ctrl z 돌아가기

ctrl shift z 역으로 돌아가기

ctrl 3 / 4 주석추가/헤제

ctrl a

* 덧셈
* 뺄셈

‘\*’ 곱셈

/ 나눗셈

% 나머지 계산

// 소수점 이하 절삭

\*\* 거듭제곱 계산

%s s는 ‘string’의 첫 글자로서 문자열을 의미한다.

%d d는 ‘digit”의 첫 글자로 정수형 숫자를 의미한다.

$f f는 ‘floating point’의 첫 글자로서 실수형 숫자를 의미한다.

print() 함수에 사용되는 문자열 포맷 코드 :

**%d** : 정수형(interger)숫자

**%f** : 실수형(Floating Point)숫자

**%.nf** : 실수에서 소수점 첫째 자리(둘째 짜리에서 반올림)까지 구함

**%s** : 문자열(String)

여기서 %.nf안에 있는 n은 숫자여야 함!

자주 사용되는 이스케이프 코드

\n : 줄 바꿈

\t : 탭

\\ : 역슬래쉬(\) 출력

\’ : 단 따옴표(‘) 출력

\” : 쌍 따옴표(“) 출력

if~ 구문 : 만약 조건을 만족하면 ~ 작업을 수행하라.

if~else~ 구문 : 만약 조건을 만족하면 ~ 작업을 수행하고, 그렇지 않으면 ~ 작업을 수행하라.

if~elif~else~ 구문 : 만약 조건 1을 만족하면 작업 1을 수행하고, 조건 2를 만족하면 작업 2를 수행하고, 조건 3을 만족하면 작업 3을 수행하고 그렇지 않으면 작업 n을 수행하라.

비교 연산자 : >, <, ==, !=, <=, >=

논리 연산자 : and, or, not

비교 연산자의 종류 :

a == b : a 와 b는 같다.

a != b : a와 b는 같지 않다.

a > b : a는 b보다 크다

a >= b : a는 b보다 크거나 같다.

a < b : a는 b보다 작다.

a <= b : a는 b보다 작거나 같다.

논리 연산자의 종류

조건 1 and 조건 2 : 논리곱(and) 조건1과 조건2가 둘다 참인 경우에만 참이 된다.

조건 1 or 조건 2 : 논리합(or) 조건1 또는 조건 2 중 하나만 참이어도 참이 된다.

not 조건 : 논리부정(not) 조건이 참이면 거짓, 거짓이면 참으로 해서 논리 값을 반대로 변경한다.

영문 소문자를 대문자로 변환하는 데에는 upper()라는 함수를 사용한다.

upper() 함수는 문자열 내에서만 사용된다. upper()함수와 같이 특정 영역에서만 사용되는 함수를 메소드(Method)라고 부른다. 메소드를 사용할 때는 해당 문자열 다음에 점(.)을 찍고 함수를 사용한다.

파이썬의 반복문에는 다음의 두가지가 존재한다.

1. for문
2. while문

프로그래밍 언어에서 반복문은 같은 블록의 코드를 반복해서 수행할 때 사용된다.

반복문은 특정 문장을 반복해서 여러 번 수행할 때 사용하기도 한다.

for는 “~하는 동안”이란 의미를 갖는다. 파이썬을 포함한 많은 프로그래밍 언어에서 사용되는 for문은 주어진 조건에 따라 문장들을 반복 수행하게 된다.

| 반복 루프 | i | sum = sum + i |
| --- | --- | --- |
| 1번째 | 1 | 1 ← 0 + 1 |
| 2번째 | 2 | 3 ← 1 + 2 |
| 3번째 | 3 | 6 ← 3 + 3 |
| 4번째 | 4 | 10 ← 6 + 4 |
| 5번째 | 5 | 15 + 10 + 5 |
| 6번째 | 6 | 21 ← 15 + 6 |
| 7번째 | 7 | 28 ← 21 + 7 |
| 8번째 | 8 | 36 ← 28 + 8 |
| 9번째 | 9 | 45 ← 36 + 9 |
| 10번째 | 10 | 55 ← 45 + 10 |

10번의 반복 루프가 끝나면 최종 결과인 누적된 합계 55가 sum에 저장된다.

range() 함수가 나타내는 반복 횟수의 범위 동안 문장1, 문장2, …가 반복 수행된다. 이 때 변수는 반복 루프 동안 반복 횟수의 범위에 있는 값을 가진다.

range() 함수는 다음의 세 가지 형식으로 사용된다.

range(종료값)은 0에서 종료값-1의 정수 범위를 갖게 된다. 그리고 변수는 각 반복 루프에서 range() 범위에 있는 각각에 값을 가지게 된다. 예를 들어 range(10)은 0에서 9까지의 정수 범위를 의미한다.

range(시작값, 종료값)은 시작값에서 종료값-1의 정수 범위를 갖는다. 예를 들어 range(1,11)은 1에서 10까지의 정수 범위를 갖는다.

range(시작값, 종료값, 증가\_감소)는 시작값에서 종료값-1사이의 정수 범위를 갖는데 각 정수 사이의 간격은 증가\_감소에 의해 결정된다. 예를 들어 range(1, 11, 2)는 1에서 10까지의 정수 중에서 2씩 증가하는 범위를 나타내기 때문에 정수 1, 3, 5, 7, 9의 값을 의미한다.

for의 반복 루프에서 변수는 문자열의 각 문자 값을 가진다.

이중 for문은 for문을 이중으로 사용하는 것을 말한다.

while문은 for문과 함께 많이 사용되는 반복문이다.

while문은 조건식이 참인 동안 들여쓰기 되어 있는 문장1, 문장2, …이 반복 수행된다.

for문이나 while문을 사용하다 보면 반복 루프를 수행 중 중간에 루프를 빠져나가고 싶을 때 break문을 사용한다. 일반적으로 break문은 if문과 같이 사용되어 반복 루프가 진행되는 동안 조건식을 만족하면 반복 루프를 빠져 나가게 한다.

while문에서도 유사하게 break문을 이용하면 원하는 조건에서 반복 루프를 빠져나가게 할 수 있다.

리스트는 여러 개의 데이터 값을 하나의 변수, 즉 리스트에 담을 수 있는 데이터 구조이다. 리스트는 다음과 같이 요소들을 콤마(,)로 분리하고 대괄호[]로 둘러싸게 된다.

리스트는 정수, 실수, 문자열 등 다양한 데이터 형을 가질 수 있다.

리스트는 for문이나 while문 같은 반복문과 같이 많이 사용된다. 반복문과 반복 루프에서는 리스트의 각 요소를 반복적으로 읽어 들여 처리함으로써 리스트의 요소들을 효율적으로 다룰 수 있다.

변수는 리스트 각 요소의 값을 가지고 for 반복 루프가 진행된다.

리스트의 append() 메소드는 리스트 내부에서 사용되는 함수로 리스트 마지막에 요소를 추가하는 역할을 수행한다.

리스트의 insert() 메소드는 리스트의 특정 위치에 새로운 요소를 삽입할 때 사용된다.

리스트의 index() 메소드는 리스트의 특정 요소의 위치, 즉 인덱스 번호를 구하는 데 사용 된다.

리스트에서 요소를 삭제하는 데이는 리스트의 remove(), pop(), clear() 메소드가 사용된다.

리스트의 remove() 메소드는 리스트명의 뒤에 점(‘,’) 다음에 사용한다. 리스트명 내에 있는 요소의 값이 데이터인 요소를 리스트에서 삭제한다.

리스트의 pop() 메소드는 인덱스번호가 지시하는 요소 값을 얻은 다음 해당 요소를 리스트에서 삭제한다.

리스트의 clear() 메소드는 리스트명 내에 있는 모든 요소를 삭제한다.

두 개의 리스트를 서로 병합하려면 문자열을 병합할 때와 마찬가지로 덧셈 기호(+)를 사용한다.

리스트에서 요소들의 합계를 구할 때에는 sum() 함수를 이용한다.

* sum() 함수는 리스트에서 사용되는 함수, 즉 메소드가 아니라 파이썬의 내장 함수이다.

copy() 메소드는 리스트명의 리스트를 복제하여 새로운 리스트를 생성한다.

sort() 메소드는 리스트명 내에 있는 요소들을 오름차순으로 정렬한다. 옵션 reverse=True는 내림차순으로 요소들을 정렬하게 한다.

random 라이브러리는 불러올 때 import random을 입력한다.

| 리스트 메소드 | 의미 |
| --- | --- |
| append() | 리스트의 제일 뒤에 새로운 요소를 추가함 |
| insert() | 리스트에서 특정 인덱스 앞에 새로운 요소를 삽입함 |
| index() | 리스트에서 특정 요소의 위치인 인덱스 번호를 구함 |
| remove() | 리스트에서 특정 값을 가진 요소를 삭제함 |
| pop() | 리스트에서 특정 인덱스 번호를 가진 요소를 추출하고 그 요소를 리스트에서 삭제함 |
| clear() | 리스트의 전체 요소를 삭제함 |
| reverse() | 리스트의 요소를 거꾸로 함 |
| copy() | 리스트를 복사하여 새로운 리스트를 생성함 |
| sort() | 리스트 요소들을 오름차순(또는 내림차순)으로 정렬함 |
| 리스트에서 사용되는 파이썬의 내장함수 | 의미 |
| list() | 새로운 리스트를 생성함 |
| len() | 리스트의 길이를 구함 |
| sum() | 리스트 요소들의 합계를 구함 |
| 문자열 메소드 | 의미 |
| find() | 문자열에서 특정 문자열을 찾아 위치(인덱스 번호)를 구함 |
| replace() | 문자열에서 특정 문자열을 다른 문자열로 치환함 |
| split() | 특정 문자열을 기준으로 문자열을 쪼개서 리스트에 저장함 |
| join() | 리스트의 요소를 하나로 묶어서 문자열로 변환함 |

1. 튜플에서는 리스트의 대괄호[] 대신에 소괄호()를 사용한다.
2. 튜플에서는 리스트와는 달리 요소의 수정과 추가가 불가능하다.

튜플은 생성하려면 소괄호, ()를 이용하거나 tuple() 함수를 사용한다.

튜플에서는 리스트(또는 문자열)에서와 같은 방법으로 인덱스를 이용하여 요소를 추출할 수 있다.

튜플의 인덱스는 0부터 시작한다.

리스트에서와 마찬가지로 len() 함수를 이용하면 튜플의 길이를 쉽게 구할 수 있다.

리스트에서와 마찬가지로 덧셈 기호(+)를 이용하면 튜플을 서로 병합할 수 있다.

파이썬의 딕셔너리는 자료를 찾는 인덱스를 의미하는 키(Key)와 자료의 내용인 값(Value)을 이용하여 데이터를 관리한다. 딕셔너리에서는 다음과 같이 요소들을 중괄호 {}로 감싸게 된다.

딕셔너리 키에 값을 저장함으로써 해당 키의 값을 수정할 수 있다. 이 때 해당 키가 딕셔너리에 존재해야 한다. 만약 딕셔너리에 해당 키가 존재하지 않으면 키와 값을 쌍으로 한 새로운 요소가 딕셔너리에 추가된다.

for 루프에서 사용되는 변수는 딕셔너리의 키가 되고, 딕셔너리[변수]는 딕셔너리의 해당 키에 대응되는 값이 된다.

이중반복문(이중 for문) : for i in range():

for j in range():

함수 int()는 실수(Floating point)나 문자열(String)을 정수형 숫자로 변환한다.

함수 float()는 정수(Interger)나 문자열(String)을 실수형 숫자로 변환하는 데 사용한다.

함수 str()은 정수형이나 실수형 숫자를 문자열로 변환하는 데 사용된다.

IDLE은 ‘Intergrated Development and Learning Environment’의 약어로 파이썬의 ‘통합 개발과 학습 환경’이라는 뜻이다. IDLE은 우리말로 ‘아이들’이라고 부르는데 이 IDLE은 파이썬으로 프로그램을 개발하는 데 가장 많이 사용되는 개발 프로그램 중의 하나이다.

소스 프로그램(Source Program)은 인간이 기술한 언어, 즉 컴퓨터 키보드로 타이핑하여 작성한 프로그램을 의미한다. 다른 말로 소스 코드(Source Code)라고도 부른다. 이 소스 프로그램을 저장한 파일을 소스 파일(Source File)이라고 한다.

SyntaxError에서 ‘Syntax’는 문법을 의미한다. 따라서 이것은 문법상 오류를 뜻하는 것으로 자주 발생되는 오류 중의 하나이다.

len() 함수는 문자열의 길이를 구하는 데 사용된다.

나머지 연산자 %는 어떤 수를 나눈 나머지를 계산한다.

소수점 절삭 연산자 //는 나눗셈 결과에서 소수점 이하를 절삭하게 된다.

파이썬에서 변수의 형은 그 변수에 저장되어 있는 값에 따라 결정된다. 정수 값을 가지고 있는 변수는 정수형 변수, 실수 값을 가진 변수는 실수형 변수가 된다.

거듭제곱 연산자 \*\*는 어떤 수의 거듭제곱을 계산하는 데 사용한다.

문자열(String)은 하나 또는 여러 개의 문자로 구성된 데이터형이다. 문자열에서는 해당 문자들의 앞과 뒤에 쌍 따옴표(“) 또는 단 따옴표(‘)를 붙인다.

문자열의 위치를 나타내는 인덱스는 1이 아니라 0부터 시작한다.

문자열에서는 공백(‘ ‘)도 하나의 문자이다.

기호 +를 문자열에 사용하면 문자열을 서로 연결할 수 있다.

숫자의 곱셈의 사용되는 기호 \*가 문자열에 사용되면 문자열이 그 횟수만큼 반복된다.

문자열 포맷팅(String formatting)은 특정 포맷에 맞추어 문자열을 재구성할 때 사용된다.

파이썬에서 키보드로 데이터를 입력 받아 처리할 때는 input() 함수가 이용된다.

input() 함수는 입력을 안내하는 문구를 출력하고 그 다음 사용자가 입력하는 데이터를 변수에 저장한다.

컴퓨터에서 키보드로 입력되는 데이터는 문자열로 간주하여 변수나 파일로 저장하게 된다.

print() 함수의 가장 기본적인 사용법은 변수나 데이터를 콤마(,)로 구분하여 출력하는 것이다.

print() 함수에 사용된 키워드 sep은 ‘seperator’의 약어로서 각 학몽 사이에 삽입할 문자열을 지정하는 데 사용된다.

print() 함수를 사용할 때 다음과 같이 콤마(,)로 구분하여 출력하면 공백(“ “)이 항목 사이에 삽입된다.

파이썬에서 역슬래쉬(\)로 시작하는 문자를 이스케이프 코드(Escape code)라고 한다.

주석문은 프로그램을 짤 때 프로그램의 작성자, 작성한 날짜, 프로그램의 기능, 코드에 대한 주석, 즉 설명 글을 다는 데 사용되는 문장을 말한다.

주석 처리된 내용은 프로그램의 실행 결과에는 전혀 영향을 미치지 않는다.

파이썬에 조건문인 if문은 조건식의 참 또는 거짓에 따라 실행되는 코드가 달라 질 때 사용한다.

이런 식으로 if문에서는 조건에 따라 수행되는 문장이 달라지게 된다.

if문을 사용할 때는 if문의 콜론(:) 다음 줄에서는 반드시 들여쓰기가 되어 있어야 한다.

파이썬에서는 다른 프로그래밍 언어와 달리 if문을 사용할 때 반드시 들여쓰기를 한다. if문의 들여쓰기는 파이썬만의 독특한 방식이다.

if문의 들여쓰기 방식은 나중에 배우는 for문, while문, 함수와 클래스 정의 등에서도 그대로 적용된다.

비교 연산자와 논리 연산자는 if문(또는 뒤에서 배우는 for문, while문)의 조건식에 사용되어 조건이 참(True)인지, 거짓(False)인지를 판정하는 데 사용된다.

비교 연산자는 변수, 숫자, 문자열 등을 서로 비교하여 조건식의 참 또는 거짓을 판정한다,

논리 연산자의 종류에는 다음의 표에 나타난 것과 같이 논리곱(and), 논리합(or), 논리부정(not)의 세 가지가 있다,

and 연산자는 두 조건이 모두 참이어야만 참이고, or 연산자는 두 조건 중 하나만 참이어도 참이 되고, not 연산자는 참을 거짓으로 거짓을 참으로 변경한다.

논리 연산자 and는 두 조건이 모두 참일 때만 그 결과가 참이 된다.

논리 연산자 or은 두 조건 중 하나만 참이어도 그 결과가 참이 된다.

if문 다음 줄에 있는 문장의 앞에는 탭 키를 삽입하여 반드시 들여쓰기를 하여야 한다.

if ~ else ~ 구문은 두 가지 조건 만이 존재할 경우에 사용한다.

if 다음의 조건식이 참이면 콜론(:) 다음 줄에 들여쓰기 되어 있는 문장을(를) 수행하고 그렇지 않으면 else : 다음 줄에 있는 문장을(를) 수행한다.

영어 알파벳은 자음과 모음 둘 중의 하나에 해당 되기 때문에 if ~ else ~ 구문이 사용되는 것이다.

if ~ elif ~ else ~ 구문은 하나의 if문에 2개 이상의 조건식이 필요할 때 사용된다.

if의 조건식이 참이면 들여쓰기 되어 있는 문장 을(를) 수행하고, 그렇지 않고 elif의 조건식이 참이면 문장 을(를) 수행하고, 그렇지 않고 앞의 조건식들이 모두 거짓이면 else 다음의 문장을(를) 수행한다.

if ~ elif ~ else에서 사용되는 ‘elif’는 ‘else if’의 약어로서 ‘그렇지 않고 만약’이라는 의미이다.

프로그래밍 언어에서 반복문은 같은 블록의 코드를 반복해서 수행할 때 사용된다.

파이썬의 반복문에는 다음의 두 가지가 존재한다. :

1. for문
2. while문

for는 ‘~ 하는 동안’이란 의미를 갖는다. 파이썬을 포함한 많은 프로그래밍 언어에서 사용되는 for문은 주어진 조건에 따라 문장들을 반복 수행하게 된다.

오류가 있을 때에는 잘못된 부분을 찾아 수정하고 명령을 재실행하여 제대로 된 결과가 나오도록 해야 한다.

주피터 노트북(Jupyter Notebook)은 IDLE 프로그램과 함께 파이썬 프로그램 개발에 가장 많이 사용되는 개발 툴이다. 주피터 노트북은 파이참, 비쥬얼 스튜디오 등 다른 파이썬 개발 프로그램과 비교하여 가볍고 성능도 우수하여 널리 사용된다.

주피터 노트북은 데이터 분석, 시각화, 인공 지능 프로그램을 개발하는 데에 최적화 되어 있지만 일반적인 프로그램 개발에도 무척 유용한 프로그램이다.

주피터 노트북에서 파일을 저장하면 파일 확장자는 .ipynb가 된다. 이러한 주피터 노트북 파일은 오로지 주피터 노트북에서만 사용 가능하다.

파이썬은 직관적이고 단순한 문법 체계로 되어 있어 처음 프로그래밍을 접하는 초보자에게 가장 적합한 언어이다.

파이썬은 웹 서버, 과학적 연산, 사물 인터넷(Internet Of Things), 인공지능(Artificial Intelligence), 게임 등 IT 전문 분야의 다양한 애플리케이션 프로그램을 개발하는 데에 강력한 능력을 발휘한다.

컴퓨터에서 NULL은 값이 없는 것을 의미한다. NULL은 “”(쌍 따옴표(“)를 공백없이 붙여씀)의 표기를 사용한다.

NULL은 0이나 공백(“ “)과는 다르다. 0은 정수 0의 값을 가진다는 것을 의미하고, 공백(“ “)은 키보드의 스페이스 바로 입력하는 공백 문자를 의미한다.

역슬래쉬(\)를 키보드로 입력하려면 키보드의 엔터 키 위에 있는 ₩ 키를 누르면 된다.

쌍 따옴표(“)의 대한 이스케이프 코드인 \”를 사용하면 쌍 따옴표(“) 자체를 print() 함수로 출력할 수 있다.

print() 함수를 이용하여 쌍 따옴표(“) 내에 쌍 따옴표(“) 자체를 출력하려고 하면 오류가 발생하게 된다.

파이썬에서 역슬래쉬(\)로 시작하는 문자를 이스케이프 코드(Escape Code)라고 한다. 조건문은 조건식의 참/거짓에 따라 실행하는 코드가 달라지는 경우에 사용한다.

파이썬을 포함한 많은 프로그래밍 언어에서 사용되는 함수(Function)는 수학에서의 ‘함수’의 개념과 ‘함수’의 영어 단어인 function이 가지는 ‘기능’이라는 의미를 모두 가진다.

함수는 함수명()과 같은 형태로 쓰이는데 지금까지 우리가 사용해 온 print(), input() 등은 모두 함수이다. 또한 함수는 사용자가 함수명과 기능을 새롭게 정의해서 사용할 수도 있다.

함수 정의부와 함수 호출부로 구성되어 있다는 것을 알 수 있다.

함수 정의에서는 def 다음에 함수명()과 콜론(:)을 삽입한 후에 함수가 수행할 기능을 다음 줄에 들여쓰기 한 뒤 문장1, 문장2 … 에 기술하면 된다.

프로그램 내에서 함수명을 적어주면 함수가 호출된다. 함수가 호출되면 함수 정의 부분에서 정의된 함수명()을 실행한 다음 다시 호출한 위치로 돌아온다.

사용자 함수는 사용자가 직접 함수를 정의해서 사용하는 함수이다. 사용자 함수에는 함수의 정의 부분과 함수의 호출 부분이 존재한다.

사용자 함수와는 달리 내장 함수에서는 사용자가 직접 함수를 정의할 필요가 없다. 내장 함수는 파이썬 프로그램 설치 시 내장 함수 정의 부분의 코드들이 같이 설치 되기 때문에 사용자가 함수를 별도로 정의할 필요가 없는 것이다.

| 내장 함수 | 의미 |
| --- | --- |
| print() | 화면에 데이터 값을 출력함 |
| input() | 키보드를 통해 데이터를 입력 받음 |
| range() | 정수의 범위를 설정함 |
| list() | 리스트를 생성한 |
| int() | 문자열이나 실수형 숫자를 정수형 숫자로 변환함 |
| float() | 문자열이나 정수형 숫자를 실수형 숫자로 변환함 |
| str() | 정수형 숫자나 실수형 숫자를 문자열로 변환함 |
| type() | 데이터의 형을 구함 |

매개변수는 호출 함수에서 전달하고자 하는 값이나 변수를 전달받기 위해 함수 정의에서 사용되는 변수를 말한다.

프로그래밍에는 각 언어마다 사용할 수 있는 데이터 형이 정해져 있다.

파이썬은 단일데이터 타입인 숫자, 문자열, 논리 그리고 복합데이터 타입인 리스트, 튜플, 딕셔너리, 집합 데이터 타입을 제공한다.

다량의 데이터를 저장해야 할 때는, 한 변수에 여러 개의 값을 저장할 수 있는 복합 데이터 타입을 사용하는 것이 데이터 생성 및 관리가 편리해진다.

리스트(List)

* 여러 개의 데이터를 가진 순서형 자료
* 수정할 수 있다.
* 파이썬에서는 리스트에 서로 다른 데이터 작업들을 저장할 수 있다.

튜플(Tuple)

* 여러 개의 데이터를 가진 순서형 자료
* 수정할 수 없다.
* 리스트와 마찬가지로 인덱싱이 가능하다. 인덱싱은 **대괄호**로 하도록 주의한다,

딕셔너리(Dictionry)

* 여러 개의 데이터를 키(Key)와 값(Value)을 쌍으로 갖는 자료형
* 딕셔너리는 사전이라는 뜻으로, 키는 실마리, 벨류는 값을 뜻한다.
* 벨류는 딕셔너리 변수에 담긴 정보, 키는 리스트의 인덱스와 비슷한 역할이다.

집합(Set)

* 여러 개의 데이터를 중복으로 허용하지 않는 자료형
* 중복으로 넣은 요소는 하나로 표현된다.
* 합집합, 교집합, 차집합 연산을 할 수 있다.

리스트는 대괄호 [] 와 콤마 , 를 사용해서 만든다.

복합데이터인 리스트도 사용하기 위해서는 변수(데이터를 저장하는 공간)에 저장해주어야 한다.

저장하는 값은 같은 자료형이 아니어도 상관 없다.

리스트를 만들 때는 빈리스트로 생성하거나, 초기값을 가지고 생성할 수 있다.

리스트의 크기나 내용에 대한 정보가 없을 경우에는 빈 리스트로 만든 후 값을 추가한다.

각 요소에는 인덱스라는 고유 번호가 지정되고, 인덱스는 첫 번째 값부터 순서대로 0으로 시작된다. 인덱스는 각각의 값들의 주소라고 기억해두자

리스트는 인덱싱 및 슬라이싱으로 값에 접근할 수 있다.

인덱싱은 한 번에 한 항목, 슬라이싱은 한 번에 여러 개의 항목들 접근이 가능하다.

순서형 자료를 토대로 반복을 처리하는 for문은 리스트 자료형과 함께 자주 사용된다.

| 종류 | 설명 |
| --- | --- |
| 인덱싱 | 인덱스(Index)를 통해 개별적으로 접근 할 수 있다. |
| 리스트 안의 리스트인덱싱 | 리스트 안에 리스트가 있을 경우에도 인덱스로 접근 가능하다. |
| 음수 인덱싱 | 음수 인덱스를 사용하여 인덱싱 가능하다. |
| 슬라이싱 | 슬라이싱 기법으로 한 번에 여러 개 항목을 추출할 수 있다. |
| 음수 슬라이싱 | 음수 인덱스를 사용하여 슬라이싱이 가능하다 |
| 인덱스 생략 슬라이싱 | 시작인덱스와 끝인덱스는 각각 생략 가능하고, 모두 생략도 가능하다. |
| 특정 간격 슬라이싱 | 슬라이싱 세번째 항목에 간격을 입력해주면 특정간격으로 슬라이싱 가능하다. |

순서형 자료를 토대로 반복을 처리하는 for문은 리스트 자료형과 함께 자주 사용된다.

| 종류 | 설명 |
| --- | --- |
| in | 리스트 값들 중 원하는 데이터가 있다면 True 변환 |
| not in | 리스트 값들 중 원하는 데이터가 없다면 True 변환 |
| + | 두 개 이상의 리스트를 연결 |
| \* | 반복문 리스트를 만든다. |

파이썬에서는 리스트 자료형을 편하게 다룰 수 있는 다양한 메소드를 제공한다.

인덱싱과 슬라이싱을 이용하면, 메소드 없이도 값을 변경할 수 있다.

이외에도 len(변수명), sum(변수명), min(변수명), max(변수명)과도 유용하게 쓰인다.

| 종류 | 설명 |
| --- | --- |
| 리스트명.append(값) | 리스트 끝에 값 추가 |
| 리스트명.(insert(index,값) | 리스트 중간에 값 삽입 |
| 리스트명.extend(리스트) | 리스트 끝에 리스트 추가 |
| 리스트명.pop(index) | 특정 값 반환 후 삭제 |
| 리스트명.remove(값) | 값을 이용하여 삭제 |
| del 리스트명[index] | 인덱스를 이용하여 삭제 |
| 리스트명.clear() | 전체 값 삭제 |
| 리스트명.index(값) | 특정 항목의 인덱스 찾기 |
| 리스트명.count(값) | 특정 항목의 개수 찾기 |
| 리스트명.copy() | 리스트 통째로 복사하기 |
| 리스트명.sort() | 리스트 오름차순 정렬 |
| sorted(리스트명) | 리스트 원본 손상 없이 새롭게 오름차순 정렬 |
| 리스트명.sort(reverse=True) | 리스트 내림차순 정렬 |
| 리스트명.reverse() | 리스트 역순 정렬 |
| 문자열.split(기준문자) | 기준 문자를 기준으로 리스트로 반환한다. |

파이썬의 리스트는 리스트 안에 for반복문과 if조건문을 사용할 수 있다. 이러한 형태로 리스트를 생성하는 것을 리스트 컴프리헨션, 즉 리스트 표현식이라고 한다,

리스트 표현식에서 if 조건문을 사용하려면 for반복문 다음에 원하는 조건식을 추가해주면 된다.

map함수는 리스트의 요소를 지정된 함수로 처리해주는 기능을 가지고 있다. map함수를 사용할 경우 기존 리스트는 변경되지 않고 새로운 리스트로 생성되어 저장된다.

map함수와 input(, split()함수를 함께 사용하면 여러 개의 정수를 쉽게 리스트 형태로 저장할 수 있다.

아래 코드는 사용자가 숫자를 세 개 입력했을 경우의 코드이다.

map함수가 반환하는 것을 이터레이터이기 때문에 변수 여러 개에 각각의 값을 직접 넣어줄 수 있다.

또한 map함수의 반환값은 이터레이터이기 때문에 리스트 형태로 만들기 위해서는 반드시 list()함수와 함께 사용해주어야 한다.

Z파이썬이나 다른 언어를 사용할 때는 변수 이름을 정확하게, 제대로 지어주는것이 실사용하는되 도움이 된다.



a, b = map(int,input().split())

a=input()

b=input()

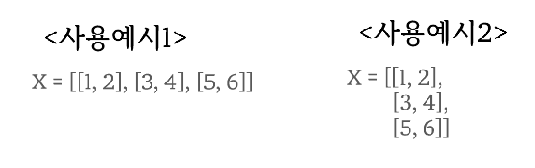
a=int(input()

a=int()

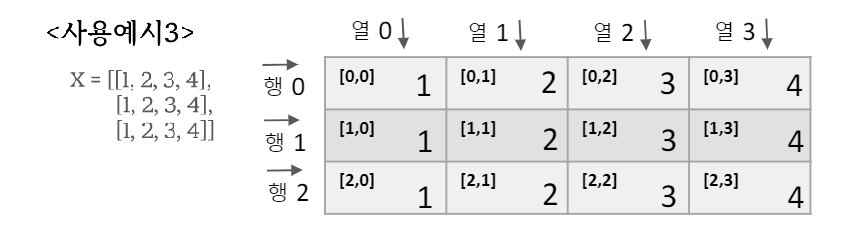
문자열은 항상 쌍따옴표(“)로 감싼다.

2차원 리스트는 리스트 안에 리스트만으로 구성된 리스트를 뜻한다.

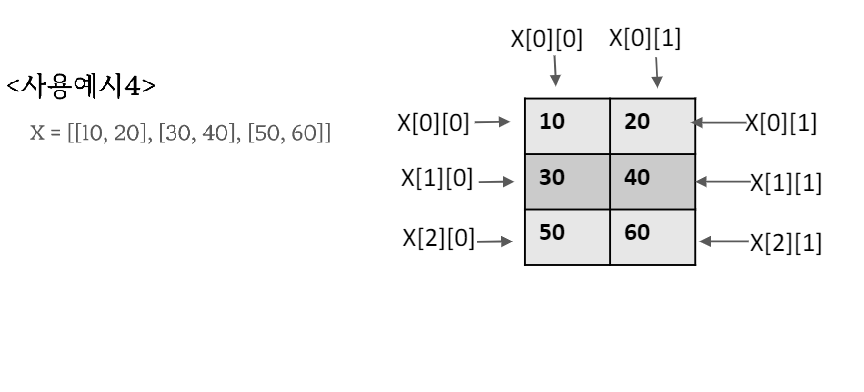
2차원 리스트는 행과 열 (가로x세로) 형태로 이루어져 있기 때문에 가로로 정의할 수도 있지만 행, 열을 구분해서 세로로 정의도 가능하다.



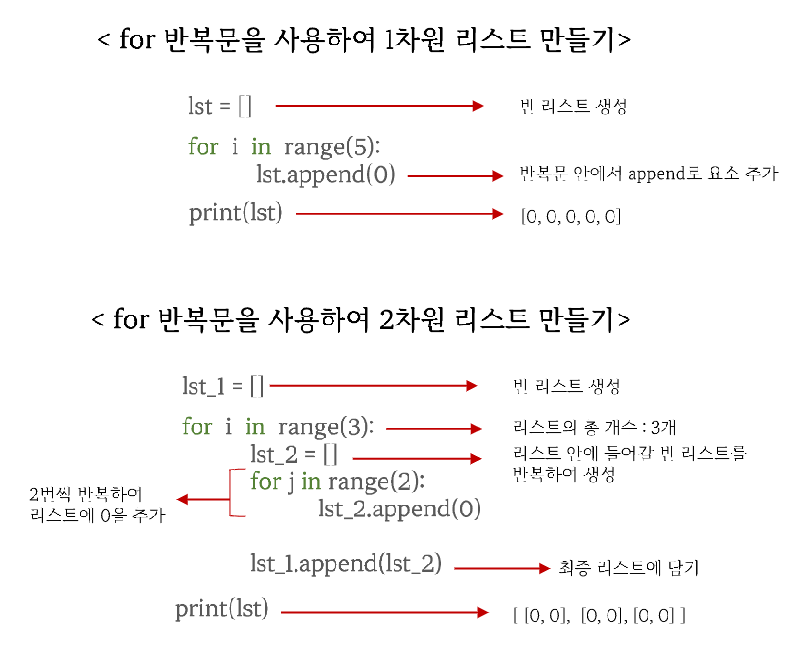
2차원 리스트는 행과 열의 형태로 이해하는 것이 각 요소에 접근하기 편하다.

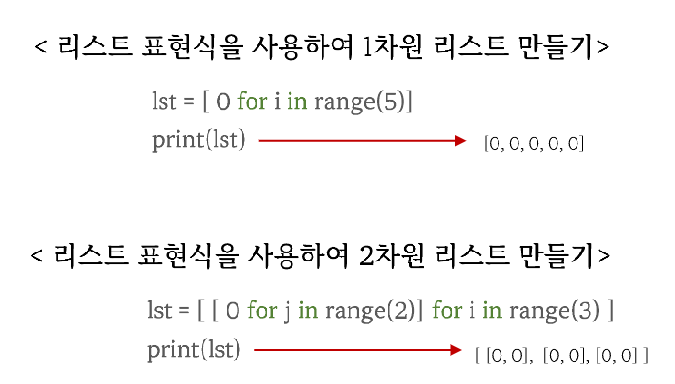


2차원 리스트에서 각 요소에 접근하는 방법은 일반 리스트에서 사용했던 인덱싱 방법과 동일하게 사용하면 된다.

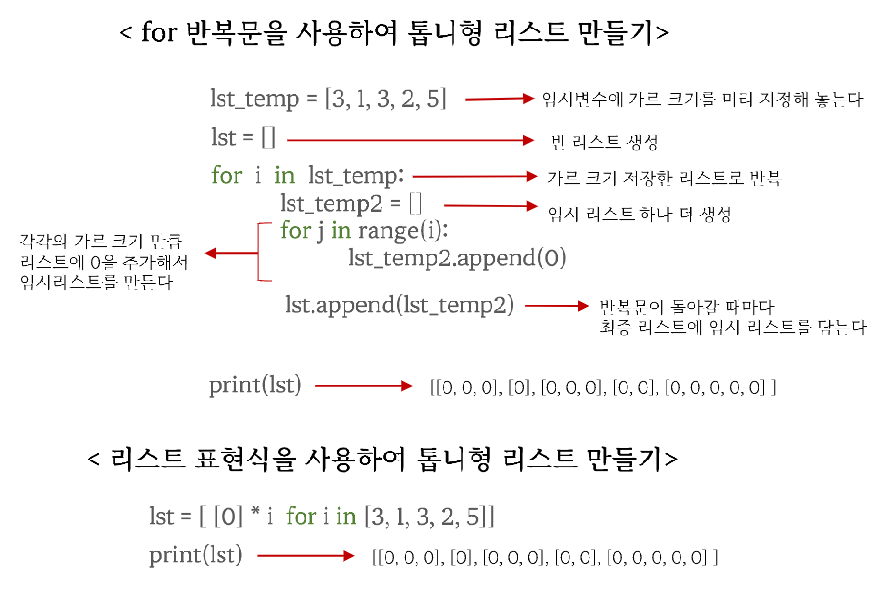


반복문을 이용하면 2차원 리스트를 쉽게 만들 수 있다.

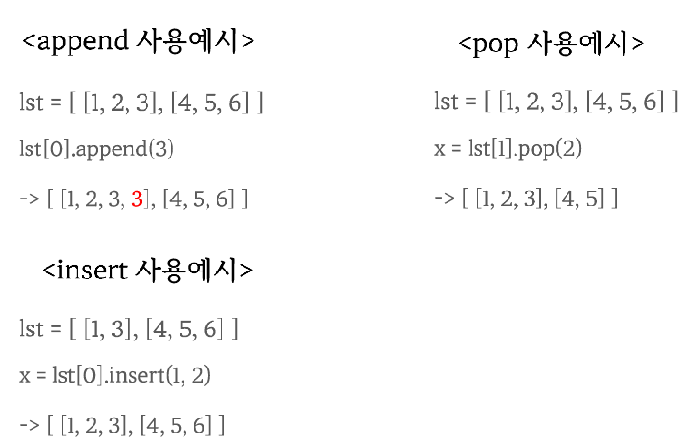
리스트 표현식을 이용하면, 조금 더 간단하게 2차원 리스트를 만들 수 있다.



리스트는 크기가 서로 다른 2차원 리스트도 담을 수 있다. 가로 크기가 불규칙한 리스트를 톱니형 리스트(jagged list)라고 한다.

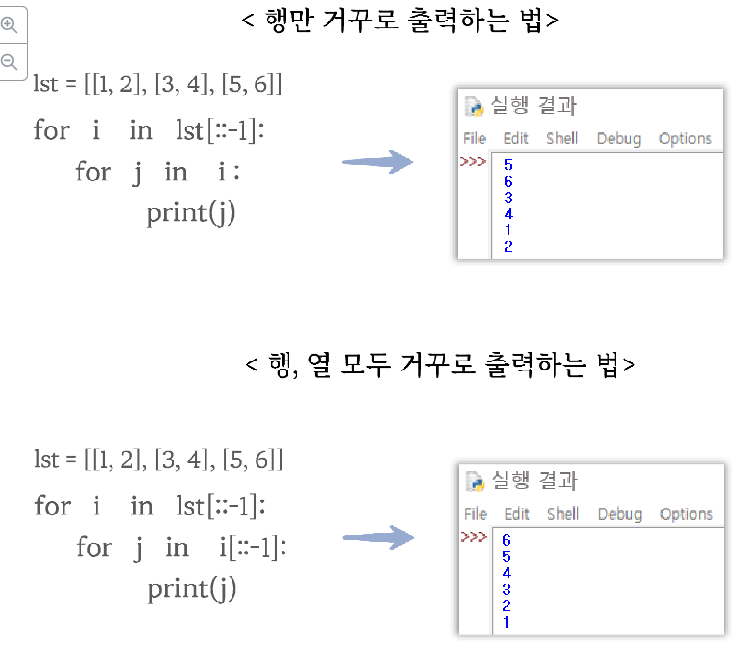


2차원 리스트 이상에서 리스트의 메소드를 사용할 경우에는 아래와 같은 규칙으로 사용하면 된다.



반복문을 이용하면 2차원 리스트의 요소를 출력할 수 있다.

반복문과 슬라이싱 기능을 이용하면 2차원 리스트의 요소를 거꾸로 출력할 수 있다.



list\_sample = [1,2,3,4,5,6,7,8,9,10]

## print(list\_sample[:5]) == print(list\_sample[0:5])

## print(list\_sample[6:] == print(list\_sample[6:10]

파이썬을 포함한 많은 프로그래밍 언어에서 사용되는 함수(Function)는 수학에서의 ‘함수’의 개념과 ‘함수’의 영어 단어인 function이 가지는 ‘기능’이라는 의미를 모두 가진다.

함수는 함수명()과 같은 형태로 쓰이는데 지금까지 우리가 사용해 온 print(), input(), range(), list(), append(), remove() 등은 모두 함수이다. 또한 함수는 사용자가 함수명과 기능을 새롭게 정의해서 사용할 수도 있다.

함수는 함수 정의부와 함수 호출부로 구성되어 있다는 것을 알수 있다.

파이썬에서 사용되는 함수는 크게 두 가지로 나눌 수 있다.

1. 사용자 함수 : 사용자 함수는 사용자가 직접 함수를 정의해서 사용하는 함수이다. 사용자 함수에는 함수의 정의 부분과 함수의 호출 부분이 존재한다.
2. 내장 함수 : 사용자 함수와는 달리 내장 함수에서는 사용자가 직접 함수를 정의할 필요가 없다. 내장 함수는 파이썬 프로그램 설치 시 내장 함수 정의 부분의 코드들이 같이 설치 되기 때문에 사용자가 함수를 별도로 정의할 필요가 없는 것이다.

매개변수는 호출 함수에서 전달하고자 하는 값이나 변수를 전달받기 위해 함수 정의에서 사용되는 변수를 말한다.

매개변수(Parameter) : 호출 함수의 인수를 정의 함수에서 전달받기 위해 사용되는 변수

인수(Argument) : 호출 함수에서 정의 함수의 매개변수에 값을 전달하기 위해 사용되는 데이터나 변수

호출 함수의 인수 개수와 정의 함수에서 사용되는 매개변수의 개수는 같아야 한다.

정의 함수에서 사용되는 매개 변수는 정의된 함수 내에서만 유효하다.

함수 정의를 제외한 부분, 즉 함수를 호출해서 사용하는 프로그램의 메인 부분을 메인 루틴(Main Routine)이라고 한다. 프로그램 시작은 메인 루틴의 제일 앞에서부터 시작된다.

일반적인 함수 정의에서는 매개변수의 개수가 고정된다. 그러나 하나의 함수에서 매개변수의 개수를 고정하지 않고 호출 함수에서 전달하는 인수의 개수에 따라 매개변수를 가변적으로 하고 싶은 경우에 사용하는 것이 매개변수 \*args이다.

\*args에서 반드시 args란 매개변수 이름을 사용할 필요는 없다.

리스트와 튜플 등의 다른 데이터 형도 정의 함수의 매개변수에 전달될 수 있다.

프로그래밍 언어에서 말하는 레퍼런스(Reference)는 데이터가 저장된 메모리 번지의 주소를 의미한다; 쉽게 생각해서 레퍼런스는 데이터가 저장된 메모리 주소라고 생각하면 된다.

정의된 함수에서 얻은 값을 return 문을 이용하여 호출한 함수 측에 반환할 수 있다. 이것을 함수 값의 반환이라고 한다.

사용자 함수를 정의해서 사용하다 보면 정의된 함수에서 사용되는 변수(지역 변수)와 메인 루틴에서 사용하는 변수(전역 변수)간에 충돌이 발생할 수 있다.이런 문제를 미연에 방지하기 위해서 지역 변수와 전역 변수의 개념을 잘 이해하고 프로그램에서 변수를 적절하게 잘 사용해야 한다.

지역 변수(Local Variable)는 정의된 함수 내에서 사용되는 변수를 말한다,

전역 변수(Global Variable)는 메인 루틴의 제일 앞에 정의된 변수를 말한다. 전역 변수는 메인 루틴과 정의된 함수 내에서 사용 가능하다.

메인 루틴에서 정의된 전역 변수와 함수 내에서 정의된 지역 변수는 변수 이름이 같을지라도 전혀 다른 변수로 처리된다.

같은 이름의 변수를 전역 변수와 지역 변수로 사용하여도 전혀 문제가 되진 않지만 두 변수가 유효한 장소가 서로 다르다는 점만 이해하고 있으면 된다.

함수 내에서 전역 변수를 정의하고 싶은 경우에는 키워드 global을 사용하면 된다.

전역 변수는 메인 루틴과 사용자 함수 내부에서도 사용가능하기 때문에 굳이 키워드 global을 사용하지 않더라고 전역 변수로 간주되기 때문이다.

그러나 정의 함수에서 전역 변수를 사용할 경우에는 사용되는 변수가 메인 루틴의 전역 변수라는 것을 명확히 하기 위해 키워드 global을 사용하는 것을 추천한다.

전역 변수를 사용할 경우에는 변수 값이 메인 루틴이나 정의 함수들에서도 변경될 수 있기 때문에 그 값을 추적하기 힘들 때가 종종 있다. 따라서 전역 변수는 꼭 필요한 경우에만 사용하고, 일반적인 경우에는 매개변수나 지역 변수를 사용하는 것이 좋은 프로그래밍 방식이라는 것을 꼭 기억하자.

| 함수 | 의미 |
| --- | --- |
| print() | 데이터나 변수를 화면에 출력함 |
| input() | 키보드로 데이터를 입력 받음 |
| type() | 변수의 데이터 형을 구함 |
| int() | 실수나 문자열을 정수로 변환함 |
| float() | 정수나 문자열을 실수로 변환함 |
| str() | 정수나 실수를 문자열로 변환함 |
| sum() | 리스트나 튜플 요소들의 합을 구함 |
| len() | 문자열, 리스트, 튜플, 딕셔너리의 길이를 구함 |
| range() | 10진수 정수를 2진수로 변환함 |
| list() | 새로운 리스트를 생성함 |
| tuple() | 새로운 튜플을 생성함 |
| dict() | 새로운 딕셔너리를 생성함 |
| ord() | 문자의 아스키(ASCII)코드 값을 구함 |
| bin() | 10진수 정수를 2진수로 변환함 |
| hex() | 10진수 정수를 16진수로 변환함 |
| round() | 반올림 값을 구함 |
| max() | 리스트나 튜플의 최댓값을 구함 |
| min() | 리스트나 튜플의 최솟값을 구함 |

키보드로 입력 받는 모든 데이터(숫자, 문자, 특수문자 등)는 문자열로 처리된다.

문자열을 저장하는 데 사용되는 컴퓨터 코드(0과 1의 조합)가 아스키 코드이다. 아스키(ASCII, American Standard Code for Information Interchange)는 영문 알파벳, 숫자, 특수문자 등에 사용되는 대표적인 문자 인코딩(Encoding)방식이다.

min() 함수는 최솟값을 구하는 데 사용된다.

min()함수의 사용법은 max() 함수와 동일하다.

선형 탐색(Linear search)은 다른 말로 순차 탐색(Sequential search)이라고도 한다. 선형 탐색에서는 찾고자 하는 요소와 리스트에 있는 요소들을 순차적으로 하나씩 비교해 나간다.

이진 탐색(Binary search)에서는 요소들이 오름차순 또는 내림차순으로 정렬되어 있어야 한다. 만약 리스트가 정렬되어 있지 않다면 이진 탐색을 적용하기 전에 리스트를 사전에 정렬해 놓아야 한다. 이진 탐색에서는 지속적으로 탐색 범위를 1/2씩 줄여가면서 해당 값을 찾는다.

이진 탐색을 이용하면 선형 탐색을 사용했을 때와 비교하여 계산량이 현저히 줄어들게 된다.

프로그래밍을 할 때 프로그램이 복잡해지고 소스 파일의 개수가 많아진 경우 하나의 파일을 여러개로 나누어 관리하면 상당히 편리해 진다. 이 때 사용하는 것이 파이썬의 모듈(Module)이다.

파이썬 모듈에서는 프로그램에서 공통적으로 사용되는 함수들을 모아서 별도의 파일에 저장하는 기능을 제공한다. 사용자는 프로그램 작성 시 필요한 모듈 파일을 불러와서 파일 내에 정의된 함수들을 쉽게 이용할 수 있다.

프로그램에서 불러오려면 import 명령을 사용한다.

import 명령을 이용하여 모듈을 불러오는 방법은 다음의 세 가지로 분류할 수 있다.

1. import ~ 구문
2. import ~ as ~ 구문
3. from ~ import ~

import ~ 구문을 사용해서 모듈을 불러오면 모듈명 다음에 점(.)을 붙인 다음 해당 모듈 함수를 사용하면 된다.

import 모듈명은 모듈명.py 파일에 정의되어 있는 변수, 함수 등을 불러온다. 그리고 프로그램 내에서 모듈 함수를 호출할 때에는 모듈명.모듈함수명()을 사용한다,

import ~ as ~ 구문에 의해 별칭이 사용되면 별칭.모듈함수명()의 형태로 모듈함수를 이용할 수 있다.

from ~ import ~ 구문으로 모듈함수명을 직접 불러다 사용하면 모듈함수명 앞에 아무것도 붙이지 않고 모듈함수명을 그대로 사용한다.

파이선에서는 수학적 연산을 위해 내장 모듈(Built-in Module)인 math 모듈을 제공한다.

| 모듈 함수 | 의미 |
| --- | --- |
| math.sin() | sin() 값을 반환함 |
| math.cos() | cos() 값을 반환함 |
| math.tan() | tan() 값을 반환함 |
| math.ceil() | 실수 값을 무조건 올림한 정수 값을 반환 |
| math.floor() | 실수 값을 무조건 내림한 정수 값을 반환 |
| math.fsum() | 리스트, 튜플 등의 합계를 구함 |
| math.log() | 자연 로그 값을 반환 |
| math.log10() | 밑을 10으로 한 로그 값을 반환 |
| math.pow() | 거듭 제곱 값을 반환 |
| math.sqrt() | 제곱근(Square Root) 값을 반환 |
| math.pi | 원주율 |

이외에 더 많이 있다,

math.sin(x) 함수는 x의 싸인(Sine) 값을 구하는 데 사용된다.

math.cos(x) 함수는 x의 코사인(Cosine) 값을 반환한다.

math.tan(x) 함수는 x의 탄젠트(Tangent) 값을 반환한다.

math.ceil(x) 함수는 x보다 작지 않은 최대 정수 값을 반환한다. 즉, 무조건 올림한 정수 값을 얻는다.

math.floor(x) 함수는 x보다 크지 않은 최소 정수 값을 반환한다. 즉 무조건 내림한 정수 값을 얻게 된다.

round(x) 함수는 x를 반올림한 정수 값을 반환한다.

፠ round() 함수는 파이썬 자체의 내장 함수이기 때문에 math 모듈이 필요하지 않다.

math.fsum(x) 함수는 리스트나 튜플 내에서 같은 데이터 형을 가진 x에 있는 요소들의 합계를 반환한다.

◻ math.fsum() 함수는 합계를 계산할 때 실수(Floating point)를 기본으로 한다. 따라서 math.fsum() 함수의 실행 결과는 모두 실수형으로 표시된다.

math.log(x)와 math.log10(x)는 각각 x의 자연 로그와 밑이 10인 로그의 값을 반환한다.

math.pow(x,y) 함수는 x의 y승, 즉 x\*\*y의 값을 반환한다.

math.sqrt(x) 함수는 x의 제곱근 값을 반환한다.

파이썬에서는 컴퓨터가 가지고 있는 시계를 이용하기 위한 time 모듈을 제공한다.

| 함수 | 의미 |
| --- | --- |
| time.time() | UTC 표준 시를 기준으로 한 현재 시간 구함 |
| time.gmtime() | UTC 초 단위의 시간을 struct\_time 구조로 변환함 |
| time.localtime() | 현지 시간(Local Time)을 구함 |
| time.ctime() | UTC 초 단위의 시간을 문자열로 변환함 |
| time.strftime() | 일시를 포맷 기호를 이용하여 특정 포맷으로 변환함 |
| time.sleep() | 일정 시간만큼 지연시킴 |

컴퓨터에서 현재 시간을 얻기 위해서는 time 모듈의 time() 함수를 이용한다.

time.time()함수는 UTC(GMT+0) 기준으로 19070년1월1일0분0초로부터 경과된 시간을 구하는 데 사용된다. 이와 같이 시간을 계산하는 것을 타임스탬프(Timestamp)라고 한다. time.time()의 실행 결과에 나타난 숫자가 현재 시간에 대한 타임스탬프이다.

GMT(Greenwich Mean Time)는 그리니치 표준 시라고 하는데 영국 런던에 소재한 그리니치 천문대를 기준으로 한 시간을 의미한다. UTC는 GMT의 기점, 즉 GMT+0를 나타내고, 우리나라(서울) 시간의 기준은 GMT+9, 즉 GMT 보다 9시간 늦은 시간이 된다.

time.gmtime()은 타임스탬프를 struct\_time 구조로 하여 보여준다.

타임스탬프를 현지 시간으로 변경하는 데에는 time.localtime() 함수를 이용한다.

타임스탬프를 간단하게 문자열로 변환하여 시간을 알아보기 위해서는 time.ctime() 함수를 이용하면 된다.

time.strftime() 함수를 이용하여 현재 시간을 특정 포맷의 문자열로 변환할 수 있다.

| 포맷 기호 | 의미 |
| --- | --- |
| %Y | 네 자리 년도 |
| %y | 두 자리 년도 |
| %m | 월 |
| %d | 일 |
| %A | 요일 |
| %a | 간략 요일 |
| %H | 시(24시) |
| %l | 시(12시) |
| %p | AM 또는 PM |
| %M | 분 |

time.sleep() 함수는 프로그램의 실행을 일정 시간 지연시키고 싶은 경우 사용한다.

| 함수 | 의미 |
| --- | --- |
| datetime.timedelta() | 일시에 대한 산술 연산이 가능하게 포맷 변환 |
| datetime.date.today() | 오늘의 날짜 구함 |
| datetime.datetime.now() | 현재의 날짜와 시간 구함 |
| datetime.datetime.strftime() | 날짜와 시간을 포맷 기호로 특정 포맷으로 변환 |

프로그래밍을 하다보면 날짜와 시간에 대한 산술 연산이 필요한 경우가 생긴다. 이 때 사용하는 것이 datetime.timedelta() 함수이다.

datetime.timedelta() 함수는 기간을 컴퓨터 내부 시간 포맷인 마이크로 초로 일시를 저장한다. 이를 이용하여 3일 3시 30분과 5일 5시 40분 간의 시간에 대한 차이를 구한다. 이와 같이 datetime.timedelta()를 이용하면 날짜와 시간 간에 서로 덧셈이나 뺄셈 등의 산술 연산이 가능하게 된다.

datetime 모듈에 있는 date.today() 함수를 이용하면 컴퓨터가 가지고 있는 오늘의 날짜를 얻을 수 있다.

파이썬의 random 모듈은 파이썬 설치 시 기본적으로 설치되는 내장 모듈로서 임의의 수를 발생시키거나 리스트의 요소 중 임의의 수를 선택하는데 사용된다.

random 모듈은 게임, 시뮬레이션, 테스팅, 보안 등의 프로그램을 개발할 때 주로 사용된다.

random 모듈에는 다양한 모듈 함수가 존재한다.

| 함수 | 의미 |
| --- | --- |
| random.random() | 0에서 1 사이의 임의의 실수를 반환함 |
| random.uniform() | 주어진 두 수 사이의 임의의 실수를 반환함 |
| random.randint() | 주어진 영역 사이의 임의의 정수를 반환함 |
| random.choice() | 리스트, 튜플, 문자열 등에서 임의로 선택한 요소를 반환함 |
| random.shuffle() | 리스트를 임의의 순서로 섞음 |

random.random에서 0과 1사이의 범위는 0은 포함되고 1은 포함되지 않는다.

random.uniform(x,y)에서 x는 범위에 포함되고 y는 포함되지 않는다.

random.randint(x,y)에서 x와 y는 모두 범위에 포함된다.

텍스트 파일(.txt)은 메모장과 같은 텍스트 에디터에서 글자를 저장하는 기본 포맷이다.

open() 함수의 파일 모드

| 파일 모드 | 의미 |
| --- | --- |
| “r” | 읽기 모드, 파일을 읽을 때 사용 |
| “w” | 쓰기 모드, 파일에 데이터를 저장할 때 사용  ፠파일이 존재하지 않으면 새로운 파일을 생성하고, 해당 파일이 존재하면 기존 파일에 덮어씀 |
| “a” | 추가 모드, 기존에 파일에 데이터를 추가할 때 사용 |

텍스트 파일에서 사용 가능한 모든 문자를 인코딩, 즉 각 문자에 대해 코드를 부여하는 것을 유니코드(Unicode)라고 한다.

UTF-8 인코딩 방식은 유니코드 인코딩에서 일반적으로 널리 사용되는 세계 표준 방식 중의 하나이다. UTF-8에서는 유니코드 문자열을 8비트 단위와 코드로 매핑하게 된다.

파이썬에서 텍스트 데이터를 다루는 경우에 있어서 줄 바꿈을 할 때 사용하는 것이 개행 문자 “\n”이다. 개행 문자는 다른 말로 라인 피드(Line Feed)라고 부른다.

파이썬에서 파일을 삭제하는 데에는 OS 모듈과 os.remove() 함수를 이용한다.

폴더를 삭제하는 것은 파일 삭제 방법과 유사한 방법으로 하면 되는데, os.remove() 대신에 os.mdir() 함수를 사용한다.

CSV(Comma Separated Values)는 다음과 같이 콤마로 구분된 텍스트로 구성된 파일을 말한다. CSV파일은 기상청, 통계청, 행정 기관 등에서 제공하는 공공 데이터의 기본 포맷이기도 하다.

리스트(또는 튜플 등)에서 데이터를 가져오는 next() 함수의 사용 서식은 next(반복 객체)이다.

리스트, 튜플 등 여러 개의 요소를 가진 객체를 반복 객체(Iterable Object)라고 한다. next() 함수는 반복 객체에서 요소를 하나씩 가져오는 데 사용된다.

JSON은 ‘JavaScript Object Notation’의 약어로 자바스크립트 문법에 기반한 데이터의 표현 방식이다. JSON은 그 구조가 단순하고 유연하기 때문에 시스템 간에 데이터를 교환 할 때 많이 사용한다. 특히 웹 브라우저와 웹 서버 사이에 데이터를 교환하는 데 많이 사용된다.

파이썬에서는 JSON을 다룰 수 있는 표준 라이브러리를 제공하고 있기 때문에 파이썬의 리스트, 튜플, 딕셔너리 등의 데이터를 JSON으로 변환(JSON 인코딩) 할 수 있다. 반대로 JSON 타입의 데이터를 파이썬의 데이터 형으로 쉽게 변환(JSON 디코딩)할 수도 있다.

예외(Exception)는 작성한 프로그램을 실행했을 때 예기치 않게 발생되는 오류와 같은 상황을 의미한다. 파이썬에서는 이러한 예외, 즉 오류에 대처하고 처리하는 방법을 제공한다,

파이썬에 예외 처리를 위해 다음의 세 개의 블록이 존재한다.

1. try 블록 : 오류가 발생하는지를 테스트하는 블록
2. except 블록 : 오류를 처리하는 블록
3. finally 블록 : try와 except 블록이랑 상관없이 무조건 실행되는 블록

객체지향 프로그래밍(Object-Oriented Programming)은 소프트웨어를 만드는 가장 효과적인 방법 중의 하나이다. 객체지향에서는 클래스(Class)를 정의하는 것부터 시작한다. 클래스가 정의되면 객체(Object)를 생성하고 이 객체를 이용하여 프로그램을 작성해 나간다.

클래스(Class)는 객체지향 프로그래밍(Object-Oriented Programming)의 핵심 요소 중 하나이다. 클래스를 이용하면 복잡한 프로그램을 좀 더 쉽고 체계적으로 작성하고 관리 할 수 있다.

클래스는 속성(Attribute)와 메소드(Method)로 구성된다.

클래스를 만들어 놓으면 필요할 때 언제든지 객체를 쉽게 생성할 수 있다.

◾ 클래스 : 속성과 메소드로 구성되는 객체 생성에 사용되는 툴이다.

◾ 객체 : 클래스로부터 생성되며 해당 클래스의 속성과 메소드를 가진다.

◾ 속성 : 클래스와 객체 내부에서 사용되는 변수를 의미한다.

◾ 메소드 : 클래스와 객체 내부에서 사용되는 함수를 의미한다.

클래스의 메소드에서 사용되는 매개변수 self는 객체에서 메소드를 호출할 때 해당 객체를 전달받는 데 사용된다.

객체는 다른 말로 인스턴스(Instance)라고 부르기 때문에 self.name과 같은 속성을 인스턴스 속성이라고 한다.

클래스로부터 생성되는 것을 객체라고 하고, 그 객체가 실제 컴퓨터 메모리에 할당되어 사용될 때의 객체를 인스턴스(Instance)라고 부른다.

생성자(Constructor)는 \_\_init\_\_() 메소드를 의미하는데 이것은 객체를 생성할 때 자동으로 호출되어 객체를 초기화하는 데 사용된다.

클래스에서 사용되는 속성(Attribute)에는 클래스 속성(Class attribute)와 인스턴스 속성(Instance attribute)이 있다. 클래스 속성은 클래스에 속해 있는 속성으로 파생된 객체에서 그 값이 유효하다. 그러나 인스턴스 속성은 해당 인스턴스, 즉 객체에서만 그 값이 유효하다.

클래스 속성은 말 그대로 클래스에 소속되어 있기 때문에 해당 클래스에서 파생된 하위 객체들에서 속성 값이 공유된다.

인스턴스 속성은 객체마다 속성 값이 다르다

클래스에서 상속(Inheritance)은 다른 클래스에 있는 속성과 메소드를 상속받아 클래스를 정의할 수 있는 기능을 제공한다. 상속을 해주는 클래스를 부모 클래스(Parent Class)라 하고 부모 클래스로부터 상속을 받는 클래스를 자식 클래스(Child Class)라 한다.

super()는 부모 클래스에서 정의된 생성자를 가져와 자식 클래스에 포함시키는 역할을 수행한다.

부모 클래스와 자식 클래스에 동일한 이름의 메소드가 존재하면 자식 클래스에 있는 메소드가 부모 클래스에 앞서 동작하게 된다. 이것을 메소드 오버라이딩(Method Overriding)

이라고 한다.

부모 클래스와 이름이 같은 메소드를 자식 클래스에서 재정의하면 자식 클래스의 메소드가 부모 클래스의 메소드에 우선하여 실행된다. 이것을 메소드의 오버라이딩(Overriding)이라고 한다.

파이썬의 CSV 모듈에는 데이터를 읽어오기 위한 reader() 함수와 데이터를 저장하기 위한 writer() 함수가 있다.

๏ csv.reader() : CSV 파일에서 데이터를 읽어오는 함수

๏ csv.writer() : CSV 파일에 데이터를 저장하는 함수

주피터 노트북을 데이터분석을 위해서 사용하는 경우에는 주피터 노트북 파일을 저장할 때 CSV 파일을 저장한 곳과 같은 폴더에 저장해야 한다.

for 반복문을 사용하면 CSV 파일에 저장된 데이터를 한 줄씩 읽어올 수 있다.

데이터는 완전무결한 것이 아니라는 점을 꼭 기억하여 데이터를 맹신하지 않는다.

헤더(header)는 데이터 파일에서 여러 가지 값들이 어떤 의미를 갖는지 표시한 행을 말한다.

많은 경우, 헤더는 데이터의 첫 번째 줄의 위치하여 두 번째 줄부터 나타나는 데이터의 속성을 설명한다.

헤더를 통해 각 열의 데이터가 어떤 의미를 갖는지 알 수 있기 때문에 매우 중요한 데이터이다.

헤더를 별도로 저장하려면 next() 함수를 사용할 수 있다.

데이터분석의 대부분의 문제 해결 방법:

1. 데이터를 읽어온다.
2. 순차적으로 확인하고 싶은 데이터를 확인한다.
3. 그 데이터의 저장하고 싶은 데이터(예를 들어 가장 높은, 가장 작은 등)을 저장한다.
4. 최종 저장된 데이터를 출력한다.

탐색과 비교의 과정의 예시:

1. 기준이 되는 값을 설정한다.
2. 기준값과 새로운 값을 비교한 후, 기준값보다 더 큰 값이 나타나면 기준값을 업데이트한다.(이 과정에서 ‘기준이 되는 값’을 저장할 공간인 ‘변수’가 필요하다.

matplotlib의 pyplot 에서 plot()함수는 직선 또는 꺾은선 형태의 그래프를 그릴 때 사용할 수 있는 명령어이다.

plot() 함수에 입력된 리스트의 값이 y축의 값으로 입력되며, x축 값은 자동으로 0부터 1씩 증가하는 정수로 입력되는 것을 알 수 있습니다.

plot() 함수에서 x축 데이터와 y축 데이터의 개수가 맞지 않으면 다음과 같은 에러가 발생합니다.

plot() 함수에 두 개의 리스트를 입력했을 때는 첫 번째 리스트가 x축 값이고, 두 번째 리스트가 y축 값으로 입력된다.

plot() 함수에 기본 그래프를 그리려면 보통 다음 세 단계를 거친다.

1. import matplotlib.pyplot as plt 혹은 from matplotlib import pyplot as plt로 라이브러리 불러오기(plt로 별명을 지정하는 것은 선택이나 거의 필수라는 불문율이 있는게 함정이다.)
2. plt.plot([x축 데이터],[y축 데이터]):plot() 함수에 데이터 입력하기
3. plt.show(): 그래프 보여주기

그래프에 제목을 넣는 함수는 title() 함수이다. 따라서 plot.title(“제목에 넣을 문자열”)과 같이 명령을 입력하면 제목이 보인다.

그래프가 의미하는 바를 구별할 때 보통 범례(legend(를 넣는다.

범례는 보통 두 개 이상의 데이터를 표시할 때 사용한다.

범례를 넣기 위해 plot() 함수를 사용는데 plot() 함수에 label이라는 속성의 레이블 값으로 원하는 문자열을 넣어주고, 그래프를 그리기 전에 legend()함수를 실행시키면 레이블 값이 범례로 나타난다. 두 가지 범례를 넣으려면 plot() 함수를 두 개 사용하면 된다.

범례의 위치는 자동으로 결정되지만, 직접 지정할 수도 있습니다. 다음과 같이 legend() 함수안의 loc이라는 속성에 값을 넣어\주면 된다. 이때 loc에 들어갈 숫자는 원하는 위치에 따라 0~10까지 넣을 수 있다.

matplotlib 라이브러리에서는 그래프의 색상을 직접 지정할 수 있다. 원하는 색상으로 바꾸려면 color 속성을 간단히 추가하면 된다. color 속성을 생략하면 자동으로 색상으로 설정된다.

plot() 함수는 기본적으로 직선으로 그래프를 그린다. 그래프의 선 모양(line style)을 다양한 형태로 바꾸고 싶을 때는 linestyle 속성에 원하는 선 모양을 지정하면 된다.이때 linestyle 속성 대신 ls라고 작성할 수도 있다.

색과 선 모양을 ‘(색상)(선모양)’을 동시에 적는 형태로 코드를 작성할 수도 있다.

plot() 함수에 marker 속성을 설정하면 선 형태가 아닌 점 형태로 그래프를 그릴 수 있다. 이때 색상의 마커(marker) 모양을 한번에 설정할 수도 있다. 마커 모양으로는 섬, 점, 동그라미, 삼각형, 별 모양 등 여러 가지가 있다.

마찬가지로 색상과 선 모양과 마커 모양을 동시에 설정하고 싶을 때는 ‘(색상)(마커모양)(선모양)’순으로 코드를 작성할 수도 있다.

그래프 크기를 조정하기 위해서는 figure() 함수의 figsize 속성 값을 변경하여 그래프의 크기를 조절할 수 있다. figsize=(가로 길이, 세로 길이) 형식으로 크기를 설정할 수 있다.

단위는 인치이다. 참고로 1인치는 2.54 센티미터이다.

데이터는 보는 관점에 따라 다르게 해석될 수 있습니다.

히스토그램은 자료의 분포 상태를 직사각형 모양의 막대 그래프로 나타낸 것으로, 데이터의 빈도에 따라 높이가 결정된다. hist() 함수를 사용하면 데이터를 히스토그램을 그릴 수 있다.

주사위 시뮬레이션은 다음과 같은 과정으로 진행된다.

1. 주사위를 굴린다.
2. 나온 결과를 기록한다
3. 1~2 과정을 n번 반복한다.
4. 주사위의 눈이 나온 횟수를 히스토그램으로 그린다.

실험 또는 관찰에 의해 얻은 통계 비율을 ‘통계적 확률’이라 하고, 각각의 사건이 일어날 가능성이 모두 같다고 가정하여 얻은 통계 비율을 ‘수학적 확률’이라고 합니다. 주사위를 던지는 횟수를 늘릴수록 특정 숫자가 나오는 횟수가 전체의 1/6에 가까워진다고 예상할 수 있다. 이러한 법칙을 ‘큰 수의 법칙’이라고 한다.

상자 그림은 가공하지 않은 자료를 그대로 이용하는 것이 아니라, 자료에서 얻어낸 최댓값, 최솟값, 상위 ¼, 2/4(중앙), 3/4에 위치한 값을 보여주는 그래프이다.

파이썬의 in 연산자를 사용하면 조금 더 효율적으로 코드를 작성할 수 있다.

‘A in B’는 A가 B 안에 존재하면 참, 그렇지 않으면 거짓이 된다. 즉 똑같은 값이 있는지 찾아서, 그 결과를 True 또는 False로 반환한다. 한 글자라도 다르면 해당 값이 없다고(False) 판단한다.

bar() 함수는 막대그래프를 표현하는 명령어이다. 막대그래프에서 막대의 길이는 각 데이터의 크기를 의미한다.

bar() 함수에는 두 가지 값이 입력된다. 첫 번째는 막대를 표시할 위치이고, 두 번째는 막대의 높이이다. 그리고 이 두 종류의 값들은 개수가 일치해야 한다.

bar(막대를 표시할 위치, 막대의 높이)

range() 함수를 사용하여 막대그래프의 위치를 표현할 수도 있다.

barh() 함수는 막대그래프를 수평 방향으로 그린다.

산점도는 가로축과 세로축을 기준으로 두 요소가 서로 어떤 관계를 맺고 있는지 파악하기 쉽게 나타낸 그래프이다.

산점도는 scatter() 함수를 사요하여 표현할 수 있다. 또한 버블 차트를 그릴 때도 사용할 수 있다. 함수로 버블 차트를 나타내려면 기존 코드에서 size를 의미하는 s 속성을 추가하고 원하는 크기를 입력한다.

colorbar() 함수를 사용하면 그래프 옆에 컬러바를 추가할 수 있다.

scatter() 함수에 c 속성을 추가하여 표현하고 싶은 색상의 개수를 설정하면 각 데이터에 해당하는 컬러바의 색으로 정해진다.

cmap이라는 컬러맵 속성을 사용하면 컬러바에 사용될 색상의 종류를 정할 수 있다.

matplotlib 라이브러리에서 사용할 수 있는 다양한 컬러맵의 종류 하이퍼링크:

<https://matplotlib.org/tutorials/color/colormaps.html> (라고 하는데 404 에러) (대체 주소는 여기로 : <https://matplotlib.org/stable/tutorials/colors/colormaps.html>)

alpha 속성으로 투명도를 조절할 수 있다. alpha 속성값의 범위는 0부터 1까지이며, 0에 가까울수록 투명하고, 1에 가까울수록 불투명하다.

그래프를 이미지로 저장하려면 savefig() 함수를 사용하면 되는데, 이때 이미지 파일의 이름과 형식을 지정할 수 있다.

ndarray에서 nd는 N - Dimensional, 즉 ‘N차원’이라는 의미이고 array는 ‘배열’이라는 의미이다.

numpy 라이브러리에서는 ndarray 라는 특별한 데이터 타입의 배열이 사용된다.

배열을 생성하기 위해 꼭 리스트가 있어야 하는 것은 아니다.

arange() 함수에 값을 하나만 입력할 경우, 0부터 해당 숫자보다 1만큼 작은 정수까지 저장된 배열이 만들어진다. 값을 두 개 입력할 경우, 첫 번째 숫자부터 두 번째 숫자보다 1만큼 작은 정수까지 저장된 배열이 만들어진다. 값을 세 개 입력할 경우, 첫 번째 숫자부터 두 번째 숫자보다 1만큼 작은 정수까지 범위에서 세 번째 숫자만큼의 간격을 둔 숫자가 저장된 배열이 만들어진다.

또 다른 숫자를 생성하는 방법으로 numpy의 linspace() 함수를 사용할 수 있다.

numpy의 linspace() 함수는 특정 구간을 쪼개어 값을 생성한다는 점에서 arange() 함수와 비슷하다. 하지만 arange() 함수에서는 간격을 지정하고 linspace() 함수에서는 구간의 개수를 지정한다는 점에서 차이가 있다.

마스크(mask) 기능은 어떤 조건에 부합하는 데이터만 선별적으로 저장하기 위한 기능이다.

numpy에서는 배열 전체에 조건을 적용해서 배열에서 조건에 부합하는 데이터는 True, 그렇지 않은 데이터는 False인 마스크가 생성된다.

pandas는 주로 pd라는 별명을 사용한다.

read\_html() 함수는 웹 페이지에서 테이블 형태의 데이터를 추출하는 함수이다.

iloc은 데이터의 순서에 따라 접근하는 인덱스 방식인데, 콤마를 중심으로 앞은 행, 뒤는 열에 접근하는 것이다.

sort\_values에서 ascending 속성을 변경함으로써 정렬할 수 있다. 만약 오름차순으로 정렬하고 싶으면 False를 True로 바꾸면 된다.

pandas에서 불러온 표를 엑셀 파일로 저장하려면 to\_excel()함수를 사용하고, 괄호 안에는 파일이름(xlsx확장자까지 넣어서)을 지정한다,

pandas는 panel datas(패널 자료)의 약자로 파이썬을 활용한 데이터 분석에서 가장 많이 활용되는 라이브러리이다.

1차원 배열 형태의 데이터 구조를 Series라고 부르고, 2차원 배열 형태의 데이터 구조를 DataFrame이라고 부른다.

df라는 변수는 DataFrame이라는 뜻으로 사용된다.

2차원 데이터일 경우 pandas에서는 기본적으로 행 방향을 축으로 계산합니다. 하지만 경우에 따라 열 방향을 축으로 계산하기도 한다. 이때 열 방향을 축으로 계산하려면 axis를 1로 설정하면 된다.(axis=0은 행 방향을 나타내며 0이 기본 값이다.)

^ 사실은 numpy 기능인데 pandas에서도 된다고 한다.

head() 함수는 많은 데이터 중 처음 5개의 데이터만 확인하고 싶을 때 사용하는 함수로 많은 양의 데이터의 형태를 확인할 때 유용하다.

데이터 프레임을 CSV 파일로 저장하려면 to\_csv()함수를 쓰며 to\_excel() 함수와 사용법이 같다.

pandas에서 csv 데이터를 읽어오려면 read\_csv() 함수를 활용하면 된다.

df.index.str.contains() 함수는 데이터 프레임의 인덱스 문자열에 원하는 문자열이 포함된 행을 찾아낸다.

pandas에서는 기본적으로 데이터 정렬은 본질적이다. 라벨과 데이터 간의 연결은 사용자가 명시적으로 해제하지 않는 한 해제되지 않는다.

%matplotlib inline을 설정하면 matplotlib.pyplot의 show 함수를 호출하지 않아도 주피터 노트북 안에서 그래프가 표시된다.

DataFrame의 loc속성을 이용하면 특정 인덱스의 데이터만 가져올 수 있다.(loc 속성은 label-location의 약자로 레이블을 지정해 특정 인덱스의 데이터만 가져오는 데 쓰인다.)

os모듈은 Operating System의 약자로서 운영체제에서 제공되는 여러기능을 파이썬에서 수행할 수 있게 해준다.

현재 경로를 구하려면 os모듈의 getcwd 함수를 사용하면 된다.

listdir 함수의 인자로 경로를 전달하는 경우 해당 경로에 존재하는 파일과 디렉터리 목록을 구할 수 있다.

plt.subplots()을 통해 한 사진에 그래프를 몇 개 그릴지 정할 수 있다.

파이썬 패키지 관리가 어려운 이유는 다음과 같다.

* 1. 의존성 관리: 파이썬은 수많은 패키지가 존재하며, 각 패키지는 독립적으로 의존성을 가지고 있다. 이러한 의존성을 관리하고 해결하는 것은 복잡할 수 있다. 특히, 서로 다른 패키지가 동일한 의존성의 충돌하는 버전을 요구하는 경우가 있을 수 있다.
* 호환성 문제: 일부 패키지는 서로 또는 특정 버전의 파이썬과 호환되지 않을 수 있다. 이로 인해 패키지를 함께 설치하거나 사용할 때 충돌과 오류가 발생할 수 있다.
* 설치 및 설정: 패키지를 설치하는 과정은 가상 환경 구성, 다양한 설치 도구 (pip, conda)와 시스템 종속성 해결 등 여러 단계를 필요로 한다. 이는 초보자나 패키지 관리 프로세스에 익숙하지 않은 사용자에게는 혼란스러울 수 있다.
* 버전 관리 및 업데이트: 패키지는 시간이 지남에 따라 발전하며, 새로운 버전은 변경 사항을 도입하거나 특정 기능을 폐기할 수 있다. 패키지 버전을 관리하고 다른 패키지나 기존 코드베이스와의 호환성을 유지하는 것은 복잡할 수 있다. 특히, 대규모 프로젝트에서 작업하는 경우에는 더욱 어려울 수 있다.
* 제한된 표준화: 파이썬은 내장 패키지 관리자인 pip를 제공하지만, 생태계에서는 엄격한 표준화가 부족하다. 서로 다른 패키지는 설치 지침이 다를 수 있으며, 버전 관리 방식이 다를 수 있고, 의존성 관리 방법이 다를 수 있다.

하지만 파이썬 패키지 관리를 개선하기 위해 많은 노력이 기울여져 왔다. pip, conda와 같은 도구들은 패키지 관리 과정을 상당히 단순화시켜 주었다. 또한, PyPI (Python Package Index)와 Conda Forge와 같은 온라인 패키지 저장소는 다양한 패키지를 미리 빌드하여 필요한 의존성을 찾고 설치하는 것을 용이하게 해주고 있다.