

강원대학교
AI 소프트웨어학과

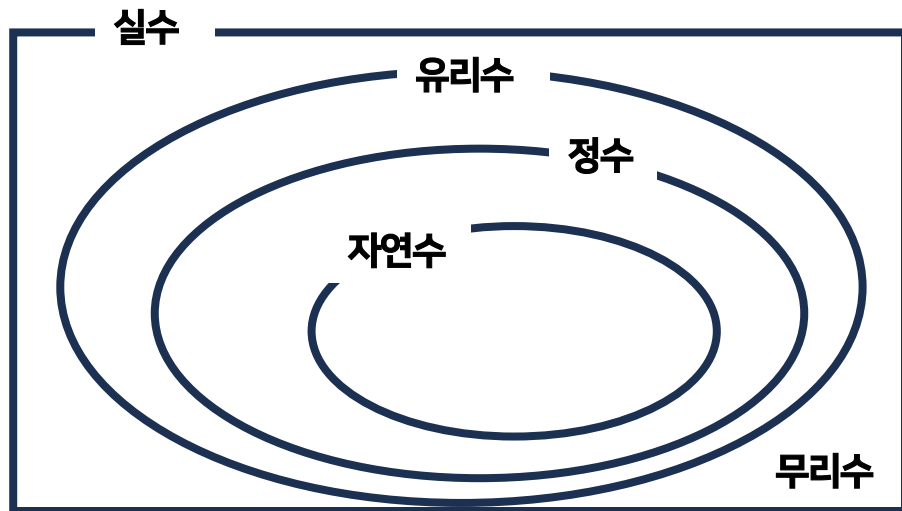
데이터 전처리

- 데이터의 구조 -

- 데이터란?
 - 이론을 세우는 데 기초가 되는 사실, 또는 바탕이 되는 자료
 - 관찰이나 실험, 조사로 얻은 사실이나 자료
 - 컴퓨터가 처리할 수 있는 문자, 숫자, 소리, 그림 따위의 형태로 된 자료
 - 데이터는 신호, 기호, 숫자, 문자 등으로 기록 됨
 - 정보를 위한 기초적인 자료를 말함
 - 정보는 데이터를 가공하지 않은 경우

단순구조

- 자료형 : 정수, 실수, 문자, 문자열 등
- 정수 : 양의 정수, 0, 음의 정수 or 자연수, (-)자연수
- 실수 : 유리수 → 정수와 분수가 존재, 소수로 나타내면 유한 소수나 순환 소수
무리수 → 간단한 분수로 고칠 수 없는 수, 소수점 아래의 수가 반복되지 않고 무한히 계속되는 소수



단순구조

- 문자(Character) : 숫자이외의 정보를 표현하는 방법
- 문자열(String) : 둘 이상의 문자의 결합
- 문자를 컴퓨터가 이해할 수 있는 숫자로 변환 → 인코딩

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	Null	32	20	SPACE	64	40	@	96	60]
1	1	SOH (start of heading)	33	21	!	65	41	A	97	61	a
2	2	STX (start of text)	34	22	"	66	42	B	98	62	b
3	3	ETX (end of text)	35	23	#	67	43	C	99	63	c
4	4	EOF (end of file)	36	24	\$	68	44	D	100	64	d
5	5	ENQ (enquiry)	37	25	%	69	45	E	101	65	e
6	6	ACK (acknowledge)	38	26	&	70	46	F	102	66	f
7	7	BEL (bell)	39	27	'	71	47	G	103	67	g
8	8	BS (backspace)	40	28	(72	48	H	104	68	h
9	9	HT (horizontal tab)	41	29)	73	49	I	105	69	i
10	A	LF (line feed, new line)	42	2A	*	74	4A	J	106	70	j
11	B	VT (vertical tab)	43	2B	+	75	4B	K	107	71	k
12	C	FF (form feed, new page)	44	2C	,	76	4C	L	108	72	l
13	D	CR (carriage return)	45	2D	-	77	4D	M	109	73	m
14	E	SO (shift out)	46	2E	.	78	4E	N	110	74	n
15	F	SI (shift in)	47	2F	/	79	4F	O	111	75	o
16	10	DLE (data link escape)	48	30	0	80	50	P	112	76	p
17	11	DC1 (device control 1)	49	31	1	81	51	Q	113	77	q
18	12	DC2 (device control 2)	50	32	2	82	52	R	114	78	r
19	13	DC3 (device control 3)	51	33	3	83	53	S	115	79	s
20	14	DC4 (device control 4)	52	34	4	84	54	T	116	80	t
21	15	NAK (negative acknowledge)	53	35	5	85	55	U	117	81	u
22	16	SYN (synchronous idle)	54	36	6	86	56	V	118	82	v
23	17	END (end of transmission)	55	37	7	87	57	W	119	83	w
24	18	CAN (cancel)	56	38	8	88	58	X	120	84	x
25	19	EM (end of medium)	57	39	9	89	59	Y	121	85	y
26	1A	ESC (escape)	58	3A	:	90	5A	Z	122	86	z
27	1B	FS (file separator)	59	3B	;	91	5B	[123	87	{
28	1C	GS (group separator)	60	3C	<	92	5C	\	124	88	
29	1D	RS (record separator)	61	3D	=	93	5D]	125	89	}
30	1E	US (unit separator)	62	3E	>	94	5E	^	126	90	~
31	1F	UNIT SEPARATOR	63	3F	?	95	5F	_	127	7F	[DEL]

American
Standard
Code for
Information
Interchange



Dec	Hex	Oct	Char	Dec	Hex	Oct	Html	Chr	Dec	Hex	Oct	Html	Chr	Dec	Hex	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	#32;	Space	64	40	100	#64;	@	96	60	140	#96;]
1	1	001	SOH (start of heading)	33	21	041	#33;	!	65	41	101	#65;	A	97	61	141	#97;	a
2	2	002	STX (start of text)	34	22	042	#34;	"	66	42	102	#66;	B	98	62	142	#98;	b
3	3	003	ETX (end of text)	35	23	043	#35;	#	67	43	103	#67;	C	99	63	143	#99;	c
4	4	004	EOF (end of transmission)	36	24	044	#36;	\$	68	44	104	#68;	D	100	64	144	#100;	d
5	5	005	ENQ (enquiry)	37	25	045	#37;	%	69	45	105	#69;	E	101	65	145	#101;	e
6	6	006	ACK (acknowledge)	38	26	046	#38;	&	70	46	106	#70;	F	102	66	146	#102;	f
7	7	007	BEL (bell)	39	27	047	#39;	'	71	47	107	#71;	G	103	67	147	#103;	g
8	8	010	BS (backspace)	40	28	050	#40;	(72	48	110	#72;	H	104	68	150	#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	#41;)	73	49	111	#73;	I	105	69	151	#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	#42;	*	74	4A	112	#74;	J	106	70	152	#106;	j
11	B	013	VT (vertical tab)	43	2B	053	#43;	+	75	4B	113	#75;	K	107	71	153	#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	#44;	,	76	4C	114	#76;	L	108	72	154	#108;	l
13	D	015	CR (carriage return)	45	2D	055	#45;	-	77	4D	115	#77;	M	109	73	155	#109;	m
14	E	016	SO (shift out)	46	2E	056	#46;	.	78	4E	116	#78;	N	110	74	156	#110;	n
15	F	017	SI (shift in)	47	2F	057	#47;	/	79	4F	117	#79;	O	111	75	157	#111;	o
16	10	020	DLE (data link escape)	48	30	060	#48;	0	80	50	120	#80;	P	112	76	160	#112;	p
17	11	021	DC1 (device control 1)	49	31	061	#49;	1	81	51	121	#81;	Q	113	77	161	#113;	q
18	12	022	DC2 (device control 2)	50	32	062	#50;	2	82	52	122	#82;	R	114	78	162	#114;	r
19	13	023	DC3 (device control 3)	51	33	063	#51;	3	83	53	123	#83;	S	115	79	163	#115;	s
20	14	024	DC4 (device control 4)	52	34	064	#52;	4	84	54	124	#84;	T	116	80	164	#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	#53;	5	85	55	125	#85;	U	117	81	165	#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	#54;	6	86	56	126	#86;	V	118	82	166	#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	#55;	7	87	57	127	#87;	W	119	83	167	#119;	w
24	18	030	CAN (cancel)	56	38	070	#56;	8	88	58	130	#88;	X	120	84	170	#120;	x
25	19	031	EM (end of medium)	57	39	071	#57;	9	89	59	131	#89;	Y	121	85	171	#121;	y
26	1A	032	SUB (substitute)	58	3A	072	#58;	:	90	5A	132	#90;	Z	122	86	172	#122;	z
27	1B	033	ESC (escape)	59	3B	073	#59;	;	91	5B	133	#91;	[123	87	173	#123;	{
28	1C	034	FS (file separator)	60	3C	074	#60;	<	92	5C	134	#92;	\	124	88	174	#124;	
29	1D	035	GS (group separator)	61	3D	075	#61;	=	93	5D	135	#93;]	125	89	175	#125;	}
30	1E	036	RS (record separator)	62	3E	076	#62;	>	94	5E	136	#94;	^	126	90	176	#126;	~
31	1F	037	US (unit separator)	63	3F	077	#63;	?	95	5F	137	#95;	_	127	91	177	#127;	[DEL]

아스키(ASCII) : 미국 국립 표준협회(ANSI, American National Standards Institute)

현대는 Unicode를 더 많이 사용

데이터 타입

- 컴퓨터 시스템과 프로그래밍 언어에서 문자, 실수, 정수, 논리값 등의 여러 종류의 데이터를 식별 가능하게 하는 것
- 데이터의 형태, 의미, 크기와 해당 자료형의 값이 저장되는 방식

Example	Type
"A", "Hello"	문자(character)
10, 20, 1.178	실수(float, numeric)
5, 10, 2	정수(integer)
TRUE, FALSE	논리값(logical)

수치 자료형 (예: 1, 0, -10)

문자 자료형 (예 : 작은 따옴표 안의 문자, 큰 따옴표 안의 문자)

문자 자료형 (예 : "3", "5.1")

불린(Boolean) 자료형 (예 : True, False)

: → 콜론

"", '' → 큰따옴표, 작은따옴표

, → 쉼표

자료형을 저장하고, 쓰는 법(수치형)

```
a=1  
b=2  
c=3  
d=4
```

ex) a = 1

변수 a 에 1을 저장

```
a+b, a-b, a*c
```

(3, -1, 3)

```
print(a+b)  
print(a+b, a-b, a*c)
```

3

3 -1 3

% : 나머지 연산

ex) 5%2 = 1

5/2 했을 때, 몫 2, 나머지 1

```
print(a+b)  
print(d/b)  
print(d%c)
```

2

2.0

1

```
In [97]: a=input()
```

```
10
```

```
In [98]: type(a)
```

```
Out[98]: str
```

```
In [101]: a=float(input())
```

```
10
```

```
In [102]: type(a)
```

```
Out[102]: float
```

```
In [99]: a=int(input())
```

```
10
```

```
In [100]: type(a)
```

```
Out[100]: int
```

Input을 이용해 변수를 저장할 수 있음

이때, 변수들의 type을 정확하게 지정해 주는 것이 중요함

코드	설명
<code>\n</code>	문자열 안에서 줄을 바꿀 때 사용
<code>\t</code>	문자열 사이에 탭 간격을 줄 때 사용
<code>\\</code>	문자 <code>\</code> 를 그대로 표현할 때 사용
<code>\'</code>	작은따옴표(')를 그대로 표현할 때 사용
<code>\"</code>	큰따옴표(")를 그대로 표현할 때 사용
<code>\r</code>	캐리지 리턴(줄 바꿈 문자, 현재 커서를 가장 앞으로 이동)
<code>\f</code>	폼 피드(줄 바꿈 문자, 현재 커서를 다음 줄로 이동)
<code>\a</code>	벨 소리(출력할 때 PC 스피커에서 '뽕' 소리가 난다)
<code>\b</code>	백 스페이스
<code>\000</code>	널 문자

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

이중에서 활용빈도가 높은 것은 `\n`, `\t`, `\\`, `\'`, `\"`이다. 나머지는 프로그램에서 잘 사용하지 않는다.

```
a=123
b=-123
c=0
```

정수형 : 소수점으로 표현하지 않는 수. Python에서 int로 정수를

```
type(a)
```

```
int
```

```
a=1.2
b=-2.1
```

실수형 : 소수점으로 표현해야 하는 수 Python에서 float로 실수를 표현

```
type(b)
```

type(a) : a의 자료형을 반환함. 예시 : int, float, str, object 등

```
float
```

```
a=1.24e-2
```

en(n은 숫자) : 10의 n승을 의미함

```
a
```

예시: $1.24e-2 : 1.24 \times 10^{-2} = 0.0124$

```
0.0124
```

```
b=1.24E2
```

```
b
```

```
124.0
```

```
a+b #더하기 연산
```

```
124.0124
```

```
a-b #빼기 연산
```

```
-123.9876
```

```
a*c #곱하기 연산
```

```
0.0
```

```
a**c #제곱 연산
```

```
1.0
```

```
7%3 #나머지 반환
```

```
1
```

```
3%7 #나머지 반환
```

```
3
```

```
7/4 #나누기
```

```
1.75
```

```
7//4 #몫을 반환
```

```
1
```

논리형(Boolean 타입)

- True, False의 값을 갖는 자료형
- None, 공백, 0인 경우에 False이고 이외의 값은 True
- 비교연산자, 논리연산자의 결과값으로 반환됨

✓ [18] 10<15

True

✓ [19] 10<=5

조건이 맞을 시에는 True 반환

조건이 맞지 않을 시에는 False 반환

False

✓ [20] 10>61

False

✓ [18] print(bool(1))

0 이외의 값이 들어가게 되면 True

True

✓ [19] print(bool(0))

나머지 값이 들어가게 되면 False

False

✓ [20] print(())

()

✓ [21] print(bool(""))

빈 문자열 False

False

✓ [22] print(bool("안녕"))

비어 있지 않은 문자열 True

True

논리형(Boolean 타입)

- True, False의 값을 갖는 자료형
- None, 공백, 0인 경우에 False이고 이외의 값은 True
- 비교연산자, 논리연산자의 결과값으로 반환됨

✓
0초

[23] 1 == 1

True

a = b : a에 b를 대입한다(변수선언)**a == b : a와 b는 같다(조건식)**✓
0초

[24] 1 == 0

False

✓
0초

[30] 1 != 1 # 같지 않다

a != b : a와 b는 같지 않다✓
0초

[31] not 1 == 1 # 같지 않다

False

not a == b : 두 숫자가 같지 않을 때✓
0초 1 == 1 and 1 == 2**and : 두 조건이 모두 만족****or : 두 조건 중 하나만 만족**

False

✓
0초

[33] 1 == 1 or 1 == 2

True

문자형(String)

```
"Life is too short, You need Python"
```

```
'Life is too short, You need Python'
```

1

```
"a"
```

```
'a'
```

2

```
"123"
```

```
'123'
```

```
food = "Python's favorite food is perl"
```

3

```
type(food)
```

```
str
```

str(문자형) : Python에서 문자형을 표현할 때 큰따옴표 혹은 작은따옴표 안에

1

문자를 넣음

따옴표 안에 숫자를 넣으면 숫자(int, float)로 인식 안함.

2

문자로 인식

type(a) : a의 자료형을 반환함.

3

문자의 자료형 : str

```
python='python'
```

```
a="python"
```

```
b=2
```

4

```
a*b
```

5

```
'pythonpython'
```

```
print("="*50)
print("My Program")
print("="*50)
```

```
=====
My Program
=====
```

str형 * int형 : str형인 문자를 int형 숫자만큼 반복 5

모든 유형의 객체 집합 (all kinds of objects)

리스트 (List)

동일한 유형의 데이터 (same type of objects)

배열 (Array)

(2차원 이상, more than 2 dimensions)

행렬 (Matrix)

(2차원, 2 dimension)

벡터 (Vector)

(1차원, 1 dimension)

스칼라 (Scala)

(구성인자 1개, 1 element)

요인
(Factor)

(범주형
데이터,
Categorical
data)

서로 다른 유형의 데이터 (different type of objects)

데이터 프레임 (Dataframe)

(2차원, 2 dimension)

숫자형
(numeric)

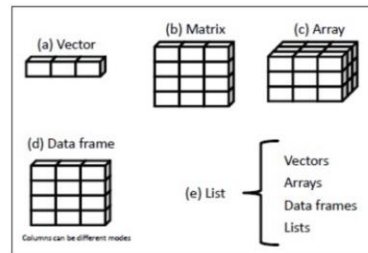
문자형
(character)

...

논리형
(logical)

Data structures

- a) Vector
- b) Matrix
- c) Array
- d) Data frame
- e) List



스칼라 : 일반적으로 숫자(예: 정수 또는 부동 소수점), 문자, Bool 또는 컬렉션이 아닌 기타 값(예: 목록 또는 부동 소수점)이 될 수 있는 단일 값을 나타냄

스칼라는 가장 간단한 형태의 데이터로, 하위 구조가 없는 단일 데이터 단위를 나타냄

A="안녕하세요 지금은 데이터 전처리 수업 시간입니다."

A=10

A=3.14

A=TRUE


스칼라 : 일반적으로 숫자(예: 정수 또는 부동 소수점), 문자, Bool 또는 컬렉션이 아닌 기타 값(예: 목록 또는 부동 소수점)이 될 수 있는 단일 값을 나타냄

스칼라는 가장 간단한 형태의 데이터로, 하위 구조가 없는 단일 데이터 단위를 나타냄

A="안녕하세요 지금은 데이터 전처리 수업 시간입니다."

A="안녕하세요 지금은 데이터 전처리 수업 시간입니다."

0 1 2 3 4 5 6 7 8 9 10 26


A=123

선형구조-리스트

- 데이터의 유형상관 없이 저장하고, 저장된 데이터들을 그룹화할 수 있는 데이터 구조
- 숫자, 문자, 논리값 ... 등등 다양한 데이터 유형의 요소가 포함될 수 있음

리스트명 = [요소1, 요소2, 요소3, ...]

A = [1, 2, 3, 4, 5]

type(A)

리스트명 = list[요소1, 요소2, 요소3, ...]

B = list[1, 2, 3, 4, 5]

type(B)



실행은 되지만
사용하지 않음

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

리스트명 = [요소1, 요소2, 요소3, ...]

a="독고영재, 50"

b="김철수, 30"

리스트명 = ["독고영재, 50", "김철수, 30", ...]

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

리스트명 = [**요소1**, **요소2**, **요소3**, ...]

리스트는 문자와 같이 각각의 공간을 가진다.

A="안녕하세요 지금은 0000 수업 시간입니다."

0 1 2 3 4 5 6 7 8 9 10 23

A = [10, 20, 30, ...]

0

1

2

...

튜플 : 불변성(한 번 생성되면 해당 요소를 추가, 제거 또는 변경할 수 없음)은 특정 상황에서 여러 가지 효율성과 이점을 제공함

튜플 = (**요소1**, **요소2**, **요소3**, ...)

튜플은 리스트와 같이 각각의 공간을 가진다.

A = [10, 20, 30, ...]
 0 1 2 ...

A = (10, 20, 30, ...) → A=list((10, 20, 30))
 0 1 2 ...

프로그램에서 벡터란?

- 값을 저장하고, 조작할 수 있는 기본 데이터 구조
- 숫자, 문자 또는 논리 값과 같은 **동일한 데이터 유형**의 요소를 보유할 수 있는 1차원 배열
- Python의 벡터는 [**요소1**, **요소2**, **요소3**, ...]로 표현할 수 있음

List(리스트)

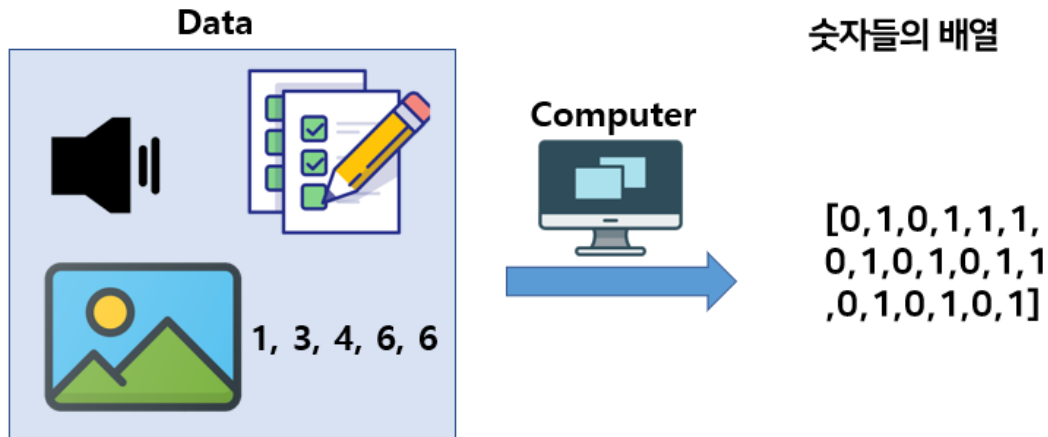
- 자료를 순서대로 한 줄로 저장하는 자료구조, **동일한 데이터 유형이 아닌** 모든 데이터 유형을 하나의 리스트에 저장 가능
- 여러 자료가 일직선으로 서로 연결된 선형 구조(리스트에 있는 데이터는 몇 번째 인지 의미를 가짐)

Array(배열)

- 단일 타입으로 구성되는 자료구조

리스트는 계산이 안돼...
다양한 타입을 담을 수 있기 때문에.

계산할 수 있도록 만들기 array

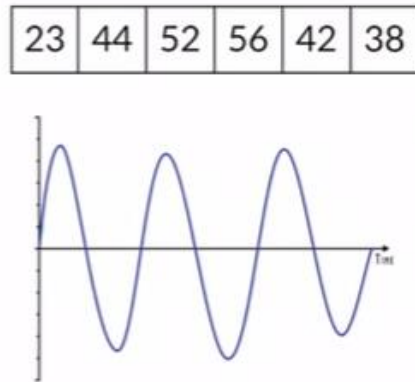
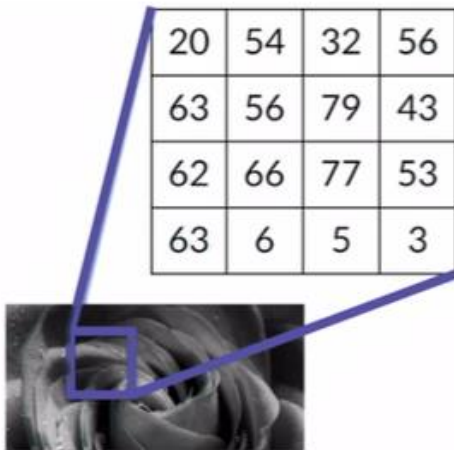


대규모 다차원 배열

- 데이터의 대부분은 숫자 배열로 볼 수 있음
- 흑백 이미지는 픽셀의 밝기와 명암을 2차원 배열로 표현할 수 있고 소리 같은 경우는 1차원 배열로 나타낼 수 있음

List(리스트)와 Array(배열)

- List는 [1,2,"Kim",2.5,True,False]와 같은 실수형, 정수형, 문자열과 같은 다양하게 관계없이 구성이 가능함
- array(배열)는 모두 단일 타입으로 구성됨



cmd (기존 cmd 와는 다른 cmd 창에서)

pip install numpy

```
import numpy as np
```

패키지명

패키지명을 요약해 불러올 이름

1차원 배열 or 벡터

```
a=np.array([1,2,3])
```

```
b=np.array([5,6,7])
```

```
c=a+b
```

```
type(c)
```

```
c.shape
```

리스트

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

```
b=[5,6,7]
```

```
c=a+b
```

```
type(c)
```

```
import numpy as np
```

패키지명

패키지명을 요약해 불러올 이름

2차원 배열 or 행렬

```
a=np.array([[1,2,3], [4,5,6]])
```

```
b=np.array([[5,6,7], [8,9,10]])
```

```
c=a+b
```

```
type(c)
```

```
c.shape
```


(시험문제나옴) 파일 새로 만들기 > 경로 부분에 cmd 치고 가상환경 활성화

```
import numpy as np
```

패키지명

패키지명을 요약해 불러올 이름

> 주피터 노트북

실행

> 파일 저장

n차원 배열

```
a=np.array([[[1,2,3], [4,5,6], [7,8,9]]])
```

```
a.shape
```

```
→(1,3,3)
```

하나의 3차원 배열, 안에 3개의 배열, 안에 3개의 요소

문자열은 인덱싱을 사용할 수 있음

인덱싱 : 연속적인 객체들에(예: 리스트, 튜플, 문자열) 범위를 지정해 선택해서 객체들을 가져오는 방법
및 표기법을 의미함

A="안녕하세요 지금은 데이터 전처리 수업 시간입니다."

A[시작범위:직전범위]

A[0:4] = 안녕하세

문자열은 인덱싱을 사용할 수 있음

인덱싱 : 연속적인 객체들에(예: 리스트, 튜플, 문자열) 범위를 지정해 선택해서 객체들을 가져오는 방법
및 표기법을 의미함

A="안녕하세요 지금은 데이터 전처리 수업 시간입니다."

A[-1] = . → 맨뒤는 0으로 할 수 없으므로 -1 부터

A[-5:] = 간입니다.

03 데이터의 인덱싱(format)

```
print("안녕하세요\n반갑습니다")
```

Wn : 줄바꿈

안녕하세요
반갑습니다



Int 형 : %d, str형 : %s, float형 : %f

```
print("이번시험의 성적으로 %d점을 맞았습니다." % 3)
```

ex) "~%d~" % 3 : %d 자리에 3을 넣어 줌

이번시험의 성적으로 3점을 맞았습니다.

```
print("이번시험의 성적으로 %0.3f점을 맞았습니다." % 3.333333)
```

%0.3f : 소수점 3번째까지만 보여줌

이번시험의 성적으로 3.333점을 맞았습니다.

```
print(("이번시험의 성적으로 {}점을 맞았습니다.").format(100))
```

이번시험의 성적으로 100점을 맞았습니다.

```
print(("이번시험에서 수학은 {}점 영어는 {}점 국어는 {}점을 맞았습니다.").format(100, 60, 60))
```

이번시험에서 수학은 100점 영어는 60점 국어는 60점을 맞았습니다.

(~{}~{}~{}).format(a,b,c)

{ }의 개수와 format뒤 ()안의 매개변수 개수가 같아야 함

맨 앞 {}부터 차례로 format뒤의 값을 넣어줌

format : 문자열을 formatting 하는 방법으로 문자열 중간중간 특정 변수의 값을 넣어주기 위해 사용되는 것

format(중괄호 사이에 들어갈 값)

A="I am a "
B="boy"

A+B→I am a boy

A="I am a {}"
B="boy"

A.format(B)

format은 print에서도 활용되고 단순 문자열에서도 사용됨

format(중괄호 사이에 들어갈 값)

Print("{} {}".format(첫번째 값, 두번째 값))

"I am a {}".format(첫번째 값) → format은
따옴표로도 가능

format은 print에서도 활용되고 단순 문자열에서도 사용됨

```
print("{:.2f}, {:.3f}".format(3.1415, 3.1415))
```



```
3.14,  
3.141
```

Replace : 특정 문자열을 찾아서 다른 문자열로 대체 가능함

변수.replace("기존의 문자", "변환하고 싶은 문자")

url = "https://www.youtube.com/"

str = url.replace("https://", "")

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

$A = [10, 20, 30, 40, 50, 60, 70, 80, 90]$

$A[\text{시작범위:직전범위}]$

$A[0:4] = \text{안녕하세}$



$A[\text{시작범위:직전범위}]$

$A[0:4] = [10, 20, 30, 40]$

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

$A = [10, 20, 30, 40, 50, 60, 70, 80, 90]$

$A[0] = 10$

$A[5] = 60$

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

$A = ["\text{독고영재}, 50", "\text{김철수}, 30"]$

$A = "\text{안녕하세요 지금은 데이터 전처리 수업 시간입니다.}"$

$A[\text{시작범위:직전범위}]$

$A[0] = \text{독고영재}, 50$ $A[0][0:4] = \text{독고영재}$

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

변수[:변수.index("직전까지 문자")]

a="독고영재, 50" b="김철수, 30"

a[:a.index(",")] b[:b.index(",")]

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

`a = ["독고영재, 50", "김철수, 30"]`

`b = ["김철수", 20, "김영희, 25"]`

`a[:a.index(",")]` 는

이용해서 독고영재 놓아주세요

`b[:b.index(",")]`

김영희



어떻게 바꿀까?

리스트는 다양한 type을 가질 수 있음

```
a=["hi", 10, "80", "김철수", 50, 50.6]
```

```
type(a[1]) → int
```

```
type(a[3]) → str
```

```
type(a[5]) → float
```

리스트를 추가하는 방법

```
a=["hi", 10, "80", "김철수", 50, 50.6]
```

```
a.append(추가하고 싶은 새로운 값)
```

```
a.append(1)
```

```
a=["hi", 10, "80", "김철수", 50, 50.6, 1]
```

리스트의 값을 삭제하는 방법

```
a=["hi", 10, "80", "김철수", 50, 50.6]
```

```
a.pop(제거하고 싶은 값의 위치)
```

```
a.pop(1)
```

```
a=["hi", "80", "김철수", 50, 50.6, 1]
```


리스트의 값을 삭제하는 방법

```
a=["hi", 10, "80", "김철수", 50, 50.6]
```

```
a.index(50)
```

```
a.pop(4)
```

```
a=["hi", 10, "80", "김철수", 50.6]
```

리스트의 값을 삭제하는 방법

```
a=["hi", 10, "80", "김철수", 50, 50.6]
```

```
b=a.index("80")
```

```
a.pop(b)
```

```
a=["hi", 10, "김철수", 50, 50.6]
```

배열의 값을 추가하는 방법

```
a=np.array([1,2,3,4])
```

```
a=np.append(a, [4, 5])
```

```
a=np.insert(a, 1, [7, 8])
```

 처음 나오는 값에 적용함

배열의 값을 제거하는 방법

```
a=np.array([1,2,3,4,5])
```

```
a=np.delete(a, [1, 3])
```



삭제하고 싶은 값의 위치

```
a= a[a<= 4]
```

```
a= a[a!=4]
```

배열 수직으로 합치기

```
array1 = np.array([[1, 2], [3, 4]])  
array2 = np.array([[5, 6], [7, 8]])  
merged_array = np.vstack((array1, array2))  
print(merged_array)
```

배열 수평으로 합치기

```
array1 = np.array([[1, 2], [3, 4]])  
array2 = np.array([[5, 6], [7, 8]])  
merged_array = np.hstack((array1, array2))  
print(merged_array)
```

딕셔너리 : 키-값 으로 쌍을 저장하는 효율적인 자료구조로 빠른 데이터 조회, 수정, 삭제 가능
키값은 중복이 불가능하며 하나의 키 값을 정해 키 값 안에 변수값을 넣어주는 방법

$A = \{\text{키값} : \text{변수값}, \text{키값} : \text{변수값}\}$

$A[\text{키값}] = \text{변수값}$

$A[\text{키값}] = \text{변수값}$

딕셔너리 : 하나의 키 값을 정해줘 키 값 안에 변수를 넣어주는 방법

A={1: "김철수", 20: "박영희"}

A[1] = 김철수

A[20] = 박영희

A.get(1) = 김철수

A.get(3) = None

딕셔너리의 키값을 설정하는것에 있어 꼭 수치값이 아니라도 괜찮음

$A = \{ "1-A": "김철수", "20-B": "박영희" \}$

$A["1-A"] = \text{김철수}$

$A["20-B"] = \text{박영희}$

$A.get("1-A") = \text{김철수}$

$A.get("20-B") = \text{박영희}$

딕셔너리의 추가 삭제

$A = \{ "1-A": "김철수", "20-B": "박영희" \}$

추가 : $A["2-A"] = "영희"$

삭제 : $\text{del } A["20-B"]$

하나의 키 값과 다수의 값을 가지는 리스트로 생성 가능(모든 자료형에 대해 저장 가능)

```
A={  
    "1-A": [1,2,3,4,5],  
    "20-B": ["A","B","C","D","E"]  
}
```

- 1) 하나의 리스트를 만들고, 해당 리스트의 값의 결과가 첫번째 :10, 두번째 :20, 세번째 : 30, 네번째 : [1,2,3] 값이 들어가도록 만드시오.
- 2) 해당 리스트를 인덱싱해 [1,2,3]의 값이 도출되게 하시오.
- 3) 수학점수 : 20, 30, 40, 50, 60, 영어점수 : 50, 70, 50, 60, 90 으로 두개의 리스트로 생성하시오.
- 4) <https://www.youtube.com/> 를 www.naver.com으로 변경하시오.
- 5) 해당 리스트의 값을 계산해 수학점수와 영어점수 각각의 값의 평균을 계산하시오.
- 6) 1,2,100,4,5, 900 해당 값을 가지는 배열을 생성하시고, 100과 900을 제거하시오.
- 7) 문제 3번의 값을 딕셔너리 형태로 저장하시오.
- 8) 3번의 리스트를 딕셔너리 형태로 저장하고, 수학점수 첫번째 값과, 영어점수 세번째 값의 합을 계산하시오.