국가공무원인재개발원

파이썬 실무 활용 기초 과정

2021년

<목 차>

제 목	01장 파이썬 설치 및 실행	
상세내용	파이썬 설치 및 간단한 출력 살펴표기	

1. 파이썬 프로그램이란?

- 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어•파이썬이라는이름을 자신이 좋아하는 코미디 쇼인 "몬티파이썬의날아다니는 서커스 (MontyPython'sFlyingCircus)"에서 따왔다고 함
- 사전적 의미는 고대 신화에 나오는 파르나소스산의 동굴에 살던 큰 뱀을 뜻하며, 아폴로 신이 델파이에서 파이썬을퇴치했다는 이야기가 전해지고 있다.
- 대부분의 파이썬 책 표지와 아이콘이 뱀 모양으로 그려져 있는 이유가 여기에 있다.

[파이썬 프로그램 장점]

- opensource 로서 다양한 패키지(모듈, 라이브러리)를 전세계에서 개발한다.
- 범용적이며 프로그램 학습이 타 프로그램에 비해 쉽다.

[파이썬 프로그램 단점]

- 대단히 빠른 속도를 요구하거나 하드웨어를 직접 건드려야 하는 프로그램에는 부적합 (C, C++ 이용)
- 2.X 버전과 3.X 버전이 호환되지 않는다.

Jan 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	^	C	17.38%	+1.61%
2	1	v	Java	11.96%	-4.93%
3	3		Python	11.72%	+2.01%
4	4		C++	7.56%	+1.99%
5	5		C#	3.95%	-1.40%
6	6		Visual Basic	3.84%	-1.44%
7	7		JavaScript	2.20%	-0.25%
8	8		PHP	1.99%	-0.41%
9	18	^^	R	1.90%	+1.10%
10	23	^^	Groovy	1.84%	+1.23%

[https://www.tiobe.com/tiobe-index/ 2021년도 1월. 프로그래밍 언어인기지표]

2. 파이썬으로 작업가능한 영역

가. 시스템 유틸리티 제작”

파이썬은 운영체제(UNIX, Windows 등)의 시스템 명령어들을 이용할 수 있는 각종 도구를 갖추고 있기 때문에 이를 바탕으로 갖가지 시스템 유틸리티를 만드는 데 유리하다. 실제로 여러분은 시스템에서 사용 중인 서로 다른 유틸리티성 프로그램들을 하나로 뭉쳐서 큰 힘을 발휘하게 하는 프로그램들을 무수히 만들어낼 수 있다.

나. GUI프로그래밍

GUI(Graphic User Interface) 프로그래밍이란 쉽게 말해 윈도우 창처럼 화면을 보며 마우스나 키보드로 조작할 수 있는 프로그램을 만드는 것이다. 파이썬으로 GUI 프로그램을 만드는 것은 다른 언어를 이용해 만드는 것보다 훨씬 쉽다. 대표적인 예로 파이썬 프로그램을 설치할때 함께 설치되는 기본 모듈인 Tkinter(티케이인터)를 이용해 만드는 GUI 프로그램을 들 수 있다. 실제로 Tkinter를 이용한 파이썬 GUI 프로그램의 소스 코드는 매우 간단하다. Tkinter를 이용하면 단 5줄의 소스 코드만으로도 윈도우 창을 띄울 수 있다.

다. 웹프로그래밍

일반적으로 익스플로러나 크롬, 파이어폭스와 같은 브라우저를 이용해 인터넷을 사용하는데, 누구나 한 번쯤 웹 서핑을 하면서 게시판이나 방명록에 글을 남겨 본 적이 있을 것이다. 그러한 게시판이나 방명록을 바로 웹 프로그램이라고 한다. 파이썬은 웹 프로그램을 만들기에 매우 적합한 도구이며 실제로 파이썬으로 제작된 웹사이트는 셀 수 없을 정도로 많다.

라. 수치연산 프로그래밍

사실 파이썬은 수치 연산 프로그래밍에 적합한 언어는 아니다. 수치가 복잡하고 연산이 많다면 C같은 언어로 하는 것이 더 빠르기 때문이다. 하지만 파이썬에는 Numeric Python이라는 수치 연산 모듈이 제공된다. 이 모듈은 C로 작성되었기 때문에 파이썬에서도 수치 연산을 빠르게 할 수 있다.

마. “데이터 분석, 사물 인터넷(IoT)

파이썬으로 만들어진 판다스(Pandas)라는 모듈을 이용하면 데이터 분석을 더 쉽고 효과적으로 할 수 있다. 데이터 분석을 할 때 아직까지는 데이터 분석에 특화된 "R"이라는 언어를 많이 사용하고 있지만, 판다스가 등장한 이후로 파이썬을 이용하는 경우가 점점 증가하고 있다. 사물 인터넷 분야에서도 파이썬은 활용도가 높다. 한 예로 라즈베리파이(Raspberry Pi)는 리눅스 기반의 아주 작은 컴퓨터이다. 라즈베리파이를 이용하면 홈시어터나 아주 작은 게임기 등 여러 가지 재미있는 것들을 만들 수 있는데 파이썬은 이 라즈베리파이를 제어하는 도구로 사용된다. 예를 들어 라즈베리파이에 연결된 모터를 작동시키거나 램프에 불이 들어오게 하는 일들을 파이썬으로 할 수 있다.

3. 파이썬으로 작업불가능한 영역

가. 시스템과 밀접한 프로그래밍 영역

파이썬으로 도스나 리눅스 같은 운영체제, 엄청난 횟수의 반복과 연산을 필요로 하는 프로그램 또는 데이터 압축 알고리즘 개발 프로그램 등을 만드는 것은 어렵다. 즉, 대단히 빠른 속도를 요구하거나 하드웨어를 직접 건드려야 하는 프로그램에는 어울리지 않는다.

나. 모바일 프로그래밍

파이썬은 구글이 가장 많이 애용하는 언어이지만 파이썬으로 안드로이드 앱(App)을 개발하는 것은 아직 어렵다. 안드로이드에서 파이썬으로 만든 프로그램들이 실행되도록 지원하긴 하지만 이것만으로 앱을 만들기에는 아직 역부족이다. 아이폰 앱을 개발하는 것 역시 파이썬으로는 할 수 없다.

4. 파이썬 프로그램의 라이브러리(패키지, 모듈)

파이썬 프로그램 특징

- ✓ 인공지능관련 다양한 모듈 제공
- ✓ 공개소프트웨어(무료)
- ✓ 다양한 무료 모듈 (전세계인이 개발공유)
- ✓ 타언어에 비해 접근성과 사용성이 용이하다. (문법이 쉬움)

모듈: 자주 사용하는 코드나 유용한 코드를 논리적으로 묶어서 관리하고 사용할수 있도록 하는것임. 000.py가 하나의 모듈이 됨

- ✓ 수치해석-: NumPy, SciPy, SymPy
- ✓ 데이터탐색: Pandas, MDP, Orange
- ✓ 시계열/회귀분석: Statsmodels, Filterpy, Hmmlearn
- ✓ 분류인식: Scikit-Learn
- ✓ 고속계산: Teano, Tensorflow

- ✓ 베이지안 모형: PyMC3,
- ✓ 딥러닝: Keras, Lasagne, Blocks
- ✓ 영상신호처리: Pillow, Scikit-image
- ✓ 문서처리: NLTK, Gensim
- ✓ 음향신호처리: PyAudio-Analysis, LibRosa
- ✓ 확률적 그래프 모형: LibPGM, Pgmpy

모듈 / 라이브러리/ 패키지

구글을 검색해 봅니다.

[1] 파이썬 doc
[2] 파이썬 라이브러리 종류
[3] 파이썬 시각화 라이브러리
[4] 파이썬 이미지 처리 라이브러리
[5] 파이썬 데이터분석 라이브러리
[6] 파이썬 인공지능 라이브러리
[7] 파이썬 머신러닝 라이브러리

구글을 검색해 봅니다.

[1] 파이썬 matplotlib
[2] 파이썬 pillow
[3] 파이썬 pandas

구글을 검색해 봅니다.

[1] filetype:pdf matplotlib
[2] filetype:pdf numpy

구글을 검색해 봅니다.

[1] filetype:pdf matplotlib
[2] filetype:pdf numpy

5. 파이썬 프로그램 설치 및 실행

▶방법1: 파이썬 프로그램을 설치 (http://www.python.org/downloads)
▶방법2: 파이썬의 기본기능을 탑재한 콘다 프로그램을 이용하여 설치

5-1. 컴퓨터 환경 확인

. 파이썬 배포판으로서 파이썬 인터프리터, 표준 라이브러리 모듈, 디버거, 테스팅 도구, 통합 개발 환경(IDE), 사용자 설명서 등을 한 번에 설치할 수 있도록 묶어놓은 패키지를 말한다. ActivePython, CPython, Jython, WinPython 등이 있으나 가장 유명한 건 Anaconda이다.

Anaconda(아나콘다)는 일반적인 용도의 개발을 위한 파이썬 인터프리터 뿐만 아니라 Numpy, Pandas, Matplotlib 등 1500개가 넘는 데이터 분석 패키지를 함께 제공하기 때문에 인기가 많다. 패키지 관리도 conda라는 시스템을 통해 쉽게 할 수 있다.

그런데 정작 아나콘다를 설치하려고 하면 너무 무겁다. 함께 설치되는 패키지가 정말 너무 많기 때문에 실제로 설치하려면 하드 용량도 3GB 정도 잡아먹고 설치 시간도 오래 걸린다.

Miniconda(미니콘다)는아나콘다의 라이트 버전으로서기본적인 요구 사항만 포함하고 있어, 패키지를 그때그때 설치해서 사용하는 단점은 있지만 저용량으로 conda 시스템을 사용할수 있다.

5-2. 콘다 프로그램 설치

콘다 프로그램 다운로드 사이트에서 본인 컴퓨터 운영체제에 맞는 콘다 프로그램을 다운로드 받습니다. (Python 3.8 버전으로 받음)

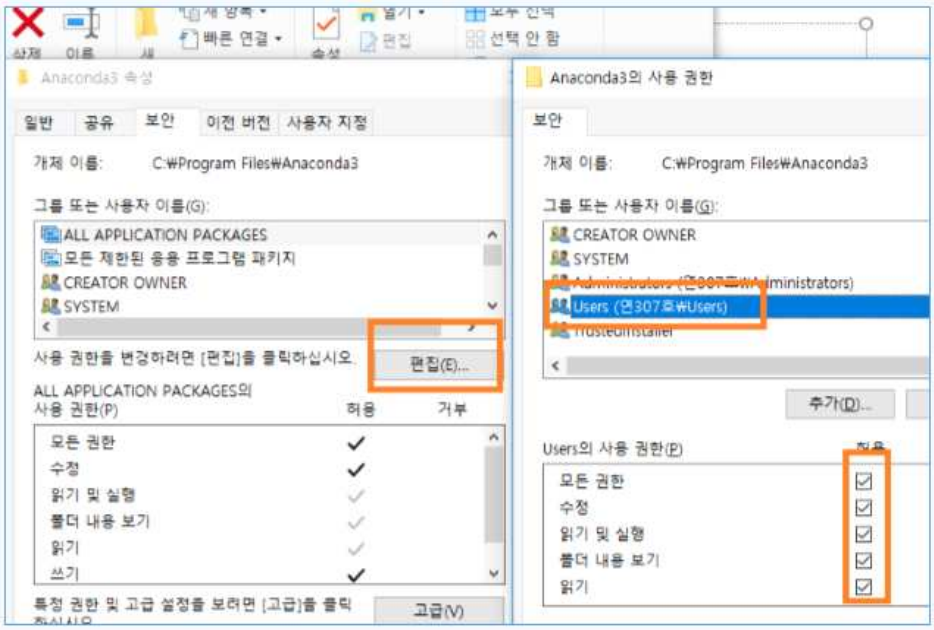
https://docs.conda.io/en/latest/miniconda.html

* 주의:
  본인 계정이 한글인경우에는 반드시 콘다프로그램 설치시 [All Uswrs]를 선택하여 설치

5-3. 라이브러리 설치 보안 옵션 확인

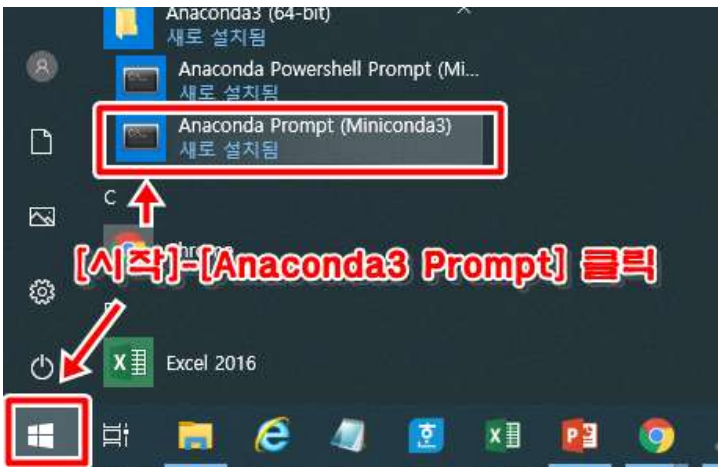
파이썬은 필요한 라이브러리(모듈, 패키지)를 다운받아서 사용하는 일이 대부분인 프로그램임으로 라이브러리가 설치될수 있도록 수정옵션을 활성화 해야 한다.
특히 c드라이브에 설치된 파이썬은 접근권한을 꼭 확인하여 보안옵션을 수정한다.

▪ 아나콘다 폴더권한 변경(C드라이브에 다운받은 경우만)



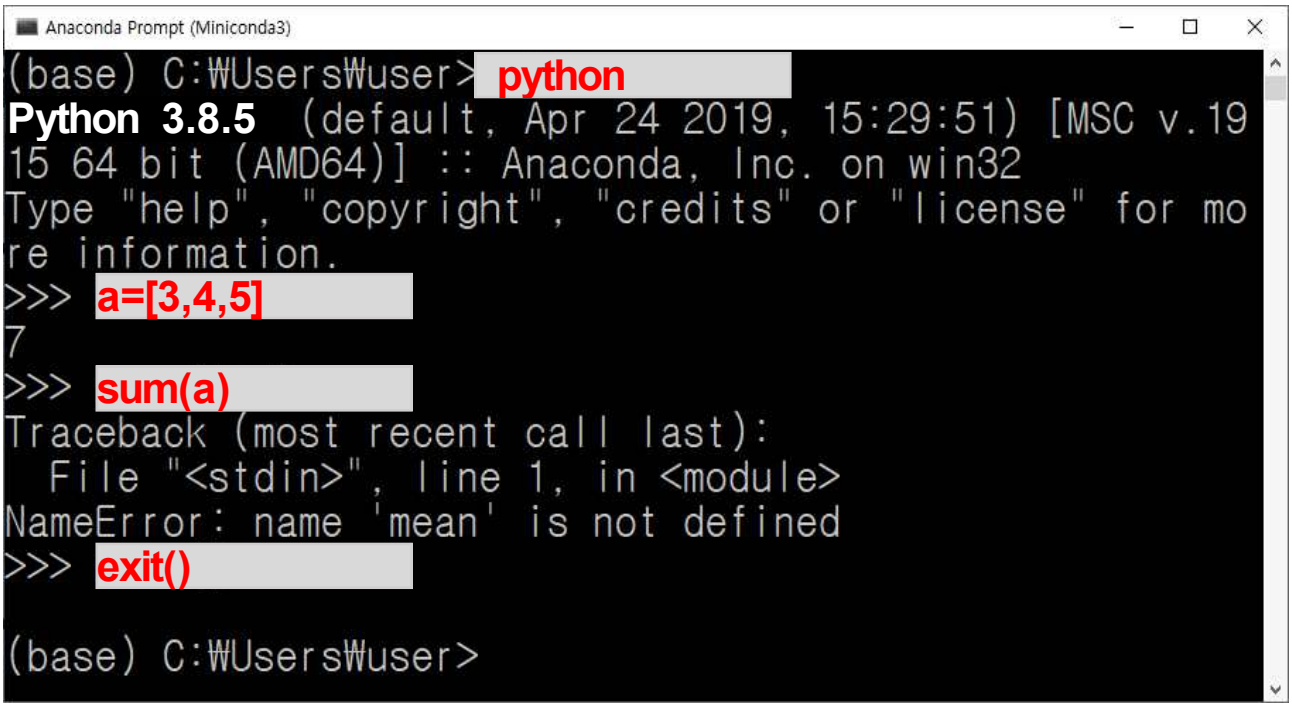
5-4. 파이썬 프로그램 실행

[시작]-> [anaconda prompt] 실행 한뒤 cmd 화면에서 python을 입력합니다.



실행된 파이썬 인터프리터 창에서 쉘표시 >>> 표시가 나올 때 명령어를 입력하여 결과를 확인합니다.

* 인터프리터언어: 명령어 입력후 실행시 대화형처럼 실행가능한 프로그래밍



* cmd 창에서 python을 실행하고 아래 명령어를 실행합니다.
파이썬에서 # 표시부터는 실행하지 않습니다. (주석문임)

```
>>> userList=[]
>>> userList.append(3)
>>> userList
>>> userList.append(5)
>>> userList
>>> userList.append('~파!이!썬')
>>> userList
>>> userList[2]
>>> tmp=userList[2]
>>> import re                    # 파이썬 내장함수
>>> re.sub('~','',tmp)
>>> re.sub('[^ㄱ-힣]','',tmp)
>>> tmp
```

파이썬내장함수) https://docs.python.org/ko/3/library/functions.html

5-5. 라이브러리(패키지) 설치

파이썬 쉘에서 아래와 같이 입력하고 Add(덧셈)을 실행해보면 리스트안에 있는 문자열의 연결이 실행됩니다. 리스트의 합,곱등 수치연산과 관련된 기본 통계를 실행하고자 할 때 내장함수에서 제공되지 않는 자료는 패키지 프로그램을 다운로드 하여 실행합니다.

```
>>> Val_1=[1,2]
>>> Val_2=[3,4]
>>> Val_1+Val_2
결과: [1,2,3,4]
>>> exit()
```

콘다 프롬프트에서 'coda install numpy'를 입력하여 실행합니다.
Y/N을 선택하는 입력창에서 모두 Y를 눌러 실행을 완료합니다.

```
(base) C:₩Users₩Administrator>conda install numpy
```

콘다 프롬프트 창에서 python을 실행합니다.

```
>>> Val_1=[1,2]
>>> Val_2=[3,4]
>>> import numpy
>>> numpy.add(Val_1,Val_2)
결과: array(4,6)
>>> import numpy as np
>>> np.add(Val_1,Val_2)
```

※ 모듈이 설치되지 않았을 때 표시되는 메세지

No module named 모듈이름

※ 라이브러리 제거는 콘다프롬프트 창에서 conda unistall로 제거가능함.
conda uninstall 라이브러리이름

5-6. 파이썬 가상환경 설정

외부에서 다운받는 라이브러리(패키지, 모듈)를 프로젝트에 따라 폴더(가상환경)을 만들어서 관리하면 추후 라이브러리 이동이 편리합니다.
아래와 같이 작업하면 작업환경(envs) 폴더 하위에 가상환경 폴더가 생성되며
가상환경 폴더 하위의 lib->site-packages 폴더에 패키지가 다운로드 됩니다.

C:\Users\Administrator**miniconda3\envs\test\Lib\site-packages**

A 콘다프로그램 업데이트

(base) C:₩Users₩user> **conda update conda**
Proceed ([y]/n)? y

B 가상환경 설정

가상환경: 작업별 공간임. 프로그램 다운로드(모듈) 및 저장등을 지정한 폴더에 모아놓는 공간임.

방1 이름:nhi	방2 이름: Test	방3 이름: Pro1	방마다 물건을 채워야함. (인터넷다운로드, 모듈) 그방에서만 사용할수 있음. 방은 폭파할수 있음.

(base) C:₩Users₩user> **conda create -n test**
→ 사용자 방이름

C 생성된 가상환경 확인

(base) C:₩Users₩user> **conda info --envs**

D 생성된 가상환경 삭제

(base) C:₩Users₩user> **conda remove --name test -all**

E 가상환경 활성화

(base) c:₩User₩user> **activate test**
→ 사용자 방이름

(test) C:₩Users₩user>
→ activate 방이름

F 가상환경 종료

(test) c:₩User₩user>**deactivate**
(base) c:₩User₩user>
→ 기본 작업환경 세팅

5-7. 필수 라이브러리 설치

콘다 cmd 환경의 설치된 가상환경 아래의 라이브러리를 설치합니다.
(test) > 라이브러리 설치하기

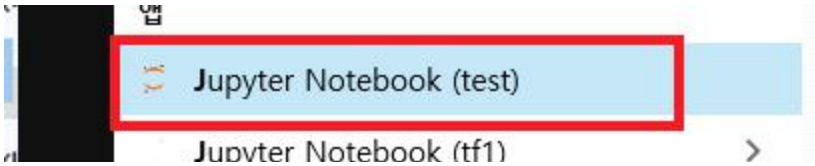
라이브러리	설명
conda instll jupyter	파이썬 에디터 창
conda instll numpy	자료를 행열구조로 바꾸어 주는 라이브러리
conda install matplotlib	기본차트
conda install seaboarn	분석에 특화된 차트
conda install PILLOW	이미지 처리

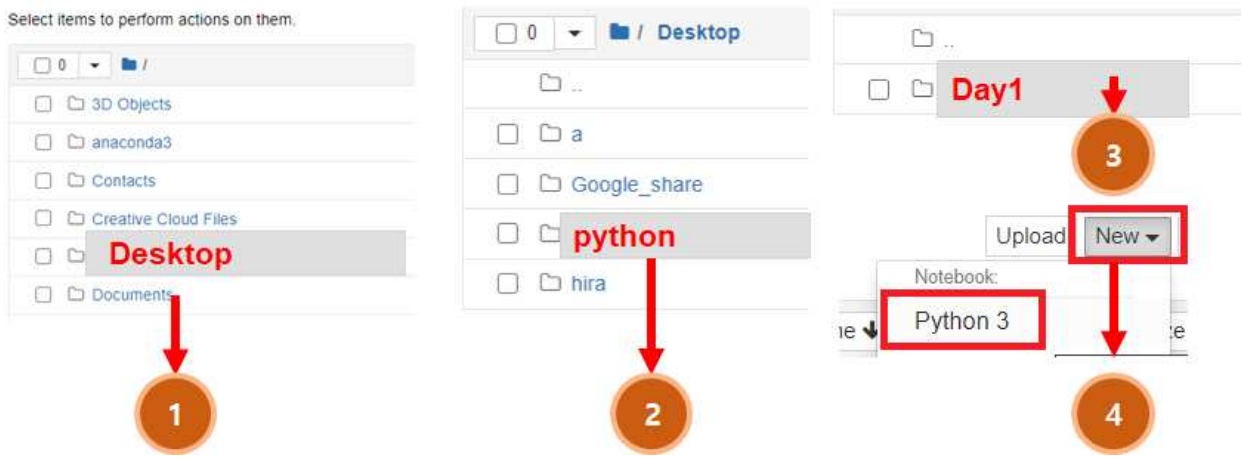
5-8. 주피터 노트북 에디터 창을 이용한 파이썬 실행

※ 준비: 크롬웹브라우저

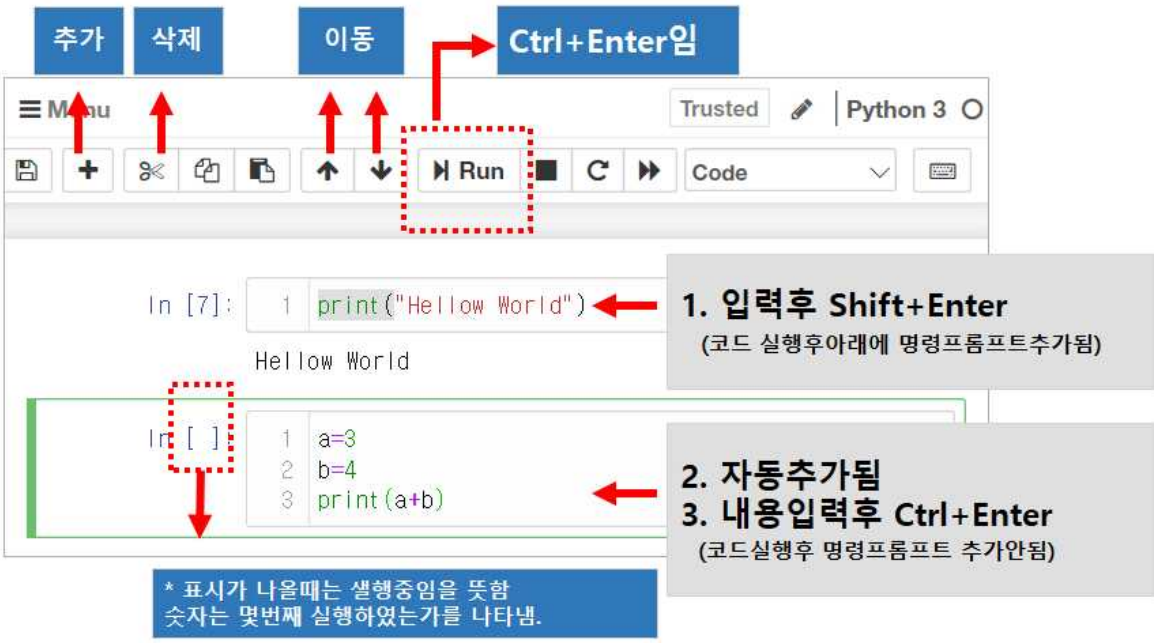
※ 바탕화면에 작업자료 python 폴더가 있는지 확인한후 작업을 시작합니다.

콘다 cmd를 종료한뒤 윈도우 시작키에서 Jupyter Notebook(test)를 클릭하여 실행합니다.



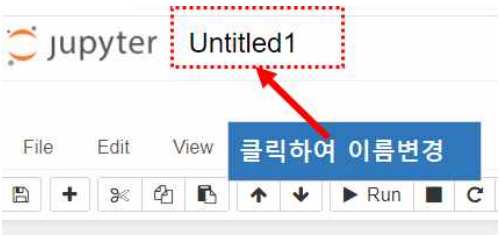


- **화면크기 조정**: Ctrl키 누르고 마우스 가운데 휠 올리고 내리고



※ 파일이름 변경

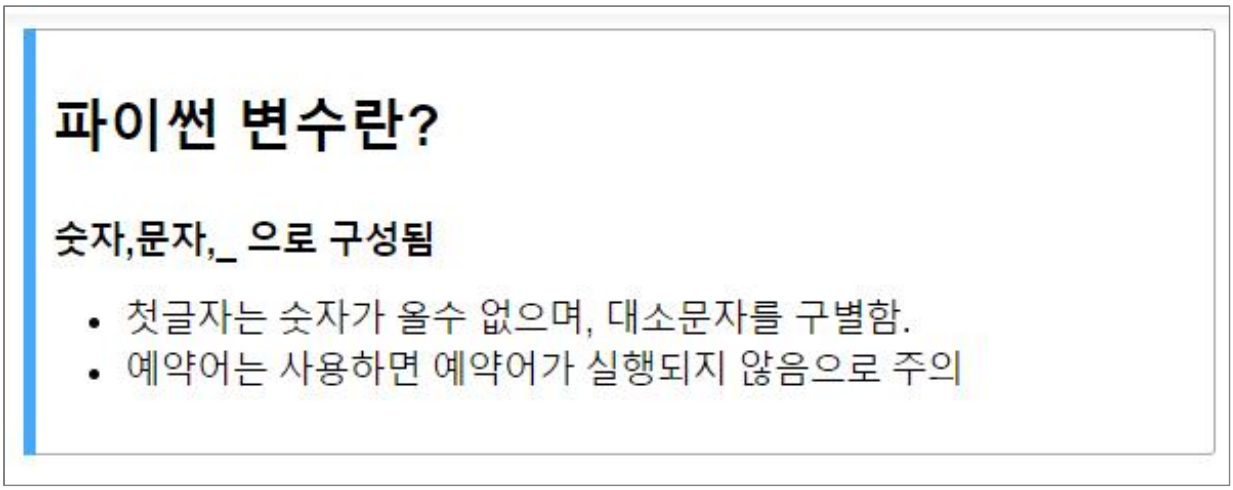
저장은 주피터노트북에서만 실행되는 ipynb 파일로 저장할수 있으며 실행시 사용자가 저장 버튼을 클릭하지 않아도 자동 저장됩니다.



※ 주의 : 주피터 노트북을 실행한후 작업표시줄에 있는 주피터노트북 커널을 종료하면 프로그램 실행한됨. 또한 주피터 노트북을 크롬에서 종료한뒤 반드시 작업표시줄의 커널프로그램도 종료해야함.

5-9. 주피터 노트북 마크다운

주피터 노트북에서 문서작성이 가능한 마크다운은 노트창을 선택하고 Markdown으로 설정한뒤 #과 -을 입력하여 문서를 작성함.
의 개수에 따라 글자크기가 지정되며 문서역할이 가능함. (#의 개수가 많아질수록 글자는 작아짐)
마크다운노트창에는 이미지도 붙여넣기 가능함.



제 목	02장 기본문법	
상세내용	변수와 자료형	

1. 변수

▶ 변수의 이름 작성 규칙

- 변수명에는 문자, 숫자, 언더바(_)를 포함할 수 있으며 숫자는 가장 처음에 등장할 수 없음.
- 파이썬에선 대소문자를 구분하기함.
- 파이썬 지정 단어(예약어)

A 파이썬 변수 특징

숫자, 문자형을 변수에 미리 설정해야 동작하는 일부 프로그램과 달리 파이썬은 실행 시점에 변수의 type(형)이 정해지기 때문에 동적 프로토타이핑 언어라고 말한다.

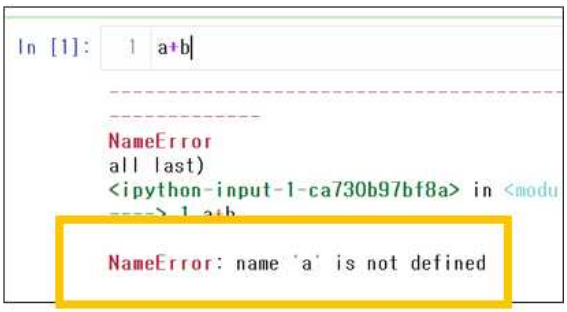
예: 비주얼베이직언어에서는
Dim Age as Byte
==> Age변수는 0~255까지 숫자만 입력가능한 변수임을 미리 설정(선언)함
==> Age="20세" 로 입력하면 에러발생
==> 프로그램 개발후 변수의 성격은 변경못할수 있음.
초기개발단계에서 잘 확립해야함.

B 변수명작성규칙

- 영문자(대, 소문자 구분), 숫자, 언더바(_)를 사용할 수 있다.
- 첫 자리에는 숫자를 사용할 수 없다.
- 파이썬 키워드는 변수 명으로 사용할 수 없다.

잘못된 변수명: A-3, A 3, 3A, A.3, if

C 같은 변수명 사용시 마지막에 값이 기록됨



```
In [2]: a-1=3
  File "<ipython-input-2-37d7798e4cb8>", line 1
    a-1=3
       ^
SyntaxError: can't assign to operator
```

```
In [3]: a.1=3
  File "<ipython-input-3-681ed9be2b01>", line 1
    a.1=3
      ^
SyntaxError: invalid syntax
```

```
In [4]: 1a=3
  File "<ipython-input-4-249f51bc5abe>", line 1
    1a=3
    ^
SyntaxError: invalid syntax
```

```
a b=3
  File "<ipython-input-2-2d3d3c0c8ec9>", line 1
    a b=3
      ^
SyntaxError: invalid syntax
```

예약되어 있는 단어를 변수로 사용하면 안됨, 예약되어 있는 단어는 초록색으로 표시됨
아래의 내용을 입력하고 shift+Enter로 노트창을 추가하면서 실행해봅니다.

```
print(3)
```
```
3
```

→ print는 화면에 값을 뿌려주는 출력구문

```
print=4
```

```
print(3)
```

→ 변수=값 일때 왼쪽의 변수명이 초록색으로 나오면 안됨

```
TypeError                                 Traceback (most recent call last)
<ipython-input-5-ce36b08be018> in <module>
----> 1 print(3)

TypeError: 'int' object is not callable
```

→ 예약어 print를 변수로 사용해서 생기는 에러

```
print
```
```
4
```

→ print하면 변수로 인지해서 print에 할당된 4가 출력됨

```
del print
```

→ 변수를 삭제하고 다시 작업함.

```
print
```
```
<function print>
```

```
print(3)
```
```
3
```

※ 사용자변수로 사용할수 없는 예약어는 아래의 명령으로 확인할수 있음.

```
import keyword
keyword.kwlist
```

파이썬 변수는 대소문자를 구별하며, Enter의 행 띄어쓰기는 사용자 마음이나
들여쓰기(스페이스칸)는 주의하여야함.

[메모]

2. 출력문

```
>>>print(1+2)
▶결과: 3

>>> print('1+2')
▶결과: 1+2

>>> print('안녕'+'하세요')
▶결과: 안녕하세요

>>> print('안녕'+1+'하세요')
▶결과: Error

>>> print('안녕' + str(1) + '하세요')
▶결과: 안녕1하세요

>>> tmp='프로그래밍'파이썬'
▶ 결과: Error
  File "<ipython-input-6-fae761b56141>", line 1
    tmp=' 프로그래밍'파이썬'
         ^
SyntaxError: invalid character in identifier

>>> tmp="'프로그래밍'파이썬'"
>>> print(tmp)
▶결과: 프로그래밍'파이썬

>>> tmp1='파이썬' ; tmp2=2 ; tmp3=3.57
>>> print('%s과정 %d일차 자료 %f개'%(tmp1,tmp2,tmp3))
▶결과: 2일차 자료 3일차 자료
```

※ 구글검색 '파이썬 출력문'

※ 달력 내장함수로 print와 변수를 실습하여 봅니다.

```
# 파이썬에서 별도의 conda install 설치없이 제공하는 calender 모듈 사용하기
# 내장함수라고 함.

import calendar as cal
year=2021
c = cal.calendar(year)
print(str(year) + '해 달력')
print(c)
```

▶결과

```
2021해 달력
                                  2021

      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th F
r Sa Su
                1  2  3       1  2  3  4  5  6  7       1  2  3  4
5  6  7
 4  5  6  7  8  9 10       8  9 10 11 12 13 14       8  9 10 11 1
2 13 14
11 12 13 14 15 16 17      15 16 17 18 19 20 21      15 16 17 18 1
9 20 21
18 19 20 21 22 23 24      22 23 24 25 26 27 28      22 23 24 25 2
6 27 28
25 26 27 28 29 30 31                                29 30 31
```

```
#calender 모듈 사용하기

import calendar as cal

print(" 달력 확인 ")

year =2021
month =8
print(cal.month(year, month))
```

```
#calender 모듈 사용하기

import calendar as cal

print(" 달력 확인 ")

year =int(input(" 태어난 년도 : "))
month =int(input(" 태어난 월 : "))
print(cal.month(year, month))
```

3. 자료형과 수식계산

3-1. 산술연산자의 종류

연산자	예시	기능
+	a + b	a와 b를 서로 더한다
-	a - b	a와 b를 서로 뺀다
*	a * b	a와 b를 서로 곱한다
/	a / b	a를 b로 나눈다
//	a // b	a를 b로 나눈 몫을 가져온다
%	a % b	a를 b로 나눈 나머지를 가져온다
**	a ** b	a의 b 제곱

3-2. 숫자 자료형

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8진수	0o34, 0o25
16진수	0x2A, 0xFF

※ 참고: 파이썬에서 %는 나머지값을 구하는 연산자로만 사용됨.

```
근무시간_분=200
시간=int(200/60)
분=200%60
print('근무시간은 %d 시간 %d 분입니다' %(시간,분))
```

▶결과: 근무시간은 3 시간 20 분입니다

```
In [17]: a=3.3.4
  File "<ipython-input-17-d2ccd1f44a70>", line 1
    a=3.3.4
        ^
SyntaxError: invalid syntax

In [18]: b=3,200
         b
Out[18]: (3, 200)

In [19]: a=+3+3
         a
Out[19]: 6

In [21]: a=*3+3
  File "<ipython-input-21-262add8bcfad>", line 4
SyntaxError: can't use starred expression here

In [22]: a=3*20%
  File "<ipython-input-22-a31649299b09>", line 1
    a=3*20%
        ^
SyntaxError: invalid syntax
```

숫자형의 정수형과 실수형의 연산속도와 결과를 확인해봅니다.

```
1 a=3
2 b=2.
3 a+b
```

```
5.0
```

```
1 import time
2 startTime=time.time()
3 print(3+2)
4 endTime=time.time()-startTime
5 print(endTime)
```

time은 내장모듈로서 별도로 설치작업 필요없음.

```
5
0.0
```

```
1 import time
2 startTime=time.time()
3 print(3+2.)
4 endTime=time.time()-startTime
5 print(endTime)
```

```
5.0
0.0009932518005371094
```

```
1 ver=3.7
2 print("파이썬 버전은 %d 입니다" % ver)
```

```
파이썬 버전은 3 입니다
```

```
1 ver=3.7
2 print("파이썬 버전은 %.2f 입니다" % ver)
```

```
파이썬 버전은 3.70 입니다
```

참고) 숫자자료형

■ **정수형(Integer)**

- int 라는 녀석은 정수(Integer)의 약자
- 10진수, 2진수, 8진수, 16진수

```
>>> A = 5
>>> B = 10
>>> print(A, B)
5 10

>>> print(0b10, 0o10, 0x10)
2 8 16

>>> bin(202020)
'0b110001010100100100'
>>> oct(20114)
'0o47222'
>>> hex(401000)
'0x61e68'
```

■ **실수형(Float)**

- float은 부동 소수점(Floating point)의 약자
- 부동 소수점이란 정수처럼 소숫점이 고정

```
>>> A = 3.14
>>> B = 5.1402020319
>>> print(A, B)
3.14 5.1402020319

#지수형
>>> 0.23193202032E8
23193202.032
>>> 1412232.22E-10
0.000141223222

#진법 변환에 따른 오차
>>> 5.4 + 1.2
6.6000000000000005
>>> 1.4 - 1.1
0.2999999999999998
```

참고) 숫자자료형

■ **복소수(Complex)**

- 복소수란 실수와 허수가 합해져 이루어지는 수
- 단위가 i가 아니라 j 또는 J

```
>>> x = 7 - 3j
>>> type(x)
<class 'complex'>
>>> x.imag
-3.0
>>> x.real
7.0
>>> x.conjugate()
(7+3j)
```

- 위 예제에서 imag는 복소수의 허수 부분을 돌려주고, real은 복소수의 실수 부분을 되돌려줍니다.
- conjugate 함수는 복소수의 켤레 복소수를 되돌려줍니다.

참고) 자료타입확인

```
>>> type(33)
<class 'int'>

>>> type(3.14)
<class 'float'>

>>> type(['1', '2', '3'])
<class 'list'>

>>> type(33331491491492)
<class 'int'>
```

3-3. 문자 자료형

문자열(String)
문자열은 문자들의 모임, 문자를 나열한 것을 문자열이라고 함.

▶ 이스케이프 문자: 별도의 기능을 제공하는 특수한 문자

이스케이프 문자	설명
₩n	개행(Newline, 줄바꿈)
₩t	탭(Tab)
₩0	NULL 문자
₩₩	문자 '₩'
₩'	단일 인용부호(')
₩"	이중 인용부호(")

```
>>> print('AAA\nB\tCDE')
>>> print('C:\\ABC.txt')
>>> print('A\'B')
>>> print('A\"B')
```

▶ + 연산자는 문자열을 서로 연결하는 기능
숫자와 문자는 숫자에 str함수로 문자로 변환하여 결합한다.(+)
숫자 1->001, 2를 ->002, 3을 ->003으로 출력하려면 zfill명령을 사용한다.

```
>>> 'ABC' + 'DEFGHI'
>>> 'Python!' * 3
>>> 'A'+str(3)
>>> 'A'+str(3).zfill(5)
```

▶ 문자열 관련 내장함수(메서드)

- 대문자로 변환하는 upper, 소문자로 변환하는 lower

```
>>> s = 'abCdeFg'
>>> s.upper()
'ABCDEFG'
>>> s.lower()
'abcdefg'
```

- 문자열 위치를 찾아내는 find

```
>>> s = 'It`s always such a pleasure'
>>> s.find('way')
7
>>> s.find('such', 13)
-1
>>> s.find('the')
-1
```

- 부분 문자열의 발생 횟수를 알려주는 count

```
>>> s = 'we are invincible, we are unique'
>>> s.count('we')
2
>>> s.count('in')
2
```

- 새로운 문자열로 교체하는 replace

```
>>> s = 'AB-C-D-EFGH'
>>> s.replace('-', '')
'ABCDEFGH'
```

- 좌우 공백을 제거하는 strip

```
>>> s = '       1234 56    '
>>> s.strip()
'1234 56'
```

- 문자열을 분리하는 split

```
>>> s = '1/2/3/4/5/6'
>>> s.split('/')
['1', '2', '3', '4', '5', '6']
>>> lst = s.split('/')
>>> lst[3]
'4'
```

- 문자열을 결합하는 join

```
>>> lst = ['1', '2', '3', '4', '5', '6']
>>> sep = "-"
>>> sep.join(lst)
'1-2-3-4-5-6'
```

[참고] format함수

```
>>> weight = 62.53
>>> print("몸무게:", format(weight, '.1f'))
>>> print("돈:", format(12931401, ',d'))

>>> '{} {}'.format('홍길동', 34)
'홍길동 34'

>>> '{0} {1} {2}'.format(12, 34, 56)
'12 34 56'
>>> '{2} {2} {1} {0}'.format(12, 34, 56)
'56 56 34 12'

>>> '{} / {} = {:.2f}'.format(5, 2, 5 / 2)
'5 / 2 = 2.50'
>>> '{0} / {1} = {2:.4f}'.format(13, 3, 13 / 3)
'13 / 3 = 4.3333'

>>> lst = [30, 40, 50, 80, 90, 100]
>>> 'lst[4] = {0[4]}'.format(lst)
'lst[4] = 90'

>>> '제 나이는 {age}살이고, 제 몸무게는 {weight} kg 입니다.'.format(age = 19,
weight = 72.5)
'제 나이는 19살이고, 제 몸무게는 72.5 kg 입니다.'

>>> '{0:6s}'.format('cat')
'cat    '
>>> '{0:5d}'.format(334)
'  334'

>>> '{0:<6d}'.format(1234)
'1234   '
>>> '{0:>6d}'.format(1234)
'  1234'

>>> '{0:07d}'.format(1234)
'0001234'

>>> '{0:#o} {0:#x}'.format(123)
'0o173 0x7b'
```

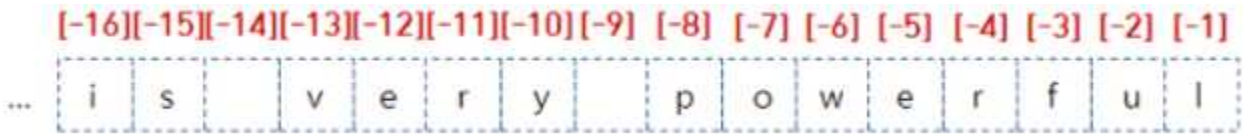
4. 자료 인덱싱과 슬라이싱

4-1. 자료 인덱싱

- 문자열에서 원하는 위치에 있는 문자를 마음대로 꺼낼 수 있다
- 슬라이싱 번호가 양수일때는 시작위치부터 찾음


```
>>> var = "python is very powerful"
>>> var[4]
>>> var[11]
>>> var[0]
>>> var[1]
```

- 슬라이싱 번호가 음수일때는 마지막 위치부터 찾음

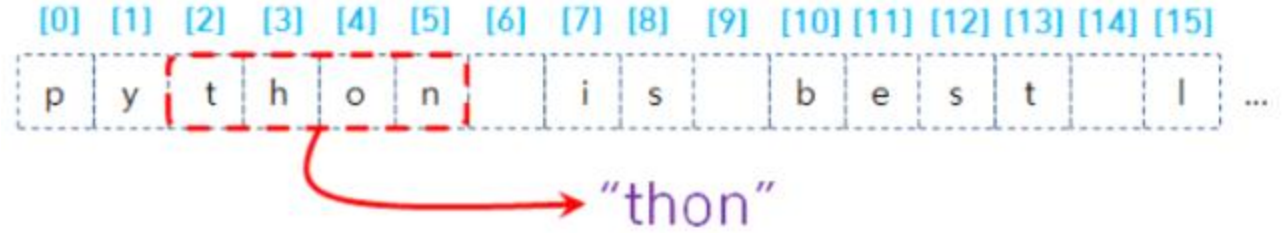



```
>>> var = "python is very powerful"
>>> var[-2]
>>> var[-10]
>>> var[-16]
```

```
>>> var = "ABCDE"
>>> var[2] = 'G'
Traceback (most recent call last):
  File "<pyshell#66>", line 1, in <module>
    var[2] = 'G'
TypeError: 'str' object does not support item assignment
```

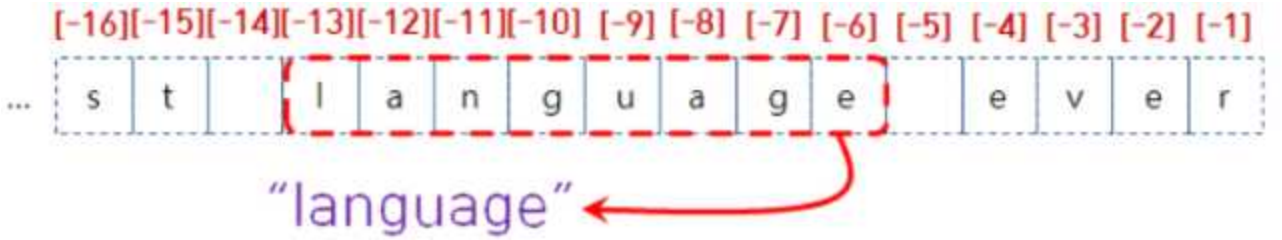
4-2. 슬라이싱(Slicing)

- 문자열의 시작과 끝지점을 지정하여 2개이상의 문자를 처리할수 있다.
- 슬라이싱 번호가 양수일때는 시작위치부터 찾음



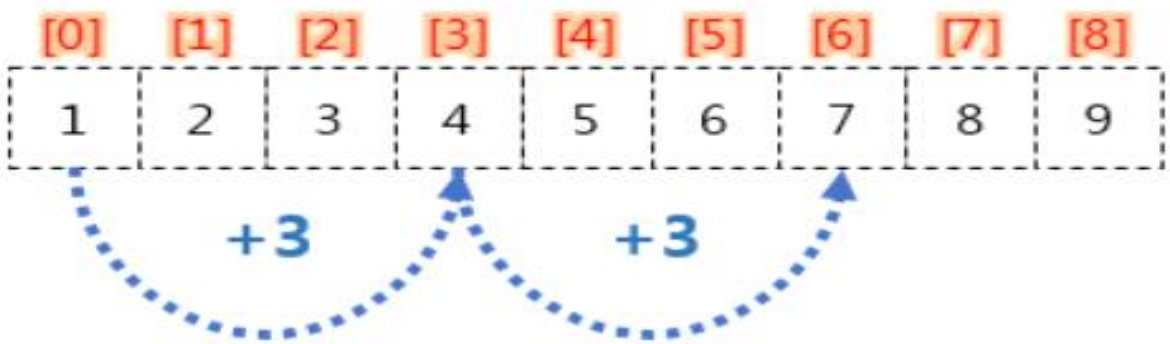

```
>>> var = "python is best language ever"
>>> var[2:6]
>>> var[3:7]
>>> var[0:15]
```

- 슬라이싱 번호가 음수일때는 마지막 위치부터 찾음




```
>>> var = "python is best language ever"
>>> var[-13:-5]
>>> var[-15:-6]
```

```
>>> var = "0123456789"
>>> var[2:]
'23456789'

>>> var[:5]
'01234'

>>> var[:]
'0123456789'

>>> temp = "123456789"

>>> temp[::3]    # 0번위치, +3 해서 3번위치
'147'

>>> temp[::-1]
'987654321'

>>> temp[::-2]
'97531'
```



제 목	03장 이미지처리를 이용한 기본문법 정리	
상세내용 1)	PILLOW 모듈	

```
1  from PIL import Image
2  # 이미지 열기
3  tmp = Image.open('./picture/pic1.jpg')
4
5  # 이미지 크기 출력
6  print(tmp.size)
7
```

```
(512, 271)
```

```
1  from PIL import Image
2  # 이미지 열기
3  fileName='pic1'
4  tmp = Image.open('./picture/' + fileName + '.jpg')
5
6  # 이미지 크기 출력
7  print(tmp.size)
```

```
(512, 271)
```

[메모]

```
fileName='pic1'
tmp=Image.open('./picture/' + fileName + '.jpg')

tmpResize=tmp.resize((200,200))
tmpResize.save('./picture/200' + fileName + '.jpg')

print('---end ---')
```

```
---end ---
```

```
fileName='pic1'
tmp=Image.open('./picture/' + fileName + '.jpg')

size=300
tmpResize=tmp.resize((size,size))
tmpResize.save('./picture/' + str(size) + fileName + '.jpg')
print('---end ---')
```

```
---end ---
```

[메모]

[미션1]

pic1, pic2, pic3를 읽어서 이미지를 90도로 회전하고
rot90pic1.jpg / rot90pic2.jpg / rot90pic3.jpg 로 저장합니다.

[미션2]

PIL함수(PILLOW)의 명령을 이용하여 pic1 / pic2 /pic3의 가로,세로 길이를 출력하여 봅니다

제 목	04장 조건문과 반복문	
상세내용 1)	조건문 if, 반복문 for	

1. 조건문(Condition Statements)

조건의 결과에 따라서 실행되는 문장이 결정

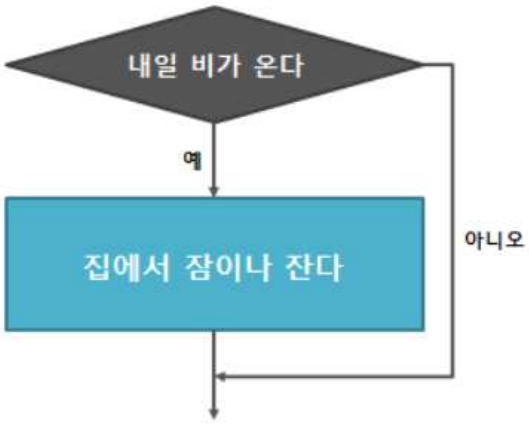
파이썬에서는 if a in b / if a not in b 구문도 있음.

1-1. 조건문 형식

- if문을 사용하면 조건식이 참(True)이냐 거짓(False)이냐에 따라 실행되는 문장을 다르게 만들 수 있습니다.

```
if 조건식:
    문장
```

```
if 내일 비가 온다:
    집에서 잠이나 잔다
```



〈순서도(Flow Chart)를 통한 예〉

```
>>> money = 1000
>>> if money >= 500:
    print("돈이 500원 이상 있습니다.")
돈이 500원 이상 있습니다.
```

1-2. else문

- if~else문의 구조

```
if 조건식:
    문장1
else:
    문장2
```

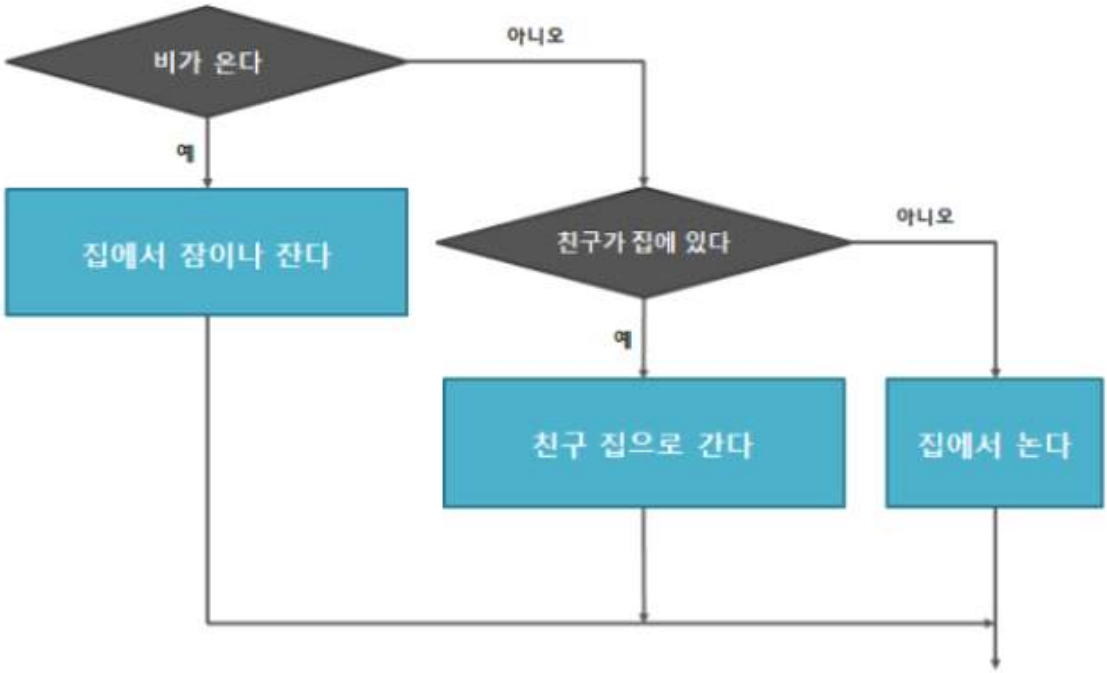
〈순서도(Flow Chart)를 통한 예〉

```
>>> money = 430
>>> if money >= 500:
    print("돈이 500원 이상 있습니다.")
else:
    print("돈이 500원 미만입니다.")
돈이 500원 미만입니다.
```

1-3. elif문

- if문과 else문 사이에 여러번 사용

```
if 조건식:
    문장
elif 조건식:
    문장
else:
    문장
```



〈순서도(Flow Chart)를 통한 예〉

```
>>> money = 500
>>> if money > 500:
        print("가지고 있는 돈이 500원보다 많습니다.")
    elif money == 500:
        print("가지고 있는 돈이 500원입니다.")
    else:
        print("가지고 있는 돈이 500원보다 적습니다.")
```

▶결과: 가지고 있는 돈이 500원입니다.

2. 반복문(Loop)

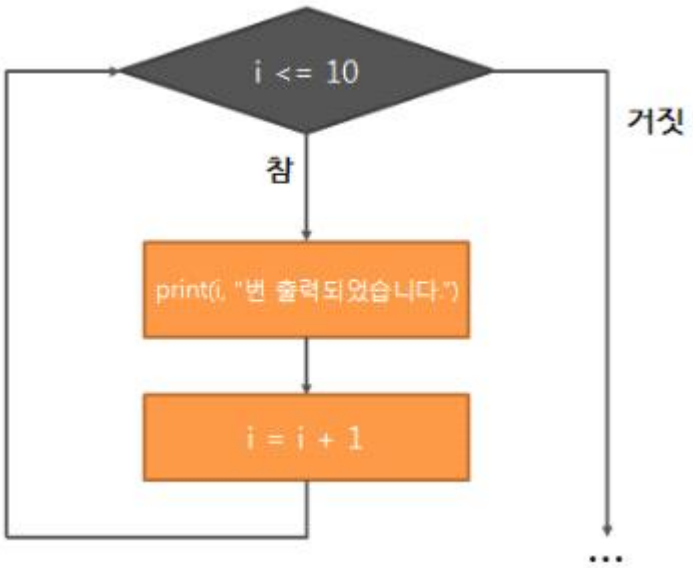
동일 작업을 반복 수행

2-1. while문

- i조건식이 참이라면 while문 내부에 있는 블록의 문장을 실행하고, 거짓이면 반복을 멈추고 빠져나옵니다.

```
while 조건식:
    문장
```

```
>>> i = 1
>>> while i <= 5:
    print(i, "번 출력되었습니다.")
    i += 1

1 번 출력되었습니다.
2 번 출력되었습니다.
3 번 출력되었습니다.
4 번 출력되었습니다.
5 번 출력되었습니다.
```



2-2. break문

- break문을 만나 바로 반복문을 빠져나오게 됩니다.

```
>>> i = 1
>>> while i <= 10:
    if i == 7:
        break
    print(i, "번 출력되었습니다.")
    i += 1

1 번 출력되었습니다.
2 번 출력되었습니다.
3 번 출력되었습니다.
4 번 출력되었습니다.
5 번 출력되었습니다.
6 번 출력되었습니다.
```

2-3. continue문

- continue문 아래의 문장은 실행하지 않고 반복문의 시작 지점으로 되돌아갑니다.

```
>>> i = 0
>>> while i <= 10:
    i += 1
    if i % 2 == 1: continue
    print(i, "번 출력되었습니다.")

2 번 출력되었습니다.
4 번 출력되었습니다.
6 번 출력되었습니다.
8 번 출력되었습니다.
10 번 출력되었습니다.
```

3. for문

3-1. 변수의 자료 할당

```
for 변수 in 순서형 자료:
    문장
```

```
>>> lst = [1, 3, 5, 7, 9]
>>> for i in lst:
    print(i)

1
3
5
7
9
```

- 리스트 lst의 요소를 하나하나씩 가져와 변수 i에 넣어서 i의 값을 출력하고 있음

```
>>> lst = [1, 2, 3, 4, 5, 6, 7, 8]
>>> for i in lst:
    if i % 2 == 1: continue
    if i == 6: break
    print(i, "번 출력되었습니다.")

2 번 출력되었습니다.
4 번 출력되었습니다.
```

- i를 2로 나눈 뒤의 나머지가 1인 경우에 시작 지점으로 돌아가고 i가 6과 같으면 반복문을 탈출하게끔 작성

```
>>> 좋아하는음식리스트=[]
>>> for i in range(10):
        음식은=input('좋아하는 음식?')
        좋아하는음식리스트.append(음식은)
>>> print(좋아하는음식리스트)
```

[여러장의 이미지를 for로 높이와 너비를 읽어서 csv로 저장합니다]

```
from PIL import Image
dic=[]
fileName=['pic1','pic2','pic3']
for i in fileName:
    im=Image.open('./picture/' + i + '.jpg')
    dic.append({'파일명':im.filename,'사진높이':im.height,'사진너비':im.width})

import pandas as pd
df=pd.DataFrame(dic,columns=['파일명','사진높이','사진너비'])
df
df.to_csv('사진자료정리.csv')
```

3-2. range 함수

- for문에 이용, 숫자의 범위를 가지는 range 객체를 돌려보내며 이를 통하여 for문 내에서 위치를 가지고 요소를 가져오거나 변경할 수 있습니다.

```
>>> for i in range(0,10,2):
    print(i, end=" ")

0 1 2 3 4 5 6 7 8 9
```

```
>>> lst = [1, 2, 3, 4, 5]
>>> for i in range(len(lst)):
    if i % 2 == 0:
        lst[i] *= 2
    else:
        lst[i] *= -2
    print(lst[i], end=" ")

2 -4 6 -8 10
```

[메모]

3-3. 파이썬 Comprehension

iterable한 오브젝트를 생성하기 위한 방법중 하나로 파이썬에서 사용할 수 있는 유용한 기능중 하나 List Comprehension (LC)를 확인하여 봅니다.

```
#### List comprehension
#### 0~10까지의 숫자를 리스트에 넣기
tmp = [i   for i in range(11)]
print(tmp)

## 위의 LC 기능을 사용하지 않으면 아래와 같이 작업해야함.
#  tmp=[]
#  for x in range(11):
#      tmp.append(x)
#  print(tmp)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
#### List comprehension
#### 0~10까지의 숫자중 짝수값(x%2 는 2로 나누었을때 나머지값 구하는 계산식) 만 리스트에 넣기
tmp = [i for i in range(11) if i%2==0 ]
print(tmp)

# 위의 LC 기능을 사용하지 않으면 아래와 같이 작업해야함.
#tmp=[]
#for x in range(11):
#      if x%2==0:
#            tmp.append(x)
#print(tmp)
```

```
a=['리스트1 나눌자료','이 자료를 빈칸 단위로 나누어 보는 작업을 합니다.']
tmp=[i.split(' ') for i in a]
tmp

## 위의 기능을 사용하지 않으면 아래와 같이 작업해야함
#tmp=[]
#for i in a:
#    tmp.append(i.split(' '))
#print(tmp)
```

```
[['리스트1', '나눌자료'], ['이', '자료를', '빈칸', '단위로', '나누어', '보는', '작업을', '합니다.']]
```

```
### 2차원리스트 for
##
자료=[ ['리스트1 나눌자료','이 자료를'],['빈칸 단위로','나누어 보는 작업을 합니다.']]

tmp1=[[문장단위로나눔.split(' ')  for 문장단위로나눔 in 리스트단위로나눔] for 리스트단위로나눔 in 자료  ]
tmp1


# 위의 기능을 사용하지 않으면 아래와 같이 작업해야함.
#tmp1=[]
#for 리스트단위로나눔 in 자료:
#    for 문장단위로나눔 in 리스트단위로나눔:
#        tmp1.append(문장단위로나눔.split(' '))

#tmp1
```

3-5. 반복문의 중첩

- 여러 개의 반복문이 중첩되어 사용될 수 있습니다.

```
>>> while i <= 9:
    j = 1
    while j <= 9:
        print(i, " * ", j, " = ", i * j)
        j += 1
    i += 1

2  *  1  =  2
2  *  2  =  4
…
9  *  8  =  72
9  *  9  =  81
```

```
>>> for i in range(2, 10):
    for j in range(1, 10):
        print(i, " * ", j, " = ", i * j)

2  *  1  =  2
2  *  2  =  4
…
9  *  8  =  72
9  *  9  =  81
```

** 제어문과 반복문을 이용한 이미지 로고 합성
강아지 사진위에는 ‘dog.png’ / 고양이 사진위에는 ‘cat.png’를 추가합니다.

```
import os
path_dir='./images'
fileList=os.listdir(path_dir)
fileList

dog=Image.open('dog.png')
area_dog=(0,0,dog.width,dog.height)

cat=Image.open('cat.png')
area_cat=(0,0,cat.width,cat.height)


for i in fileList:
    gubun=i[:3]
    im=Image.open('./images/' + i)

    if gubun=='강아지':
        im.paste(dog,area_dog)

    else:
        im.paste(cat,area_cat)

    im.save('./img_logo/logo_'+ i)
```

dog와 cat의 정보를 리스트에 넣어서 관리

```
:
1
2  dog=Image.open('dog.png')
3  cat=Image.open('cat.png')
4
5  area=[[0,0,dog.width,dog.height],[0,0,cat.width,cat.height]]
6  print(area[0])
7  print(area[1])
```

```
[0, 0, 222, 105]
[0, 0, 334, 141]
```

```
:
1  for i in fileList:
2      gubun=i[:3]
3      im=Image.open('./images/' + i)
4
5      if gubun=='강아지':
6          im.paste(dog,area[0])
7
8      else:
9          im.paste(cat,area[1])
10
11     im.save('./img_logo/logo_'+ i)
```

제 목	05장 리스트 구조의 이해	
상세내용	단일, 이중 리스트	

1. 리스트

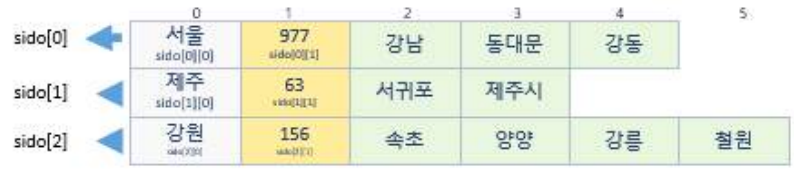
구글검색 '파이썬 리스트' 로 검색하여 아래에 정의를 작성합니다.

아래와 같이 sido라는 저장장소에 시도이름,인구수,지역명을 입력할 때 리스트구조를 사용한다면

sido

```
sido=[
    ['서울',977,'강남','동대문','강동'],
    ['제주',63,'서귀포','제주시'],
    ['강원',156,'속초','양양','강릉','철원']
]
```

	0	1	2	3	4	5
sido[0]	서울 sido[0][0]	977 sido[0][1]	강남	동대문	강동	
sido[1]	제주 sido[1][0]	63 sido[1][1]	서귀포	제주시		
sido[2]	강원 sido[2][0]	156 sido[2][1]	속초	양양	강릉	철원



▶파이썬에서 리스트 인덱싱에서 [:] 이 있을때는 예) [:3] 은 0에서 (3-1) 까지 즉 0,1,2/ [1:5] 는 1에서 5-1까지 즉 1,2,3,4

```
1 print(sido[0][1:5])  #서울의 인구수및 지역출력
2
3 # 제주의 인구수와 지역을 출력하시오
4 코드작성
5 # 강원의 인구수와 지역을 출력하시오.
6 코드작성
7
```

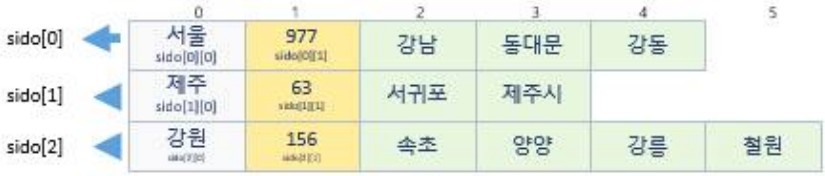
```
[977, '강남', '동대문', '강동']
```

▶구글 검색 '파이썬 리스트 개수'

```
1 print(sido[0][1: 코드작성 ])
2 print(sido[1][1: 코드작성 ])
3 print(sido[2][1: 코드작성 ])
```

```
[977, '강남', '동대문', '강동']
[63, '서귀포', '제주시']
[156, '속초', '양양', '강릉', '철원']
```

[메모]



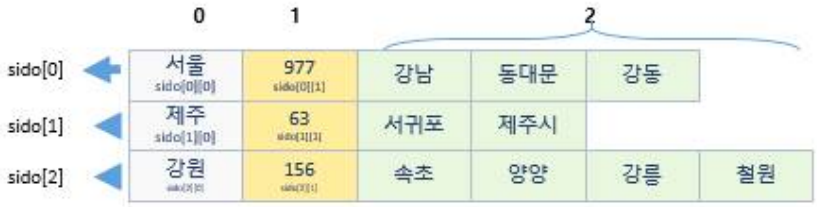

```
serachJi='강원'
for i in range(3):
   cnt=len(sido[i])

   if serachJi==sido[i][0]:
      print(sido[i][1:cnt])
```

```
[156, '속초', '양양', '강릉', '철원']
```

```
1 serachJi='강원'
2 for i in range(3):
3     cnt=len(sido[i])
4     if serachJi==sido[i][0]:
5         print('검색지역은 ->' + sido[i][0])
6         print('인구수는 ->' + str(sido[i][1]))
7         print('지역은->' + str(sido[i][2:cnt]))
8
9
```

```
검색지역은 ->강원
인구수는 ->156
지역은->['속초', '양양', '강릉', '철원']
```

[메모]



리스트 입력시 시도, 인구수, 지역에서 지역을 또 한개의 리스트로 만든다면

```
sido=[
    ['서울',977,['강남','동대문','강동']],
    ['제주',63,['서귀포','제주시']],
    ['강원',156,['속초','양양','강릉','철원']]
]

print(sido[0][2])
print(sido[1][2])
print(sido[2][2])
```

```
['강남', '동대문', '강동']
['서귀포', '제주시']
['속초', '양양', '강릉', '철원']
```

```
serachJi='강원'
for i in range(3):
    cnt=len(sido[i])

    if serachJi==sido[i][0]:
        print(sido[i][1:cnt])
```

코드변경해봅니다.

[메모]

참고)

```
###################
# 문자열 자료
###################
tmp='a'
list_='eake'
print('문자열의길이는 %d 입니다.'%len(list_))
#print('문자열의길이는 ' + str(len(list_)) + ' 입니다')
print(list_[1])   #1번째위치,

# 이자료에서는 e는 [0], a는[1]

if tmp in list_:
    print('*****')
else:
    print('not fo')
```

1 len(list_)는 4개, 인덱싱 값=>
list_='eake'

0	1	2	3
e	a	k	e

```
###################
# 1차원 리스트자료
###################

list_=['a','cba','ce']
print('리스트내의 원소 갯수는 %d 입니다.'%len(list_))
print(list_[1])

tmp='c'

for i in list_:
    if tmp in i:
        print(i+'***')
    else:
        print(i+'xxxx')
```

2 len(list_)는 3개, 인덱싱 값=>
list_=['a','cba','ce']

0	1	2
a	c b a	c e

```
###################
# 2차원 리스트자료
###################

list_=[['a','cba','ce'],[1,3]]
print('리스트내의 리스트 갯수는 %d 입니다.'%len(list_))
#print( list_[1])

tmp='re'
for i in list_:

    #읽어들인 i값리스트를 그 리스트안의 원소단위로 나눔
    for j in i:
        print(j)
        #if tmp in i:
```

3 len(list_)는 2개, 인덱싱 값=>
list_=[['a','cba','ce'],[1,3]]

Part 0			Part 1		Pa Pa
0	1	2	0	1	Pa Pa
a	c b a	c e	1	3	

for i in list ==> Part 0, Part 1이 들어감
for j in i: ==> Part 0의 0, 1, 2 / Part 1의 0, 1이 들어감

제 목	05장 예외처리와 함수	
상세내용 1)	예외처리와 사용자 정의함수	

1. 예외 처리(Exception Handling)

- 예기치 못한 상황으로 에러가 발생하여 비정상 종료 해결

▶ 문제 발생의 예

```
>>> 2013 * (1229/0)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    2013 * (1229/0)
ZeroDivisionError: division by zero

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

▶ try~except

```
try:
  예외를 유발할 수 있는 구문
except <예외 종류>:
  예외 처리를 수행하는 구문
```

- try절의 영역에는 예외가 발생할 수 있는 구문이 들어가며 except절의 영역에서는 try절에서 예외가 발생하였을때 잡아서 처리를 수행하는 구문이 들어갑니다.

```
try:
    a = 10 / 0
except ZeroDivisionError:
    print('나누는 수는 0이 될 수 없습니다!')

▶결과
나누는 수는 0이 될 수 없습니다!
```

```
try:
    a = int(input("첫번째 숫자를 입력하세요: "))
    b = int(input("두번째 숫자를 입력하세요: "))
    print("a + b = ", a + b)
except ValueError:
    print('값이 적절하지 않습니다.')
```

▶결과

```
첫번째 숫자를 입력하세요: 50
두번째 숫자를 입력하세요: 이십
값이 적절하지 않습니다.
```

```
try:
    a = int(input("피제수를 입력하세요: "))
    b = int(input("제수를 입력하세요: "))
    print("a / b = ", a / b)

except ValueError:
    print('값이 적절하지 않습니다.')
except ZeroDivisionError:
    print('제수는 0이 될 수 없습니다!')
```

▶결과

```
피제수를 입력하세요: 10
제수를 입력하세요: 0
제수는 0이 될 수 없습니다!
```

```
try:
    a = int(input("피제수를 입력하세요: "))
    b = int(input("제수를 입력하세요: "))
    print("a / b = ", a / b)
except (ValueError, ZeroDivisionError):
    print('제수가 0이거나 값이 적절하지 않습니다.')
```

▶결과

```
피제수를 입력하세요: 10
제수를 입력하세요: 영
제수가 0이거나 값이 적절하지 않습니다.
```

```
try:
    a = 50 / "이십"
except TypeError as e:
    print('예외:', e.args[0])
```

▶결과

```
예외: unsupported operand type(s) for /: 'int' and 'str'
```

▶ else

- except절이 예외가 발생하면 처리하는 영역이라면, else절은 예외가 발생하지 않았을 경우에 작업이 수행되는 영역입니다.
- else절은 모든 except 절의 가장 마지막에 등장하여야 하며, 필요에 의해 else절을 달 수도 있고 달지 않을 수도 있습니다.

```
try:
  예외를 유발할 수 있는 구문
except <예외 종류>:
  예외 처리를 수행하는 구문
else:
  예외가 발생하지 않을 경우 수행할 구문
```

```
try:
    f = open('test.txt', 'r')
except IOError:
    print('파일을 열지 못했습니다.')
else:
    print('test.txt:', f.read())
    f.close()
```

▶결과

```
test.txt: 테스트 파일!
```

▶ finally

- 예외가 발생하든 말든 실행되는 영역

```
try:
  예외를 유발할 수 있는 구문
except <예외 종류>:
  예외 처리를 수행하는 구문
finally:
  예외 발생 여부과 상관없이 항상 실행되는 구문
```

```
try:
    a = 10 / 0
except ZeroDivisionError:
    print('제수는 0이 될 수 없습니다!')
finally:
    print('무조건 실행되는 영역!')

▶결과
제수는 0이 될 수 없습니다!
무조건 실행되는 영역!
```

▶ raise

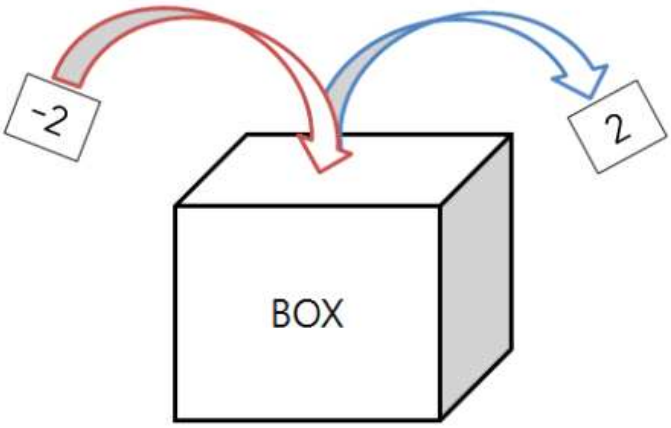
- 의도적으로 개발자가 예외를 발생시켜야 할 경우
- raise 구문을 통해 NameError 예외를 강제로 발생시킴

```
raise[예외]
raise[예외(데이터)
```

```
try:
    a = int(input('피제수를 입력하세요: '))
    b = int(input('제수를 입력하세요: '))
    if a <= 0 or b <= 0:
        raise ArithmeticError('피제수 혹은 제수가 0 이하일 수 없습니다.')

except ArithmeticError as e:
    print('예외 발생:', e.args[0])

▶결과
피제수를 입력하세요: 2030
제수를 입력하세요: -10
예외 발생: 피제수 혹은 제수가 0 이하일 수 없습니다.
```

2. 함수(Function)

값을 함수에 집어넣으면, 함수는 결과값을 되돌려줍니다.



▶ 함수형식

```
def 함수명(입력파라미터):
    문장1
    문장2
    [return 리턴값]
```

```
>>> def absolute(n):
    if n < 0:
        n = -n
    return n
```

```
>>> absolute(3)
3
>>> absolute(0)
0
>>> absolute(-1)
1
```

```
>>> def total(a, b):
    return a + b
>>> total(1, 5)
6
>>> total(100, 20000)
20100
```

▶ 2개 이상의 값 반환

- a와 b를 곱한 값과 나눈 값을 튜플 형태의 값으로 반환

```
>>> def mul_div(a, b):
    return a * b, a / b
>>> mul_div(4, 5)
(20, 0.8)
```

- 반환되는 튜플 값을 따로따로 저장하려면 아래와 같이 호출

```
>>> def mul_div(a, b):
    return a * b, a / b
>>> mul, div = mul_div(4, 5)
>>> print(mul, div)
20 0.8
```

▶ 스코핑 룰(Scoping rule)

- 변수의 생존 범위에 관련된 규칙

```
>>> n = 10
>>> def func(n):
    n = n * 10
>>> func(n)
>>> print(n)
10
```

```
>>> n = 10
>>> def func():
    global n        #전역변수
    n = n * 10
>>> func()
>>> print(n)
100
```

▶ 가변 인자 목록

- 인자의 갯수가 정해지지 않은 가변 인자를 전달하는 방법

```
def 함수명(*인자):
    문장
    ...
```

- varl가 튜플 형태로 처리되는 가변 인자 목록이며, 이를 for문으로 순회하여 요소에 접근하여 출력하도록 하는 함수 func를 정의

```
>>> def func(*varl):
    for i in varl:
        print(i, end=' ')
>>> func(1, 2)
1 2
>>> func(4, 5, 6)
4 5 6
>>> func(4)
4
```

▶ 기본 인자값

- 함수의 인자에 기본 값을 지정해 줄 수 있습니다.

```
>>> def mul(a, b = 10):
    return a * b
>>> mul(3)
30
>>> mul(4, 5)
20
```

```
>>> def mul(a = 10, b):
    return a * b
SyntaxError: non-default argument follows default argument
```

제 목	06장 csv자료 처리 및 시각화	
상세내용 1)	pandas, numpy, matplotlib	

1. Pandas 패키지를 이용한 데이터 프레임 자료 처리

- 테이블 형식의 데이터 (tabular, rectangular grid 등으로 불림)를 다룰 때 사용한다.

▶ 판다스의 기능

- 축의 이름을 따라 데이터를 정렬할 수 있는 자료 구조(data structure). 다양한 소스에서 가져온 다양한 방식으로 색인된 데이터를 핸들링 가능한 기능
- 통합된 시계열 기능
- 시계열, 비시계열 데이터 모두 다룰 수 있는 자료 구조
- 산술연산 및 한 축(column을 의미하는 듯)의 모든 값을 더하는 등의 축약연산이 가능
- 누락된 데이터의 유연한 처리 기능
- SQL 같은 관계연산 수행 기능

* pandas dataframe은 다양한 데이터 타입으로부터 만들 수 있다.
* ndarray, dictionary, dataframe, series, list의 예를 들고 있다.
  (IPython의 display 함수는 IPython 쉘 환경에서 pandas dataframe을 테이블 형식으로 표현해준다.)

▶ Pandas Dataframe 만들기

```
import pandas as pd
import numpy as np
import pandas as pd
df = pd.DataFrame([[1, 2, 3], [4, 5, 6]])
df
```

* Series의 경우 pandas에서 제공하는 데이터 타입인데, index가 있는 1차원 배열이며 문자, 논리형, 숫자 모든 데이터 타입이 들어갈 수 있다.
* dataframe의 한 컬럼, 한 컬럼이 series이다.

▶ loc와 iloc를 이용한 데이터 프레임 인덱싱

```
df = pd.DataFrame({"A":[1,4,7], "B":[2,5,8], "C":[3,6,9]})

# Use 'iloc[]' to select a row
display(df.iloc[0])
display(df.loc[0])
display(df.ix[0])

# Use 'loc[]' to select a column
display(df.loc[:,'A'])
display(df['A'])

# 특정 row, column을 선택하기
display(df.ix[0]['A'])
display(df.loc[0]['B'])
```

* column을 조회할때 df['column'], row를 조회할 때는 df.ix[index]를 많이 사용한다.

▶ PANDAS 그룹별 평균 구하기

```
import pandas as pd

data = pd.DataFrame( {"cluster" : [1,1,2,1,2,3], "org":['a','a','h','c','d','w'],
        "time":[8,6,34,23, 74,6] })
data.groupby(['cluster'], as_index=False).mean()
data.groupby(['cluster', 'org'], as_index=False).mean()
```

cluster	org	time
1	a	8
1	a	6
2	h	34
1	c	23
2	d	74
3	w	6

	cluster	time
0	1	12.333333
1	2	54.000000
2	3	6.000000

	cluster	org	time
0	1	a	7
1	1	c	23
2	2	d	74
3	2	h	34
4	3	w	6

▶ Dataframe에 열, 행, 인덱스 추가하기

- 인덱스 설정하기

```
# Print out your DataFrame `df` to check it out
df = pd.DataFrame({"A":[1,4,7], "B":[2,5,8], "C":[3,6,9]})
display(df)

# Set 'C' as the index of your DataFrame
df = df.set_index('A')
display(df)
```

- 인덱스 접근하기 (ix와 iloc의 차이점)

```
print(df.ix[7])
print(df.iloc[1])
```

```
df = pd.DataFrame(data=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]), index= [2, 'A', 4],
columns=[48, 49, 50])

display(df)
# Pass `2` to `loc`
print(df.loc[2])

# Pass `2` to `iloc`
print(df.iloc[2])

# Pass `2` to `ix`
print(df.ix[2])
```

- 행 추가하기

```
df = pd.DataFrame(data=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]), index= [2.5,
12.6, 4.8], columns=[48, 49, 50])
display(df)

# There's no index labeled `2`, so you will change the index at position `2`
df.ix[2] = [60, 50, 40]
display(df)

# This will make an index labeled `2` and add the new values
df.loc[2] = [11, 12, 13]
display(df)
```

- Append를 이용해 행 추가하기

```
df = pd.DataFrame(data=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]), columns=[48, 49, 50])
display(df)

a = pd.DataFrame(data=[[1,2,3]], columns=[48,49,50])
display(a)

df = df.append(a)
df = df.reset_index(drop=True)
display(df)
```

- 열 추가하기

```
df = pd.DataFrame(data=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]), columns=['A', 'B', 'C'])

# Study the DataFrame `df`
display(df)

# Append a column to `df`
df.loc[:, 'D'] = pd.Series(['5', '6', '7'], index=df.index)

# Print out `df` again to see the changes
display(df)

df['E'] = pd.Series(['5', '6', '7'], index=df.index)
display(df)
```

▶ Dataframe의 인덱스, 컬럼, 데이터 삭제하기

-인덱스 삭제
인덱스를 지워야할 경우는 그렇게 많지 않을 것이다. 주로 reset_index를 이용해서 index를 리셋하는 것을 많이 사용한다. 혹은 index의 이름을 삭제하고 싶다면 del df.index.name을 통해 인덱스의 이름을 삭제할 수 있다.

-컬럼 삭제
drop 명령어를 통해 컬럼 전체를 삭제할 수 있다. axis=1은 컬럼을 뜻한다. axis=0인 경우, 로우를 삭제하며 이것이 디폴트이다. inplace의 경우 drop한 후의 데이터프레임으로 기존 데이터프레임을 대체하겠다는 뜻이다. 즉, 아래의 inplace=True는 df = df.drop('A', axis=1)과 같다.

- 인덱스 삭제

```
df = pd.DataFrame(data=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]), columns=['A', 'B', 'C'])
display(df)

# Drop the column with label 'A'
# drop axis의 경우 column이면 1, row이면 0이다.
df.drop('A', axis=1, inplace=True)
display(df)
```

- 행 삭제(중복 row 삭제)

```
df = pd.DataFrame(data=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [40, 50, 60], [23, 35, 37]]),
                  index= [2.5, 12.6, 4.8, 4.8, 2.5],
                  columns=[48, 49, 50])

display(df)

df = df.reset_index()
display(df)

df = df.drop_duplicates(subset='index', keep='last').set_index('index')
display(df)
```

- 인덱스를 통한 행 삭제
  drop 명령어를 통해 특정 index를 가진 row를 삭제할 수 있다. df.index[1] 명령어는 1번 째 위치에 있는 index를 가져온다. 가져온 이 index를 drop에 인풋으로 넣어주면 해당 index를 가진 row를 삭제할 수 있다.

```
# Check out your DataFrame `df`
df = pd.DataFrame(data=np.array([[1, 2, 3], [1, 5, 6], [7, 8, 9]]), columns=['A', 'B', 'C'])
display(df)

# Drop the index at position 1
print(df.index[1])
print(df.drop(df.index[1]))
print(df.drop(0))
```

▶ 누락 데이터 처리

- 누락된 데이터를 처리하는 일은 데이터 분석 애플리케이션에서 흔이 있는 일
- 누락 데이터를 가능한 쉽게 처리
- 모든 기술통계는 누락된 데이터를 배제하고 처리
- 누락된 데이터를 실수든 아니든 모두 NaN(Not a Number)으로 취급
- 누락된 값을 쉽게 찾을 수 있는 파수병 역할

```
#누락된 데이터 처리하기
from pandas import Series, DataFrame
import pandas as pd

#Series == 고정 길이의 정렬된 사전형
string_data = Series(['aardvark', 'artichoke', np.nan, 'avocado'])
string_data
string_data.isnull()

# Python 내장 None 또한 null 취급
string_data[0] = None
string_data.isnull()
```

```
#누락된 데이터 골라내기
from pandas import Series, DataFrame
import pandas as pd
from numpy import nan as NA

data = Series([1, NA, 3.5, NA, 7])
data.dropna()
data[data.notnull()]

data = DataFrame([[1., 6.5, 3.], [1., NA, NA], [NA, NA, NA], [NA, 6.5, 3]])
cleaned = data.dropna()
data

# NA가 하나라도 있으면 기본적으로 제외해서 보여줌
cleaned
data.dropna(how='all')
data[4] = NA
data
# axis=0 is row.
data.dropna(axis=0, how='all')
# axis=0 is row.
data.dropna(axis=1, how='all')
#axis => 연산을 수행할 축. DataFrame에서 0은 로우고 1은 칼럼이다.
```

2. Pandas 패키지를 이용한 외부자료(.csv) 읽고 처리

▶ 파일처리

- 텍스트 파일 이용하는 방법

함수	설명
read_csv	파일, URL 또는 파일과 유사한 객체로부터 구분된 데이터를 읽어온다. 데이터 구분자는 쉼표(,)를 기본으로 한다.
read_table	파일, URL 또는 파일과 유사한 객체로부터 구분된 데이터를 읽어온다. 데이터 구분자는 탭('\t')을 기본으로 한다.
read_fwf	고정폭 칼럼 형식에서 데이터를 읽어온다(구분자가 없는 데이터)
read_clipboard	클립보드에 있는 데이터를 읽어오는 read_table 함수. 웹페이지에서 표를 긁어올 때 유용하다.

```
from pandas import DataFrame, Series
import pandas as pd

df = pd.read_csv('ex1.csv')
pd.read_csv(ex1.csv', header=None)

# 원래 있던 Column명 무시하고 내가 원하는 Column명 설정
pd.read_csv('ex1.csv', names=[5,6,7,8,9])
pd.read_csv('ex1.csv', names=['a1', 'b1', 'c1', 'd1', 'message1'])
df

# csv는 DataFrame으로 읽어온다.
type(df)
pd.read_table('ex1.csv', sep=',')
pd.read_table('ex1.csv', sep=',', header=None)

# header 자동 생성
pd.read_csv('ex2.csv', header=None)

# header 옵션이 없을시 header를 첫번째 줄로 이용
pd.read_csv('ex2.csv')

# Column명 추가
pd.read_csv('ex2.csv', names=['a', 'b', 'c', 'message'])

names = ['a', 'b', 'c', 'd', 'message']
pd.read_csv('ex2.csv', names=names)

# message -> index
pd.read_csv('ex2.csv', names=names, index_col='message')

pd.read_csv('ex2.csv', names=names, index_col='a')
#csv_mindex.csv 확인
```

참고: 외래관광객 실태조사 자료 읽고 처리

외래관광객 실태조사 분석모델 설정

ftour.csv

```
import pandas as pd
df=pd.read_csv('ftour.csv',encoding='euc-kr')
display(df.head())

display(df.info())

```

```
C:₩Users₩Administrator₩miniconda3₩lib₩site-packages₩IPython₩core₩interactiveshell.py:3146: DtypeWarning: Columns (2,73,76) have mix
d types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

	id	gender	age	edu	job	nat	other	count	month	info1	...	accom	shop1	shop2	shop3	prd1	prd2	prd3	expense	card	cash
0	584	2	0	1	8	일본		1	3	3	...	1	7	11		5	19		480.975000	0.000	480.97500(
1	840	1	3	2	2	싱가포르	1	18	11	3	...	1	7			5			646.888000	323.444	323.44400(
2	524	1	1	3	8	프랑스		1	6	3	...	3	6	7	8	9	10	11	560.650000	0.000	560.65000(
3	282	1	1	2	8	미국		2	2	2	...	4	4	6	7	9	12	16	509.225468	0.000	509.22546(
4	562	1	2	3	6	미국	1	1	8	3	...	2	12			21			409.000000	0.000	409.00000(

외래관광객 실태조사 분석모델 설정

분석: 거주국가(nat)에 따른 동반유형(member)과 만족활동(bestact) 관계 분석

거주국가(관광객 국가, nat) 분석

```
df['nat'].describe()
```

```
count     13501
unique       19
top          중국
freq       2878
Name: nat, dtype: object
```

```
df['nat'].unique()
```

```
array(['일본', '싱가포르', '프랑스', '미국', '호주', '홍콩', '말레이시아', '인도', '영국', '중동',
       '필리핀', '중국', '러시아', '인도네시아', '캐나다', '베트남', '독일', '태국', '대만'],
      dtype=object)
```

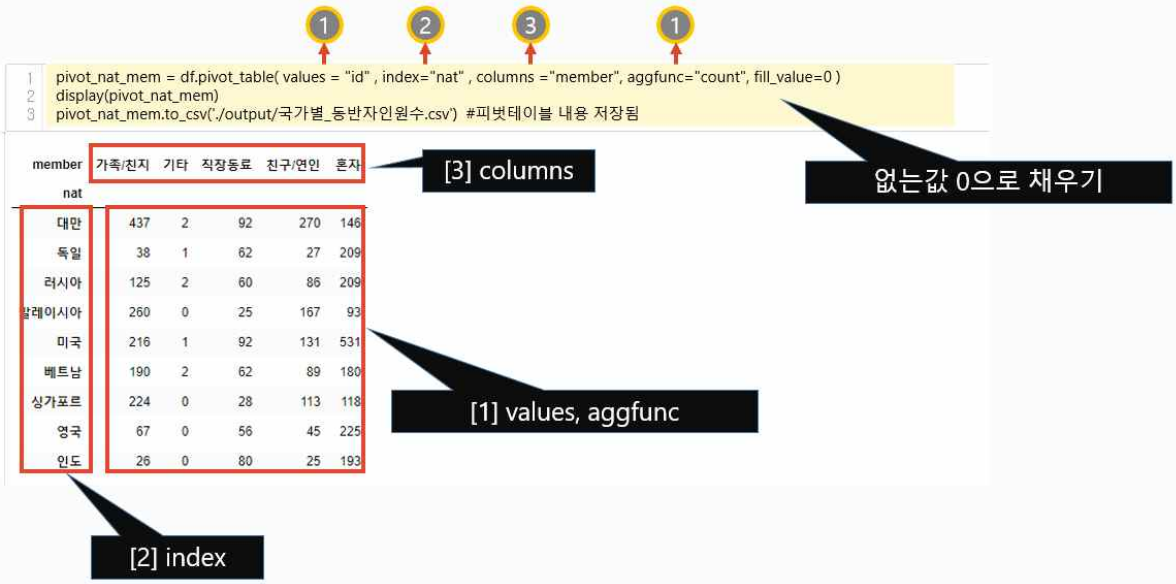
```
df.groupby('nat').nat.count()
```

```
nat
대만         947
독일         337
러시아        482
말레이시아      545
미국         971
베트남        523
싱가포르       483
영국         393
인도         324
인도네시아      440
일본        1543
중국        2878
중동         434
캐나다        446
태국         658
```

외래관광객 실태조사 분석모델 설정

분석: 거주국가(nat)에 따른 동반유형(member)과 만족한활동(bestact) 관계 분석

거주국가(nat)별 동반유형(member)의 인원수 (피벗테이블을 이용한 집계)



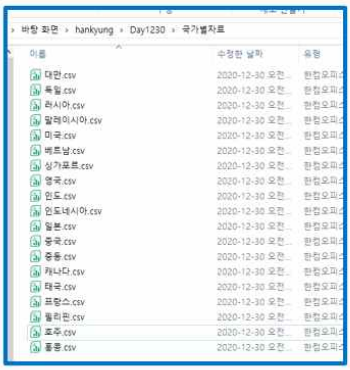
외래관광객 실태조사 분석모델 설정

분석: 거주국가(nat)에 따른 동반유형(member)과 만족한활동(bestact) 관계 분석

국가별자료를 검색하여 '국가별자료폴더안에 프랑스.csv로 저장합니다.'

아래 구문을 for로 변경하면 국가별로 자료를 나누어 저장하는 자동화를 실행할수 있습니다.

```
조건='프랑스'
tmp=df[df['nat']==조건]
tmp.head()
tmp.to_csv('./국가별자료/'+ 조건 + '.csv')

조건='미국'
tmp=df[df['nat']==조건]
tmp.head()
tmp.to_csv('./국가별자료/'+ 조건 + '.csv')
```



제 목	7장 시각화(pyplot)	
상세내용 1)	시각화	

1. matplotlib 라이브러리

▶ pyplot 모듈

- 그래프를 그리는 데 사용되는 다양한 함수를 포함

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

```
plt.plot([1, 2, 3, 4])
plt.show()
```

- y축 값 1, 2, 3, 4를 지나는 직선이 그려진 것을 확인
- plot 함수를 호출할 때 x 축 값을 따로 지정하지 않으면 자동으로 실숫값이 할당
- y=x2 (x>=0) 그래프

```
x = range(0, 100)
y = [v*v for v in x]
plt.plot(x, y)
plt.show()
```

```
plt.plot(x, y, 'ro')
plt.show()
```

- 주요 색상

문자	색상
b	blue(파란색)
g	green(녹색)
r	red(빨간색)
c	cyan(청록색)
m	magenta(마젠타색)
y	yellow(노란색)
k	black(검은색)
w	white(흰색)

마커	의미
o	circle(원)
v	triangle_down(역 삼각형)
^	triangle_up(삼각형)
s	square(네모)
+	plus(플러스)
.	point(점)

▶ Figure와 subplots

- 한 화면에 여러 개의 그래프를 그리는 객체와 메서드입니다.
- subplot의 개수는 add_subplot 메서드의 인자를 통해 조정할 수 있습니다.

```
#2x1 subpolt
fig = plt.figure()
ax1 = fig.add_subplot(2, 1, 1)
ax2 = fig.add_subplot(2, 1, 2)
plt.show()
```

- add_subplot 메서드를 호출하면 AxesSubplot 객체가 생성
- Axes가 하나의 subplot과 유사한 개념

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax1 = fig.add_subplot(2, 1, 1)
ax2 = fig.add_subplot(2, 1, 2)

x = range(0, 100)
y = [v*v for v in x]

ax1.plot(x, y)
ax2.bar(x, y)
plt.show()
```

- 첫 번째 subplot (ax1)에는 plot 함수를 호출한 그래프
- 두 번째 subplot (ax2)에는 bar라는 함수를 호출한 그래프

▶ 라벨 및 범례 표시

- x축과 y축의 값이 어떤 데이터인지를 표시

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

x = np.arange(0.0, 2 * np.pi, 0.1)
sin_y = np.sin(x)
cos_y = np.cos(x)

fig = plt.figure()
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)

ax1.plot(x, sin_y, 'b--')
ax2.plot(x, cos_y, 'r--')
ax1.set_xlabel('x')
ax1.set_ylabel('sin(x)')

ax2.set_xlabel('x')
ax2.set_ylabel('cos(x)')

plt.show()
```

- matplotlib.pyplot 함수의 크기 및 한글깨짐 전역변수로 설정

```
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [20, 6]
plt.rc('font', family='Malgun Gothic')
plt.rcParams['axes.unicode_minus'] = False  #  한글폰트 사용시 그래프에서 마이너스 폰트 깨지는 문제에 대한 대처
```

▶ 참고: 외국인 관광객 데이터를 이용하여 거주국가(nat)에 따라 체류기간(period)과, 지출경비(expense), 전반적만족도(overal) 의 차이 분석

외래관광객 실태조사 분석모델 설정

ftour.csv

```
import pandas as pd
df=pd.read_csv('ftour.csv',encoding='euc-kr')
display(df.head())

display(df.info())
```

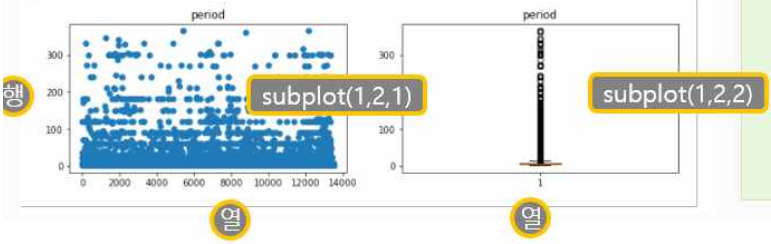
```
C:\Users\Administrator\miniconda3\lib\site-packages\IPython\core\interactiveshell.py:3146: DtypeWarning: Columns (2,73,76) have mix
d types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

	id	gender	age	edu	job	nat	other	count	month	info1	...	accom	shop1	shop2	shop3	prd1	prd2	prd3	expense	card	cash
0	584	2	0	1	8	일본		1	3	3	...	1	7	11		5	19		480.975000	0.000	480.97500(
1	840	1	3	2	2	싱가포르	1	18	11	3	...	1	7			5			646.888000	323.444	323.44400(
2	524	1	1	3	8	프랑스		1	6	3	...	3	6	7	8	9	10	11	560.650000	0.000	560.65000(
3	282	1	1	2	8	미국		2	2	2	...	4	4	6	7	9	12	16	509.225468	0.000	509.22546(
4	562	1	2	3	6	미국	1	1	8	3	...	2	12			21			409.000000	0.000	409.00000(

```
df['period'].describe()
```

```
count    13501.000000
mean        14.616399
std         34.456984
min          1.000000
25%          4.000000
50%          5.000000
75%          8.000000
max        365.000000
Name: period, dtype: float64
```

```
import matplotlib.pyplot as plt
#차트크기 조정
plt.rcParams['figure.figsize'] = [12, 3]

plt.subplot(1,2,1)
plt.scatter(df.index, df['period'], label = "df")
plt.title('period')

plt.subplot(1,2,2)
plt.boxplot(df['period'])
plt.title('period')
plt.show()
```



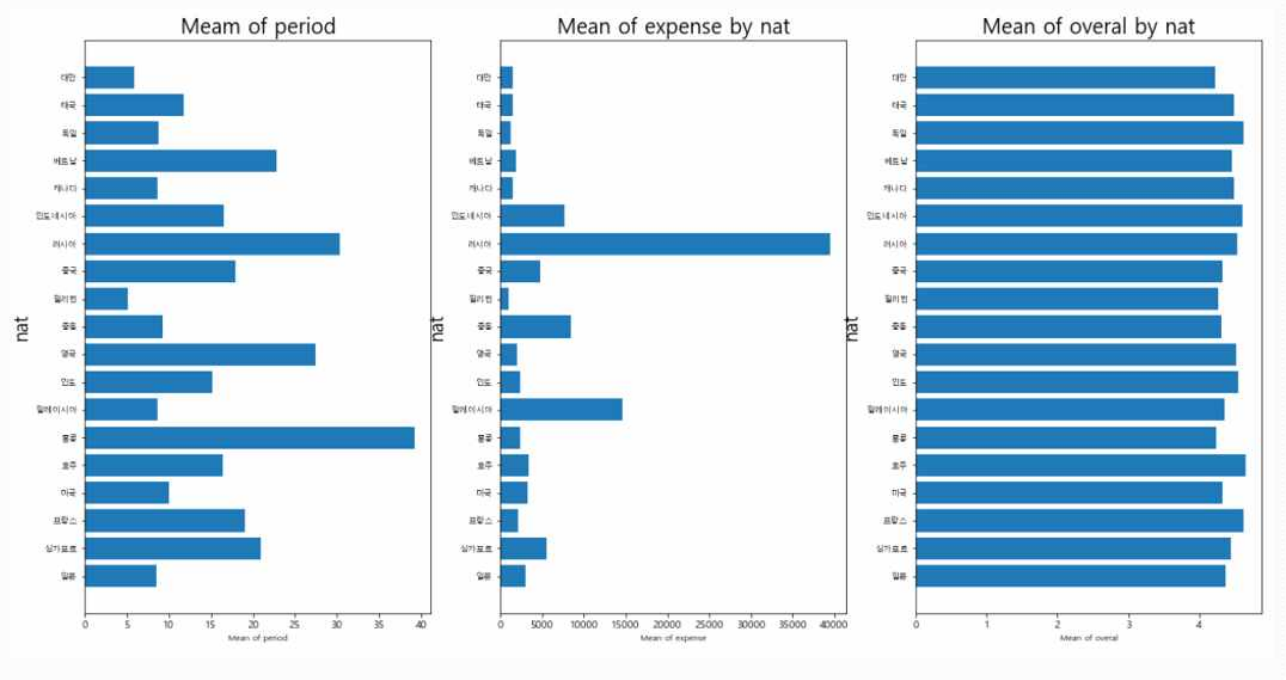
subplot은 차트창나누기
subplot(행의갯수,열의갯수,내위치값)

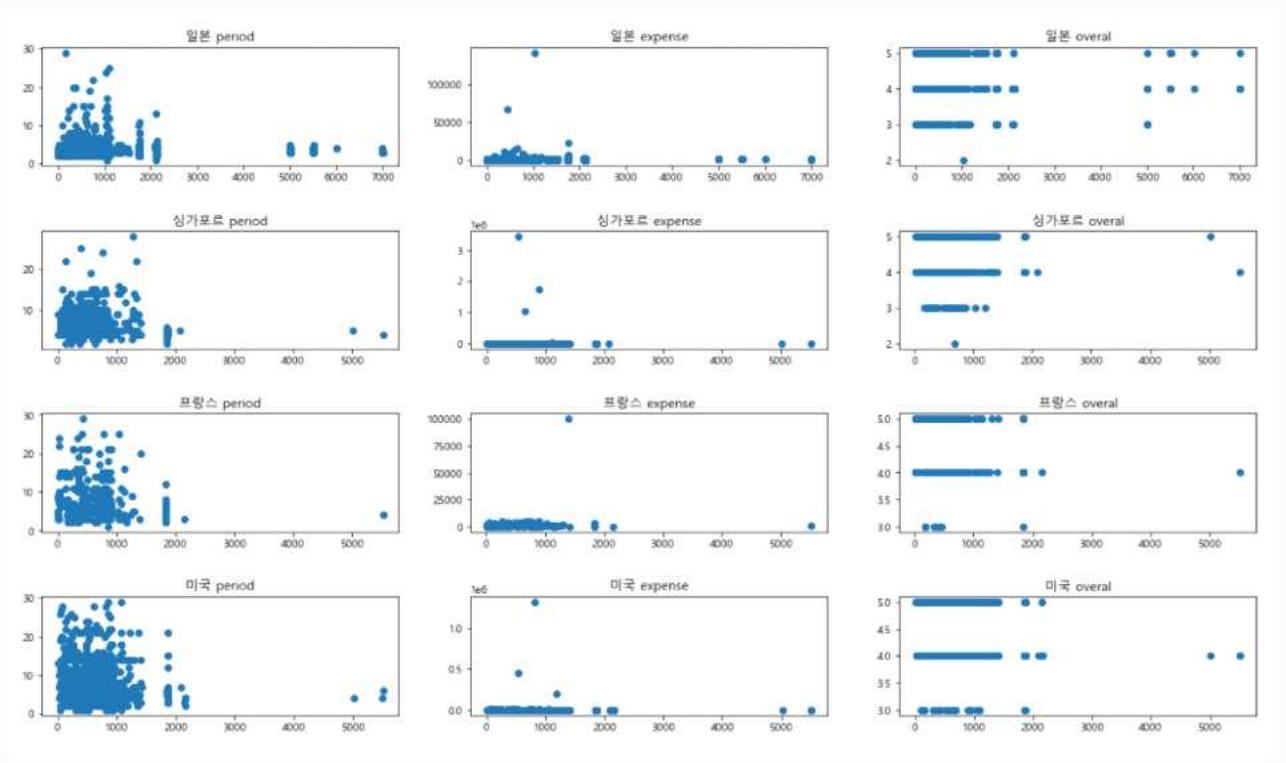
subplot(1,2,1)

1행2열로나눈 차트창에서 1번째위치차트임

---

subplot(3,4,1) ~ subplot(3,4,12)


Meam of period
Mean of expense by nat
Mean of overal by nat
nat
Mean of period
Mean of expense
Mean of overal

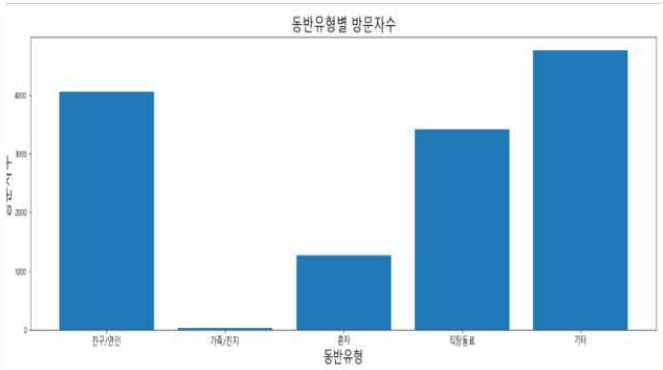

일본 period
일본 expense
일본 overal
싱가포르 period
싱가포르 expense
싱가포르 overal
프랑스 period
프랑스 expense
프랑스 overal
미국 period
미국 expense
미국 overal

외래관광객 실태조사 분석모델 설정

분석: 거주국가(nat)에 따른 동반유형(member)과 만족한활동(bestact) 관계 분석

동반유형(member 변수) 분석

```
In [52]: df['member'].describe()
Out[52]: count     13501
         unique        5
         top        혼자
         freq       4757
         Name: member, dtype: object

In [53]: df['member'].unique()
Out[53]: array(['친구/연인', '가족/친지', '혼자', '직장동료', '기타'], dtype=object)

In [54]: df.groupby('member').nat.count()
Out[54]: member
         가족/친지    4048
         기타         24
         직장동료     1264
         친구/연인    3408
         혼자       4757
         Name: nat, dtype: int64
```



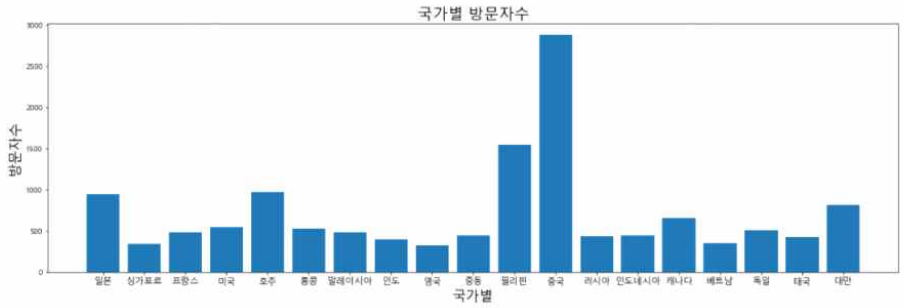
외래관광객 실태조사 분석모델 설정

분석: 거주국가(nat)에 따른 동반유형(member)과 만족한활동(bestact) 관계 분석

거주국가(관광객 국가, nat 변수) 분석

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [20, 6]
plt.rc('font', family='Malgun Gothic')
plt.rcParams['axes.unicode_minus'] = False # 한글폰트 사용시 그래프에서 마이너스 폰트 깨지는 문제에 대한 대처
```

```
tmp=df.groupby('nat').nat.count()
label = df['nat'].unique()
index = np.arange(len(label))

plt.bar(index, tmp)
plt.title('국가별 방문자수', fontsize=20)
plt.xlabel('국가별', fontsize=18)
plt.ylabel('방문자수', fontsize=18)
plt.xticks(index, label, fontsize=12)

plt.show()
```

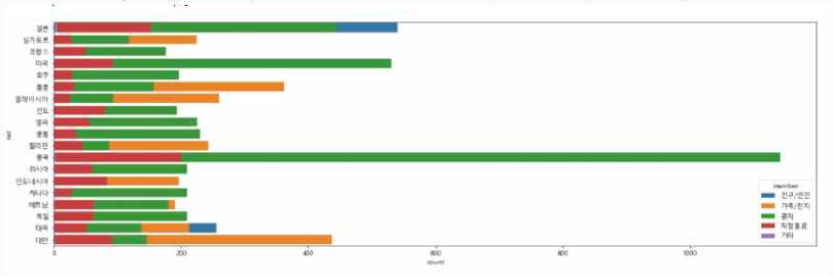


외래관광객 실태조사 분석모델 설정

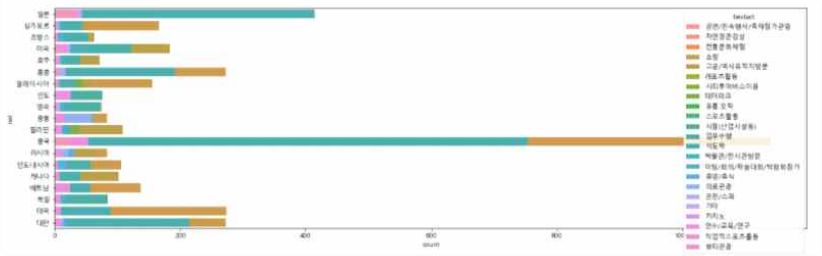
분석: 거주국가(nat)에 따른 동반유형(member)과 만족한활동(bestact) 관계 분석

[미션] 거주국가에 따른 동반유형을 seaborn의 countplot를 이용하여 다음과 같이 누적막대가 나올수 있도록 지정합니다.

sns.countplot(y='nat', hue='member', dodge=False, data=df)



sns.countplot(y='nat', hue='bestact', dodge=False, data=df)



제 목	8장 시각화	
상세내용	지도 시각화	

※ 강의시간에 실습자료료만 제공합니다.

제 목	9장 Open API연결	
상세내용	지도 시각화	

※ 강의시간에 실습자료료만 제공합니다.