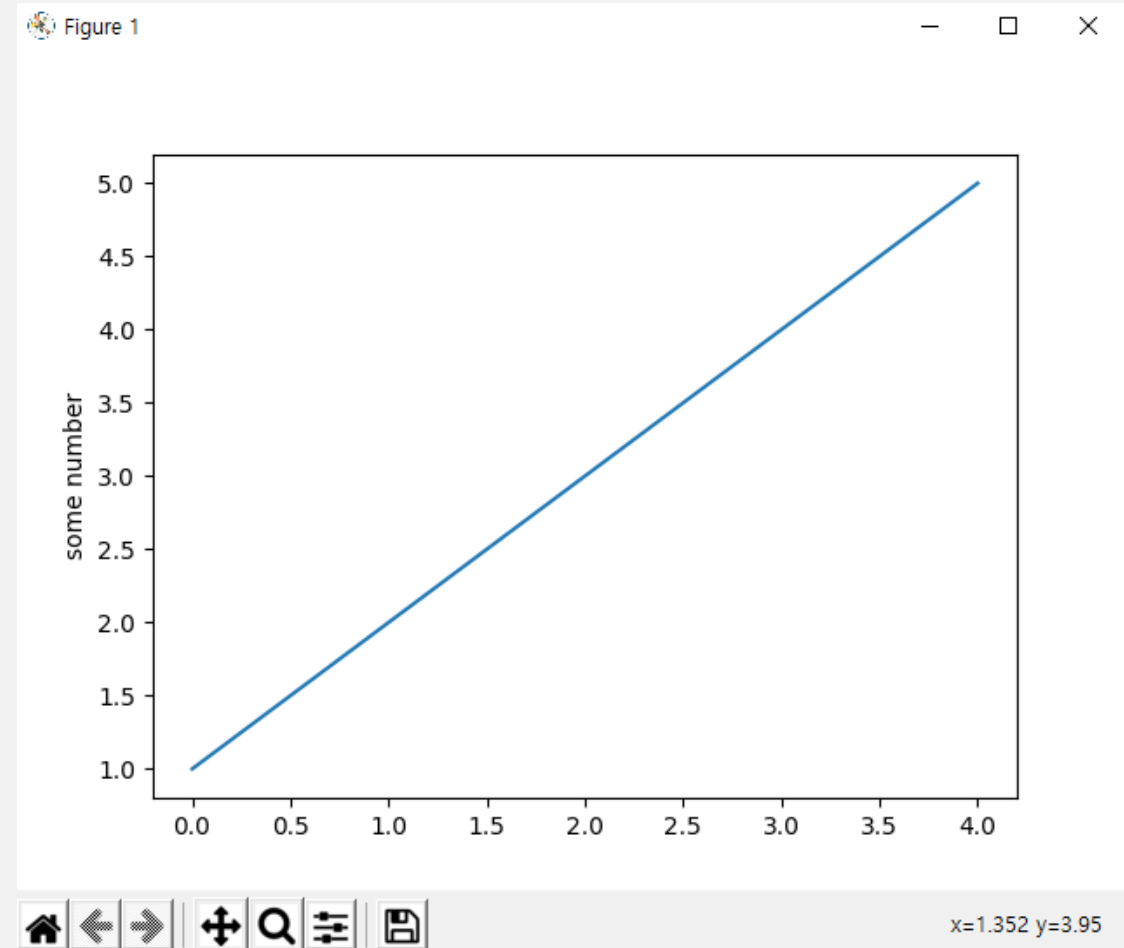
## 7. 데이터 분석 라이브러리 사용

### - matplotlib

Graph는 figure 객체에 생성됨  
pyplot 객체를 통해 Figure에 그래프가 그려짐

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4,5],[1,3,5,7,9]) # x, y
plt.plot([1,2,3,4,5],[10,20,30,40,80])
plt.ylabel('some number')
plt.show()
```





## 7. 데이터 분석 라이브러리 사용

---

### - matplotlib

figure에 여러가지 그래프를 그리는 방법  
subplot을 활용하여 plot을 생성

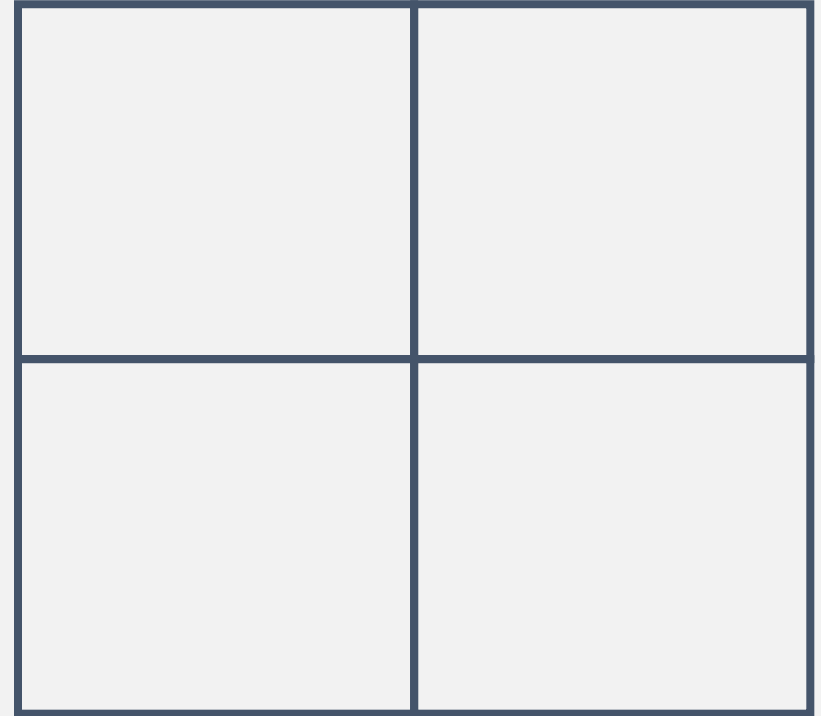
`plt.subplot( A, B, C )`

A : num of row [행]

B : num of col [열]

C : position [ 위치 ]

2x2 subplot 생성



## 7. 데이터 분석 라이브러리 사용

---

### - matplotlib

Subplot 을 활용하여 2개의 그래프를 표현

`plt.subplot( 2, 1, C )`

```
x_1 = range(100)
y_1 = [value for value in x_1]

x_2 = range(100)
y_2 = [value**2 for value in x_2]

plt.subplot(2,1,1)
plt.plot(x_1,y_1)
plt.subplot(2,1,2)
plt.plot(x_2, y_2)
plt.show()
```

2x1 subplot 생성



## 7. 데이터 분석 라이브러리 사용

### 다양한 차트 사용법

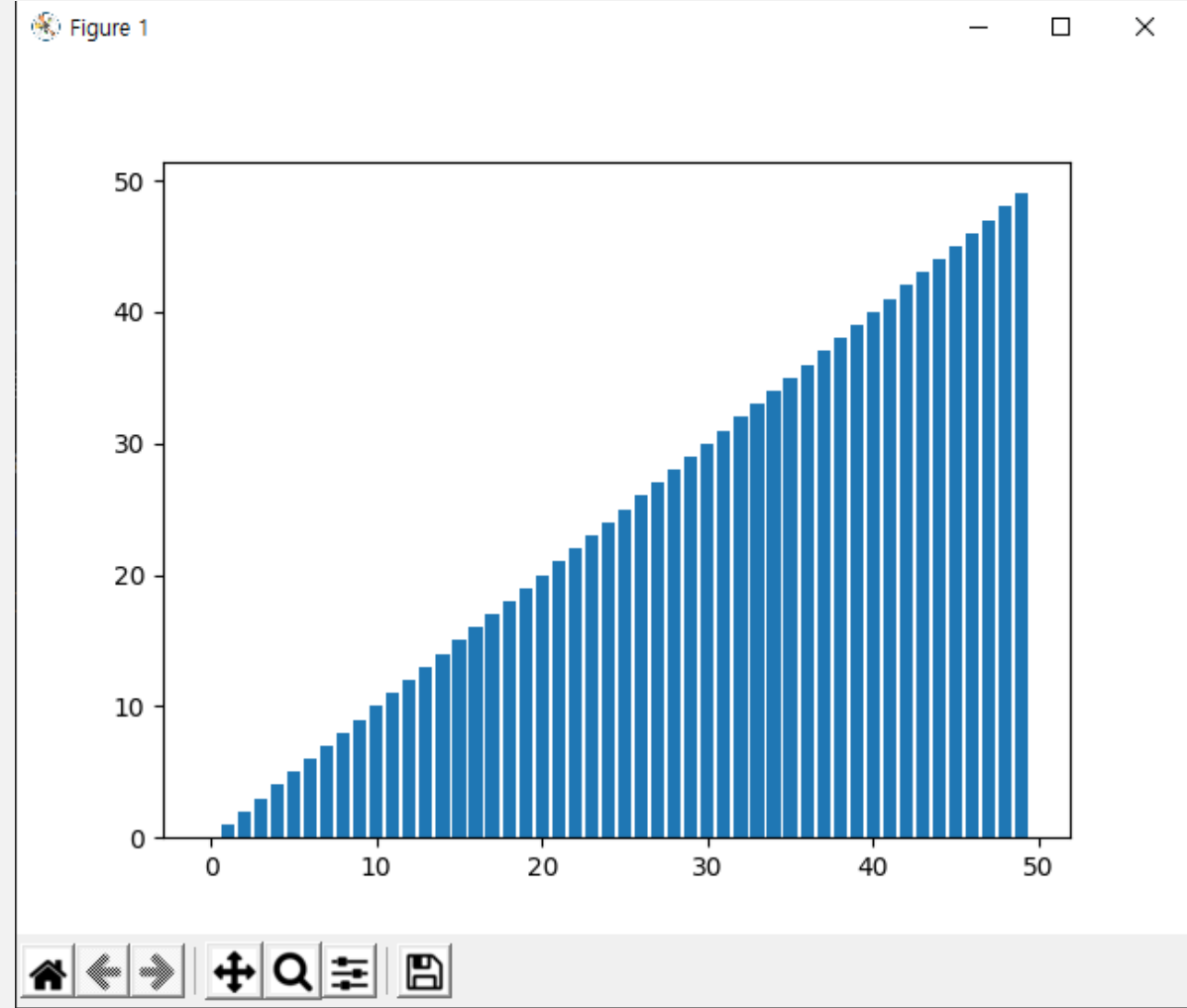
#### 1) Bar Chart

```
x_1 = range(50)
y_1 = [value for value in x_1]

plt.bar(x_1, y_1)
plt.show()
```

Option

width = 0.5 (막대 그래프 폭 설정)  
color = 'r', 'g', 'b' ... 와 같이 색 설정 가능  
align='center' or 'edge'



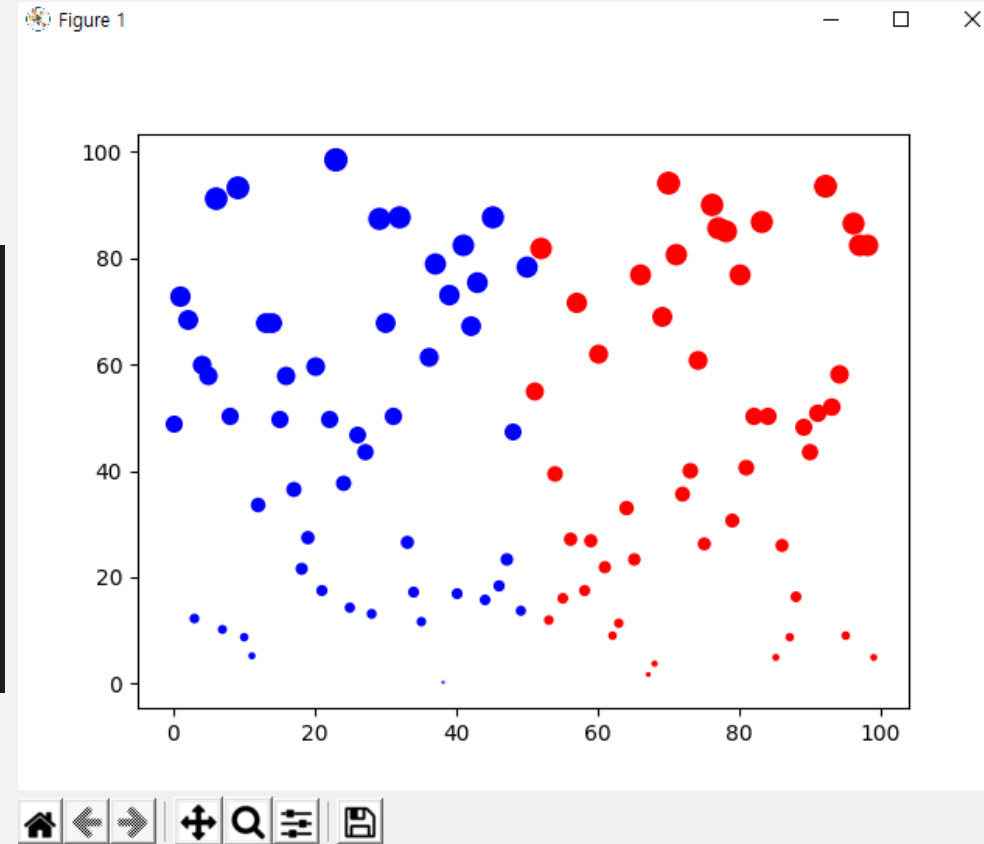
## 7. 데이터 분석 라이브러리 사용

### 다양한 차트 사용법

#### 2) Scatter

두 변수의 관계를 보여주는 자료 표시 방법

```
x_2 = range(100)
y_2 = np.random.rand(100) * 100
for i in range(0,100):
    if i > 50:
        plt.scatter(x_2[i],y_2[i], color = 'r', s = y_2[i])
    else:
        plt.scatter(x_2[i],y_2[i], color = 'b', s = y_2[i])
plt.show()
```



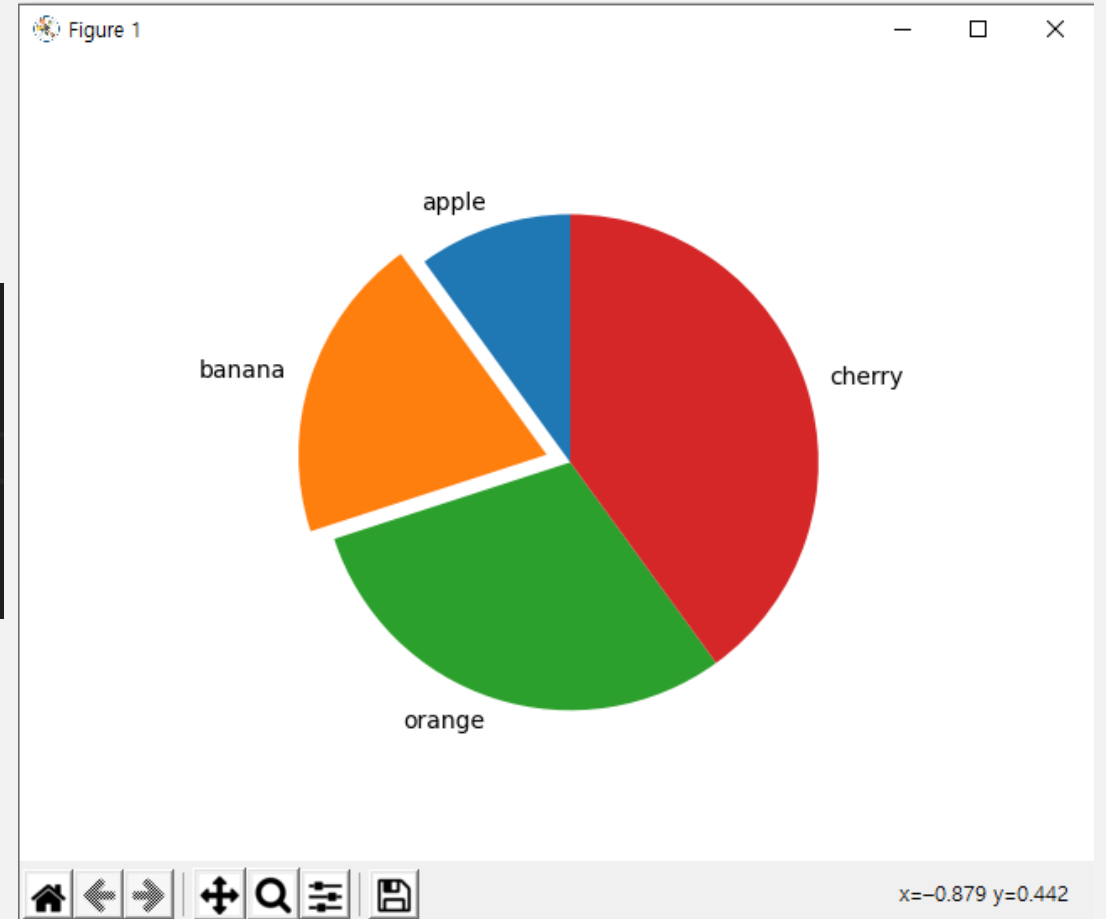
## 7. 데이터 분석 라이브러리 사용

### 다양한 차트 사용법

#### 3) Pie Chart

비율을 한눈에 볼 수 있다는 장점

```
labels = ['apple', 'banana', 'orange', 'cherry']  
size = [10, 20, 30, 40]  
explode = (0, 0.1, 0, 0)  
  
plt.pie(size, explode=explode, labels=labels, startangle=90)  
plt.show()
```



## 7. 데이터 분석 라이브러리 사용

### 다양한 차트 사용법

Matplotlib.org

Tutorials에 있는 다양한 시각화 그래프를

참조하여 나만의 그래프를 만들어 보기



Version 3.3.1

InstallationDocumentationExamplesTutorialsContributing

home | contents » Gallery

### Gallery

This gallery contains examples of the many things you can do with Matplotlib. Click on any image to see the full image and source code.

For longer tutorials, see our [tutorials page](#). You can also find [external resources](#) and a [FAQ](#) in our [user guide](#).

#### Lines, bars and markers



Stacked bar chart



Grouped bar chart with labels



Horizontal bar chart



Broken Barh



Plotting categorical variables



Plotting the coherence of two signals



CSD Demo



Curve with error band

## 8. 데이터 분석 예제

---

- 공공 데이터 포털
- 문제 정의
- 데이터 분석
- 데이터 전처리
- 시각화

# 데이터 분석 이해

## DIKW 피라미드



DATA

의미가 부여 안된 객관적 사실  
가공하기전 데이터 자체



Information

데이터 구조화를 통해  
유의미한 정보로 분류  
데이터간의 상관관계 속에  
의미를 부여한다.



Knowledge

데이터를 기반으로  
의사결정에 활용  
예)  
B지역에 이용자가 학생이 많아  
판매가 증진된다.



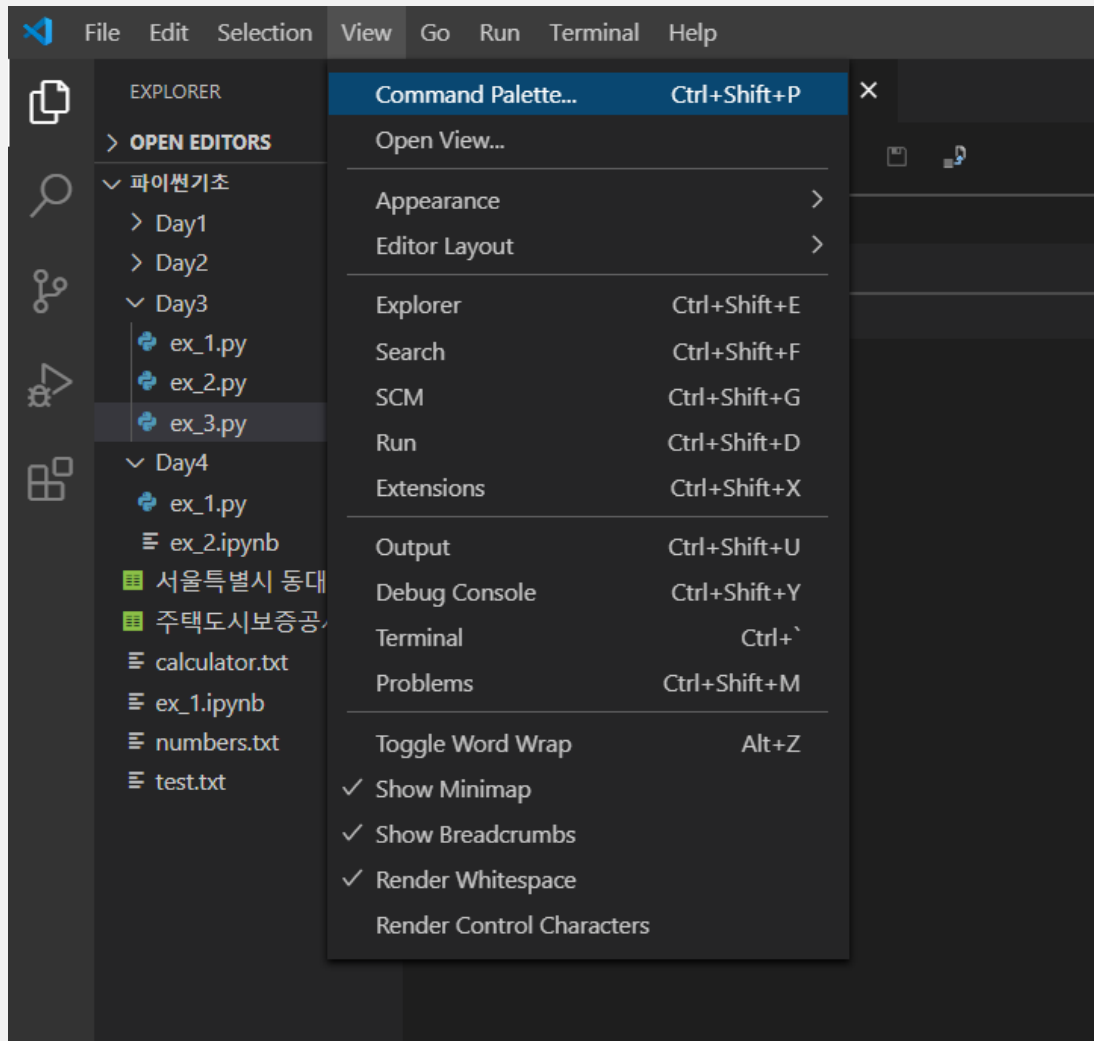
Wisdom

전체 서비스에  
응용하여  
전략적인 관리체계  
또는 문제 해결



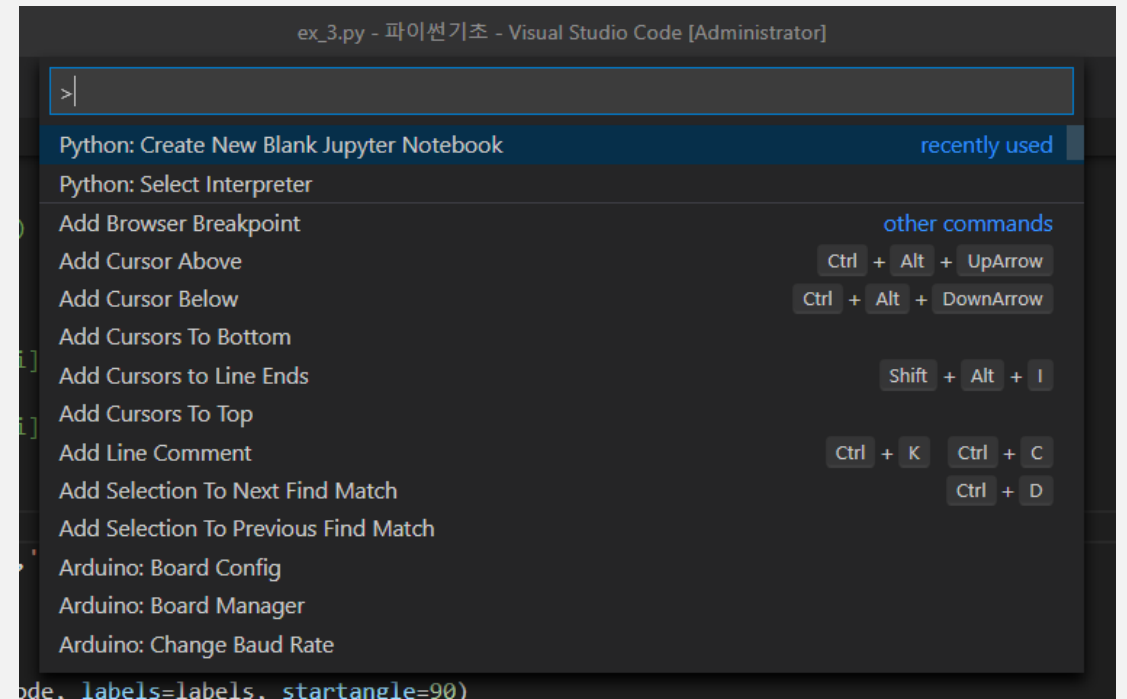
## 8. 데이터 분석 예제

### 개발환경 설치 : Jupyter Notebook



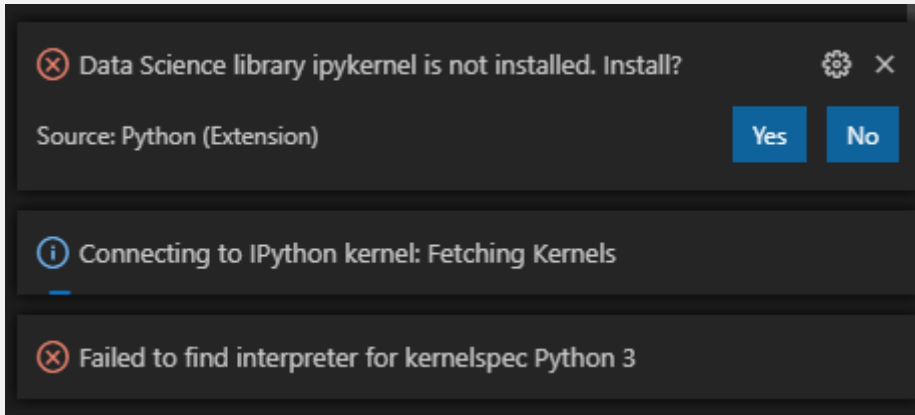
입력 >>

Python Create New Blank Jupyter Notebook



## 8. 데이터 분석 예제

### 개발환경 설치 : Jupyter Notebook



오른쪽 하단 알림창 Yes 클릭

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

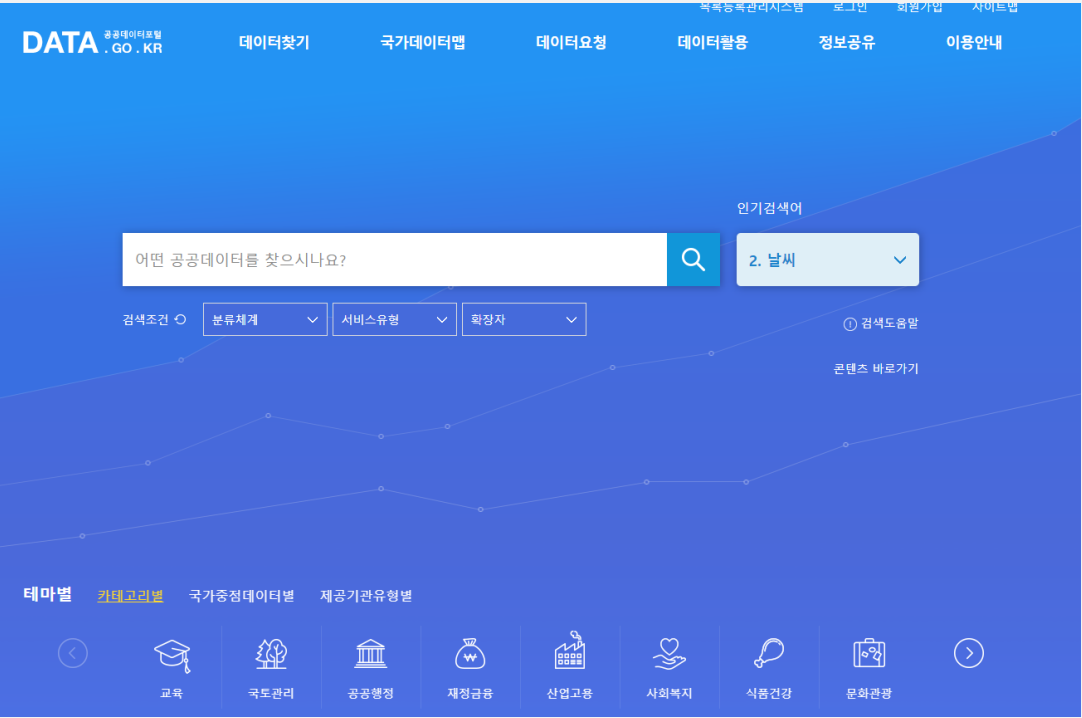
Collecting pywin32>=1.0; sys_platform == "win32"
  Downloading pywin32-228-cp38-cp38-win32.whl (8.4 MB)
  |████████████████████| 8.4 MB 6.4 MB/s
Collecting parso<0.8.0,>=0.7.0
  Downloading parso-0.7.1-py2.py3-none-any.whl (109 kB)
  |████████████████████| 109 kB 6.8 MB/s
Collecting wcwidth
  Downloading wcwidth-0.2.5-py2.py3-none-any.whl (30 kB)
Installing collected packages: tornado, six, python-dateutil, pyzmq, ipython-genutils, decorator, traitlets, pywin32, jupyter-core, jupyter-client, colorama, ll, ipython, ipykernel
```

가설

부동산 가격의 상승은  
아파트 분양가에도 영향이 있을 것이다.

# 8. 데이터 분석 예제

## 데이터 수집 - 공공데이터 포털 >> 아파트 분양가격 검색



<https://www.data.go.kr/data/15060028/fileData.do>

DATA공공데이터포털GO . KR

데이터찾기

국가데이터맵

데이터요청

데이터활용

정보공유

이용안내

파일데이터 상세

CSV주택도시보증공사\_전국 신규 민간아파트 분양가격 동향

주택분양보증을 받아 분양한 전체 민간 신규아파트 분양가격 동향

0

관심

다운로드

파일데이터 정보

파일데이터명	주택도시보증공사_전국 신규 민간아파트 분양가격 동향_20200430		
분류체계	사회복지 - 주택	제공기관	주택도시보증공사
관리부서명	주택도시금융연구원	관리부서 전화번호	051-955-5336
보유근거		수집방법	
업데이트 주기	월간	차기 등록 예정일	2020-06-22
매체유형	텍스트	전체 행	4675
확장자	CSV	다운로드(바로그가기)	124
데이터 한계		키워드	주택분양보증,아파트,분양가
등록	2020-05-20	수정	2020-05-20
제공형태	공공데이터포털에서 다운로드(원문파일등록)		

## 8. 데이터 분석 예제

---

### 데이터 가져오기

pandas, numpy, matplotlib 활용

plt.rc 함수를 활용하여 폰트 설정

- window User : 'Malgun Gothic'
- Max User : 'Apple Gothic'

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

plt.rc('font', family='Malgun Gothic')
plt.rc('axes', unicode_minus=False)

df = pd.read_csv('주택|도시보증공사_전국 신규 민간아파트 분양가격 동향_20200430.csv', encoding='cp949')
```

## 8. 데이터 분석 예제

---

### 데이터 분석

내가 가져온 데이터가 어떻게 구성되어 있는지 확인

shape : 데이터 셋의 구조를 확인

info : 데이터 요약

head / tail or sample : 데이터 샘플 확인

isnull : 데이터 결측치 확인

df['컬럼명'] : 데이터 컬럼 확인

```
df = pd.read_csv('주택도시보증공사_전국 신규 민간아파트 분양가격 동향_20200430.csv', encoding='cp949')
print(df.shape)
print(df.info())
print(df.head())
print(df.tail())
print(df.isnull().sum())
```

## 8. 데이터 분석 예제

---

### 데이터 전처리

데이터 분석에 편리하도록 데이터의 타입이나 컬럼 값들을 수정 하거나 변경

`astype(dtype)` : 데이터 타입을 변경

`to_numeric(컬럼명, errors='coerce')` : 데이터에 빈칸 변경할 때 에러나는 경우

```
df['분양가격 (㎡)'] = pd.to_numeric(df['분양가격 (㎡)'], errors='coerce')
df['평당가격'] = df['분양가격 (㎡)'] * 3.3
print(df.head())
```

## 8. 데이터 분석 예제

---

### 데이터 전처리

내부 내용의 일부 글자를 통일

unique() : 어떤 값들이 분류 되어 있는지 확인

str 문자열 변경에 사용하는 replace 함수를 사용해서 수정

```
# print(df["규모구분"].unique())
df["전 용면적 "] = df["규모구분"].str.replace("전 용면적 ", "")
df["전 용면적 "] = df["전 용면적 "].str.replace("초과 ", "~")
df["전 용면적 "] = df["전 용면적 "].str.replace("이하 ", "")
df["전 용면적 "] = df["전 용면적 "].str.replace(" ", "").str.strip()
print(df["전 용면적 "])
```



## 8. 데이터 분석 예제

---

### 데이터 전처리

분석하고자 하는 데이터들만 추출하여 메모리 사용량을 최소화

`drop(columns = ['컬럼명'])` : 삭제하고자 하는 열을 삭제  
또는 `drop(['컬럼명'], axis = 1)` : axis는 (0 : 행, 1 : 열)

```
df = df.drop(columns=["규모구분"])  
print(df.info())
```

메모리 사용량이 얼마만큼 줄어 들었는지 확인

## 8. 데이터 분석 예제

---

### 데이터 그룹화

데이터를 그룹화 하여 데이터를 집계하고, 시각화 해보기

`groupby(["인덱스로 사용할 컬럼"])["계산할 컬럼"].연산()`

\*연산 : [ `sum()` : 합계 / `min()` : 최소값 / `max()` : 최대값

`mean()` : 평균값 / `median()` : 중간값

`describe()` : 연산 모두 실행

```
print(df.groupby(['전용면적']).mean())
print(df.groupby(['전용면적'])['평당가격'].mean())
print(df.groupby(['지역명', '전용면적'])['평당가격'].mean().unstack())

g = df.groupby(['지역명', '연도'])['평당가격'].mean().unstack().round()
print(g)
```

## 8. 데이터 분석 예제

---

### 데이터 시각화 해보기

그룹화 한 데이터를 선그래프와 막대그래프로 그려보자

plot () 함수를 활용 하여 데이터 그리기

\* 함수 옵션

line() : line 그래프 / bar() : 막대 그래프 / hist() : 히스토그램

box() : box plot / kde() : 커널밀도추정 or density / area() : area plot

```
# 지역별 평단 가격
g = df.groupby(['지역명', '연도'])['평당가격'].mean().unstack().sort_values(by='지역명', ascending=True)
g.plot.bar(rot=0, figsize=(10,5))
plt.show()
```