

파이썬 기초~응용

『2과목 :』 빅데이터 분석 도구

2025.06.23-07.04(10일, 70시간)

Prepared by Daekyeong Kim

Ph.D.

1. 빅데이터 분석 도구 -소개
 - 빅데이터 분석 도구 개요
 - 실습 환경 구축 Building a Data Analysis and AI(Artificial Intelligence) practice environment
 - Python 작업 몇 가지 규칙
2. 빅데이터 분석 도구- fundamental
 - Python 계산, 변수와 자료형
 - Python 자료구조
 - Python's Statement
 - 함수
 - 객체와 클래스
 - Moduler과 패키지
 - 파이썬의 예외처리와 내장함수
 - 파이썬 외장함수
 - 라이브러리
 - Python 과 Library
3. 빅데이터 분석 도구- advanced
 - 파이썬 표준 입/출력
 - 웹-Flask 개발 환경 만들기 등
4. Self 점검
 - 필기 평가
 - 실기 평가

『2-2』

빅데이터 분석 도구 -fundamental

- Python 계산, 변수와 자료형
- Python 자료구조
- Python's Statement
- 함수
- 객체와 클래스
- Moduler과 패키지
- **파이썬의 예외처리와 내장함수**
- 파이썬 외장함수
- 라이브러리



학습목표

- 이 워크샵에서는 파이썬의 예외처리와 내장함수를 할 수 있습니다.

눈높이 체크

- 파이썬의 예외처리와 내장함수를 알고 계신가요?

1. 예외처리

예외처리란?

- 프로그래밍을 하다 보면 많은 에러가 발생
 - 이런 에러를 처리하고 싶을 때 "try, except"를 사용
 - 에러 유형

파일	소스코드
실습환경	py3_10_basic f = open("not exist file", 'r')
소스코드	4/0 a = [1,2,3] a[4]
결과값1	Traceback (most recent call last): File "c:\DEV\python_works\day2-pm.py", line 276, in <module> f = open("not exist file", 'r') FileNotFoundError: [Errno 2] No such file or directory: 'not exist file'
결과값2	4/0 ZeroDivisionError: division by zero
결과값3	a[4] IndexError: list index out of range
비고	



1. 예외처리

에러 처리하기 (try, except)

```
try:
```

```
...
```

```
except [발생에러[as 에러메시지변수]]:
```

```
...
```

에러가 발생하면
except문을 실행

에러 발생 시 except문에 미리 정해놓은
에러와 일치할 때만 except문 실행

에러메시지를 담은 변수 하나를 더 생성하게
하는 방법

[] 안의 내용은 생략 가능

1. 예외처리

Exceptions - try-except

- 함수 `getMean()`은 시퀀스에 값이 포함되어 있는지 여부를 테스트하기 위한 `if` 문을 포함하지 않는다.

파일	소스코드
실습환경	<pre>py3_10_basic def getMean(numericValues): return sum(numericValues)/len(numericValues) my_list2 = []</pre>
소스코드	<pre>try: print("Output #1: {}".format(getMean(my_list2))) except ZeroDivisionError as detail: print("Output #2 (Error): {}".format(float('nan'))) print("Output #3 (Error): {}".format(detail))</pre>
결과값1	<pre>Output #2 (Error): nan Output #3 (Error): division by zero</pre>
비고	<p><code>my_list2</code> 목록에 있는 것처럼 시퀀스가 비어 있는 경우 함수를 적용하면 <code>ZeroDivisionError</code>가 발생한다. <code>try-except</code> 블록을 사용하려면 실행하려는 코드를 <code>try</code> 블록에 넣는다. 그런 다음, 예외 블록을 사용하여 잠재적 오류를 처리하고 유용한 오류 메시지를 프린트한다. 어떤 경우에는 예외에 연관된 값이 있다. 예외 라인에 <code>as</code> 구문을 포함하고 예외 값에 지정한 이름을 인쇄하여 예외 값에 액세스할 수 있다. <code>my_list2</code>에는 값이 포함되어 있지 않기 때문에 <code>nan</code>을 출력한 다음 <code>Error: float 나눗셈을 0으로 출력하는 except 블록이 실행된다.</code></p>

1. 예외처리

에러 처리하기 (try, except)

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("나누기 전용 계산기입니다.") num1 = int(input("첫 번째 숫자를 입력하세요 : ")) num2 = int(input("두 번째 숫자를 입력하세요 : ")) print("{0} / {1} = {2}".format(num1, num2, int(num1/num2)))</pre>
결과값1	<pre>나누기 전용 계산기입니다. 첫 번째 숫자를 입력하세요 : 6 두 번째 숫자를 입력하세요 : 0 Traceback (most recent call last): File "c:\DEV\python_works\day2-pm.py", line 299, in <module> print("{0} / {1} = {2}".format(num1, num2, int(num1/num2))) ZeroDivisionError: division by zero</pre>
비고	



파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>try: print("나누기 전용 계산기입니다.") num1 = int(input("첫 번째 숫자를 입력하세요 : ")) num2 = int(input("두 번째 숫자를 입력하세요 : ")) print("{0} / {1} = {2}".format(num1, num2, int(num1/num2))) except ValueError: print("에러! 잘못된 값을 입력하였습니다.") except ZeroDivisionError as err: print(err)</pre>
결과값1	<pre>나누기 전용 계산기입니다. 첫 번째 숫자를 입력하세요 : 6 두 번째 숫자를 입력하세요 : 0 division by zero</pre>
비고	

1. 예외처리

에러 처리하기 (try, except)

파일

소스코드

실행환경

py3_10_basic

소스코드

```
try:
    print("나누기 전용 계산기입니다.")
    nums = []
    nums.append(int(input("첫 번째 숫자를 입력하세요 : ")))
    nums.append(int(input("두 번째 숫자를 입력하세요 : ")))
    # nums.append(int(nums[0] / nums[1])) # 계산 결과를 리스트에 추가
    print("{0} / {1} = {2}".format(nums[0], nums[1], nums[2]))
except ValueError:
    print("에러! 잘못된 값을 입력하였습니다.")
except ZeroDivisionError as err:
    print(err)
except Exception as err:
    print("알 수 없는 에러가 발생하였습니다.")
    print(err)
```

결과값1

```
나누기 전용 계산기입니다.
첫 번째 숫자를 입력하세요 : 6
두 번째 숫자를 입력하세요 : 0
알 수 없는 에러가 발생하였습니다.
list index out of range
```

비고

1. 예외처리

에러 처리하기 (try, except)

파일

소스코드

실습환경

py3_10_basic

소스코드

```
class BigNumberError(Exception):
    def __init__(self, msg):
        self.msg = msg

    def __str__(self):
        return self.msg

try:
    print("한 자리 숫자 나누기 전용 계산기입니다.")
    num1 = int(input("첫 번째 숫자를 입력하세요: "))
    num2 = int(input("두 번째 숫자를 입력하세요: "))
    if num1 >= 10 or num2 >= 10:
        raise BigNumberError("입력값 : {0}, {1}".format(num1, num2))
    print("{0} / {1} = {2}".format(num1, num2, int(num1 / num2)))
except ValueError:
    print("잘못된 값을 입력하였습니다. 한 자리 숫자만 입력하세요.")
except BigNumberError as err:
    print("에러가 발생하였습니다. 한 자리 숫자만 입력하세요.")
    print(err)
finally: # 에러 발생 여부 상관 없이 항상 실행
    print("계산기를 이용해 주셔서 감사합니다.")
```

결과값1

한 자리 숫자 나누기 전용 계산기입니다.
첫 번째 숫자를 입력하세요: 10
두 번째 숫자를 입력하세요: 5
에러가 발생하였습니다. 한 자리 숫자만 입력하세요.
입력값 : 10, 5
계산기를 이용해 주셔서 감사합니다.

비고



1. 예외처리

에러 처리하기 (try, except)

<pre>try: 수행 문장 except: 에러 처리</pre>	<pre>try: 수행 문장 finally: 마지막 수행</pre>	<pre>try: 수행 문장 except: 에러 처리 else: 정상 동작</pre>	<pre>try: 수행 문장 except: 에러 처리 else: 정상 동작 finally: 마지막 수행</pre>
---	---	---	---

2. 문자열 처리 함수

파이썬으로 문자열 위치 알기 (find, index)-값이 있을 때

파일	소스코드
실습환경	py3_10_basic company = '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'
소스코드	print(company.find('전자')) print(company.index('전자'))
결과값1	2 2
비고	

2. 문자열 처리 함수

파이썬으로 문자열 위치 알기 (find, index)-값이 없을 때

파일	소스코드
실습환경	<pre>py3_10_basic company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'</pre>
소스코드	<pre>print(company.find('네이버')) print(company.index('네이버'))</pre>
결과값1	-1
결과값2	<pre>Traceback (most recent call last): File "c:/DEV/python-works/Test.py", line 101, in <module> print(company.index('네이버')) ValueError: substring not found</pre>
비고	

2. 문자열 처리 함수

파이썬으로 문자열 위치 알기 (find, index)-예외처리

파일	소스코드
실습환경	<pre>py3_10_basic company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'</pre>
소스코드	<pre>try: print(company.index('전자')) except ValueError: print("-1")</pre>
결과값	2
	<pre>company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'</pre>
소스코드	<pre>try: print(company.index('네이버')) except ValueError: print("-1")</pre>
결과값	-1
비고	

2. 문자열 처리 함수

파이썬으로 문자열 위치 알기 (find, index)-값을 계속 호출

파일	소스코드
실습환경	<pre>py3_10_basic company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'</pre>
소스코드	<pre>index = 0 while index > -1: index = company.find('전자', index) if index > -1: print(index) index += len('전자')</pre>
결과값1	<pre>2 7</pre>
비고	

2. 문자열 처리 함수

파이썬으로 문자열 위치 알기 (find, index)-값을 계속 호출

파일	소스코드
실습환경	py3_10_basic
	company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'
소스코드	<pre>index = 0 while index > -1: try: index = company.index('전자', index) print(index) index += len('전자') except ValueError: break</pre>
결과값1	2 7
비고	

3. 내장 함수

내장 함수?

- import 필요 없이 python에서 바로 사용할 수 있는 함수

list of python builtins



전체

이미지

동영상

쇼핑

뉴스

더보기

도구

검색결과 약 474,000개 (0.44초)

도움말: 한국어 검색결과만 검색합니다. 환경설정에서 검색 언어를 지정할 수 있습니다.



python.org

<https://docs.python.org> > library > functions

Built-in Functions — Python 3.11.2 documentation

The **Python** interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.,,,, **Built-in** ...

[bytearray\(\)](#) · [bytes\(\)](#) · [dict\(\)](#)

3. 내장 함수

내장 함수?

- import 필요 없이 python에서 바로 사용할 수 있는 함수

Python » English » 3.11.2 » 3.11.2 Documentation » The Python Standard Library » Built-in Functions

Quick search Go | previous | next | modules | index

Previous topic

Introduction

Next topic

Built-in Constants

This Page

Report a Bug

Show Source

Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

Built-in Functions			
A abs() aiter() all() any() anext() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip() __import__()

abs(*x*)

Return the absolute value of a number. The argument may be an integer, a floating point number, or an object implementing `__abs__()`. If the argument is a complex number, its magnitude is returned.

3. 내장 함수

내장 함수?

- import 필요 없이 python에서 바로 사용할 수 있는 함수

함수	설명
abs	abs(x)는 숫자값을 입력값으로 받았을 때, 그 숫자의 절대값을 돌려주는 함수
chr	정수 형태의 아스키코드값을 입력으로 받아서 그에 해당하는 문자를 출력하는 함수
dir	객체가 가지고 있는 변수나 함수를 리스트 형태로 보여준다.
divmod	두 개의 숫자를 입력값으로 받았을 때 그 몫과 나머지를 튜플의 형태로 반환하는 함수
enumerate	입력값으로 시퀀스자료형(리스트, 터플, 문자열)을 입력으로 받아 첫번째로 그 순서값, 두번째 로 그 순서값에 해당되는 시퀀스 자료형의 실제값을 반환
eval	입력값으로 실행가능한 문자열(1+2, 'hi' + 'a' 같은 것)을 입력으로 받아서 문자열을 실행한 결과값을 반환하는 함수
filter	함수와 리스트를 입력으로 받아서 리스트의 값이 하나씩 함수에 인수로 전달될 때, 참을 반환시키는 값만을 따로 모아서 리스트의 형태로 반환하는 함수이다.
hex	입력으로 정수값을 받아서 그 값을 십육진수값(hexadecimal)로 변환하여 돌려주는 함수
id	객체를 입력값으로 받아서 객체의 고유값(레퍼런스)을 반환하는 함수
input	사용자 입력을 받는 함수이다
int	string 형태의 숫자나 소수점 숫자 등을 정수의 형태로 반환시켜 돌려준다.

3. 내장 함수

내장 함수?

함수	설명
isinstance	입력값으로 인스턴스와 클래스 이름을 받아서 입력으로 받은 인스턴스가 그 클래스의 인스턴스인지를 판단
lambda	함수를 생성할 때 사용되는 예약어로 def와 동일하나 보통 한줄로 간결하게 만들어 사용할 때사용한다. def를 쓸 정도로 복잡하지 않거나 def를 쓸 수 없는 곳에 쓰인다.
len	인수로 시퀀스 자료형(문자열, 리스트, 터플)을 입력받아 그 길이(요소의 개수)를 돌려주는 함수
list	인수로 시퀀스 자료형을 입력받아 그 요소를 똑같은 순서의 리스트로 만들어 돌려주는 함수
map	함수와 시퀀스 자료형(리스트, 터플, 문자열)을 입력으로 받아서 시퀀스 자료형의 각각의 요소가 함수의 입력으로 들어간 다음 나오는 출력값을 묶어서 리스트로 돌려주는 함수
max	시퀀스 자료형(문자열, 리스트, 터플)을 입력받아 그 최대값을 돌려주는 함수
min	시퀀스 자료형을 입력받아 그 최소값을 돌려주는 함수
type	type(object)은 인수로 객체를 입력받아 그 객체의 자료형이 무엇인지 알려주는 함수.
oct	정수 형태의 숫자를 8진수 문자열로 바꾸어 돌려주는 함수
open	파일 이름과 읽기 방법을 입력받아 파일 객체를 돌려주는 함수이다.
ord	문자의 아스키 값을 돌려주는 함수이다.
pow(x,y)	x의 y승을 한 결과값을 돌려주는 함수이다.
range	인수로 정수값을 주어 그 숫자에 해당되는 범위의 값을 list 형태로 돌려주는 함수



3. 내장 함수

내장 함수?

함수	설명
repr	객체를 출력할 수 있는 문자열 형태로 변환하여 돌려주는 함수이 변환된 값은 주로 eval 함수의 입력으로 쓰인다.
sorted	입력으로 받은 시퀀스 자료형을 소트한 후 그 결과를 리스트로 리턴하는 함수이다.
str	str(object)은 객체를 출력할 수 있는 문자열 형태로 변환하여 돌려주는 함수이다.
tuple	tuple(sequence)은 인수로 시퀀스 자료형을 입력받아 터플 형태의 자료로 바꾸어 돌려준다
zip	zip 함수는 동일한 갯수의 요소값을 갖는 시퀀스 자료형을 묶어주는 역할.

3. 내장 함수

내장 함수?

abs(x)

- x의 절댓값을 돌려주는 함수

```
>>> abs(3)
3
```

```
>>> abs(-3)
3
>>> abs(-1.2)
1.2
```

all(x)

- 반복 가능한(iterable) 자료형 x가 모두 참이면 True, 하나라도 거짓이면 False 반환

```
>>> all([1, 2, 3])
True
```

```
>>> all([1, 2, 3, 0])
False
```

any(x)

- x가 모두 거짓이면 False, 하나라도 참이면 True 반환

```
>>> any([1, 2, 3, 0])
True
```

```
>>> any([0, ""])
False
```

3. 내장 함수

내장 함수?

▪ chr(x)

- 아스키(ASCII) 코드를 입력받아 코드에 해당하는 문자 반환

```
>>> chr(97)
'a' ← 아스키 코드 97은 소문자 a
>>> chr(48)
'0' ← 아스키 코드 48은 숫자 0
```

▪ dir(x)

- 객체가 자체적으로 가지고 있는 변수나 함수 반환

```
>>> dir([1, 2, 3])
['append', 'count', 'extend', 'index', 'insert', 'pop',...]
>>> dir({'1':'a'})
['clear', 'copy', 'get', 'has_key', 'items', 'keys',...]
```

▪ divmod(a, b)

- a를 b로 나눈 몫과 나머지를 튜플 형태로 반환

```
>>> divmod(7, 3)
(2, 1) ← 7 나누기 3의 몫은 2, 나머지는 1
```

3. 내장 함수

내장 함수?

■ enumerate(x)

- '열거하다'라는 뜻
- 순서가 있는 자료형(리스트, 튜플, 문자열)을 입력으로 받아 인덱스 값을 포함하는 `enumerate` 객체 반환

```
>>> for i, name in enumerate(['body', 'foo', 'bar']):
...     print(i, name)
...
0 body
1 foo
2 bar
```

■ eval(expression)

- 실행 가능한 문자열(`expression`)을 입력으로 받아 문자열을 실행한 결과값 반환
- 문자열로 파이썬 함수나 클래스를 동적으로 실행할 때 사용

```
>>> eval('1+2')
3
>>> eval("'hi' + 'a'")
'hia'
>>> eval('divmod(4, 3)')
(1, 1)
```




3. 내장 함수

내장 함수?

▪ `filter(f, iterable)`

- Iterable 자료형의 요소가 함수 `f`에 입력되었을 때 반환 값이 참인 것만 묶어서 반환

```
#filter1.py
def positive(x):
    return x > 0

print(list(filter(positive, [1, -3, 2, 0, -5, 6])))
```

결괏값: [1, 2, 6]

▪ `lambda`도 사용 가능

```
>>> list(filter(lambda x: x > 0, [1, -3, 2, 0, -5, 6]))
```

▪ `hex(x)`

- 정수 값을 입력받아 16진수로 변환

```
>>> hex(234)
'0xea'
>>> hex(3)
'0x3'
```



3. 내장 함수

내장 함수?

▪ id(object)

- object(객체)의 고유 주소 값 반환

```
>>> a = 3
>>> id(3)
135072304
>>> id(a)
135072304
>>> b = a
>>> id(b)
135072304
```

— 3, a, b는 모두 같은 객체를 가리킴

▪ input([prompt])

- 사용자 입력을 받는 함수
- 문자열 인자 생략 가능
- 매개변수로 문자열을 주면 프롬프트 띄움

```
>>> a = input() ← 사용자가 입력한 정보를 변수 a에 저장
hi
>>> a
'hi'
>>> b = input("Enter: ") ← Enter: 프롬프트를 띄우고 사용자 입력을 받음
Enter: hi
```



3. 내장 함수

내장 함수?

▪ int(x)

- 문자열 형태의 숫자나 소수점이 있는 숫자 등을 정수 형태로 변환

```
>>> int('3') ← 문자열 형태 '3'  
3  
>>> int(3.4) ← 소수점이 있는 숫자 3.4  
3
```

▪ int(x, radix)

- radix 진수 문자열을 10진수로 변환

```
>>> int('1A', 16)  
26
```

▪ instance(object, class)

- object: 인스턴스 / class: 클래스 이름
- 인스턴스가 클래스의 인스턴스인지 판단하여 참이면 True, 거짓이면 False 반환

```
>>> class Person: pass ← 아무 기능이 없는 Person 클래스 생성  
...  
>>> a = Person() ← Person 클래스의 인스턴스 a 생성  
>>> isinstance(a, Person) ← a가 Person 클래스의 인스턴스인지 확인  
True  
>>> b = 3  
>>> isinstance(b, Person) ← b가 Person 클래스의 인스턴스인지 확인  
False
```



3. 내장 함수

내장 함수?

▪ len(x)

- 입력값의 길이(요소의 전체 개수) 반환

```
>>> len("python")  
6
```

```
>>> len([1,2,3])  
3
```

▪ list(iterable)

- Iterable 자료형을 리스트로 변환

```
>>> list("python")  
['p', 'y', 't', 'h', 'o', 'n']  
>>> list((1,2,3))  
[1, 2, 3]
```

▪ map(f, iterable)

- Iterable 자료형의 각 요소를 함수 f가 수행한 결과를 묶어서 반환

```
>>> def two_times(x): return x*2  
...  
>>> list(map(two_times, [1, 2, 3, 4]))  
[2, 4, 6, 8]
```

- lambda 활용 가능

```
>>> list(map(lambda a: a*2, [1, 2, 3, 4]))  
[2, 4, 6, 8]
```



3. 내장 함수

내장 함수?

▪ max(iterable)

- Iterable 자료형의 최대값 반환

```
>>> max([1, 2, 3])  
3
```

```
>>> max("python")  
'y'
```

▪ min(iterable)

- Iterable 자료형의 최솟값 반환

```
>>> min([1, 2, 3])  
1
```

```
>>> min("python")  
'h'
```

▪ oct(x)

- 정수 형태의 숫자를 8진수 문자열로 변환

```
>>> oct(34)  
'0o42'  
>>> oct(12345)  
'0o30071'
```



3. 내장 함수

내장 함수?

▪ `open(filename, [mode])`

- `filename`: 파일 이름 / `mode`: 읽기 방법
- `mode`를 생략하면 기본값인 읽기 전용 모드(`r`)로 파일 객체 반환
- `b`는 `w`, `r`, `a`와 함께 사용

모드	설명
<code>w</code>	쓰기 모드로 파일 열기
<code>r</code>	읽기 모드로 파일 열기
<code>a</code>	추가 모드로 파일 열기
<code>b</code>	바이너리 모드로 파일 열기

```
>>> f = open("binary_file", "rb")
```

```
f = open("./datasets/Test.txt", "rb")
```

▪ `ord(c)`

- 문자의 아스키 코드 값 반환

```
>>> ord('a')  
97
```

```
>>> ord('0')  
48
```

▪ `pow(x, y)`

- `x`의 `y` 제곱한 결괏값 반환

```
>>> pow(2, 4)  
16 ← 2의 4 제곱
```

```
>>> pow(3, 3)  
27 ← 3의 3 제곱
```

3. 내장 함수

내장 함수?

▪ range([start,] stop [,step])

- 입력받은 숫자에 해당하는 범위 값을 iterable 객체로 반환

1) 인수가 하나일 경우

- 시작 숫자를 지정해 주지 않으면 range 함수는 0부터 시작

```
>>> list(range(5))  
[0, 1, 2, 3, 4]
```

2) 인수가 2개일 경우

- 시작 숫자와 끝 숫자
- 끝 숫자는 해당 범위에 포함되지 않음

```
>>> list(range(5, 10))  
[5, 6, 7, 8, 9] ← 끝 숫자 10은 포함되지 않음
```

3) 인수가 3개일 경우

- 세 번째 인수는 숫자 사이의 거리

```
>>> list(range(1, 10, 2))  
[1, 3, 5, 7, 9] ← 1부터 9까지, 숫자 사이의 거리는 2  
>>> list(range(0, -10, -1))  
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9] ← 0부터 -9까지, 숫자 사이의 거리는 -1
```



3. 내장 함수

내장 함수?

▪ `round(number[, ndigits])`

- 숫자를 입력받아 반올림해 주는 함수

```
>>> round(4.6)
5
>>> round(4.2)
4
```

- ndigits는 반올림하여 표시하고 싶은 소수점의 자리수

```
>>> round(5.678, 2)
5.68
```

▪ `sorted(iterable)`

- 입력값을 정렬한 후 그 결과를 리스트로 반환

```
>>> sorted([3, 1, 2])
[1, 2, 3]
>>> sorted(['a', 'c', 'b'])
['a', 'b', 'c']
>>> sorted("zero")
['e', 'o', 'r', 'z']
>>> sorted((3, 2, 1))
[1, 2, 3]
```




3. 내장 함수

내장 함수?

▪ str(object)

- 객체를 문자열 형태로 변환

```
>>> str(3)
'3'
```

```
>>> str('hi'.upper())
'HI'
```

▪ sum(iterable)

- 리스트나 튜플의 모든 요소의 합 반환

```
>>> sum([1,2,3])
6
```

```
>>> sum((4,5,6))
15
```

▪ tuple(iterable)

- Iterable 자료형을 튜플 형태로 변환

```
>>> tuple("abc")
('a', 'b', 'c')
>>> tuple([1, 2, 3])
(1, 2, 3)
>>> tuple((1, 2, 3))
(1, 2, 3)
```

3. 내장 함수

내장 함수?

▪ type(object)

- 입력값의 자료형 반환

```
>>> type("abc")
<class 'str'> ← "abc"는 문자열 자료형
>>> type([])
<class 'list'> ← []는 리스트 자료형
>>> type(open("test", 'w'))
<class '_io.TextIOWrapper'> ← 파일 자료형
```

▪ zip(*iterable)

- 동일한 개수로 이루어진 자료형을 묶어주는 역할을 하는 함수

```
>>> list(zip([1, 2, 3], [4, 5, 6]))
[(1, 4), (2, 5), (3, 6)]
>>> list(zip([1, 2, 3], [4, 5, 6], [7, 8, 9]))
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
>>> list(zip("abc", "def"))
[('a', 'd'), ('b', 'e'), ('c', 'f')]
```

『2-2』 빅데이터 분석 도구 –fundamental

- Python 계산, 변수와 자료형
- Python 자료구조
- Python's Statement
- 함수
- 객체와 클래스
- Moduler과 패키지
- 파이썬의 예외처리와 내장함수
- 파이썬 외장함수
- 라이브러리





1. Syllabus

학습목표

- 이 워크샵에서는 대략 200개가 넘는 파이썬 모듈(파이썬 외장함수(Library))을 알 수 있습니다.

눈높이 체크

- 파이썬 외장함수(Library)을 알고 계신가요?



1. 프로그램 작성 방법

스탠드얼론 프로그램

- 대화식 인터프리터 코드 작성

```
>>>print("This interpreter code works!")
```

- 첫 번째 standalone 프로그램

Sa-1st.py

```
print("This standalone program works!")
```

```
(py3_10_basic) PS C:\DEV\PycharmProjects\python_works> python .\Sa-1st.py  
This standalone program works!
```

- 두 번째 standalone 프로그램

Sa-2nd.py

```
import sys  
print('Program arguments:', sys.argv)
```

```
(py3_10_basic) PS C:\DEV\PycharmProjects\python_works> python Sa-2nd.py la la la  
Program arguments: ['Sa-2nd.py', 'la', 'la', 'la']
```

2. 모듈과 import

파이썬 모듈

- 현재 파이썬 3.0 버전에서는 대략 200개가 넘는 모듈을 지원
 - 문자열(string), 날짜(date), 시간(time), 십진법(decimal), 랜덤(random)
 - 파일(file), os, sqlite3, sys, xml, email, http 등등
- 간단하게 모듈을 사용 할 수 있음
- 모듈을 사용하는 이유
 - 코드의 재 사용성
 - 코드를 이름공간으로 구분하고 관리 할 수 있음.
 - 복잡하고 어려운 기능을 포함하는 프로그램을 간단하게 만들 수 있다.

2. 모듈과 import

파이썬 모듈


list of python modules

list of python modules




 전체

 이미지

 동영상

 뉴스

 쇼핑

 더보기

도구

검색결과 약 123,000,000개 (0.44초)



python.org

<https://docs.python.org/py-modindex>

Python Module Index — Python 3.11.2 documentation

Python Module Index ; base64, RFC 4648: Base16, Base32, Base64 Data Encodings; Base85 and Ascii85 ; bdb, Debugger framework. ; binascii, Tools for converting ...

[Testing for Python keywords](#) · [Future](#) · [Abc — Abstract Base Classes](#) · [Base64](#)

관련 질문 :

How many Python modules are there?



2. 모듈과 import

파이썬 모듈

list of python modules

Python » English » 3.11.2 » 3.11.2 Documentation » Python Module Index

Quick search | modules | index

Python Module Index

[_](#) | [a](#) | [b](#) | [c](#) | [d](#) | [e](#) | [f](#) | [g](#) | [h](#) | [i](#) | [j](#) | [k](#) | [l](#) | [m](#) | [n](#) | [o](#) | [p](#) | [q](#) | [r](#) | [s](#) | [t](#) | [u](#) | [v](#) | [w](#) | [x](#) | [z](#)

__future__	Future statement definitions
__main__	The environment where top-level code is run. Covers command-line interfaces, import-time behavior, and <code>"__name__ == '__main__'"</code> .
_thread	Low-level threading API.
a	
abc	Abstract base classes according to 'pep:3119'.
aifc	Deprecated: Read and write audio files in AIFF or AIFC format.
argparse	Command-line option and argument parsing library.
array	Space efficient arrays of uniformly typed numeric values.
ast	Abstract Syntax Tree classes and manipulation.
asynchat	Deprecated: Support for asynchronous command/response protocols.
asyncio	Asynchronous I/O.
asyncore	Deprecated: A base class for developing asynchronous socket handling services.
atexit	Register and execute cleanup functions.
audioop	Deprecated: Manipulate raw audio data.

b	
base64	RFC 4648: Base16, Base32, Base64 Data Encodings; Base85 and Ascii85
bdb	Debugger framework.
binascii	Tools for converting between binary and various ASCII-encoded binary representations.
bisect	Array bisection algorithms for binary searching.
builtins	The module that provides the built-in namespace.
bz2	Interfaces for bzip2 compression and decompression.

2. 모듈과 import

Sys

- 명령행에서 인수를 전달

구분	내용	비고
실습환경	py3_10_basic	└─MyModules └─argv_test.py
소스코드	<pre>import sys print(sys.argv)</pre>	
결과값1	<pre>(py3_10_basic) PS C:\DEV\PycharmProjects\python_works\MyModules> python .\argv_test.py I need Python. ['.\argv_test.py', 'I', 'need', 'Python.'] (py3_10_basic) PS C:\DEV\PycharmProjects\python_works\MyModules></pre>	
비고		

2. 모듈과 import

Sys

● 강제 스크립트 종료

구분	내용	비고
실습환경	py3_10_basic	MyModules argv_test.py
소스코드		
결과값1	<pre>(py3_10_basic) C:\DEV\PycharmProjects\python_works\MyModules>python Python 3.10.9 packaged by conda-forge (main, Jan 11 2023, 15:15:40) [MSC v.1916 64 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license" for more information. >>> import sys >>> sys.exit() (base) C:\DEV\python_works\MyModules></pre>	
비고		

● 자신이 만든 모듈 불러서 쓰기

```
import sys
sys.path.append("C:\DEV\PycharmProjects\python_works\MyModules>")
```

 Springer

● 시스템의 환경 변수

구분	내용	비고
실습환경	py3_10_basic	
소스코드	<pre>import os print(os.environ['PATH'])</pre>	<pre>import os print(os.environ['PATH'])</pre>
결과값1		C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbe ip\;C:\Program Files\Git\cmd;C:\Program Files\Microsof ocal\Programs\Python\Python310\;C:\Users\k8s\AppData\L
비고		

● 현재 자신의 디렉토리 위치를 돌려줌

구분	내용	비고
실습환경	py3_10_basic	
소스코드	<pre>import os print(os.getcwd())</pre>	
결과값1	c:\DEV\PycharmProjects\python_works	
비고		

2. 모듈과 import

시간 표현 방법

- 컴퓨터에서 시간 표현 방법
 - 타임스탬프(Time Stamp)
 - UTC(Universal Time Coordinated, 협정세계시)
 - 그리니치 평균시
 - LST(Local Standard Time, 지방표준시)
 - 일광절약 시간제(Daylight Saving Time, DTS)
- struct_time 시퀀스 객체
 - 사람이 이해하는 표현 방식

속성	내용	속성	내용
tm_year	년도	tm_min	분
tm_mon	월	tm_sec	초
tm_mday	일	tm_wday	요일
tm_hour	시	tm_yday	누적 날짜

2. 모듈과 import

시간 표현 방법

- time 모듈

- 시간을 표현하는데 사용
- 주요 모듈 함수

- time.time()

1970년 1월 1일 자정 이후로 누적된 초를 float 단위로 반환

- time.gmtime([secs])

UTC 기준의 struct_time 시퀀스 객체로 반환

- time.localtime([secs])

지방표준시 기준의 struct_time 시퀀스 객체를 반환

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import time print(time.time())</pre>
결과값1	1680394800.7794971
비고	

2. 모듈과 import

시간 표현 방법

- time 모듈

- 알아보기 쉬운 날짜와 시간 형태의 값을 반환하여 주는 함수
- time.sleep 함수는 보통 루프 안에서 많이 쓰이는데 일정한 시간 간격을 주기위해서 주로 사용됨

구분	내용
실습환경	py3_10_basic print(time.ctime()) #알아보기 쉬운 날짜와 시간 형태의 값을 반환하여 주는 함수
소스코드	# time.sleep 함수는 보통 루프 안에서 많이 쓰이는데 일정한 시간 간격을 주기 위해서 주로 사용됨 for i in range(10): print(i) time.sleep(1)
결과값1	Sun Apr 2 09:22:29 2023
결과값1	0 1 2 3 4 5 6 7 8 9
비고	

2. 모듈과 import

시간 표현 방법

- datetime 모듈

- 기념일과 같은 날짜, 시간 연산을 위하여 사용
- 주요 클래스

- datetime.date

일반적으로 사용되는 그레고리안 달력(Gregorian Calendar)의
년, 월, 일을 표현

- datetime.time

시간을 시, 분, 초, 마이크로 초, 시간대(Time zone)로 표현

- datetime.datetime

date 클래스와 time 클래스의 조합으로 구성

- datetime.timedelta

두 날짜 혹은 시간 사이의 기간을 표현

2. 모듈과 import

시간 표현 방법

- datetime 모듈

- date 객체와 datetime 객체 간 차이 > ' 오늘 날짜와 년, 월, 일 요소 출력 '

구분	내용
	py3_10_basic
실습환경	from math import exp, log, sqrt import re from datetime import date, time, datetime, timedelta
소스코드	today = date.today() print("Output #1: today: {0!s}".format(today)) print("Output #2: {0!s}".format(today.year)) print("Output #3: {0!s}".format(today.month)) print("Output #4: {0!s}".format(today.day)) current_datetime = datetime.today() print("Output #5: {0!s}".format(current_datetime))
결과값1	Output #1: today: 2023-04-02 Output #2: 2023 Output #3: 4 Output #4: 2 Output #5: 2023-04-02 09:26:11.583746
비고	

2. 모듈과 import

시간 표현 방법

- timedelta 함수를 사용하여 날짜 객체에 특정 시간을 더하거나 빼는 방법

구분	내용
	py3_10_basic
실습환경	<pre>from math import exp, log, sqrt import re from datetime import date, time, datetime, timedelta</pre>
소스코드	<pre>today = date.today() one_day = timedelta(days=-1) yesterday = today + one_day print("Output #1: yesterday: {0!s}".format(yesterday)) eight_hours = timedelta(hours=-8) print("Output #2: {0!s} {1!s}".format(eight_hours.days, eight_hours.seconds))</pre>
결과값1	<pre>Output #1: yesterday: 2023-04-01 Output #2: -1 57600</pre>
비고	

2. 모듈과 import

시간 표현 방법

- `strftime` 함수를 사용하여 날짜 개체에서 특정 형식의 문자열

구분	내용
	<code>py3_10_basic</code>
실행환경	<code>from math import exp, log, sqrt</code> <code>import re</code> <code>from datetime import date, time, datetime, timedelta</code>
소스코드	<code>today = date.today()</code> <code>print("Output #1: {:s}".format(today.strftime('%m/%d/%Y')))</code> <code>print("Output #2: {:s}".format(today.strftime('%b %d, %Y')))</code> <code>print("Output #3: {:s}".format(today.strftime('%Y-%m-%d')))</code> <code>print("Output #4: {:s}".format(today.strftime('%B %d, %Y')))</code>
결과값1	Output #1: 04/02/2023 Output #2: Apr 02, 2023 Output #3: 2023-04-02 Output #4: April 02, 2023
비고	

2. 모듈과 import

시간 표현 방법

구분	내용
	py3_10_basic
실습환경	<pre>from math import exp, log, sqrt import re from datetime import date, time, datetime, timedelta</pre>
소스코드	<pre>today = date.today() # 날짜를 나타내는 문자열에서 # 특정 형식의 datetime 객체 생성 date1 = today.strftime('%m/%d/%Y') date2 = today.strftime('%b %d, %Y') date3 = today.strftime('%Y-%m-%d') date4 = today.strftime('%B %d, %Y') # 두 개의 datetime 객체와 두 개의 날짜 객체 # 날짜 형식이 다른 4개의 문자열 기반 print("Output #1: {!s}".format(datetime.strptime(date1, '%m/%d/%Y'))) print("Output #2: {!s}".format(datetime.strptime(date2, '%b %d, %Y'))) # 날짜 부분만 표시 print("Output #3: {!s}".format(datetime.date(datetime.strptime(date3, '%Y-%m-%d')))) print("Output #4: {!s}".format(datetime.date(datetime.strptime(date4, '%B %d, %Y'))))</pre>
결과값1	<pre>Output #1: 2023-04-02 00:00:00 Output #2: 2023-04-02 00:00:00 Output #3: 2023-04-02 Output #4: 2023-04-0</pre>
비고	

2. 모듈과 import

시간 표현 방법

- Quiz

문제 : 덧셈을 1억번 수행하는데 몇 초가 걸리는지 한번 체크
0부터 시작해서 1, 2, 3, ..., 99,999,999까지 모두 더하는 코드입니다. 위 코드를 실행했더니, 사람이 직접 1억번의 덧셈을 하려면 한 번의 덧셈에 1초씩 걸린다고 쳐도

1억초 = 약 1666667분 = 약 27778시간 = 약 1157일 = 약 3년

이 걸립니다.

컴퓨터로 하면 얼마나 걸릴까요?

2. 모듈과 import

시간 표현 방법

- Quiz

구분	내용
실행환경	py3_10_basic
소스코드	<pre>import timeit start_time = timeit.default_timer() sum = 0 for i in range(100000000): sum += i terminate_time = timeit.default_timer() print("%f초 걸렸습니다." % (terminate_time - start_time))</pre>
결과값1	12.305259초 걸렸습니다.
비고	

2. 모듈과 import

Random

- 임의의 객체 중 임의의 값을 선택
 - 임의의 정수, 실수를 생성

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import random print(random.random()) print(random.uniform(3,4)) print([random.randrange(20) for i in range(10)]) L1=list(range(10)) print(L1) print(random.randint(1,10))</pre>
결과값1	<pre>0.7334947729109645 3.427641308774751 [5, 4, 17, 17, 8, 14, 12, 6, 16, 14] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] 9</pre>
비고	<p>0.0 ~ 1.0 미만의 임의의 값 생성 괄호 안 두 수 사이의 실수 중에서 난수값을 리턴 random.randrange(1, 46) 1 ~ 46 미만의 임의의 값 1에서 10사이의 정수사이에서 난수값을 돌려줌</p>

2. 모듈과 import

Random 연습

1. lotto 번호 추출

구분	내용
실습환경	py3_10_basic
소스코드	<pre>from random import * print(int(random() * 45) + 1) print(randrange(1, 46)) print(randint(1, 45))</pre>
결과값1	9 17 11 30 36 42
비고	

2. 모듈과 import

Random 연습

1. lotto 번호 추출-중복되지 않은 6자리

구분	내용
실습환경	py3_10_basic import random
소스코드	<pre>lotto = [] rnd_num = random.randint(1, 45) for i in range(6): while rnd_num in lotto: rnd_num = random.randint(1, 45) lotto.append(rnd_num) lotto.sort() print("로또번호: {}".format(lotto))</pre>
결과값1	로또번호: [3, 21, 27, 35, 37, 44]
비고	

2. 모듈과 import

Random 연습

1. lotto 번호 추출-Python random 라이브러리의 sample() 함수를 사용

구분	내용
실습환경	py3_10_basic import random
소스코드	lotto = random.sample(range(1, 45), 6) print("로또번호: {}".format(lotto))
결과값1	로또번호: [37, 4, 29, 35, 42, 16]
비고	

1. lotto 번호 추출-파이썬의 numpy 라이브러리를 사용

구분	내용
실습환경	py3_10_basic 실습환경 구성 후 ... 수행해 보기 import numpy
소스코드	lotto = numpy.random.choice(range(1, 46), 6, replace=False) print("로또번호: {}".format(lotto))
결과값1	로또번호: [37 9 18 4 21 38]
비고	

2. 모듈과 import

Random 연습

1. lotto 번호 추출-[4, 5, 6] 을 포함한 로또 번호 생성

구분	내용
실습환경	py3_10_basic import random def fill_random_number(numbers: list): rand_num = random.randint(1, 45) for i in range(len(numbers), 6): while rand_num in numbers: rand_num = random.randint(1, 45) numbers.append(rand_num) numbers.sort()
소스코드	def make_lotto(**kwargs): lotto = [] if kwargs.get("include"): include = kwargs.get("include") lotto.extend(include) fill_random_number(lotto) return lotto print("로또번호: {}".format(make_lotto(include=[4,5,6])))
결과값1	로또번호:[4, 5, 6, 22, 35, 44]
비고	중복되지 않은 수로 빈자리를 채워주는 fill_random_number 라는 이름의 함수를 하나 작성

2. 모듈과 import

Random 연습

1. lotto 번호 추출-원하지 않은 숫자를 제외한 파이썬 로또 번호 생성

구분	내용
실습환경	<pre>py3_10_basic import random def fill_random_number(numbers: list): rand_num = random.randint(1, 45) for i in range(len(numbers), 6): while rand_num in numbers: rand_num = random.randint(1, 45) numbers.append(rand_num) numbers.sort()</pre>
소스코드	<pre>def make_lotto(**kwargs): lotto = [] if kwargs.get("exclude"): exclude = kwargs.get("exclude") while(len(lotto) != 6): fill_random_number(lotto) lotto = list(set(lotto) - set(exclude)) return lotto print("로또번호: {}".format(make_lotto(exclude =[4,5,6])))</pre>
결과값1	로또번호:[1, 3, 12, 44, 14, 27]
비고	파이썬의 집합 특성을 이용해서 집합 A - 집합 B = 차집합을 구해 구현

2. 모듈과 import

Random

- Quiz
- 가위 바위 보 게임
 - 사용자가 가위, 바위, 보 중에서 하나를 선택하고 컴퓨터도 난수로 가위, 바위, 보 중에서 하나를 선택한다. 사용자의 선택과 컴퓨터의 선택을 비교하여서 승패를 화면에 출력한다.
- 구현 화면

(가위, 바위, 보) 중에서 하나를 선택하세요: 가위
사용자: 가위 컴퓨터: 바위
컴퓨터가 이겼음!

2. 모듈과 import

Random

● Quiz 답

구분

내용

실습환경

py3_10_basic

```
import random;

player = input("(가위, 바위, 보) 중에서 하나를 선택하세요: ");

number = random.randint(0,2);
if (number == 0):
    computer = "가위";
elif (number == 1):
    computer = "바위";
else:
    computer = "보";
print("사용자: ", player, "컴퓨터: ", computer)
```

소스코드

```
if (player == computer):
    print("비겼음!");
elif (player == "바위"):
    if (computer == "보"):
        print("컴퓨터가 이겼음!");
    else:
        print("사용자가 이겼음!");
elif (player == "보"):
    if (computer == "바위"):
        print("사용자가 이겼음!");
    else:
        print("컴퓨터가 이겼음!");
elif (player == "가위"):
    if (computer == "바위"):
        print("컴퓨터가 이겼음!");
    else:
        print("사용자가 이겼음!");
```

결과값1

(가위, 바위, 보) 중에서 하나를 선택하세요: 가위
사용자: 가위 컴퓨터: 바위
컴퓨터가 이겼음!

비고

2. 모듈과 import

Quiz

- 파이썬 코딩 대회를 주최합니다.
 - 참석률을 높이기 위해 댓글 이벤트를 진행하기로 하였습니다.
 - 댓글 작성자들 중에 추첨을 통해 1명은 치킨, 3명은 커피 쿠폰을 받게 됩니다.
 - 추첨 프로그램을 작성하시오.
1. 조건1 : 편의상 댓글은 20명이 작성하였고 아이디는 1~20이라고 가정
 2. 조건2 : 댓글 내용과 상관 없이 무작위로 추첨하되 중복은 불가
 3. 조건3 : random 모듈의 shuffle 과 sample 을 활용

(출력 예제)
-- 당첨자 발표 --
치킨 당첨자 : 1
커피 당첨자 : [2, 3, 4]
-- 축하합니다 --

2. 모듈과 import

답

구분	내용
실습환경	<pre>py3_10_basic from random import *</pre>
소스코드	<pre>users = range(1, 21) users = list(users) shuffle(users) winners = sample(users, 4) print("-- 당첨자 발표 -- ") print("치킨 당첨자 : {0}".format(winners[0])) print("커피 당첨자 : {0}".format(winners[1:])) print("-- 축하합니다 --")</pre>
결과값1	<pre>-- 당첨자 발표 -- 치킨 당첨자 : 17 커피 당첨자 : [15, 9, 14] -- 축하합니다 --</pre>
비고	

2. 모듈과 import

Math

- math 모듈은 삼각함수, 제곱근, 로그함수 등 수학과 관련된 기능이 들어 있는 내장 모듈.
- dir() 함수를 이용해 모듈에 어떠한 함수 혹은 데이터가 들어 있는지 알 수 있다.

구분	내용
실행환경	py3_10_basic import math
소스코드	print(dir(math)) print(math.pow(2, 10)) print(math.pi)
결과값1	['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc'] 1024.0 3.141592653589793
비고	

2. 모듈과 import

Math

함수(Function)	설명(Description)	사용 예(Example)
<code>abs(x)</code>	값 x의 절대값을 반환한다.	<code>abs(-2) → 2</code>
<code>max(x1, x2, ...)</code>	x1, x2, ... 중 가장 큰 값을 반환한다.	<code>max(1, 5, 2) → 5</code>
<code>min(x1, x2, ...)</code>	x1, x2, ... 중 가장 작은 값을 반환한다.	<code>min(1, 5, 2) → 1</code>
<code>pow(a, b)</code>	a^b 을 반환한다. (= <code>a**b</code>)	<code>pow(2,3) → 8</code>
<code>round(x)</code>	소수점을 가까운 정수까지 반올림, 버림하여 반환한다. 소수점이 5인 경우 앞자리가 짝수면 버리고 홀수면 반올림 한다.	<code>round(5.4) → 5</code> , <code>round(5.6) → 6</code>
<code>round(x, n)</code>	소수점 아래 n번째 이하를 반올림, 버림한 뒤 반환한다.	<code>round(5.466, 2) → 5.47</code>

```
print(abs(-3))      # → 3
print(abs(-3.5))    # → 3.5
```

```
print(max(2, 3, 4, 6))  # → 6
print(min(2, 3, 4))     # → 2
```

```
print(pow(2, 3))       # → 8 (2^3)
```

```
print(round(3.51))      # → 4 (정수 반올림)
print(round(3.1456, 3)) # → 3.146 (소수점 셋째 자리 반올림)
```

2. 모듈과 import

Math

함수(Function)	설명(Description)	사용 예(Example)
<code>fabs(x)</code>	실수 x 의 절대값을 반환	<code>fabs(-2) → 2.0</code>
<code>ceil(x)</code>	x 를 가까운 정수까지 올림 한 뒤 반환한다.	<code>ceil(2.1) → 3</code> , <code>ceil(-2.1) → -2</code>
<code>floor(x)</code>	x 를 가까운 정수까지 버림 한 뒤 반환한다.	<code>floor(2.1) → 2</code> , <code>ceil(-2.1) → -3</code>
<code>exp(x)</code>	e^x 의 값을 반환한다.	<code>exp(1) → 2.71828</code>
<code>log(x)</code>	자연로그(natural logarithm) x 의 값을 반환한다.	<code>log(2.71828) → 1.0</code>
<code>log(x, base)</code>	밑을 $base$ 로 하는 로그 x 의 값을 반환한다.	<code>log(100, 10) → 2.0</code>
<code>sqrt(x)</code>	x 의 제곱근을 반환한다.	<code>sqrt(4.0) → 2</code>
<code>sin(x)</code>	x 의 사인(sin) 값을 반환한다. x 는 라디안으로 나타낸다.	<code>sin(3.14159/2) → 1</code> , <code>sin(3.14159) → 0</code>
<code>asin(x)</code>	x 의 역사인(asin) 값을 반환한다.	<code>asin(1.0) → 1.57</code> , <code>asin(0.5) → 0.523599</code>
<code>cos(x)</code>	x 의 코사인(cos) 값을 반환한다. x 는 라디안으로 나타낸다.	<code>cos(3.14159/2) → 0</code> , <code>cos(3.14159) → -1</code>
<code>acos(x)</code>	x 의 역코사인 값을 반환한다.	<code>acos(1.0) → 0</code> , <code>acos(0.5) → 1.0472</code>
<code>tan(x)</code>	x 의 탄젠트(tan) 값을 반환한다. x 는 라디안으로 나타낸다.	<code>tan(3.14159/4) → 1</code> , <code>tan(0.0) → 0</code>
<code>degrees(x)</code>	각 x 를 라디안(radians)에서 각도로(degrees)로 바꾼다.	<code>degrees(1.57) → 90</code>
<code>radians(x)</code>	각도 x 를 라디안으로 바꾼다.	<code>radians(90) → 1.57</code>

2. 모듈과 import

Math

import math

지수, 로그, 제곱근

```
print("exp(1.0)      = ", math.exp(1))
print("log(e)        = ", math.log(math.e))
print("log10(10,10)  = ", math.log(10, 10))
print("sqrt(4.0)     = ", math.sqrt(4.0))
```

```
# e^1
# ln(e)
# log_10(10)
# √4
```

삼각함수

```
print("sin(π/2)      = ", math.sin(math.pi / 2))
print("cos(π/2)      = ", math.cos(math.pi / 2))
print("tan(π/2)      = ", math.tan(math.pi / 2))
```

```
# sin(90°)
# cos(90°)
# tan(90°) → 매우 큰 값
```

각도 변환

```
print("degrees(1.57) = ", math.degrees(1.57))
print("radians(90)   = ", math.radians(90))
```

```
# rad → deg
# deg → rad
```

```
exp(1.0)      = 2.718281828459045
log(e)        = 1.0
log10(10,10)  = 1.0
sqrt(4.0)     = 2.0
sin(π/2)      = 1.0
cos(π/2)      = 6.123233995736766e-17
tan(π/2)      = 1.633123935319537e+16
degrees(1.57) = 89.95437383553924
radians(90)   = 1.5707963267948966
```

『2-2』

빅데이터 분석 도구 -fundamental

- Python 계산, 변수와 자료형
- Python 자료구조
- Python's Statement
- 함수
- 객체와 클래스
- Moduler과 패키지
- 파이썬의 예외처리와 내장함수
- 파이썬 외장함수
- 라이브러리





1. Syllabus

학습목표

- 이 워크샵에서는 애플리케이션에 필요한 라이브러리를 선정할 수 있습니다.
- 기타 외부 라이브러리(3rd Party Library)

눈높이 체크

- 파이썬 표준 라이브러리(Standard Library)를 알고 계신가요?
- 파이썬 외부 라이브러리(3rd Party Library)를 알고 계신가요?



1. 라이브러리 개요

1. 라이브러리란?

- 라이브러리는 프로그램을 효율적으로 개발할 수 있도록 필요한 기능이 구현된 프로그램을 모은 집합체이며, 일반적으로 설치 파일과 도움말, 예시 코드 등을 제공합니다.
- 1. 라이브러리 개념과 목적
 - 라이브러리란 영어로 도서관을 의미하며 코드의 재사용 및 부품화, 기술 유출 방지를 위하여 하나 이상의 Subroutine 혹은 Function들의 집합으로 일반적으로 컴파일되어 실행이 가능한 Object Code 형태로 제공됩니다.
- 2. 외부 라이브러리 개요 구성
 - 일반적으로 사용 방법을 위한 문서와 도움말, 예시 코드, 함수, 클래스, 값, 자료형 사양을 포함할 수 있습니다.
 - ① 도움말: 라이브러리에 대한 설명 및 사용 방법 등에 대한 도움말 문서
 - ② 설치 파일: 라이브러리를 연동하기 위해 제공되는 설치 파일
 - ③ 예시 코드: 애플리케이션 구현 시 라이브러리를 쉽게 연동할 수 있도록 제공되는 샘플 코드



1. 라이브러리 개요

2. 라이브러리 개요의 유형

1. 별도 설치 여부에 따른 유형

- 표준 라이브러리(Standard Library)

- 프로그래밍 언어와 함께 제공되는 라이브러리로 표준 라이브러리를 이용하면 별도의 파일 설치 없이 다양한 기능을 이용할 수 있습니다. 각 프로그래밍 언어의 표준 라이브러리는 기본 기능 이외에 날짜와 시간 등의 다양한 추가 기능을 이용할 수 있는 API를 제공합니다.

- 외부 라이브러리(3rd Party Library)

- 별도의 파일을 설치하여 사용할 수 있는 라이브러리를 의미합니다. 누구나 개발하여 사용하거나 공유할 수 있고, 관련 커뮤니티나 인터넷 검색 등을 이용하여 공유된 라이브러리를 사용할 수 있습니다.



1. 라이브러리 개요

2. 라이브러리 개요의 유형

2. 코드 결합 방식에 따른 유형

- 정적 라이브러리(Static Library)

- 프로그램을 빌드할 때 라이브러리가 제공하는 코드를 실행 파일에 넣는 방식으로 컴파일러가 원시 소스를 컴파일할 때 참조되는 모듈입니다. 동작 코드가 실행 바이너리에 포함되어 별도의 추가 작업 없이 독립적으로 라이브러리 함수 사용이 가능하다.

- 동적 라이브러리(Dynamic Library)

- 공통적으로 필요한 기능들을 프로그램과 분리하여 필요시에만 호출하여 사용할 수 있도록 만든 라이브러리를 의미합니다. 필요할 때만 해당 라이브러리를 메모리로 불러들일 수 있어 실행 파일의 크기를 줄일 수 있고, 사용이 끝나면 메모리에서 제거되어 효율적인 메모리 사용이 가능하다. 윈도우에서는 주로 .dll 확장자, 리눅스에서는 주로 .so 확장자를 가진다.

1. 라이브러리 개요

3. 라이브러리, 프레임워크, 아키텍처, 플랫폼

- IT에서 일반적으로 사용되는 개발 용어인 프레임워크, 아키텍처, 플랫폼의 차이점과 예시를 통해 용어들 간의 상관관계를 확인할 수 있습니다.

구분	설명	예시
Library	코드 재사용 및 부품화를 위하여 필요한 기능에서 호출하여 사용할 수 있도록 제공되는 모듈의 집합	jQuery, DLL, Class, Jar 등
Framework	응용프로그램 표준 구조를 구현하기 위한 클래스와 라이브러리 모임	Spring, Django 등
Archtecture	여러 가지 컴퓨터 구성요소들에 대한 전반적인 기계적 구조와 이를 설계하는 방법	모노리틱 아키텍처, MSA 등
Platform	여러 가지 기능을 제공하는 프로그램 실행이 가능한 공통 실행환경으로 플랫폼 위에 다른 플랫폼이 존재할 수 있음	Window, Linux, JVM 등

1. 라이브러리 개요

4. 애플리케이션에 필요한 라이브러리 선정

- 프로그래밍 언어에서 제공하는 표준 라이브러리를 검색합니다.
 - 일반적인 프로그래밍 언어의 경우 날짜 및 시간, 수학 연산 및 통계, 다양한 입출력, 멀티미디어 등과 같은 다양한 라이브러리를 제공합니다.

주요 기능	설명
문자열 연산	일반적인 문자열의 조작(합치기, 자르기 등)을 처리
날짜 및 시간	현재 시각 및 날짜, 시간 등에 대한 처리
수학 모듈	수학 함수 및 통계 함수를 이용한 계산
운영체제	현재 디렉터리 등 운영체제 인터페이스 처리
파일/디렉터리	파일 조회 및 생성, 시스템 경로 조작
프로토콜	HTTP, SMTP, FTP 등의 프로토콜을 사용
멀티미디어	사운드 및 비디오 같은 멀티미디어 데이터를 처리

1. 라이브러리 개요

4. 애플리케이션에 필요한 라이브러리 선정

- 파이썬의 표준 라이브러리 문서를 확인합니다.
 - 파이썬의 경우 <https://docs.python.org/>에 접속하여 좌측 상단의 언어와 파이썬 버전을 변경하여 라이브러리 문서 확인이 가능하다.

Python » Korean 3.11.2 3.11.2 Documentation » 빠른 검색 이동 | 모듈 | 색인

내려받기

이 문서 내려받기

버전별 설명서

Python 3.12 (in development)
Python 3.11 (stable)
Python 3.10 (stable)
Python 3.9 (security-fixes)
Python 3.8 (security-fixes)
Python 3.7 (security-fixes)
Python 3.6 (EOL)
Python 3.5 (EOL)
Python 2.7 (EOL)
모든 버전

기타 자원

PEP 색인
초보자 가이드
도서 목록
오디오/비디오 토크
파이썬 개발자 지침서

Python 3.11.2 문서

Welcome! This is the official documentation for Python 3.11.2.

설명서의 파트들:

파이썬 3.11의 새로운 기능은?

2.0 이후의 모든 "새로운 기능" 문서

자습서

여기에서 시작하세요

라이브러리 레퍼런스

배개 밑에 넣어 두세요

언어 레퍼런스

문법과 언어 요소들을 설명합니다

파이썬 설정 및 사용법

여러 플랫폼에서 파이썬을 사용하는 법

파이썬 HOWTO

특정 주제에 대한 심층적인 문서

파이썬 모듈 설치하기

파이썬 패키지 색인 및 기타 소스에서 설치하기

파이썬 모듈 배포하기

다른 사람들이 설치할 수 있도록 모듈을 게시하기

확장 및 내장

C/C++ 프로그래머를 위한 자습서

파이썬/C API

C/C++ 프로그래머를 위한 레퍼런스

FAQs

자주 나오는 질문들 (답도 있습니다!)

색인 및 표:

전체 모듈 색인

모든 모듈에 빠르게 액세스합니다

일반 색인

모든 함수, 클래스, 용어

용어집

가장 중요한 용어들을 설명합니다

검색 페이지

이 문서를 검색합니다

완전한 목차

모든 섹션들과 서브섹션들을 나열합니다

출처 : 파이썬 표준 라이브러리(<https://docs.python.org/>). 20213 04. 02. 스크린샷

1. 라이브러리 개요

4. 애플리케이션에 필요한 라이브러리 선정

- 외부 라이브러리의 라이선스를 파악합니다.
 - OSS(Open Source Software)의 경우 일반적으로는 자유롭게 사용 및 복제, 배포가 가능하나, 라이선스에 따라 상업적 목적으로 이용할 수 없거나 배포 시 소스코드를 공개해야 할 수 있으니 라이선스 정책을 반드시 확인합니다.

라이선스	설명
GNU GPL 2.0(GNU General Public License)	자유 소프트웨어 재단에서 만든 라이선스로 GNU 프로젝트에서 배포한 프로그램을 사용하기 위해 작성되었으며 많은 오픈소스 소프트웨어가 채택하고 있는 라이선스. 가장 많이 알려져 있고 의무사항들도 타 라이선스에 비해 엄격한 편이며, 배포하는 경우 저작권 표시, 보증책임이 없다는 표시 및 GPL에 의해 배포된다는 사실 명시 필요
GNU Lesser GPL(LGPL) 2.1	GPL 라이선스를 사용하기만 해도 소스코드를 공개해야 합니다.는 부담 때문에 라이브러리와 모듈로의 링크를 허용한 라이선스. 배포하는 경우, 저작권 표시, 보증책임이 없다는 표시 및 LGPL에 의해 배포된다는 사실 명시 필요
Apache Licens	아파치 재단(ASF: Apache Software Foundation)의 모든 소프트웨어에 적용되며 BSD 라이선스와 비슷하여 소스코드 공개 등의 의무 없음. "Apache"라는 이름에 대한 상표권을 침해하지 않아야 합니다.는 조항이 명시적으로 포함
MIT Lisense	미국 매사추세츠공과대학교(MIT)에서 소프트웨어 공학도들을 돕기 위해 개발한 라이선스로 소프트웨어를 누구라도 무상으로 제한 없이 취급 가능
Berkeley Software Distribution(BSD) License	GPL/LGPL보다 덜 제한적이기 때문에 허용 범위가 넓으며, 가장 큰 차이점은 소스코드의 공개 의무가 없음. 수정 프로그램에 대한 소스코드의 공개를 요구하지 않기 때문에 상용 소프트웨어에 무제한 사용 가능함



1. 라이브러리 개요

4. 애플리케이션에 필요한 라이브러리 선정

- 라이선스의 정책을 확인합니다.
 - 공개 SW를 이용하는 경우 공개 SW 개발자가 지정한 조건의 범위에 따라 해당 SW를 이용해야 하며, 위반하는 경우 라이선스 위반 및 저작권 침해에 따른 법적 책임을 져야 합니다.

라이선스	무료 이용 가능	배포 허용 가능	소스코드 수정 가능	2차적 저작물 재공개 의무	독점 SW와 결합 가능
GPL	O	O	O	O	X
LGPL	O	O	O	O	O
MPL	O	O	O	O	O
BSD License	O	O	O	X	O
Apache license	O	O	O	X	O



1. 라이브러리 개요

4. 애플리케이션에 필요한 라이브러리 선정

- 라이브러리를 평가합니다.
 - 라이브러리에 대한 라이선스(이용, 배포, SW 공개 의무 등)와 제공 커뮤니티의 지원 및 활성화, 문서화 정도 등의 라이브러리 자체 영역과 요구 사항 만족 여부 및 업무적 기능 구현 가능성, 시스템의 특성, 프로젝트 상황, 유지관리 효율성 등의 프로젝트 영역을 종합적으로 검토하여 후보 라이브러리들에 대한 평가를 진행합니다. 평가 항목 및 방법, 점수에 대한 가중치는 프로젝트 상황을 고려하여 적용합니다.

평가 영역	평가 항목	항목 설명	점수
라이선스	활용	무료 이용 가능 및 상업적 활용 가능 여부	5
	배포	라이브러리 적용 SW에 대한 배포 가능 여부	3
	공개	소스코드 공개 의무 여부	1



1. 라이브러리 개요

4. 애플리케이션에 필요한 라이브러리 선정

● 라이브러리를 평가합니다.

평가 영역	평가 항목	항목 설명	점수
커뮤니티	버전	다양한 OS 및 버전 지원, 릴리즈 주기에 대한 적합성	4
	지원	개발자의 전문성, 질의에 대한 회신 여부	5
	활성화	참여자들의 활동 현황 및 다운로드, 점유율 등	5
	문서화	설치 및 사용법 등에 대한 문서 종류와 포맷	3
개발/관리	기능성	라이브러리 적용 시 요구사항 충족 여부	5
	신뢰성	오류 발생에 대한 추적 및 복구에 대한 용이성	3
	사용성	모듈 설치 및 관리, 인터페이스 활용의 용이성	4
	성능	성능 이슈에 대한 점검 및 점검 결과 제공 여부	3
	이식성	다양한 환경에 대한 적응성 제공 여부	4
	유지관리성	주석 확인 및 소스 수정 가능 여부	5
	보안성	보안 취약점에 대한 이슈 및 점검, 패치 적용 여부	4



1. 라이브러리 개요

4. 애플리케이션에 필요한 라이브러리 선정

- 라이브러리를 선정합니다.
 - 후보 라이브러리 중 가장 높은 점수의 라이브러리를 선정하고, 표준 라이브러리와 외부 라이브러리가 모두 존재하는 경우에는 별도의 외부 모듈 설치가 필요하지 않고 자체적으로 처리가 가능한 표준 라이브러리를 우선 적용합니다.
- 외부 라이브러리 사용 목록 작성 예시

라이브러리	유형	라이선스	수정 여부	사용 용도	담당자
commons-fileupload-1.2.1	jar	Apache1.1	X	파일 업로드	김갑○
commons-lang-2.0	jar	Apache2.0	X	유틸리티	김을○
CKEditor(4.5.6)	JS	GPL,LGPL	X	공지사항 관리	김병○
Mwphotobrowser	java	MIT	O	열람 뷰어	김정○



1. 라이브러리 개요

5. 라이브러리 구성 및 적용

- 모듈과 패키지

- 라이브러리는 모듈과 패키지를 모두 포함하며, 모듈의 경우 개별 파일을 지칭하고 패키지의 경우 여러 개의 파일을 모아놓은 폴더로 생각할 수 있습니다.

구분	설명	사용법
Module	한 개의 파일에서 기능을 제공합니다.	import (모듈명)
Package	여러 개의 모듈을 한 개의 폴더에 묶어서 기능을 제공합니다. 패키지명과 모듈명을 import하여 불러올 수 있습니다.	import (패키지명).(모듈명)



1. 라이브러리 개요

5. 라이브러리 구성 및 적용

- PyPI(Python Package Index)
 - Python Package Index는 파이썬 패키지들이 모여 있는 저장소를 의미하고, Python 개발자라면 누구에게나 오픈되어 있습니다. PyPI에서 권한 있는 Python 라이브러리 설치 툴이 pip이며, 파이썬 3.4 이후의 버전에는 기본으로 포함되어 있습니다.

사용법	의미
pip list	설치된 패키지를 표시합니다.
pip install <package name>	패키지를 설치합니다.
pip install --upgrade <package name>	설치된 패키지를 업그레이드합니다.
pip uninstall <package name>	설치된 패키지를 제거합니다.
pip --help	pip와 관련된 명령어를 조회합니다.



1. 라이브러리 개요

5. 라이브러리 구성 및 적용

- 파이썬의 표준 라이브러리 적용 방법을 파악합니다.
 - 한 모듈에 있는 파이썬 코드는 임포팅이라는 프로세스를 통해 다른 모듈에 있는 코드들에 대한 접근권을 획득할 수 있으며, import문 사용 방법에 따라 애플리케이션에서 라이브러리 호출 및 사용을 위한 소스코드가 달라질 수 있습니다.

적용 방법	의미	사용 코드
import os	os 모듈 전체를 불러와 사용	os.getcwd()
from os import *	os 모듈 전체를 불러오는 것은 동일하나 코드는 변수/함수만 사용	getcwd()
from os import getcwd	os 모듈 내의 특정 함수(getcwd)만 불러 와 사용	getcwd()



1. 라이브러리 개요

6. 3rd Party Library

- Numpy 넘파이

- 파이썬으로 과학 계산을 하려면 꼭 필요한 패키지.
- 다차원 배열을 위한 기능과 선형 대수 연산과 푸리에 변환 같은 고수준 수학 함수와 유사 난수 생성기를 포함
- 다음 명령으로 설치합니다.

```
pip install numpy
```

- 임포트할 때는 보통 np라는 별명으로 임포트합니다.

```
import numpy as np
```

1. 라이브러리 개요

6. 3rd Party Library

- Scipy 싸이파이
 - 과학 계산용 함수를 모아놓은 파이썬 패키지.
 - 고성능 선형 대수, 함수 최적화, 신호 처리, 특수한 수학 함수와 통계 분포등을 포함한 많은 기능 제공다음 명령으로 설치한다.

`pip install scipy`

- 임포트할 때는 보통 `sp`라는 별명으로 임포트한다.

`import scipy as sp`

1. 라이브러리 개요

6. 3rd Party Library

- matplotlib 맷플랏립

- 대표적인 과학 계산용 그래프 라이브러리.
- 선 그래프, 히스토그램, 산점도 등을 지원하며 고품질 그래프를 그려준다.
- 대표적인 과학 계산용 그래프 라이브러리.
- 선 그래프, 히스토그램, 산점도 등을 지원하며 고품질 그래프를 그려준다.
- 다음 명령으로 설치한다.

`pip install matplotlib`

- 임포트할 때는 보통 `pylab` 서브패키지를 `plt`라는 별명으로 임포트한다.

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

1. 라이브러리 개요

6. 3rd Party Library

- pandas 판다스
 - 데이터 처리와 분석을 위한 파이썬 라이브러리.
 - R의 data.frame을 본떠서 설계한 DataFrame이라는 데이터 구조를 기반으로 만들어졌다.
 - 엑셀의 스프레드시트와 비슷한 테이블 형태.
 - SQL처럼 테이블에 쿼리나 조인을 수행할 수 있습니다.
 - SQL, 엑셀 파일, CSV 파일 같은 다양한 파일과 DB에서 읽어들이 수 있습니다.
 - 다음 명령으로 설치한다.

`pip install pandas`

- 임포트할 때는 보통 pd라는 별명으로 임포트한다.

`import pandas as pd`



1. 라이브러리 개요

6. 3rd Party Library

- seaborn 패키지
 - seaborn("시본"이라고 읽는다) 패키지는 맷플롯립 패키지에서 지원하지 않는 고급 통계 차트를 그리는 통계용 시각화 기능을 제공한다.
 - 다음 명령으로 설치한다.

`pip install seaborn`

- импорт할 때는 보통 `sns`라는 별명으로 импорт한다.

`import seaborn as sns`

1. 라이브러리 개요

6. 3rd Party Library

- statsmodels 패키지
 - statsmodels("스탯츠모델즈"라고 읽는다) 패키지는 추정 및 검정, 회귀 분석, 시계열분석 등의 기능을 제공하는 파이썬 패키지다. 기존에 R에서 가능했던 다양한 회귀분석과 시계열분석 방법론을 그대로 파이썬에서 이용할 수 있습니다. 다음은 statsmodels 패키지가 제공하는 기능의 일부다.
 - 검정 및 모수추정
 - 회귀분석
 - ① 선형회귀
 - ② 강건회귀
 - ③ 일반화 선형모형
 - ④ 혼합효과모형
 - ⑤ 이산종속변수
 - 시계열 분석
 - ① SARIMAX 모형
 - ② 상태공간 모형
 - ③ 벡터 AR 모형
 - 생존분석
 - 요인분석



1. 라이브러리 개요

6. 3rd Party Library

- statsmodels 패키지

- 다음 명령으로 설치한다.

```
pip install statsmodels
```

- импорт할 때는 보통 api 서브패키지를 sm이라는 별명으로 импорт한다.

```
import statsmodels.api as sm
```



1. 라이브러리 개요

6. 3rd Party Library

- scikit-learn 패키지

- scikit-learn("사이킷런"이라고 읽는다) 패키지는 머신러닝 교육을 위한 최고의 파이썬 패키지다. scikit-learn 패키지의 장점은 다양한 머신러닝 모델을 하나의 패키지에서 모두 제공하고 있다는 점이다. 다음은 scikit-learn 패키지에서 제공하는 머신러닝 모델의 목록의 일부다.
- 데이터셋
 - ① 회귀분석, 분류, 클러스터링용 가상 데이터셋 생성
 - ② 각종 벤치마크 데이터셋
- 전처리
 - ① 스케일링
 - ② 누락데이터 처리
 - ③ 텍스트 토큰화



1. 라이브러리 개요

6. 3rd Party Library

- scikit-learn 패키지

- 지도 학습

- ① 회귀분석
- ② LDA/QDA
- ③ 서포트벡터머신
- ④ 퍼셉트론, SGD
- ⑤ KNN
- ⑥ 가우스프로세스
- ⑦ 나이브베이즈
- ⑧ 의사결정나무
- ⑨ 랜덤포레스트, 부스팅

- 비지도 학습

- ① 가우스 혼합모형
- ② 클러스터링
- ③ PCA

- 성능 최적화

- ① 교차검증
- ② 특징선택
- ③ 하이퍼파라미터 최적화



1. 라이브러리 개요

6. 3rd Party Library

- scikit-learn 패키지

- 설치와 임포트에 사용하는 이름은 sklearn이다. 다음 명령으로 설치한다.

`pip install sklearn`

- 임포트할 때는 보통 sk이라는 별명으로 임포트한다.

`import sklearn as sk`



1. 라이브러리 개요

6. 3rd Party Library

- missingno 패키지
 - pandas 데이터프레임 데이터에서 누락된 데이터를 찾고 시각화하는 기능을 제공한다.
 - 다음 명령으로 설치한다.
- `pip install missingno`
- patsy 패키지
 - pandas 데이터프레임 데이터에서 선택, 변형하는 기능을 제공한다.
 - statsmodels가 의존하는 패키지이므로 statsmodels 패키지를 설치하면 별도로 설치할 필요가 없다.



1. 라이브러리 개요

6. 3rd Party Library

- 텍스트 전처리용 패키지
 - nltk 패키지
 - spacy 패키지
 - konlpy 패키지
 - soynlp 패키지
 - gensim 패키지
- 이미지 전처리용 패키지
 - opencv 패키지
- 사운드 전처리용 패키지
 - librosa 패키지
- 지리정보 전처리용 패키지
 - geopandas 패키지



2. 외부 라이브러리 살펴보기

파이썬 라이브러리 설치 확인

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import sys print("python 버전 : {}".format(sys.version)) import pandas as pd print("pandas 버전 : {}".format(pd.__version__)) import matplotlib print("matplotlib 버전 : {}".format(matplotlib.__version__)) import numpy as np print("numpy 버전 : {}".format(np.__version__))</pre>
결과값1	<pre>python 버전 : 3.8.16 (default, Jan 17 2023, 22:25:28) [MSC v.1916 64 bit (AMD64)] Traceback (most recent call last): File "c:\DEV\python_works\day3-am.py", line 156, in <module> import pandas as pd File "C:\Users\k8s\AppData\Roaming\Python\Python38\site-packages\pandas__init__.py", line 16, in <module> raise ImportError(ImportError: Unable to import required dependencies: numpy: No module named 'numpy'</pre>
비고	



2. 외부 라이브러리 살펴보기

파이썬 라이브러리 확인

```
(py3_10_basic) PS C:\DEV\PycharmProjects\python_works>pip install numpy
Collecting numpy
  Using cached numpy-1.24.2-cp38-cp38-win_amd64.whl (14.9 MB)
Installing collected packages: numpy
```

....

```
(py3_10_basic) PS C:\DEV\PycharmProjects\python_works>pip install pandas
```

...

```
(py3_10_basic) PS C:\DEV\PycharmProjects\python_works>pip install matplotlib
```



2. 외부 라이브러리 살펴보기

Numpy

- NumPy는 대규모 다차원 지원을 추가하는 라이브러리이다. 높은 수준의 배열 및 행렬과 함께 이러한 배열에서 작동하는 수학 함수를 제공한다.
- Array Creation

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import numpy as np print(np.array([1, 2, 3])) print(np.arange(10)) print(np.linspace(0,2,4)) print(np.zeros((2,4)))</pre>
결과값1	<pre>[1 2 3] [0 1 2 3 4 5 6 7 8 9] [0. 0.66666667 1.33333333 2.] [[0. 0. 0. 0.] [0. 0. 0. 0.]</pre>
비고	



2. 외부 라이브러리 살펴보기

Numpy

● Basic Operations

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import numpy as np a = np.array([1,2,3,6]) b = np.linspace(0,2,4) c = a - b print(c) print(a**2)</pre>
결과값1	<pre>[1. 1.33333333 1.66666667 4.] [1 4 9 36]</pre>
비고	

● Universal Functions

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import numpy as np import math a = np.linspace(-np.pi, np.pi, 3) print(a) print(np.cos(a)) print(math.cos(a))</pre>
결과값1	<pre>[-3.14159265 0. 3.14159265] [-1. 1. -1.] Traceback (most recent call last): File "c:\DEV\python_works\day3-am.py", line 190, in <module> print(math.cos(a)) TypeError: only size-1 arrays can be converted to Python scalars</pre>
비고	



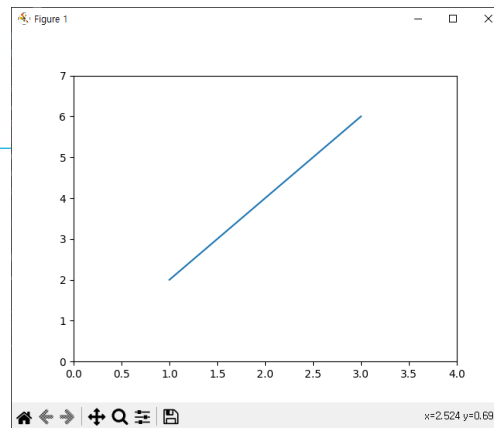
2. 외부 라이브러리 살펴보기

matplotlib

● 직선 그리기

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt import numpy as np #행렬 연산시 필요한 모듈 #(1) 직선 그리기 #plot([x좌표 리스트], [y좌표 리스트]) plt.plot([1,2,3], [2,4,6]) #axis([x축 시작, x축 끝, y축시작, y축끝]) plt.axis([0,4,0,7]) plt.show()</pre>

결과값1



비고

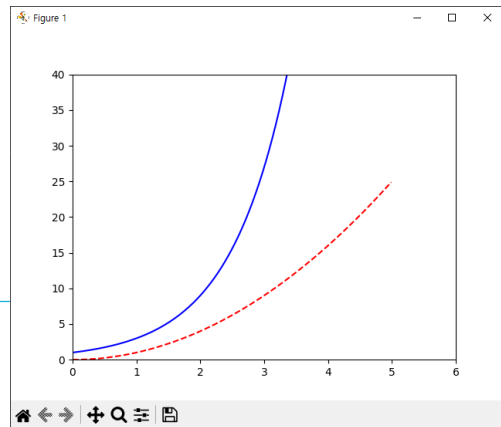


2. 외부 라이브러리 살펴보기

matplotlib

● 직선 그리기

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt import numpy as np #행렬 연산시 필요한 모듈 x = np.arange(0.0,5.0,0.01) plt.plot(x, x**2, 'r--') plt.plot(x, 3**x, 'b') plt.axis([0,6,0,40]) plt.show()</pre>



결과값1

비고



2. 외부 라이브러리 살펴보기

matplotlib

● 직선 그리기

구분

내용

실습환경

py3_10_basic

소스코드

```
x = np.arange(0.0,5.0,0.01)
print(type(x), x)
# y=x^2의 그래프
plt.subplot(221)
plt.plot(x, x**2, 'r--')
# y=3^x 그래프
plt.subplot(222)
plt.plot(x, 3**x, 'b')
# y=x 그래프
plt.subplot(223)
plt.plot(x, x, 'g-')
# y=1/(x+1) 그래프
plt.subplot(224)
plt.plot(x, 1/(x+1), 'y')
plt.show()
```

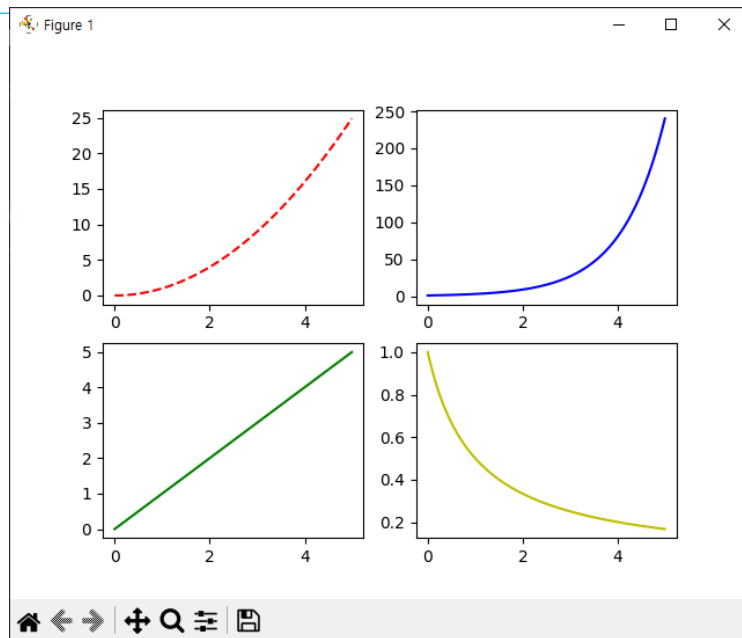


fig.add_subplot(111)은 Figure에 하나의 subplot을 추가하는 메서드입니다. 111은 하나의 행, 하나의 열, 그리고 첫 번째 subplot을 의미합니다. Matplotlib에서 subplot의 위치는 행, 열, 인덱스로 지정되는데, 이를 통해 여러 개의 subplot을 하나의 Figure에 배열할 수 있습니다.

비고



2. 외부 라이브러리 살펴보기

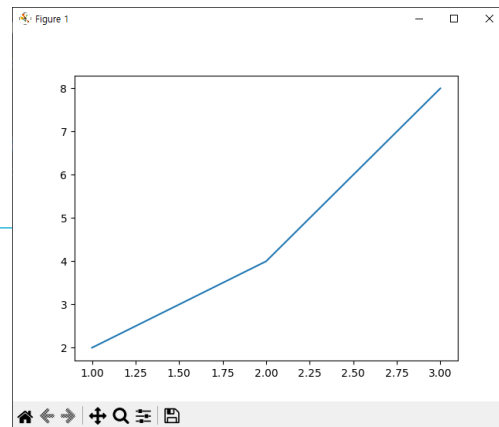
matplotlib

1. 그래프 기본

구분	내용
실습환경	py3_10_basic import matplotlib.pyplot as plt

소스코드	x = [1, 2, 3] y = [2, 4, 8] plt.plot(x, y) plt.show()
------	--

결과값1



비고



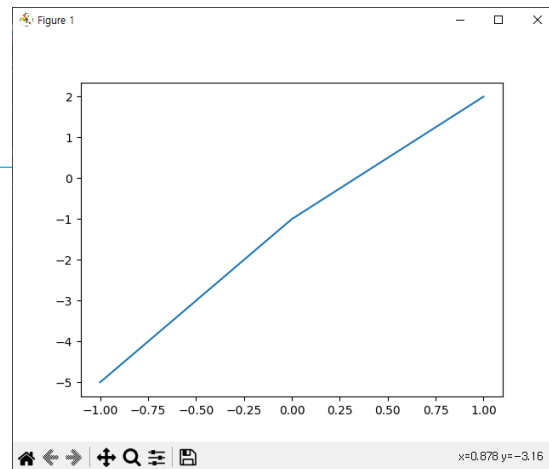
2. 외부 라이브러리 살펴보기

matplotlib

1. 그래프 기본

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt plt.plot([-1, 0, 1], [-5, -1, 2]) plt.show()</pre>

결과값1



비고

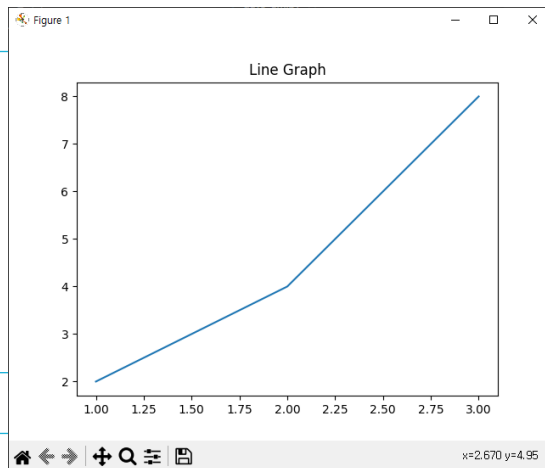


2. 외부 라이브러리 살펴보기

matplotlib

2. Title 설정

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt x = [1, 2, 3] y = [2, 4, 8] plt.plot(x, y) plt.title('Line Graph') plt.show()</pre>
결과값1	
비고	





2. 외부 라이브러리 살펴보기

matplotlib

3. 한글 처리

구분	내용
실습환경	py3_10_basic
소스코드	<pre> import matplotlib import matplotlib.pyplot as plt matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows # matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac matplotlib.rcParams['font.size'] = 15 # 글자 크기 matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결 import matplotlib.font_manager as fm fm.fontManager.ttflist # 사용 가능한 폰트 확인 print([f.name for f in fm.fontManager.ttflist]) </pre>
결과값1	<pre> ['cmsy10', 'DejaVu Sans Mono', 'STIXNonUnicode', 'STIXSizeFiveSym', 'STIXNonUnicode', 'STIXSizeFourSym', 'STIXSizeTwoSym', 'STIXGeneral', 'STIXSizeTwoSym', 'cmex10', 'DejaVu Serif', 'STIXGeneral', 'cmr10', 'DejaVu Serif Display', 'DejaVu Sans', 'cmss10', 'STIXSizeThreeSym', 'STIXNonUnicode', 'STIXSizeFourSym', 'DejaVu Serif', 'DejaVu Serif', 'STIXSizeOneSym', 'STIXSizeOneSym', 'DejaVu Sans', 'STIXNonUnicode', 'cmtt10', 'DejaVu Sans Mono', 'cmmi10', 'cmb10', 'STIXGeneral', 'DejaVu Sans Mono', 'DejaVu Sans', 'DejaVu Sans Display', 'STIXSizeThreeSym', 'STIXGeneral', 'DejaVu Serif', 'DejaVu Sans', 'DejaVu Sans Mono', 'Bahnschrift', 'SimSun-ExtB', 'Consolas', 'Microsoft PhagsPa', 'Segoe UI', 'Segoe UI', 'Palatino Linotype', 'Verdana', 'Sitka Small', 'Wingdings', 'Cambria', 'Lucida Sans Unicode', 'Arial', 'Microsoft JhengHei', 'Trebuchet MS', 'Comic Sans MS', 'Verdana', 'Nirmala UI', 'Trebuchet MS', 'Verdana', 'Candara', 'Corbel', 'Courier New', 'Malgun Gothic', 'Malgun Gothic', 'Yu Gothic', 'MV Boli', 'Nirmala UI', 'Yu Gothic', 'Candara', 'Times New Roman', 'Corbel', 'Segoe UI', 'Candara', 'Segoe Print', 'Microsoft Yi Baiti', 'Leelawadee UI', 'Times New Roman', 'Courier New', 'Constantia', 'Calibri', 'Sitka Small', 'Microsoft JhengHei', 'MingLiU-ExtB', 'Microsoft PhagsPa', 'Segoe UI', 'Batang', 'Consolas', 'Gabriola', 'Yu Gothic', 'Microsoft Tai Le', 'Gadugi', 'Courier New', 'Tahoma', 'Arial', 'Calibri', 'Nirmala UI', 'Segoe UI Symbol', 'Ebrima', 'SimSun', 'Webdings', 'Verdana', 'Microsoft New Tai Lue', 'Segoe UI', 'Georgia', 'Mongolian Baiti', 'Corbel', 'Palatino Linotype', 'Times New Roman', 'Microsoft YaHei', 'Leelawadee UI', 'Calibri', 'Constantia', 'Arial', 'Segoe UI Emoji', 'Corbel', 'Myanmar Text', 'Segoe UI Historic', 'Segoe MDL2 Assets', 'Candara', 'Comic Sans MS', 'Constantia', 'Sitka Small', 'Corbel', 'Segoe UI', 'Palatino Linotype', 'Calibri', 'HoloLens MDL2 Assets', 'Segoe UI', 'Trebuchet MS', 'Comic Sans MS', 'Microsoft New Tai Lue', 'Microsoft Tai Le', 'Georgia', 'Gadugi', 'Corbel', 'Tahoma', 'Sylfaen', 'Javanese Text', 'Ink Free', 'Courier New', 'Cambria', 'Gulim', 'Microsoft JhengHei', 'Microsoft YaHei', 'Constantia', 'Segoe UI', 'Comic Sans MS', 'Ebrima', 'Segoe Print', 'Microsoft Himalaya', 'Franklin Gothic Medium', 'Palatino Linotype', 'Lucida Console', 'Arial', 'Sitka Small', 'Calibri', 'Georgia', 'Calibri', 'Segoe Script', 'Segoe UI', 'Malgun Gothic', 'Impact', 'Trebuchet MS', 'Arial', 'Leelawadee UI', 'Franklin Gothic Medium', 'Symbol', 'MS Gothic', 'Myanmar Text', 'Segoe Script', 'Times New Roman', 'Segoe UI', 'Georgia', 'Cambria', 'Consolas', 'Segoe UI', 'Candara', 'Yu Gothic', 'Consolas', 'Segoe UI', 'Microsoft Sans Serif', 'Candara', 'Microsoft YaHei', 'Cambria' </pre>



2. 외부 라이브러리 살펴보기

matplotlib

3. 한글 처리

구분

내용

실습환경

py3_10_basic

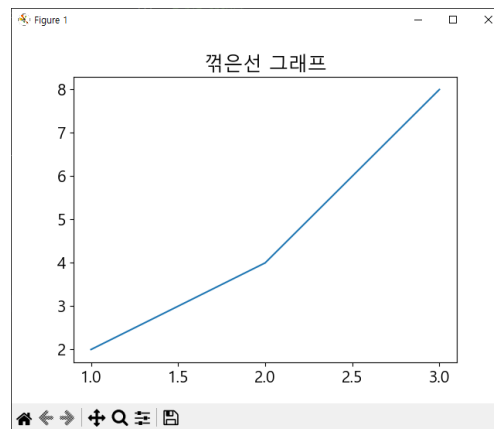
```
import matplotlib
import matplotlib.pyplot as plt
matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows
# matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac
matplotlib.rcParams['font.size'] = 15 # 글자 크기
matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스
글자가 깨지는 현상을 해결
```

소스코드

```
# import matplotlib.font_manager as fm
# fm.fontManager.ttflist # 사용 가능한 폰트 확인
# print([f.name for f in fm.fontManager.ttflist])

x = [1, 2, 3]
y = [2, 4, 8]

plt.plot(x, y)
plt.title('꺾은선 그래프')
plt.show()
```



결과값1

비고



2. 외부 라이브러리 살펴보기

matplotlib

4. 축

구분

내용

실습환경

py3_10_basic

```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows
# matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac
matplotlib.rcParams['font.size'] = 15 # 글자 크기
matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스
글자가 깨지는 현상을 해결
```

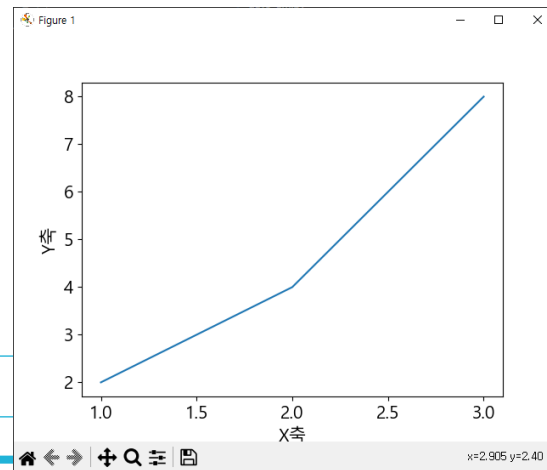
소스코드

```
x = [1, 2, 3]
y = [2, 4, 8]

plt.plot(x, y)
plt.xlabel('X축')
plt.ylabel('Y축')
plt.show()
```

결과값1

비고



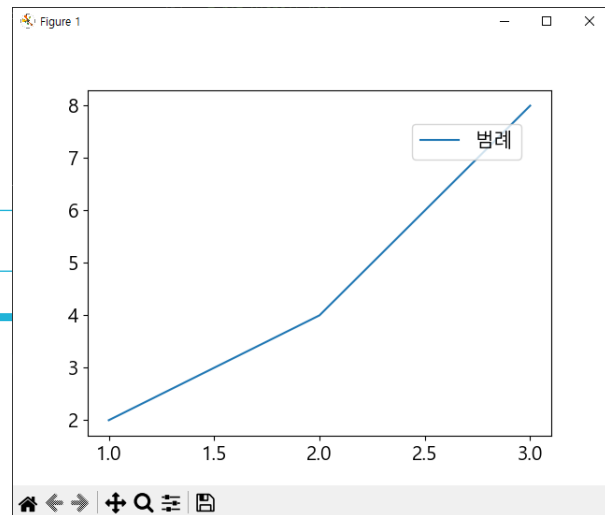


2. 외부 라이브러리 살펴보기

matplotlib

5. 범례

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt import matplotlib matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows # matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac matplotlib.rcParams['font.size'] = 15 # 글자 크기 matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결 x = [1, 2, 3] y = [2, 4, 8] plt.plot(x, y, label='범례') plt.legend(loc=(0.7, 0.8)) # x축, y축 (0~1 사이) plt.show()</pre>
결과값1	
비고	





2. 외부 라이브러리 살펴보기

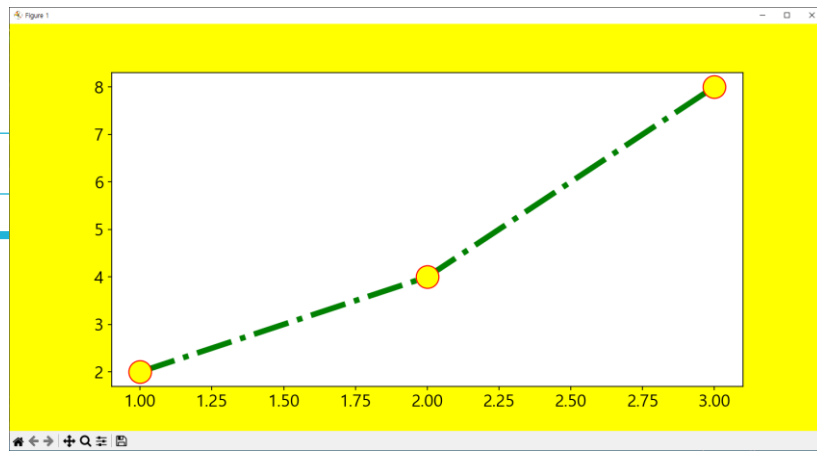
matplotlib

6. 스타일

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt import matplotlib matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows # matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac matplotlib.rcParams['font.size'] = 15 # 글자 크기 matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결 x = [1, 2, 3] y = [2, 4, 8] plt.figure(figsize=(10, 5), dpi=150, facecolor='yellow') # dots per inch, 확대 plt.plot(x, y, linewidth=5, marker='o', markersize=20, markeredgecolor='red', markerfacecolor='yellow', linestyle='-.', color='g') plt.show()</pre>

결과값1

비고





2. 외부 라이브러리 살펴보기

matplotlib

7. 파일 저장

구분	내용
실습환경	py3_10_basic
	<pre>import matplotlib.pyplot as plt import matplotlib matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows # matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac matplotlib.rcParams['font.size'] = 15 # 글자 크기 matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결</pre>

소스코드

```
x = [1, 2, 3]
y = [2, 4, 8]
```

```
plt.figure(dpi=200)
plt.plot(x, y)
plt.show()
plt.savefig('graph_200.png', dpi=100)
```

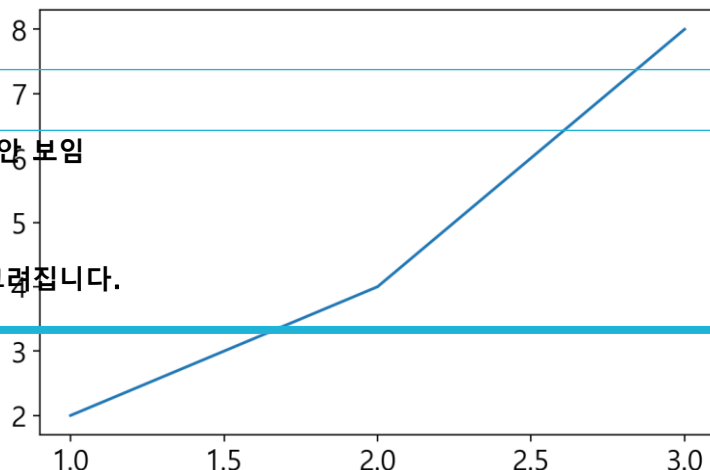
결과값1

그림을 볼 경우, 그림을 닫아야 저장이 됨. 그러나 저장된 그림이 안 보임
따라서, 위 코드 중

비고

`plt.savefig('graph_200.png', dpi=100)`
`plt.show()`처럼 위치를 변경해야 합니다.
쥬피터 노트북에서는 `plt.show()`를 쓰면 안됩니다. 그림이 2개 그려집니다.

Figure 1



이름	수정:
hello.py	2023
graph_200.png	2023



2. 외부 라이브러리 살펴보기

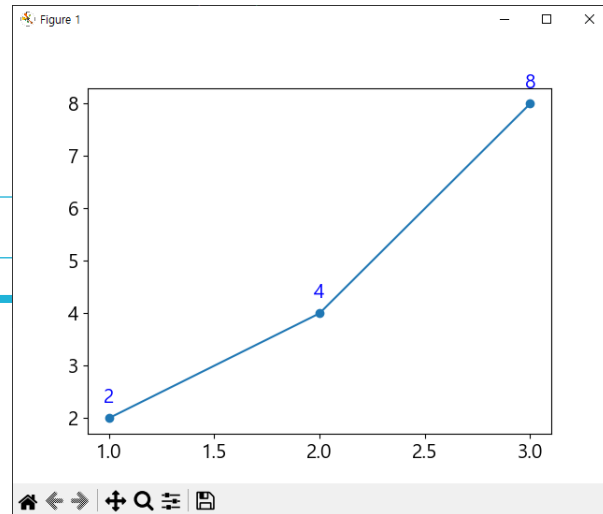
matplotlib

8. 텍스트

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt import matplotlib matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows # matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac matplotlib.rcParams['font.size'] = 15 # 글자 크기 matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결 x = [1, 2, 3] y = [2, 4, 8] plt.plot(x, y, marker='o') for idx, txt in enumerate(y): plt.text(x[idx], y[idx] + 0.3, txt, ha='center', color='blue') plt.show()</pre>

결과값1

비고





2. 외부 라이브러리 살펴보기

matplotlib

9. 여러 데이터

구분

내용

실습환경

py3_10_basic

소스코드

```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows
# matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac
matplotlib.rcParams['font.size'] = 15 # 글자 크기
matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결
```

```
days = [1, 2, 3] # 1일, 2일, 3일
az = [2, 4, 8] # (단위 : 만명) 1일부터 3일까지 아스트라제네카 접종인구
pfizer = [5, 1, 3] # 화이타
moderna = [1, 2, 5] # 모더나
```

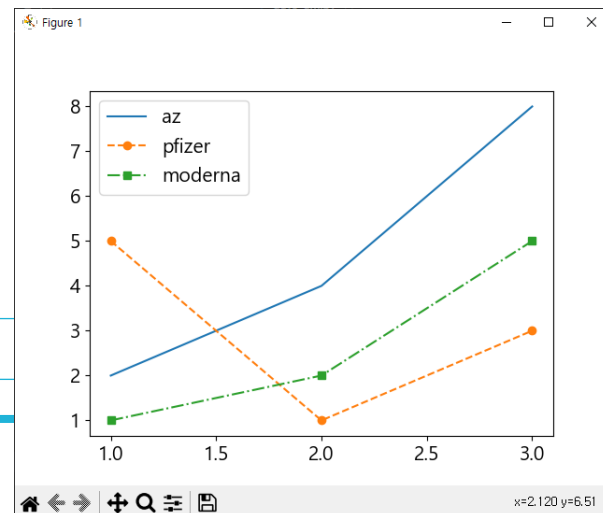
```
plt.plot(days, az, label='az')
plt.plot(days, pfizer, label='pfizer', marker='o', linestyle='--')
plt.plot(days, moderna, label='moderna', marker='s', ls='-.')
```

```
plt.legend()
```

```
plt.show()
```

결과값1

비고





2. 외부 라이브러리 살펴보기

matplotlib

9. 여러 데이터

구분

내용

실습환경

py3_10_basic

소스코드

```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows
# matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac
matplotlib.rcParams['font.size'] = 15 # 글자 크기
matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결

days = [1, 2, 3] # 1일, 2일, 3일
az = [2, 4, 8] # (단위 : 만명) 1일부터 3일까지 아스트라제네카 접종인구
pfizer = [5, 1, 3] # 화이타
moderna = [1, 2, 5] # 모더나

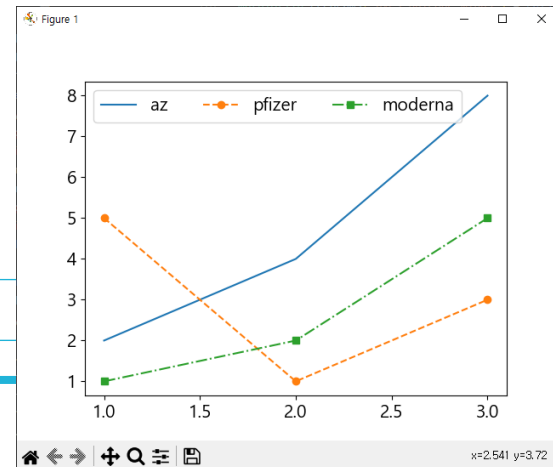
plt.plot(days, az, label='az')
plt.plot(days, pfizer, label='pfizer', marker='o', linestyle='--')
plt.plot(days, moderna, label='moderna', marker='s', ls='-.')

plt.legend(ncol=3)

plt.show()
```

결과값1

비고





2. 외부 라이브러리 살펴보기

matplotlib

10. 막대 그래프

구분

내용

실습환경

py3_10_basic

```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows
# matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac
matplotlib.rcParams['font.size'] = 15 # 글자 크기
matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스
글자가 깨지는 현상을 해결
```

소스코드

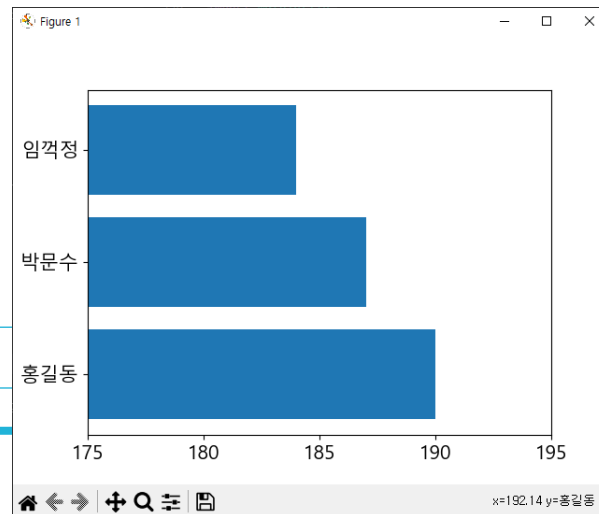
```
labels = ['홍길동', '박문수', '임꺽정'] # 이름
values = [190, 187, 184] # 키
```

```
plt.barh(labels, values)
plt.xlim(175, 195)
```

```
plt.show()
```

결과값1

비고





2. 외부 라이브러리 살펴보기

matplotlib

10. 막대 그래프

구분

내용

실습환경

py3_10_basic

소스코드

```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows
# matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac
matplotlib.rcParams['font.size'] = 15 # 글자 크기
matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결

labels = ['홍길동', '박문수', '임꺽정'] # 이름
values = [190, 187, 184] # 키

# plt.barh(labels, values)
# plt.xlim(175, 195)

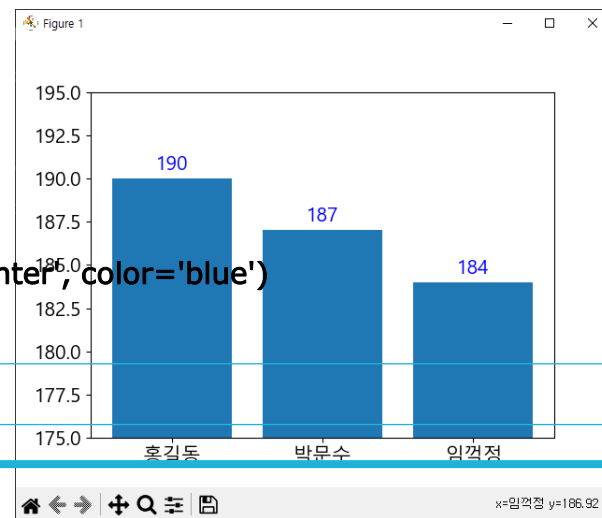
bar = plt.bar(labels, values)
plt.ylim(175, 195)

for idx, rect in enumerate(bar):
    plt.text(idx, rect.get_height() + 0.5, values[idx], ha='center', color='blue')

plt.show()
```

결과값1

비고





2. 외부 라이브러리 살펴보기

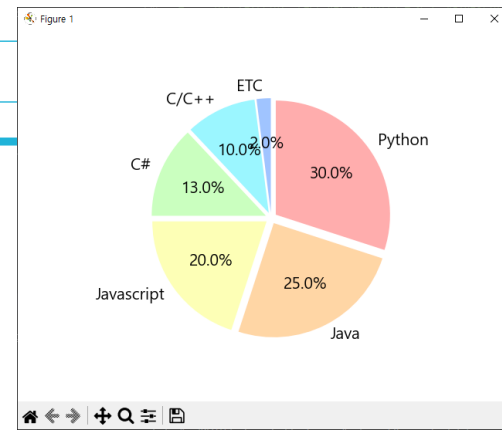
matplotlib

11. 원 그래프

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt import matplotlib matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows # matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac matplotlib.rcParams['font.size'] = 15 # 글자 크기 matplotlib.rcParams['axes.unicode_minus'] = False # 한글 폰트 사용 시, 마이너스 글자가 깨지는 현상을 해결 values = [30, 25, 20, 13, 10, 2] labels = ['Python', 'Java', 'Javascript', 'C#', 'C/C++', 'ETC'] # colors = ['b', 'g', 'r', 'c', 'm', 'y'] colors = ['#ffadad', '#ffd6a5', '#fdffb6', '#caffbf', '#9bf6ff', '#a0c4ff'] explode = [0.05] * 6 plt.pie(values, labels=labels, autopct='%1f%%', startangle=90, counterclock=False, colors=colors, explode=explode) plt.show()</pre>

결과값1

비고





2. 외부 라이브러리 살펴보기

matplotlib

Quiz

- 다음은 대한민국 영화 중에서 관객 수가 가장 많은 상위 8개의 데이터입니다.
- 주어진 코드를 이용하여 퀴즈를 풀어보시오.



2. 외부 라이브러리 살펴보기

matplotlib

● 데이터 준비

구분	내용
실습환경	<pre>py3_10_basic import pandas as pd import matplotlib.pyplot as plt import matplotlib matplotlib.rcParams['font.family'] = 'Malgun Gothic' # Windows # matplotlib.rcParams['font.family'] = 'AppleGothic' # Mac matplotlib.rcParams['font.size'] = 15 matplotlib.rcParams['axes.unicode_minus'] = False</pre>
소스코드	<pre>data = { '영화' : ['명량', '극한직업', '신과함께-죄와 벌', '국제시장', '괴물', '도둑들', '7번방의 선물', '암살'], '개봉 연도' : [2014, 2019, 2017, 2014, 2006, 2012, 2013, 2015], '관객 수' : [1761, 1626, 1441, 1426, 1301, 1298, 1281, 1270], # (단위 : 만 명) '평점' : [8.88, 9.20, 8.73, 9.16, 8.62, 7.64, 8.83, 9.10] } df = pd.DataFrame(data) print(df)</pre>
결과값1	<pre>영화 개봉 연도 관객 수 평점 0 명량 2014 1761 8.88 1 극한직업 2019 1626 9.20 2 신과함께-죄와 벌 2017 1441 8.73 3 국제시장 2014 1426 9.16 4 괴물 2006 1301 8.62 5 도둑들 2012 1298 7.64 6 7번방의 선물 2013 1281 8.83 7 암살 2015 1270 9.10</pre>
비고	

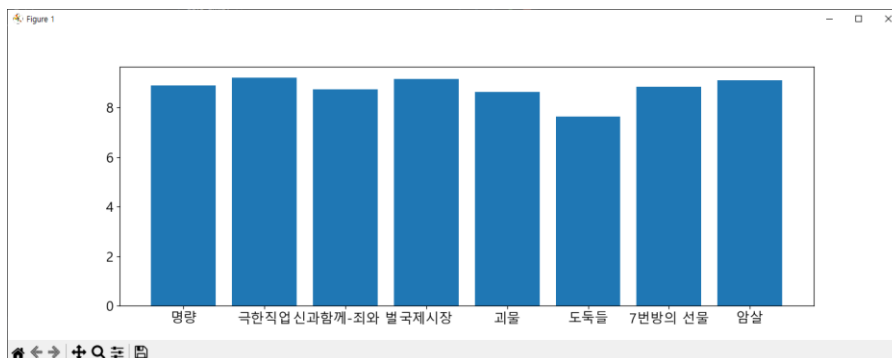


2. 외부 라이브러리 살펴보기

matplotlib

- 영화 데이터를 활용하여 x 축은 영화, y 축은 평점인 막대 그래프

구분	내용
실습환경	py3_10_basic import matplotlib.pyplot as plt
소스코드	plt.figure(figsize=(10, 5)) # 가로 10인치, 세로 5인치로 설정 plt.bar(df['영화'], df['평점']) plt.show()
결과값1	
비고	



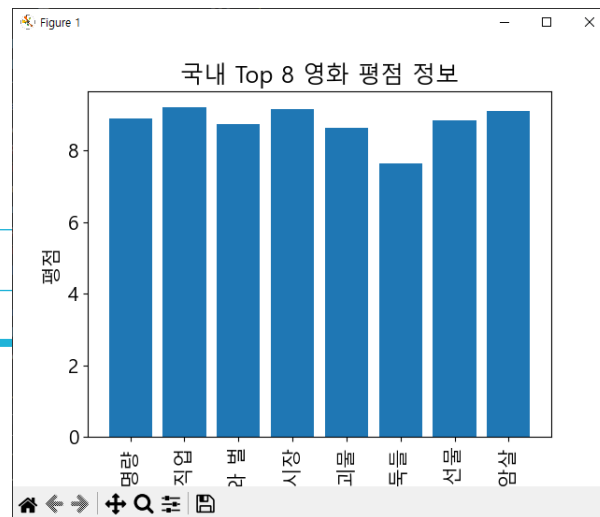


2. 외부 라이브러리 살펴보기

matplotlib

- 제목 : 국내 Top 8 영화 평점 정보
- x축 label : 영화 (90도 회전)
- y축 label : 평점

구분	내용
실습환경	py3_10_basic
소스코드	<pre>plt.bar(df['영화'], df['평점']) plt.title('국내 Top 8 영화 평점 정보') plt.xlabel('영화') plt.xticks(rotation=90) plt.ylabel('평점') plt.show()</pre>
결과값1	
비고	





2. 외부 라이브러리 살펴보기

matplotlib

- 개봉 연도별 평점 변화 추이를 꺾은선 그래프

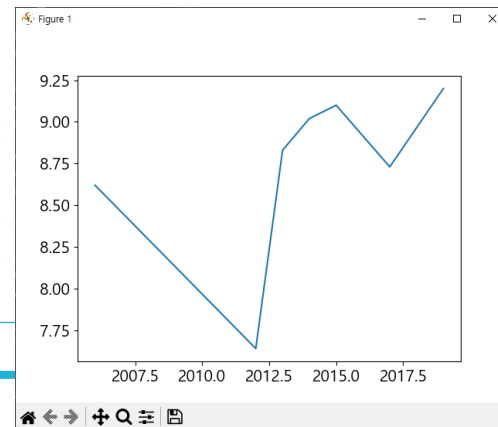
구분p	내용
실습환경	py3_10_basic
소스코드	<pre>df_group = df.groupby('개봉 연도').mean(numeric_only=True) print(df_group)</pre>
소스코드	<pre>plt.plot(df_group.index, df_group['평점']) plt.show()</pre>

결과값1

	관객 수	평점
개봉 연도		
2006	1301.0	8.62
2012	1298.0	7.64
2013	1281.0	8.83
2014	1593.5	9.02
2015	1270.0	9.10
2017	1441.0	8.73
2019	1626.0	9.20

비고

연도별 평균 데이터





2. 외부 라이브러리 살펴보기

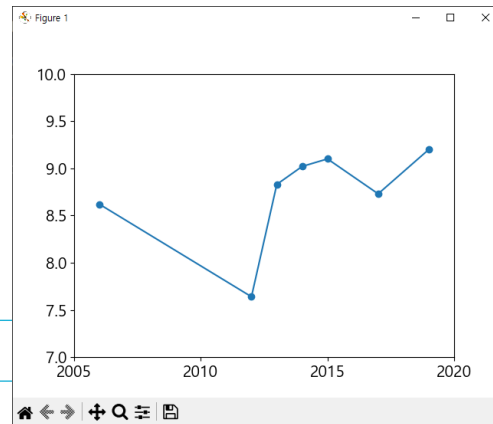
matplotlib

- marker : 'o'
- x 축 눈금 : 5년 단위 (2005, 2010, 2015, 2020)
- y 축 범위 : 최소 7, 최대 10

구분	내용
실습환경	py3_10_basic
소스코드	<pre>import matplotlib.pyplot as plt # 숫자형 컬럼만 선택해서 그룹 평균 계산 df_group = df.groupby('개봉 연도').mean(numeric_only=True) # 라인 그래프 그리기 plt.plot(df_group.index, df_group['평점'], marker='o') # X축 간격 지정 plt.xticks([2005, 2010, 2015, 2020]) # Y축 범위 설정 plt.ylim(7, 10) # 그래프 보여주기 plt.show()</pre>

결과값1

비고





2. 외부 라이브러리 살펴보기

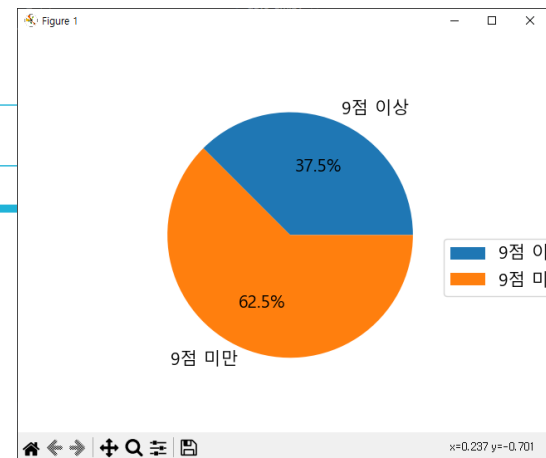
matplotlib

- 평점이 9점 이상인 영화의 비율을 확인할 수 있는 원 그래프
 - label : 9점 이상 / 9점 미만
 - 퍼센트 : 소수점 첫째자리까지 표시
 - 범례 : 그래프 우측에 표시

구분	내용
실습환경	py3_10_basic
소스코드	<pre>filt = df['평점'] >= 9.0 values = [len(df[filt]), len(df[~filt])] labels = ['9점 이상', '9점 미만'] plt.pie(values, labels=labels, autopct='%0.1f%%') plt.legend(loc=(1, 0.3)) plt.show()</pre>

결과값1

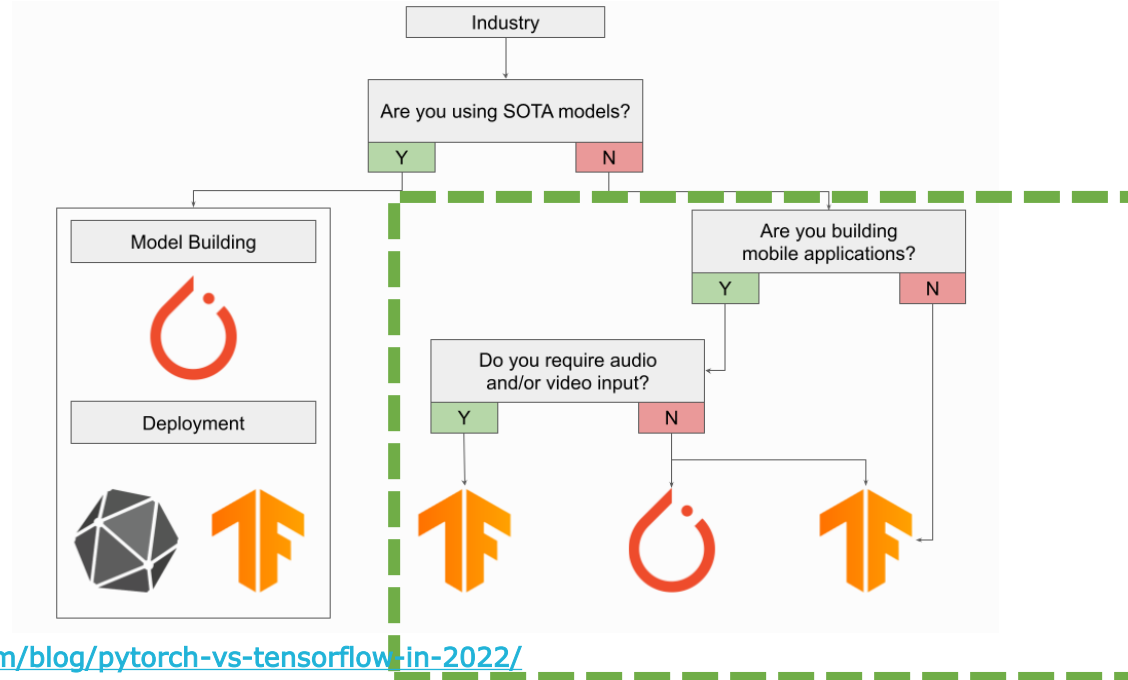
비고



3. 더 나아간 주제

Machine Learning/AI Issue

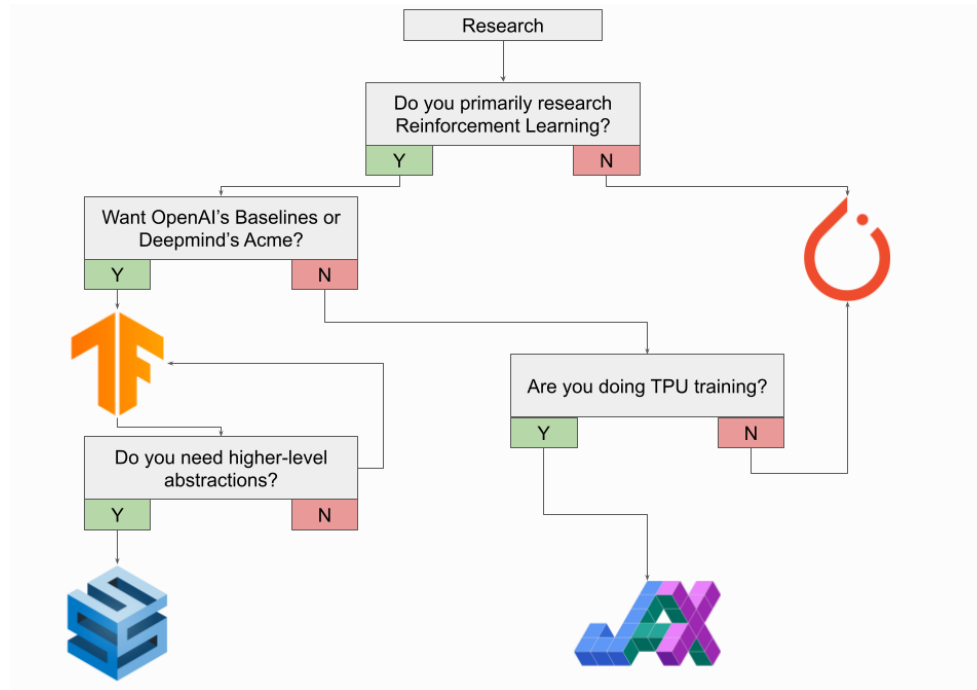
- 내가 산업체에 있다면?
 - 산업 환경에서 딥 러닝 엔지니어링을 수행하는 경우 TensorFlow를 사용하고 있을 가능성이 높으며 계속 사용해야 합니다. TensorFlow의 강력한 배포 프레임워크와 종단 간 TensorFlow Extended 플랫폼은 모델을 생산해야 하는 사람들에게 매우 중요합니다. 모델 모니터링 및 아티팩트 추적과 함께 gRPC 서버에 쉽게 배포하는 것은 업계에서 사용하기 위한 중요한 도구입니다.



3. 더 나아간 주제

Machine Learning/AI Issue

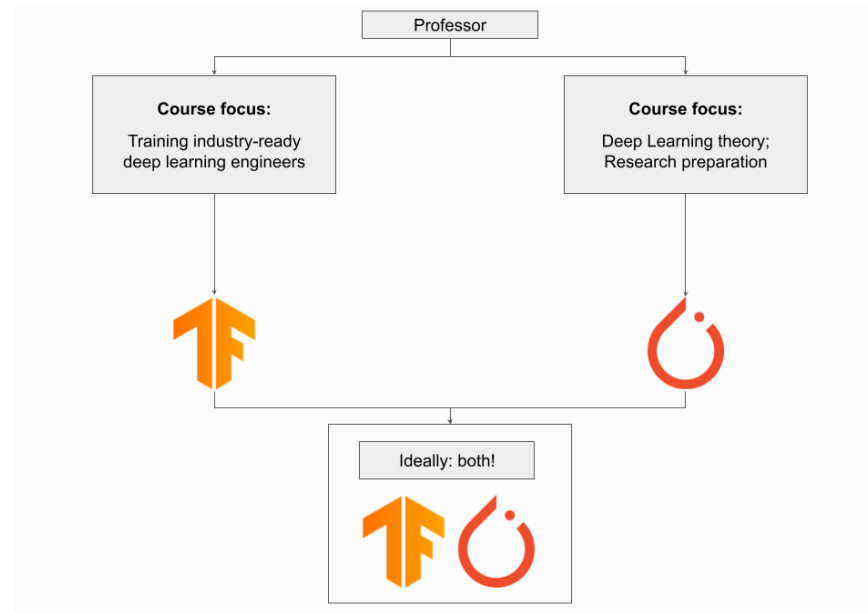
- 내가 연구원이라면?
 - 연구원이라면 거의 확실하게 PyTorch를 사용하고 있으며 현재로서는 계속 사용하고 있을 것입니다.



3. 더 나아간 주제

Machine Learning/AI Issue

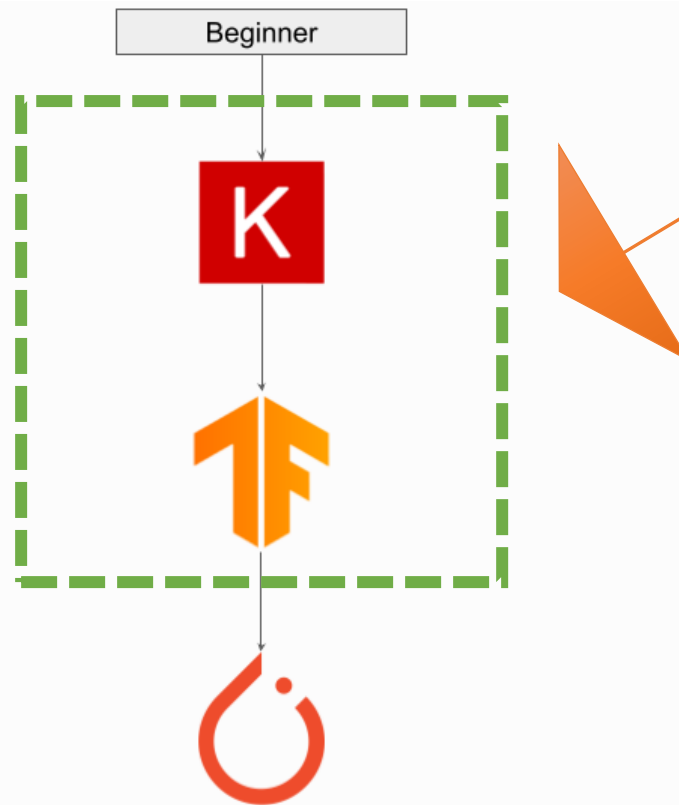
- 내가 교수라면?
 - 교육 과정의 초점이 딥 러닝 이론뿐만 아니라 전체 딥 러닝 프로세스에서 역량을 발휘하여 기초를 다질 수 있는 업계 준비 딥 러닝 엔지니어를 양성하는 것이라면 TensorFlow를 사용해야 합니다.
 - 딥 러닝 연구를 수행할 수 있도록 학생들을 준비시키기 위한 고급 학부 과정이나 초기 대학원 수준 과정을 가르치는 경우 PyTorch를 사용해야 합니다.



3. 더 나아간 주제

Machine Learning/AI Issue

- 내가 완전 초보자라면?
 - 딥 러닝에 관심이 있고 막 시작하려는 완전 초보자라면 Keras 를 사용하는 것이 좋습니다 .



THANK YOU.

