PYTHON for 데이터 분석

2019. 03

Index

❑ comments(주석)

- 코멘트(주석)란?
  - 프로그램의 의도와 기능을 잘 이해하기 위해 코드를 읽는 사용자를 위한 설명이다
- single line comments : #
  - 코멘트 하기를 원하는 줄 앞에 '#'를 사용
  - '#' 다음에는 한 칸을 띄우는게 일반적이다
  - 실행하지 않을 코드 앞에도 "#'를 넣는다
  - '#'는 한 줄 단위로 동작한다
- multi line comments : """ 설명 """
  - 삼중 따옴표 세트 안에 주석을 래핑하여 여러 줄 문자열을 사용하는 방법

TIP) 여러줄을 한꺼번에 주석처리하는 단축기

* select the desired code and press Ctrl+/ on PC, or Cmd+/ on Mac

```
# 한줄 코멘트
# math library를 import
import math
n = math.sqrt(4.0)   # sqrt를 얻기
# print(n)
print("sqrt of {} is {}".format(4.0, n))


# 여러줄 코멘트
""" If I really hate pressing `enter` and typing all th
ose hash marks, I could just do this instead """
```

❑ Indentation

- 코딩블럭을 표시하기 위해 들여쓰기를 사용한다.
- 일반적으로 tab(4)를 사용
- 코딩블럭을 시작하는 문장들
  - If, for, def, …

```
# 주어진 숫자가 홀수인지 짝수인지 출력
number = 10
if number % 2 :
    print('홀수입니다.')
else:
    print('짝수입니다.')
```

❑ import : 파이썬 모듈을 사용하기 위한 방법

- import 모듈명[,모듈명, ..., N]
- import 모듈명 as alias
  - 모듈명을 바꿔서 사용하는 경우
- from 모듈명 import 클래스/함수/변수
  - 모듈의 일부만 가져오는 경우

```
# 1. import 모듈명
   import math
   n = math.sqrt(4.0)

# 2. import 모듈명 as alias
 import numpy as np
 # 1차원 배열
 a = np.array(arr)

# 3. from 모듈명 import 클래스/함수/변수
 from datetime import datetime
 dt = datetime.now()
```

❑ dir : 해당 객체가 어떤 변수와 메소드(method) 가지고 있는지 나열

- **dir()**
  - **현재 지역 스코프에 있는 이름들의 리스트**
- **dir(objet_name)**
  - **해당 객체에 유효한 어트리뷰트의 리스트**
- **dir(object_name.function_name)**
- **참고**
  - 객체가 모듈 객체면, 리스트에는 모듈 어트리뷰트의 이름이 포함됩니다.
  - 객체가 형 또는 클래스 객체면, 리스트에는 그것의 어트리뷰트 이름과 베이스의 어트리뷰트 이름들이 재귀적으로 포함됩니다.
  - 그 밖의 경우, 리스트에는 객체의 어트리뷰트 이름, 해당 클래스의 어트리뷰트 이름 및 해당 클래스의 베이스 클래스들의 어트리뷰트 이름을 재귀적으로 포함합니다.

```
# 현재 로드되어 있는 모듈, 함수, 변수 확인
  dir()

# 문자열 객체의 변수들과 메소드 나열
  a = “hello, world”
  dir(a)
  print(a.upper())

# math의 함수, 변수 확인
  import numpy
  dir(numpy)

# np의 함수, 변수 확인
  import numpy as np
  dir(np)

# 모듈 datetime의 datetime 확인
from datetime import datetime
dir(datetime)
```

❑ print : 입력한 자료형을 출력

- 기본
  - print('Python study')
  - print("Python study")
  - print('₩'PYTHON₩' lesson')
- 큰따옴표(")로 둘러싸인 문자열은 '+' 연산과 동일하다
- 문자열 띄어쓰기는 ','로 한다

[In] print('Python study')
[Out] Python study

[In] print("welcome" "to" "'python world'")
[Out] welcometo'python world'

[In] print("welcome", "to", "'python world'")
[Out] welcome to 'python world'

❑ print : well-formed 출력

- **String modulo operator(%)**
  - **print("%4d USD = %8.1f KRW" % (50, 28422.56))**
    - ✓ [Out] : 50 USD = 28422.6 KRW
- **The string method "format"**
  - **print("{0:4d} USD = {1:.1f} KRW".format(50, 28422.56))**
    - ✓ [Out] : 50 USD($) = 28422.6 KRW(원)
- **문자열 포맷코드**

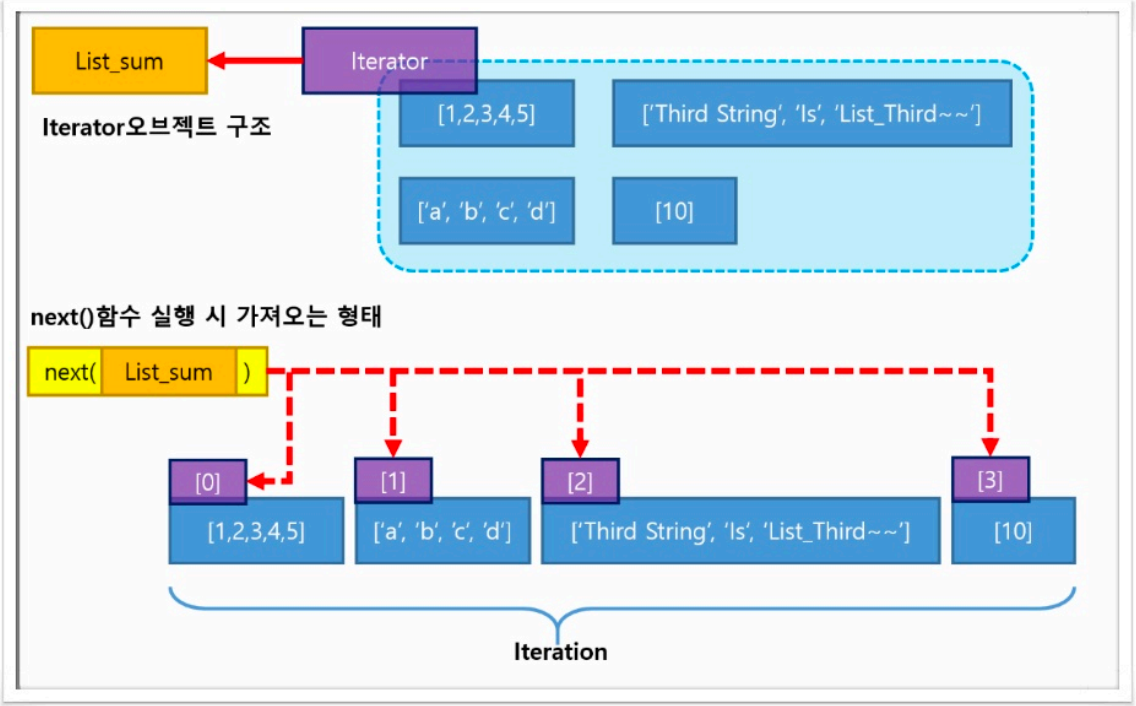
코드	설명
%s	문자열 (String)
%c	문자 1개(character)
%d	정수 (Integer)
%f	부동소수 (floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 % 자체)

❑ Python Iterator

- **Iterable**
  - member를 하나씩 반환할 수 있는 object
  - seqence type인 str, list, tuple, dict, set
  - for loop말고도 zip(), map()과 같이 sequence한 특징을 필요로 하는 작업을 하는 데 유용
  - __iter__()나 __getitem__() 메소드로 정의된 class는 모두 iterable 하다고 볼 수 있다
- **Iterator pattern**
  - 데이터 내부 구현을 노출하지 않고 포함하고 있는 요소들을 순회할 수 있는 방법을 제공하는 패턴
- **Iterator**
  - next()함수로 데이터를 순차적으로 호출이 가능한 object
  - __iter__, __next__를 구현하고 있는 오브젝트를 이터레이터라고 함

```
# range는 iterable 리스트를 생성
for in range(0,3):
    print(i)
# iterable은 아래 함수를 통해 확인 가능
[In] : import collections
       x_list = [1, 3, 5, 7, 9]
       isinstance(x_list, collections.Iterable)
[Out] : True

# 리스트 생성
x = [1, 2, 3, 4, 5]
print(type(x))
print(dir(x))

# 이터레이터 생성
iterx = iter(x)
print(type(iterx))
print(dir(iterx))
```

❑ iteration 구조



❑ Range

- **일정 간격의 정수 값을 가진 데이터를 만들어 주는 함수**
- **range(start, stop, step)**
  - - start : list 시작
  - - stop : list 끝
  - - step : 증가 값
  - stop 값은 범위에 포함되지 않음
- **예시)**
  - range(10)
  - range(0, 10)
  - range(1, 10)
  - range(10, 20)
  - range(10, 0, -1)

[In] : list(range(10))
[Out] : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[In] : list(range(0, 10))
[Out] : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[In] : tuple(range(1,10))
[Out] : (1, 2, 3, 4, 5, 6, 7, 8, 9)

[In] : list(range(10, 20, 2))
[Out] : [10, 12, 14, 16, 18]

[In] : range(10, 0, -1)
[Out] : [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

❑ Enumerate

- **iterable을 순회하면서 iterable에서 각 아이템의 인덱스를 얻어옴**
- **Enumerate(iterable)**
- **Enumerate(iterable, start)**
- **예시)**
  - enumerate(list)
  - enumerate(list, 1)

[In] :

```
seq_list = ['첫 번째', '두 번째', '세 번째', '네 번째', '다섯 번째', '여섯 번째', '일곱 번째']

enum_seq_dict = dict(enumerate(seq_list, 1))
print(enum_seq_dict)
```

[Out] :

```
{1: '첫 번째', 2: '두 번째', 3: '세 번째', 4: '네 번째', 5: '다섯 번째', 6: '여섯 번째', 7: '일곱 번째'}
```

❑ 기타

- 코드 내부에 한글을 사용가능 하게 해주는 부분입니다.
  - # -*- coding: utf-8 -*-

❑ 내장함수

- **Built-in Functions**
  - 파이썬 인터프리터에는 항상 사용할 수있는 많은 함수와 유형이 내장되어 있습니다

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

- **Built-in Functions lists and description**
  - https://docs.python.org/3/library/functions.html

❑ 주요 내장함수

- **lambda 함수(익명함수)**
  - lambda a, b: a+b
  - 함수의 이름을 작성가 필요없이 간단하게 한 줄로 사용하는 함수
- **map 함수**
  - map(funtion, iterable)
  - 입력받은 자료형의 각 요소가 함수에 의해 수행된 결과를 묶어서 map iterator 객체로 리턴
- **filter 함수**
  - filter(function, literable)
  - 두번째 인수인 반복 가능한 자료형 요소들을 첫번째 인자 함수에 하나씩 입력하여 **리턴값이 참인 것만** 묶어서 돌려준다
  - 함수의 리턴 값은 참이나 거짓이어야 한다.
- **reduce 함수 (python 3.x에서는 내장함수에서 제외됨)**
  - from functools import reduce
  - reduce(function, iterable[, initializer])
  - iterable 의 요소들을 function 에 대입하여 결국 하나의 결과값을 리턴해 주는 함수

❑ 주요 내장함수

- **zip 함수**
  - **길이가 같은 자료형을 묶어 반환합니다.**
  - list, set, dict 함수 등을 **같이 사용하여 결과를 반환합니다**

Python Data Type

❑ Data Type

타 입	설 명	표현 예
int	정수형 데이터	1, 2, 100
float	소수점을 포함한 실수	3.14159, 2.54
bool	참/거짓(숫자:0이면 False, 그 외 True, 다른 타입: empty이면 False, 그외 True)	True, False
str	문자열	'Python for data analysis'
list	리스트, 순서가 있는 배열, 수정/추가/삭제가 가능한 자료 구조	[1, 2, 3, 'a', 'b']
tuple	튜플, 순서가 있는 배열, 수정/추가/삭제가 불가능한 자료 구조	(1, 2, 3, 'a', 'b')
dict	사전, {key: value}로 구성되어있는 자료 구조	{'product': 'apple' , 'category': 'fruit', 'price': 1000}
set	집합, {key}로 구성되어있는 자료 구조	{'a', 'b', 'c'}

- 데이터 타입을 확인하기 위해서는 type()을 사용한다.
  - [In] : type(3.14159)
  - [Out] : float
  - [In] : type([1, 2, 3, 'a', 'b'])
  - [Out] : list

Python Data Type

❑ Number(숫자형)

- 숫자형태로 이루어진 자료형
  - 정수형
  - 실수형
  - 8진수
  - 16진수
- 숫자형 연산자
  - 사칙연산 : +, -, *, /
  - 나눗셈 나머지 : %
  - 나눗셈 정수 몫 : //
  - 제곱 : **

Python Data Type

❑ 문자열(String)

- **문자열 표시 : “ ”, ‘ ’**
  - **a = “Python lesson”**
- **이스케이프 코드 ₩**
  - **₩’, ₩₩, ₩n, ₩t, …**
- **문자열 더하기**
  - fullName = lastName + " " + firstName
- **문자열 곱하기**
  - print("별점(*) : " + "*" * 5
- **문자열 길이**
  - **len(a)**
- **문자열 인덱싱**
  - **a[0], a[1]**
- **문자열 슬라이싱**
  - **a[0:4]**
- **문자열 인코딩(encoding)**

이스케이프 코드

코드	설명
₩n	개행 (줄바꿈)
₩t	수평 탭
₩₩	문자 "₩"
₩'	단일 인용부호(')
₩"	이중 인용부호(")
₩r	캐리지 리턴
₩f	폼 피드
₩a	벨 소리
₩b	백 스페이스
₩000	널문자

❑ String(문자열)

메소드	설 명
문자열.startswith(prefix)	문자열이 prefix로 시작하는지 검사하여 True/False 반환
문자열.endswith(suffix)	문자열이 suffix로 끝나는지 검사하여 True/False 반환
문자열.format(value)	문자열의 특정 값을 value로 치환한다.
문자열.replace(old, new)	s에 있는 old를 new로 바꾼다.
문자열.split(sep)	sep를 구분자로 하여 문자열을 분할한다.
문자열.strip(chars)	앞이나 뒤에 나오는 공백이나 chars로 지정된 문자들을 제거한다.

❑ list(리스트)

- 여러 개의 자료를 하나의 변수로 관리할 때 사용
- “[]“(대괄호)로 표시하며, 입력된 값은 index 형태로 접근 가능
- 중첩 리스트
- 주요 기능
  - 리스트 생성
  - 리스트 확장
  - 리스트 요소 추가
  - 리스트 요소 삽입 및 삭제
  - 리스트 정렬
  - 리스트 인덱싱과 슬라이싱
  - 리스트 연산
- 예시
  - num_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
  - alphabet_list = [‘a’, ‘b’, ‘c’, ‘d’]
  - rainbow_list=['빨강', '주황', '노랑', '초록', '파랑', '남색', '보라’]
  - nested_list = [1, 2, 3, [‘a’, ‘b’, ‘c’]] # 중첩리스트

❑ list comprehension(리스트 내포)

- 리스트의 "[ ]" 안에 for 루프를 사용하여 반복적으로 표현식(expression)을 실행해서 리스트 요소들을 정의하는 특별한 용법
- 예시 : 숫자 1에서 9까지 중 짝수로 구성된 리스트를 만듬
  - [In] : list = [n for n in range(10) if n % 2 == 0]
  - print(list)
  - [Out] : [0, 2, 4, 6, 8]

❑ tuple(튜플)

- 튜플은 "()"로 표시
- 여러 개의 자료를 하나의 변수로 관리할 때 사용
- 리스트(list)와 거의 같지만, '데이터를 변경할 수 없다'는 차이가 있다.
- 따라서 .append() 등 값을 변경하는 메소드는 사용할 수 없고,
- 조회를 하는 .count(), .index() 메소드만 사용할 수 있다
- 주요 기능
  - 튜플 생성
  - 튜플 확장
  - 튜플 요소 추가
  - 튜플 요소 삽입 및 삭제
  - 튜플 정렬
  - 튜플 인덱싱과 슬라이싱
  - 튜플 연산
- 예시)
  - num_tuple = (1, 2, 3, 4, 5)
  - fruit_tuple=('apple','blueberry', 'strawberry', 'mango')

❑ dictionary(딕셔너리)

- Dictionary의 요소들은 Curly Brace "{...}"(중괄호) 를 사용하여 컬렉션을 표현
- "키(Key) - 값(Value)" 쌍을 요소로 갖는 컬렉션
- 키는 유일해야 하며 순서가 없다
- Dictionary의 키(key)는 그 값을 변경할 수 없는 Immutable 타입이어야 하며, Dictionary 값(value)은 Immutable과 Mutable 모두 가능하다.
- 예를 들어, Dictionary의 키(key)로 문자열이나 Tuple은 사용될 수 있는 반면, 리스트는 키로 사용될 수 없다.
- 예시1)
  - seq_dic = {0:'첫 번째', 1:'두 번째', 2:'세 번째', 3:'네 번째', 4:'다섯 번째', 5:'여섯 번째', 6:'일곱 번째'}
- 예시2)
  - seq_list = ['첫 번째', '두 번째', '세 번째', '네 번째', '다섯 번째', '여섯 번째', '일곱 번째']
  - enum_seq_list = list(enumerate(seq_list, 1))
  - seq_dict = dict(enum_seq_list)

❑ set(집합)

- 집합은 여러 개의 자료를 하나의 변수로 관리할 때 사용하는 자료형 중의 하나입니다.
- 기본적으로 “{}”(중괄호)를 사용한다
- 집합 자료형은 수학의 집합과 같은 성질을 가집니다. 즉, 집합은 중복된 데이터를 가질 수 없고, 순서가 없습니다.
- 따라서 순서와 관련된 인덱스기호([ ])를 사용할 수 없고, 중복 데이터를 만드는 +, *를 사용할 수 없습니다.
- 하지만, in, not in, len()은 사용할 수 있습니다.
- 예시)
  - [In] : num_set = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - [Out] : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

  - [In] : num_set = {1, 3, 5, 7, 9, 0, 2, 4, 6, 8}
  - [Out] : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

  - [In] : num_set = {1, 1, 1, 3, 3, 0, 2, 6, 6, 8}
  - [Out] : {0, 1, 2, 3, 6, 8}

Python Operators

- **Arithmetic Operators(산술 연산자)**
  - +, -, *, /, //(몫),%(나머지),**(자승)
- **Relational Operators(관계 연산자)**
  - >, >=, <, <=, ==, !=
- **Logical Operators(논리 연산자)**
  - and, or, not
- **Ternary Operators(삼항 연산자)**
  - 참인경우 값 if 조건 else 거짓인경우 값
- **Bitwise Operators(비트 연산자)**
  - &, |, ^, ~, <<, >>
- **Assignment Operators**
  - 연산과 할당을 합쳐놓은 것
  - a += b <= a = a+b
- **Membeship Operators(멤버쉽 연산자)**
  - in(포함), not in(포함되지 않음)
- **Indentity Operators**
  - is : 양쪽이 동일한 오브젝트를 가리키는지 아닌지 검사
  - is not : 양쪽이 다른 오브젝트를 가리키는지 아닌지 검사

Python Operators

❑ Ternary operators(3항 연산자)

- **참인경우 값 if 조건 else 거짓인경우 값**

```
# if 조건문
score = 70
if score >= 70:
    message = "success"
else:
    message = "failure"
print(message)

# 3항 연산자
score = 70
message = "success" if score >= 70 else "failure"
print(message)

# [Out] : success
```

Python Operators

- **Operator Precedence(연산자 우선순위)**
  - 연산 순서를 변경할 때는 괄호를 사용
  - 연산자 우선순위

순위	연산자	설명
1	(), {}, []	Tuple, Set, List, Dictionary
2	collection[index] collection[index1, index2] function(aguments ...) object.attribute	컬렉션의 첨자 슬라이싱 함수의 인수 객체의 속성 등
3	**	거듭제곱
4	+ , - , ~	단항 연산자(부호, bitwise not)
5	* / // %	곱하기, 나누기, 정수 몫, 나머지
6	+ -	더하기, 빼기
7	<< >>	시프트 연산
8	&	bitwise and
9	^	bitwise xor
10		
11	in, not in is, is not <, <=, >, >=, ==, !=	멤버 연산자 아이디 연산자 관계 연산자
12	not	논리 not
13	and	논리 and
14	or	논리 or
15	if else	삼항 연산자
16	lambda	람다 표현식

Python Control Statement

❑ if statement

- **if** 조건문:
  수행할 문장1
  수행할 문장2
  else:
  수행할 문장A
  수행할 문장B
- **if** 조건문:
  수행할 문장1
  수행할 문장2
  elif 조건문:
  수행할 문장3
  수행할 문장4
  else:
  수행할 문장A
  수행할 문장B

```
# 주어진 숫자가 홀수인지 짝수인지 출력
number = 10
if number % 2 :
    print('홀수입니다.')
else:
    print('짝수입니다.')
```

Python Control Statement

❑ for statement

- **for item in iterable:**
  반복할 구문(수행할 문장n)
- **for** item **in** range:
  반복할 구문

```
# for item in iterable 예시
x_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for i in x_list:
    print(i)

# for item in range 예시
for i in range(10):
    print(i)
```

❑ while statement

- **반복해서 문장을 수행해야 할 경우 사용**

  while <조건문>:
  <수행할 문장1>
  <수행할 문장2>
  <수행할 문장3>
- **while문 강제로 빠져 나가기**
  - **break**
- **while문의 맨 처음으로 돌아가기**
  - **continue**
- **주의 : 무한루프**

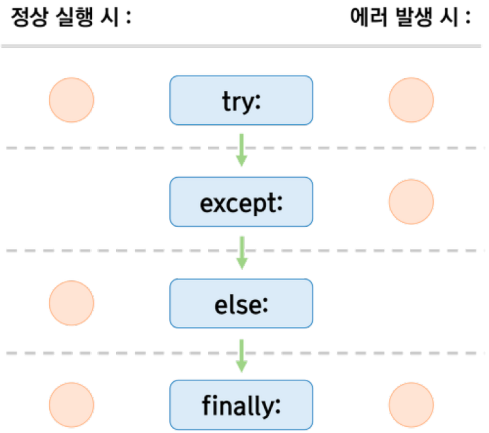
```
# 시작 시 조건을 설정한 케이스
i=1
while i <= 10:
    print(i)
    i += 1


# 무조건 시작하고 while문 안에서 조건 검사
i = 0
sum = 0
while True:
    i += 1
    if i == 5:
        continue
    if i > 10:
        break
    sum += i
    print("숫자 : {}, 합계 : {}".format(i,sum))
```

❑ 기타

- **break**
  - 가장 안쪽의 반복문을 빠져 나옴
- **continue**
  - 다음 순번의 loop 실행
- **pass**
  - 실행 할 것이 아무것도 없다는 의미
  - 아무런 동작을 하지 않고 다음 코드를 실행

❑ Exception 처리

- **예최 처리 기본 흐름 (https://gomguard.tistory.com/122)**



- **예시**

```
try:
    print(5/0)
except ZeroDivisionError:
    print('wrong division')
finally:
    print('end')
```

File Handling

- **파일 생성 및 열기**

파일 열기 모드	설 명
'r'	읽기 모드로 연다. - Default
'w'	쓰기 모드로 연다. - 기존 내용 삭제
'a'	추가 모드로 연다.
'b'	이진 모드로 연다.
't'	텍스트 모드로 연다. - Default

❑ 파일에서 읽기

- **한 줄씩 읽기**
  - **readline**
- **여러 줄을 한 번에 읽어 리스트에 저장하기**
  - **readlines**
- **파일 내용 모두 읽기**
  - **read**
- **strip ()**
  - 지정된 문자가 문자열의 시작과 끝에서 제거 된 문자열의 복사본을 반환
  - 지정하지 않은 경우는 공백문자

```
# 한 줄씩 읽기
fp = open("data/text.txt", encoding = 'utf-8')
line = fp.readline()
print(line.strip())
line = fp.readline()
print(line.strip())
f.close()

# 여러 줄을 한 번에 읽기
fp = open('data/text.txt', 'r')
lines = fp.readlines()
print(lines)
fp.close()

# 파일 내용 모두 읽기
fp = open('data/text.txt', 'r')
contents = fp.read()
print(contents)
fp.close()
```

❑ 파일에 쓰기

- **파일에 덮어 쓰기**
  - **파일 열기 모드를 'w'로 지정**
- **파일에 내용 추가하기**
  - **파일 열기 모드를 'a'로 지정**

```
# 파일에 덮어 쓰기
fp = open('data/hello.txt', 'w')
fp.write('Hello World!')
fp.close()

# 파일에 내용 추가하기
fp = open('data/text.txt', 'a')
fp.write('{}₩n'.format(2))
fp.write('{:.5f}₩n'.format(3.14159))
fp.write('Hello World!₩n')
fp.write('일산병원₩n')
fp.close()
```

❑ csv 파일 읽기

- **파일 읽기**
- **파일이 존재하지 않을 수 있으므로 exception 처리**

```
# csv 파일 읽기
import pandas as pd

datafilename = "data/oasis_longitudinal.csv"

try:
    df = pd.read_csv(datafilename, header=0, encoding ='utf-8')
    df.info()
except FileNotFoundError :
    print("파일이 존재하지 않습니다. 다시 확인 후 진행해 주세요.")
```

❑ 엑셀 사용하기

- **파이썬에서 엑셀 데이타를 핸들링하기 위해서는 openpyxl, xlrd, xlrw, pandasd 등의 외부 패키지를 설치해서 사용한다**
- **pandas를 이용한 예시**
  - import pandas as pd

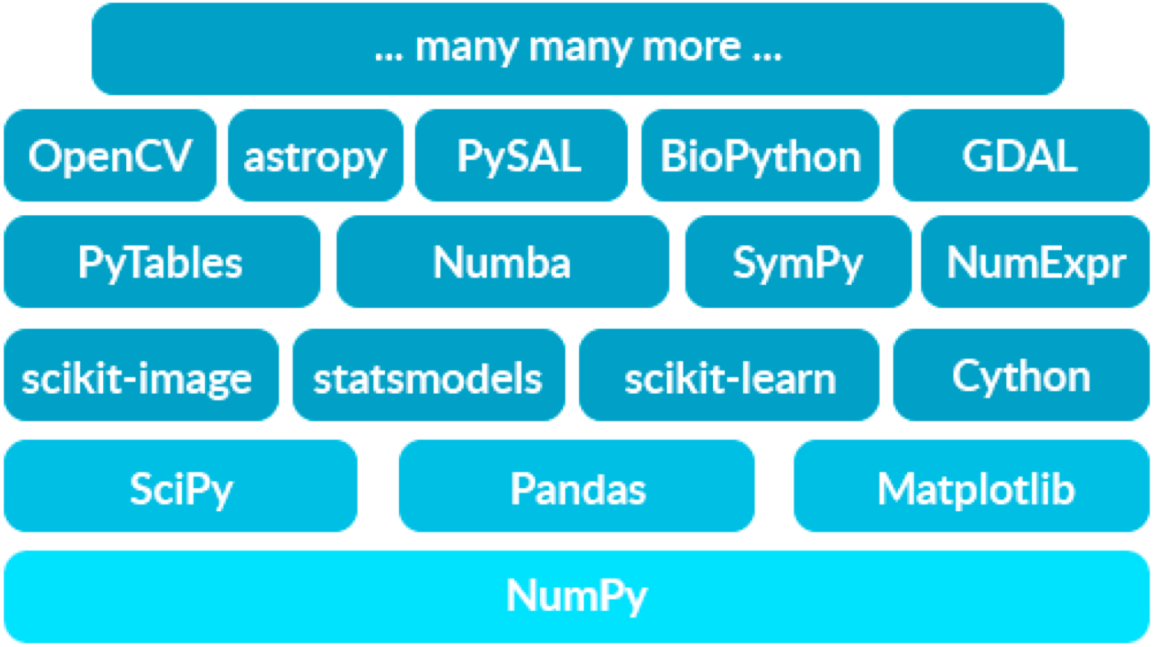
  - infilename = 'data/estate_201801.xlsx'
  - sheetname='아파트 매매 실거래가'
  - df = pd.read_excel(infilename, sheet_name=sheetname, skiprows=16, header=0, converters={'본번':str,'부번':str}, encoding=utf-8')
  - df.columns = ['Region','Sub1','Sub2','Sub3','ApartmentName','ExclusiveArea', 'ContractYM','ContractDay','SalePrice','Floor','YearBuilt','Street']

  - df.shape
  - print(df.head())

  - # SalePrice가 ','가 포함된 object type으로 되어 있는데 ','를 제거후 numeric type으로 변경
  - df['SalePrice'] = df['SalePrice'].replace(',', '', regex=True)
  - df['SalePrice'] = pd.to_numeric(df['SalePrice'])

주요 패키지

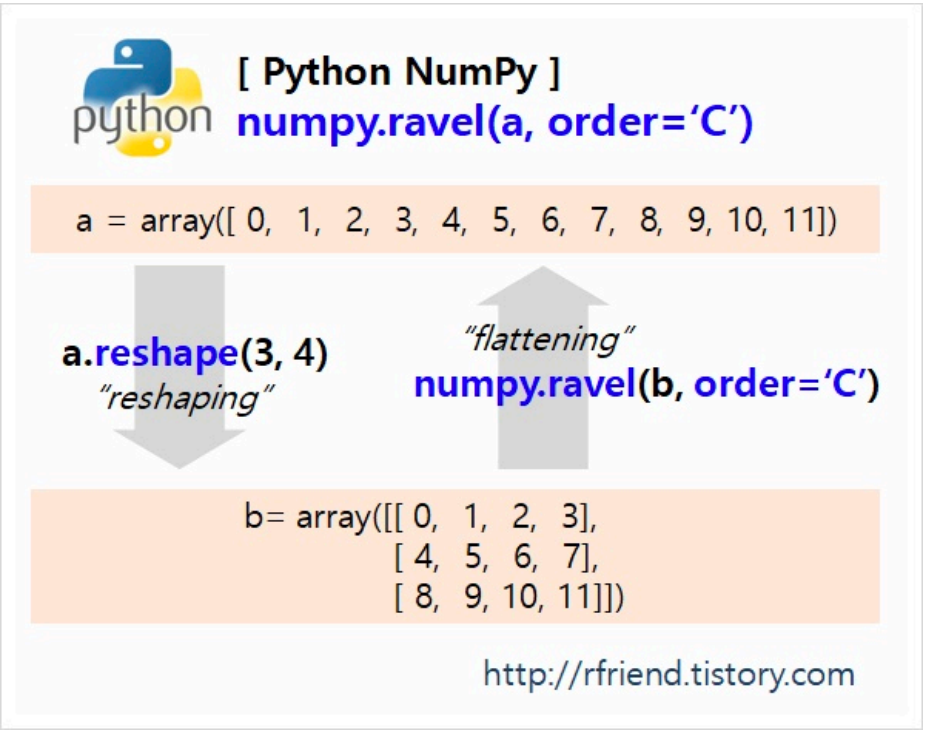
- 데이터 처리
  - Numpy
  - Pandas
- 시각화
  - Matplotlib
  - Seaborn
- 머신 러닝
  - Scikit-learn



주요 패키지

❑ numpy

- numpy는 과학 계산을 위한 라이브러리로서 다차원 배열을 처리하는데 필요한 여러 유용한 기능을 제공
- **numpy 배열**
  - **reshape**
  - **ravel**
  - **flatten**
- **numpy 백터**



- 배열 사용에서 주의할 점
  - 길이가 5인 1차원 배열과 행, 열의 갯수가 (5,1)인 2차원 배열 또는 행, 열의 갯수가 (1, 5)인 2차원 배열은 데이터가 같아도 엄연히 다른 객체라는 점이다.

주요 패키지

❑ pandas

- pandas는 데이타 분석(Data Analysis)을 위해 널리 사용되는 파이썬 라이브러리 패키지이다
- Series
  - 1차원 자료구조
  - 배열/리스트 같은 일련의 시퀀스 데이터를 받아들인다
  - 별도의 인덱스 레이블을 지정하지 않으면 자동적으로 0부터 시작되는 정수 인덱스 사용
- DataFrame
  - 2차원 자료구조
  - 행과 열이 있는 테이블 데이터(Tabular Data)를 받아들인다.
- Panel
  - 3차원 자료구조
  - Axis0(items), Axis1(major_axis), Axis2(minor_axis)등 3개 축을 가지고 있다.
  - Axis 0은 한 요소가 2차원의 DataFrame에 해당하며
  - Axis 1은 DataFrame의 행(row)에 해당하고, Axis 2는 DataFrame의 열(column)에 해당된다