



자동화를 위한 파이썬 언어 핵심 요약 (01. 자료형)

김효관

파이썬 기초 다지기

단원 개요

[단원명]

자동화를 위한 파이썬 언어 핵심 요약

[단원 소개]

- 파이썬 개발환경 구축 후 파이썬 프로그램 개발을 위한 기초를 다진다. 파이썬에서 사용하는 자료형과 핵심문법을 알면 코드작성 시 많은 노가다 작업을 피할 수 있다. 앞으로 코딩하기 즐겁게 하기 위한 기본기를 다지는 단원이다.

[교육대상]

- 데이터 분석가 / 인공지능 전문가
- 데이터 엔지니어

내용	학습내용
자료형	<ul style="list-style-type: none">- 값을 어떻게 저장하는지 실습한다.- 자주 사용하는 자료형 사용법을 실습한다.
파이썬 라이브러리 설치하기	<ul style="list-style-type: none">- 라이브러리 개념을 이해한다.- 라이브러리 설치 방법을 실습한다.
파이썬 핵심문법	<ul style="list-style-type: none">- 반복 노가다 작업을 쉽게 해결하는 방법을 실습한다.- 조건에 따라 문제를 해결하는 방법을 실습한다.- 재사용 가능한 코드를 모듈화 하는 방법을 실습한다.

Review (개요 및 파이썬 개발환경 구축)

1

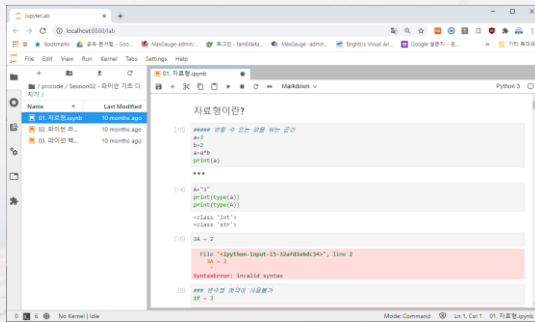
1. 최신테크 2. 개발자직군 (엔지니어/분석가/플랫폼) 3. 데모 4. 개발환경 구축

2

파이썬은 배우기 쉽고 실무에 많이 활용되는 언어이다.

3

개발환경 구축 및 환경설정 방법 확인하기.



개발 IDE (Integrated Developing Environment)

4차산업혁명 단계별로 익히는 빅데이터&인공지능(광문각, 김호관 교수)

www.youtube.com/hkcode

교육목표: Python 기본 사용법 및 프로그램 시 활용하는 자료형을 알수 있다.

CONTENTS

1 Python 코드 테스트

2 Python 자료형

3 핵심정리 및 Q&A



1. 파이썬 코드 테스트

jupyter lab 활용

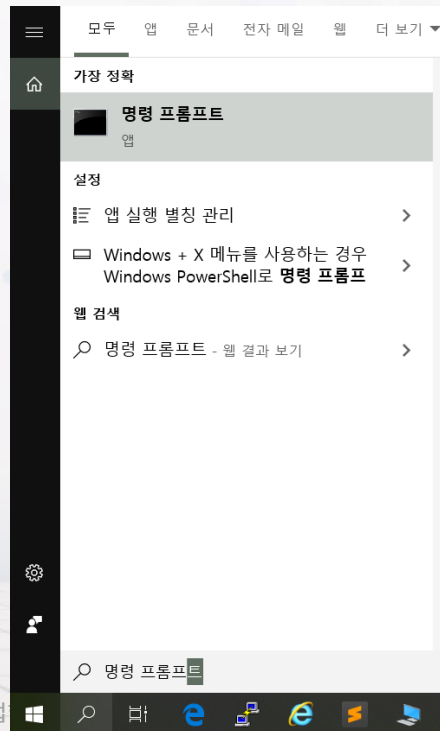
1 줄 단위의 순차적인 실행 가능한 Interactive 환경 제공

2 차트 등 데이터 시각화 용이

실행

```
명령 프롬프트
Microsoft Windows [Version 10.0.17763.973]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Wkopo>jupyter lab
```



1. 파이썬 코드 테스트

Jupyter lab 주요 단축키

모드	단축키	설명
명령 모드 Y: 코드전환 M: 마크다운 전환	Shift+M	셀 합치기
	L	Line 줄 보이기
	D+D	셀 삭제하기
	A	현재 셀 위에 셀 추가하기
	B	현재 셀 아래에 셀 추가하기
	Enter / Esc	에디트모드 진입 / 커맨드모드 진입
편집 모드	shift+tab	함수 사용예제
	. + (tab)	사용가능한 함수목록 시현
	Ctrl + Enter / Shift + Enter	그냥 실행 / 실행 후 이동
	Ctrl + /	주석 적용 / 해제
	Ctrl + Shift + '-'	셀 나누기

* 명령모드에서 H 키 누르면 단축키 목록이 시현됨

**어떤 프로그래밍 언어든
그 언어의 자료형을
알고 이해할 수 있다면 이미
그 언어의 절반을 터득한 것이나 다름없다**

2. 파이썬 자료형

자료형 이란?

1 프로그램은 연산을 위해 값을 임시/영구로 저장 필요

2 값을 저장하는 저장소 = 변수

3 변수의 유형 = 자료형



변수 생성 규칙

대소문자 구분

문자, 숫자, 밑줄로 구성 (숫자 처음 불가)

예약어 사용불가 (if, else, or, for 등)

```
##### 변환 수 있는 값을 담는 공간
a=3
b=2
a=a*b
print(a)
```

...

```
A="3"
print(type(a))
print(type(A))
```

```
<class 'int'>
<class 'str'>
```

```
3A = 2
```

File "<ipython-input-15-32afd1e6dc34>", line 2

```
3A = 2
```

^

SyntaxError: invalid syntax

2. 파이썬 자료형

Python 기본 자료형

프로그래밍을 할 때 쓰이는 숫자, 문자열 등 모든 자료형태

1	수(Number)	일반 숫자 정의	정수 (Int) / 100	실수 (float) / 12.345
2	문자열 (String)	홀 따옴표 / 쌍따옴표로 정의	"Python is my favorite"	
3	리스트 (List)	업데이트 가능한 집합	[1, 2, 3, 4, 5]	
4	튜플 (Tuple)	업데이트 불가능한 집합	(1, 2, 3, 4, 5)	
5	딕셔너리 (Dictionary)	Key-Value 정의	{ "product_name" : "led", "product_price" : 100 }	

2. 파이썬 자료형

1. 수(Number)

숫자 형태로 이루어진 자료형

예시

항목	숫자 예시	코드 예시	설명	비고
정수	123, -345, 0	<code>integer_value = 123</code>	양/음의정수, 0	-
실수	1234.45, -1234.5	<code>float_value = 123.456</code>	소수점이 포함된 숫자	

* type 확인은 `type(변수명)`으로 확인 가능하다

2. 파이썬 자료형

1. 수(Number)

사칙연산 실습

실습

```
integer_value = 123  
float_value = 12.456
```

1. 더하기 연산

```
sumation = integer_value+float_value  
print(sumation)
```

2. 곱하기 연산

```
multiple = integer_value*float_value  
print(multiple)
```

3. 나누기 연산

```
division = integer_value/float_value  
print(division)
```

4. 제곱연산

```
integer_value2 = int(float_value)  
exponential = integer_value2**2  
print(exponential)
```

5. 나머지 연산

```
mod = integer_value%int(float_value)  
print(mod)
```

현재 시간과 분을 입력값으로 받아서
두개의 값을 더하는 연산을 수행하세요

2. 파이썬 자료형

2. 문자열 (String)

문자, 단어 등으로 구성된 문자들의 집합

예시

생성방법	결과 예시	코드 예시	비고
홀 따옴표로 활용	This is 'String' value	aType = "This is 'String' value"	활용 추천
쌍 따옴표로 활용	This is "String" value	bType = 'This is "String" value'	
쌍따옴표 연속 3개 활용	This is 'String" value	cType = """This is 'String' value"""	전부 다 가능 (길어짐)

* type 확인은 type(변수명)으로 확인 가능하다

2. 파이썬 자료형

2. 문자열 (String)

문자열 기본연산 실습

예시

#문자열 더하기

```
head = "smart"
tail = "analytics"
fullString = head + " " + tail
print(fullString)
```

#문자열 곱하기

```
print("*"*20)
```

#문자열 인덱싱 0부터 시작

```
yearweek = "201801"
print(yearweek[0]) #첫번째 문자열
print(yearweek[0:4]) #0~4번째 전 문자열
print(yearweek[4:]) #4번째부터 끝까지
```

```
#문자열 더하기
head = "smart"
tail = "analytics"
fullString = head + " " + tail
print(fullString)

#문자열 곱하기 / 문자열간 + 생략 가능
print("*"*20)

#문자열 인덱싱 0부터 시작
yearweek = "201801"
print(yearweek[0]) #첫번째 문자열
print(yearweek[0:4]) #0~4번째 전 문자열
print(yearweek[4:]) #4번째부터 끝까지
```

```
smart analytics
*****
2
2018
01
```

2. 파이썬 자료형

2. 문자열 (String)

yearweek = "201801"

인덱스	[0]	[1]	[2]	[3]	[4]	[5]
	2	0	1	8	0	1
인덱스	[-6]	[-5]	[-4]	[-3]	[-2]	[-1]

예시	결과	비고
yearweek[3]	8	3번째 인덱스 문자 추출
yearweek[0:4]	2018	[0~3] 인덱스 문자 추출 * [0:4], 4는 포함되지 않음
yearweek[-1]	1	마지막 인덱스 문자 추출 * 음수 인덱스는 -1부터 시작
yearweek[-2:]	01	-2부터 끝까지 문자 추출
yearweek[:-2]	2018	처음부터 -2 전 인덱스 문자 추출

* 연속된 인덱스 값 추출 시 [2 : 4] ":" 기준으로 앞에는 시작 인덱스 뒤에는 끝 인덱스 (단, 숫자 안쓰면 맨앞 또는 맨끝)

2. 파이썬 자료형

2. 문자열 (String)

문자열 스텝업 연산 실습

Example

#문자열 개수세기 (count)

```
yearweek = "2017W28"  
print(yearweek.count('W'))
```

#문자열 위치확인 (index)

```
print(yearweek.index('W'))  
delimiter=yearweek.index('W')  
newYearweek = yearweek[:delimiter]+yearweek[delimiter+1:]  
print(newYearweek)
```

대소문자 구별함수 (upper/lower)

```
letters = "LeD_tv"  
print(letters.upper())  
print(letters.lower())
```

문자 치환 (replace)

```
repLetters = "2017W28"  
print(repLetters.replace("W", "_"))
```

문자 분리 (split)

```
yearweek_list = yearweek.split("W")  
print(yearweek_list)
```


2. 파이썬 자료형

[참조] 형변환

파이썬은 자료형을 알아서 문자를 숫자로 변경하는 등 타입변환이 가능함
예) 2017W21 -> 문자열 연산통해 “W” 제거 -> 숫자형태로 변환 후 숫자 연산

Example

#문자 → 수(정수)

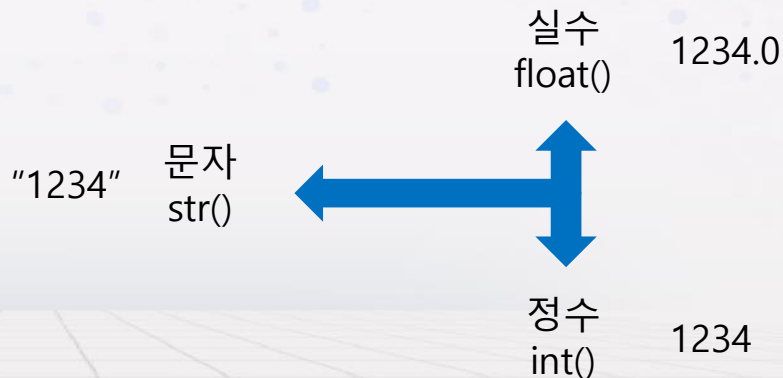
```
strValue = '1234'  
cvIntValue = int(strValue)  
print(type(cvIntValue))
```

#문자 → 수(실수)

```
strValue = '1234'  
cvFloatValue = float(strValue)  
print(type(cvFloatValue))
```

#숫자 → 문자

```
numberValue = 1234  
cvStrValue = str(numberValue)  
print(type(cvStrValue))
```



2. 파이썬 자료형

[참조] 대소문자 변환

파이썬은 대/소문자를 구분한다. 대문자는 문자열 함수(uppercase) 소문자는 문자열 함수 (lowercase)

Example

#소문자 → 대문자

```
lowerletter = " aaa "
```

```
upperletter = lowerletter.upper()
```

```
upperletter
```

#대문자 -> 소문자

```
final = upperletter.lower()
```

```
final
```

참조 [대소문자 변환]

대문자변환 및 문자 분리

```
lowerletter = "aaa"  
upperletter = lowerletter.upper()  
upperletter
```

```
'AAA'
```

```
final = upperletter.lower()  
final
```

```
'aaa'
```

4차

관 교수)

2. 파이썬 자료형

[참조] 문자열 분리

데이터 전처리 시 문자열을 특정 기준을 통해 분리하는 경우가 발생한다.

Example

#문자열 분리

```
orgletter = "2017w12"
```

```
splitter = "w"
```

```
answer = orgletter.split(splitter)
```

```
answer[0]
```

```
answer[1]
```

참조 [문자열 분리]

```
orgletter = "2017w12"
```

```
splitter = "w"
```

```
answer = orgletter.split(splitter)
```

```
answer[0]
```

```
'2017'
```

```
answer[1]
```

```
'12'
```

“SEC 20180212 250” 문자열 생성 후
stock_name, date, value로 각각 의미하는
문자열을 생성하세요

"Sec 2018W28 250" 문자열 생성 후
stock_name, date, value로 각각 의미하는
문자열을 생성하세요
(단 모두 대문자여야 하며, 'W'를 삭제해야함)

2. 파이썬 자료형

3. 리스트(List)

특정 집합 모음을 표현할 수 있는 자료형

예: TV목록 (UN42LEDTV, UN32LEDTV, UN55LEDTV,...) 집합

예시

생성방법	결과 예시	코드 예시
빈 리스트 생성	empty	emptyList = []
가격 집합 리스트 생성	[120,130,150,200,170]	pricelist = [120,130,150,200,170]
복합 리스트 생성	[SEC, 120, 130, [201712,201713]]	secInfo = ["SEC",120,130,[201712,201713]]

* type 확인은 type(변수명)으로 확인 가능하다

2. 파이썬 자료형

3. 리스트(List)

리스트 기본연산

Example

#리스트 연산

```
stockName = ["SEC"]  
priceList = [120,130]  
dateList = [201712,201713]  
secinfo = stockName+priceList+[dateList]
```

#리스트 인덱싱

```
secinfo=["SEC",120,130,[201712,201713]]  
print(secinfo[1])  
print(secinfo[3])  
print(secinfo[3][0])  
print(secinfo[0:3])
```

#리스트 길이

```
len(secinfo)
```

```
#리스트 연산  
stockName = ["SEC"]  
priceList = [120,130]  
dateList = [201712,201713]  
  
secinfo = stockName+priceList+[dateList]  
print(secinfo)  
  
#리스트 인덱싱  
#secinfo=["SEC", 120, 130, [201712, 201713]]  
  
print(secinfo[1])  
print(secinfo[3])  
print(secinfo[3][0])  
print(secinfo[0:3])  
  
['SEC', 120, 130, [201712, 201713]]  
120  
[201712, 201713]  
201712  
['SEC', 120, 130]
```

2. 파이썬 자료형

3. 리스트 (List)

scoreList = [70, 80, 60, 50, 90, 50]

인덱스

[0]	[1]	[2]	[3]	[4]	[5]
70	80	60	50	90	50
[-6]	[-5]	[-4]	[-3]	[-2]	[-1]

예시	결과	비고
scoreList[3]	50	3번째 리스트 값 추출
scoreList[0:4]	[70, 80, 60, 50]	[0~3] 리스트 추출 * [0:4], 4는 포함되지 않음
scoreList[-1]	50	마지막 인덱스 리스트 추출 * 음수 인덱스는 -1부터 시작
scoreList[-2:]	[90, 50]	-2부터 끝까지 리스트 추출
scoreList[:-2]	[70, 80, 60, 50]	처음부터 -2 전 인덱스 리스트 추출

2. 파이썬 자료형

3. 리스트(List)

리스트 값 업데이트

Example

#리스트 값 업데이트 하기

```
updateList = [120,130,140,150]
```

```
print(updateList)
```

```
updateList[2] = 180
```

```
print(updateList)
```

#리스트 값 삭제하기

```
updateList.remove(130)
```

```
del updateList[2]
```

```
print(updateList)
```

#리스트 구간 데이터 삭제하기 구간 (1 이 이후 인덱스 삭제)

```
updateList[1:] = []
```

▼ #리스트 값 업데이트 하기

```
updateList = [120,130,140,150]
```

```
print(updateList)
```

```
updateList[2] = 180
```

```
print(updateList)
```

#리스트 값 삭제하기

```
del updateList[3]
```

```
print(updateList)
```

```
[120, 130, 140, 150]
```

```
[120, 130, 180, 150]
```

```
[120, 130, 180]
```

2. 파이썬 자료형

3. 리스트(List)

리스트 관련 주요함수

Example

```
functionList = [120,130,140,150]
#리스트에 요소 추가하기
functionList.append(100)
functionList.insert(3,100) # 3번째 인덱스에 값 추가
functionList.extend([11,12])
print(functionList)
#리스트 정렬하기
functionList.sort(reverse = True)
print(functionList)
#값으로 인덱스 가져오기
print(functionList.index(150))
#리스트 자주사용하는 함수
print(len(functionList))
print(min(functionList))
print(max(functionList))
print(sum(functionList))
```

4. 리스트 관련 주요함수

```
functionList = [120,130,140,150]
#리스트에 요소 추가하기
functionList.append(100)
functionList.extend([11,12])
print(functionList)
#리스트 정렬하기
functionList.sort(reverse = False)
print(functionList)
#인덱스 가져오기
print(functionList.index(150))
#리스트 자주사용하는 함수
print(len(functionList))
print(min(functionList))
print(max(functionList))
print(sum(functionList))
```

```
[120, 130, 140, 150, 100, 11, 12]
[11, 12, 100, 120, 130, 140, 150]
6
7
11
150
663
```

2. 파이썬 자료형

3. 리스트(List)

리스트 관련 주요함수 #2

Example

#리스트에 요소 개수 세기

```
functionList = [120,130,140,150,120]
```

```
functionList.count(120) → 2 # 2개 있음
```

#리스트에 요소 빼기 (마지막 요소 빠짐)

```
functionList.pop()
```

```
#리스트 함수
functionList = [120,130,140,150]
print(functionList)
#리스트에 요소 추가하기
functionList.append(100)
print(functionList)
#리스트 정렬하기
functionList.sort()
print(functionList)
#인덱스 가져오기
print(functionList.index(150))
#리스트 자주사용하는 함수
print(len(functionList))
print(min(functionList))
print(max(functionList))
print(sum(functionList))
```

```
[120, 130, 140, 150]
[120, 130, 140, 150, 100]
[100, 120, 130, 140, 150]
4
5
100
150
640
```

[120,150,300,500,1000,100,2000]
값에서 최소 최대 값을 뺀 평균을
average 이름의 변수에 담으세요

2. 파이썬 자료형

4. 튜플(Tuple)

리스트와 유사하지만 값을 변경할 수 없다.

예: 시간 목록 (1,2,3,4,5,6,...) 집합

예시

생성방법	결과 예시	코드 예시
빈 튜플 생성	()	<code>emptyTuple = ()</code>
시간정보 생성	(1,2,3,4,5,6)	<code>timeTuple = (1,2,3,4,5,6)</code>

* type 확인은 type(변수명)으로 확인 가능하다

2. 파이썬 자료형

4. 튜플(Tuple)

튜플 기본연산

Example

#튜플 인덱싱

```
timeTuple = (1,2,3,4,5,6)
```

```
print(timeTuple[1])
```

```
print(timeTuple[3])
```

```
print(timeTuple[0:3])
```

튜플 길이

```
len(timeTuple)
```

```
emptyTuple = ()  
timeTuple = (1,2,3,4,5,6)
```

튜플 기본연산

```
#튜플 인덱싱  
timeTuple = (1,2,3,4,5,6)
```

```
print(timeTuple[1])  
print(timeTuple[3])  
print(timeTuple[0:3])
```

```
# 튜플 길이  
len(timeTuple)
```

2

4

(1, 2, 3)

6

2. 파이썬 자료형

5. 딕셔너리

(Key, Value) 형태의 자료 구조로 특정년도의 주식명을 키로 요청하면 주가총액을 value로 불러오는 형태
(2017년도 SEC, 주식총액)

예시

생성방법	코드 예시
딕셔너리 생성	<pre>dic = {"name": "sec", "id": "300000", "address": "suwon"} print(dic['name'])</pre>
딕셔너리 활용	<pre>dic.keys() dic.values() 가능 또한 list(dic.keys()) 로 형변환 가능</pre>

* type 확인은 type(변수명)으로 확인 가능하다

2. 파이썬 자료형

5. 딕셔너리

딕셔너리 관련 함수

Example

```
dic = {"name":"sec","id":"300000","address":"suwon"}
print(dic)
print(dic['name'])
```

dictionary key value 추가/변경

```
dic["name"] = "sec2"
dic["stock"] = "yes"
print(dic)
print(dic["stock"])
```

dictionary 요소 삭제하기

```
dic.pop("name")
print(dic)
```

```
dic = {"name":"sec","id":"300000","address":"suwon"}
print(dic)
print(dic['name'])
```

dictionary key value 추가하기

```
dic["name"] = "sec2"
dic["stock"] = "yes"
print(dic)
print(dic["stock"])
```

dictionary 요소 삭제하기

```
del dic["id"]
print(dic)
```

```
{'name': 'sec', 'id': '300000', 'address': 'suwon'}
sec
{'name': 'sec2', 'id': '300000', 'address': 'suwon', 'stock': 'yes'}
yes
{'name': 'sec2', 'address': 'suwon', 'stock': 'yes'}
```


2. 파이썬 자료형

5. 딕셔너리

딕셔너리 관련 고급 활용

Example

키 정의

```
tuple1 = ("미국", "LEDTV", "01")
```

```
tuple2 = ("미국", "LEDTV", "02")
```

```
dic = {tuple1: 2.5,  
       tuple2: 1.2}
```

딕셔너리 활용

```
dic["미국", "LEDTV", "01"]
```

딕셔너리 관련 고급 활용

키 정의

```
tuple1 = ("미국", "LEDTV", "01")
```

```
tuple2 = ("미국", "LEDTV", "02")
```

```
dic = {tuple1: 2.5,  
       tuple2: 1.2}
```

딕셔너리 활용

```
dic["미국", "LEDTV", "01"]
```

```
dic = {"name":"sec","id":"300000","address":"suwon"}  
에서 "id":"300000" 삭제하여 출력하세요  
{'name': 'sec', 'address': 'suwon'}
```

2. 파이썬 자료형

(추가자료) Pandas DataFrame

리스트를 활용한 데이터 분석을 위한 자료구조 생성

Example

```
import pandas as pd
# 리스트 생성
test = [10,100,1000,10000]
# 데이터프레임 변환
testDf = pd.DataFrame(test)
testDf

testDf.columns=["test"]
testDf
```

2. 리스트를 활용한 Pandas DataFrame 생성

```
import pandas as pd
# 리스트 생성
test = [10,100,1000,10000]
# 데이터프레임 변환
testDf = pd.DataFrame(test)
testDf
```

	0
0	10
1	100
2	1000
3	10000

```
testDf.columns=["test"]
testDf
```

	test
0	10
1	100
2	1000
3	10000

2. 파이썬 자료형

(추가자료) Pandas DataFrame

행/열 구조의 가장 많이 활용되는 자료구조 형태

Example

데이터 프레임 라이브러리 활용

```
import pandas as pd
```

딕셔너리를 활용한 데이터프레임 생성

```
data = {'name': ['A고객', 'B고객', 'C고객', 'D고객'],  
        'age': [27, 40, 33, 29],  
        'stock_age': [2, 10, 5, 1]}
```

```
dataFrame = pd.DataFrame(data)
```

데이터 프레임 : 스프레드시트 형태의 자료구조

```
print(dataFrame)
```

6. Pandas Dataframe

데이터 프레임 라이브러리 활용

```
import pandas as pd
```

딕셔너리를 활용한 데이터프레임 생성

```
data = {'name': ['A고객', 'B고객', 'C고객', 'D고객'],  
        'age': [27, 40, 33, 29],  
        'stock_age': [2, 10, 5, 1]}
```

```
dataFrame = pd.DataFrame(data)
```

데이터 프레임 : 스프레드시트 형태의 자료구조

```
print(dataFrame)
```

	name	age	stock_age
0	A고객	27	2
1	B고객	40	10
2	C고객	33	5
3	D고객	29	1

2. 파이썬 자료형

(추가자료) Pandas DataFrame

리스트를 활용한 데이터 분석을 위한 자료구조 생성

Example

```
import pandas as pd
date = ['16.02.29', '16.02.26', '16.02.25', '16.02.24', '16.02.23']
date2 = ['17.02.29', '17.02.26', '17.02.25', '17.02.24', '17.02.23']
```

```
date_df = pd.DataFrame(date, columns=['date2'])
date_df2 = pd.DataFrame(date2, columns=['date23'])
```

```
final = pd.concat([date_df, date_df2], axis = 1)
final
```

```
import pandas as pd
date = ['16.02.29', '16.02.26', '16.02.25', '16.02.24', '16.02.23']
date2 = ['17.02.29', '17.02.26', '17.02.25', '17.02.24', '17.02.23']
```

```
date_df = pd.DataFrame(date, columns=['date2'])
date_df2 = pd.DataFrame(date2, columns=['date23'])
```

```
final = pd.concat([date_df, date_df2], axis = 1)
final
```

	date2	date23
0	16.02.29	17.02.29
1	16.02.26	17.02.26
2	16.02.25	17.02.25
3	16.02.24	17.02.24
4	16.02.23	17.02.23

자신만의 데이터프레임을 생성해보세요

핵심정리 및 Q&A

기억합시다

1

변수는 변경 가능한 값을 저장하는 공간이다.

2

수는 사칙연산을 위해 필요하고, 특히 패턴을 찾기위해 나머지 연산을 한다.

3

자료형(문자)는 데이터를 붙이고 나누고 하는등의 데이터 조작을 위한 필수

4

리스트(집합), 딕셔너리 (키, value)활용법과 데이터프레임 생성방법 알고 넘어가기



파이썬 언어 핵심 요약 (02. 핵심문법)

김 호 관 |

2. 파이썬 기초 다지기

단원 개요

[단원명]

파이썬 기초 다지기

[단원 소개]

- 파이썬 개발환경 구축 후 파이썬 프로그램 개발을 위한 기초를 다진다. 파이썬에서 사용하는 자료형과 핵심문법을 알면 코드작성 시 많은 노가다 작업을 피할 수 있다. 앞으로 코딩하기 즐겁게 하기 위한 기본기를 다지는 단원이다.

[교육대상]

- 데이터 분석가 / 인공지능 전문가
- 데이터 엔지니어

내용	학습내용
자료형	<ul style="list-style-type: none">- 값을 어떻게 저장하는지 실습한다.- 자주 사용하는 자료형 사용법을 실습한다.
파이썬 라이브러리 설치하기	<ul style="list-style-type: none">- 라이브러리 개념을 이해한다.- 라이브러리 설치 방법을 실습한다.
파이썬 핵심문법	<ul style="list-style-type: none">- 반복 노가다 작업을 쉽게 해결하는 방법을 실습한다.- 조건에 따라 문제를 해결하는 방법을 실습한다.- 재사용 가능한 코드를 모듈화 하는 방법을 실습한다.

교육목표: 파이썬 핵심문법 및 데이터를 수집하는 방법을 익힌다.

CONTENTS

1

Python 기본문법

2

Pandas 라이브러리를 활용한 데이터 수집

3

핵심정리 및 Q&A



1. 파이썬 핵심문법

주요 문법

프로그래밍의 문법을 알면 다양한 각도로 문제를 해결할 수 있다.

1-1 반복하기

for / while

1-2 조건 판단하기

if

1-3 자주사용하는내용 함수화 하기

def testFunction(inVal) :

1. 파이썬 핵심문법

1. 반복하기 (for, while)



```
tvList = [ UN40EN001, UN40EN002, UN40EN003, UN40EN004]
```

헉! tv목록앞에 제품목록을 전부 붙여야하는데.... 어떻게하지?
하나씩 하면 되겠네..

```
preFix = "LEDTV_"  
tvList[0] = prefix + tvList[0]  
tvList[1] = prefix + tvList[1]
```

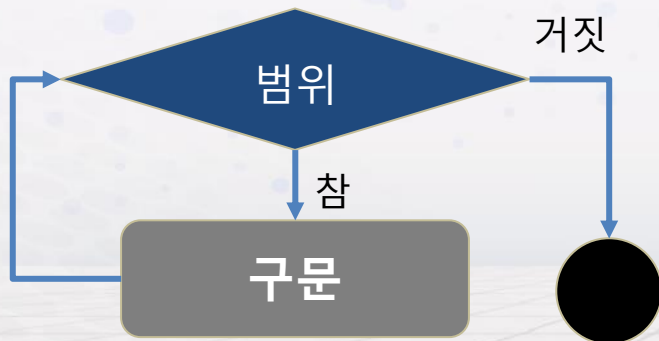
만약.. 목록이 십만개가 넘는다면??

1. 파이썬 핵심문법

1. 반복하기 (for, while)

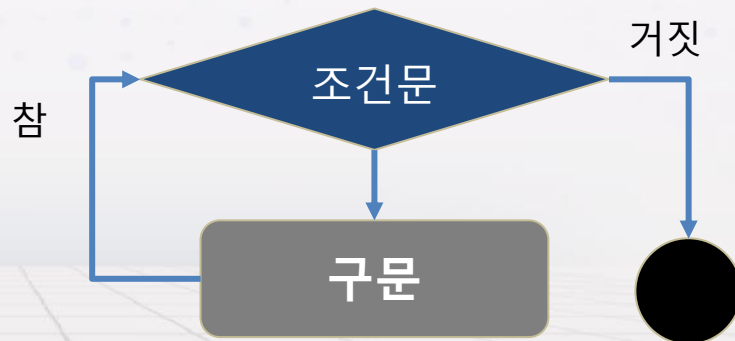
for

주어진 범위가 끝날때까지 구문 수행



while

조건문이 참인 경우 구문 수행
거짓인 경우 끝



1. 파이썬 핵심문법

참고. 비교연산자의 이해 및 논리

비교 연산자	설 명	예 시
$A == B$	A와 B가 같으면 참, 그렇지 않으면 거짓	$A == B$
$A != B$	A와 B가 다르면 참, 그렇지 않으면 거짓	$A != B$
$A > B$	A가 B보다 크면 참, 그렇지 않으면 거짓	$A > B$
$A < B$	A가 B보다 작으면 참, 그렇지 않으면 거짓	$A < B$
$A >= B$	A가 B보다 크거나 같으면 참, 그렇지 않으면 거짓	$A >= B$
$A <= B$	A가 B보다 작거나 같으면 참, 그렇지 않으면 거짓	$A <= B$

논리 연산자	설 명	예 시
$A \& B$	A 와 B가 모두참인경우 참	A 및 B 기능을 만족
$A B$	A 와 B중 하나만 참이면 참	A 또는 B 기능을 만족
$\sim A$	A가 아님	A를 제외한 항목

1. 파이썬 핵심문법

1. 반복하기 (for, while)

while (조건) loop : 조건 실패 시 조건 문 탈출
for (조건) : 조건 실패 시 조건 문 탈출

Example

줄 마지막에 백슬러시"\" 사용 시 줄이음 가능

```
tvList = [ 'UN40EN001', 'UN40EN002', 'UN40EN003', 'UN40EN004']
```

for 문

```
preFix = "LEDTV_"  
### list(range(0, 4, 1))  
for i in range(0, 4, 1):  
    tvList[i] = preFix + tvList[i]
```

범위/로직 구문을 위한
탭공백!

for문은 변수가 특정 범위안에서
구문연산 (구문은 탭으로 한칸땀)

while 문

```
preFix = "LEDTV_"  
  
i=0  
listLength = len(tvList)  
while(i < listLength ):  
    tvList[i] = preFix + tvList[i]  
    i = i+1
```

while문은 조건식을
만족하는 동안 구문연산

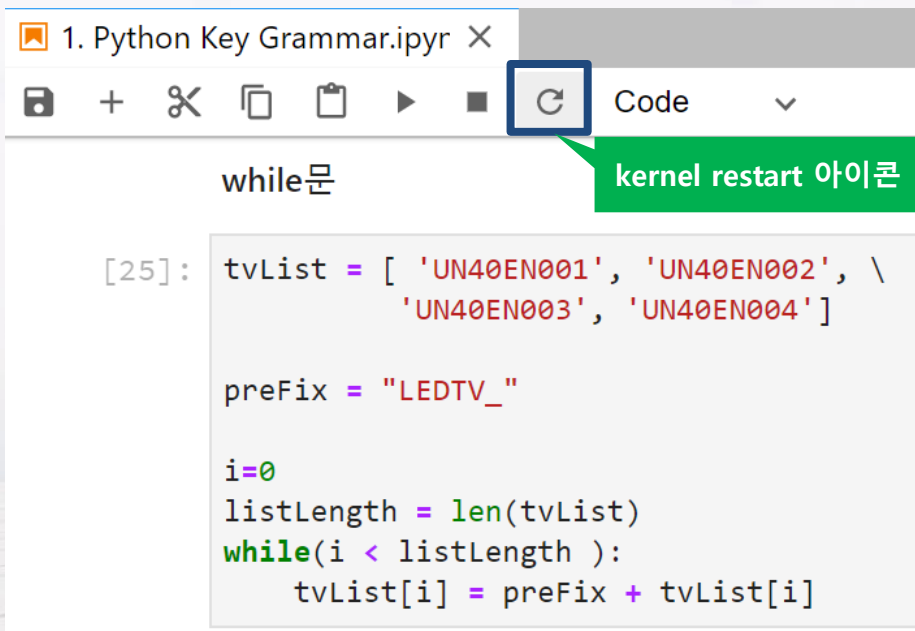
빠지면 무한 루프!

4차산업

1. 파이썬 핵심문법

1. 반복하기 (while, for)

while문에서 무한 루프에 빠지면?



1. Python Key Grammar.ipyr X

kernel restart 아이콘

```
[25]: tvList = [ 'UN40EN001', 'UN40EN002', \
               'UN40EN003', 'UN40EN004' ]

preFix = "LEDTV_"

i=0
listLength = len(tvList)
while(i < listLength ):
    tvList[i] = preFix + tvList[i]
```



```
tvList2 = [ 'UN40EN001', 'UN40EN002',  
            'UN40EN003', 'UN40EN004']  
리스트가 출력되도록 하세요
```

힌트: `print(tvList[0])`

1. 파이썬 핵심문법

2. 조건 판단하기



```
tvList = [ UN40EN001, LEDTV_UN40EN002,  
           LEDTV_UN40EN003, UN40EN004]
```

헉! tv목록앞에 제품목록을 전부 붙여야하는데.... 붙어있는게 있고 없는게 있네.. 어떻게 하지?
붙어 있는지 조건을 판단해야겠는데...

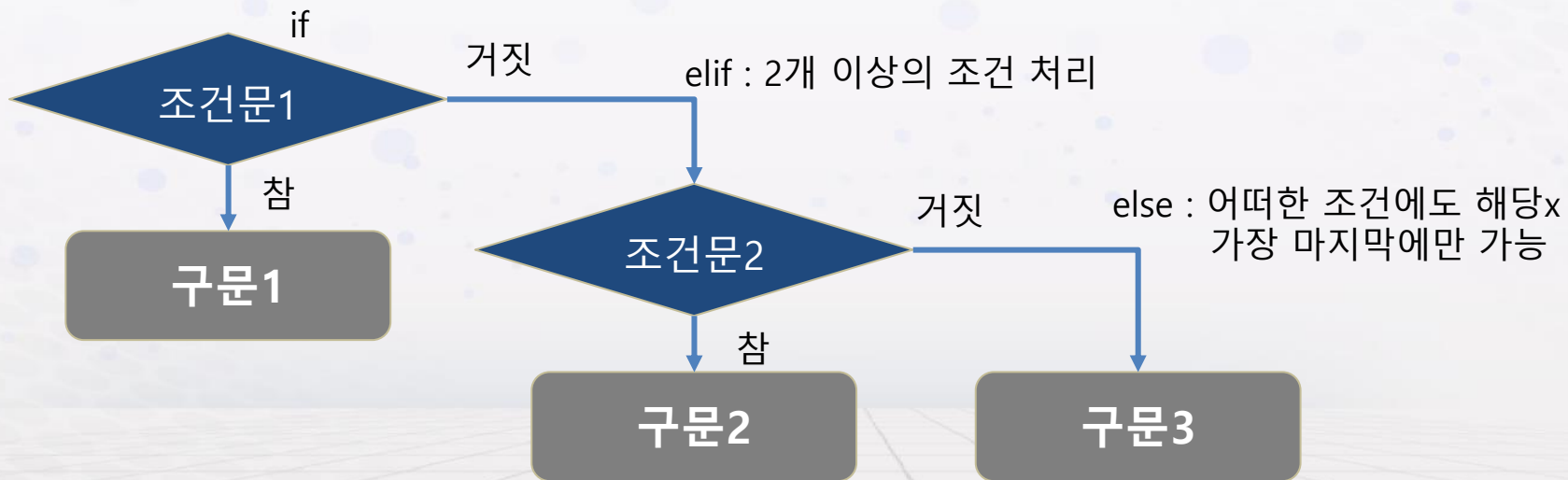
(만약, LEDTV_가 있으면 넘기고... LEDTV_가 없으면 넣고..)

조건문은 어떻게 사용할 수 있을까요?

1. 파이썬 핵심문법

2. 조건 판단하기

조건문을 평가하고 참인 경우만 구문 수행



1. 파이썬 핵심문법

참고. 비교연산자의 이해 및 논리

비교 연산자	설 명	예 시
$A == B$	A와 B가 같으면 참, 그렇지 않으면 거짓	$A == B$
$A != B$	A와 B가 다르면 참, 그렇지 않으면 거짓	$A != B$
$A > B$	A가 B보다 크면 참, 그렇지 않으면 거짓	$A > B$
$A < B$	A가 B보다 작으면 참, 그렇지 않으면 거짓	$A < B$
$A >= B$	A가 B보다 크거나 같으면 참, 그렇지 않으면 거짓	$A >= B$
$A <= B$	A가 B보다 작거나 같으면 참, 그렇지 않으면 거짓	$A <= B$

논리 연산자	설 명	예 시
$A \& B$	A 와 B가 모두참인경우 참	A 및 B 기능을 만족
$A B$	A 와 B중 하나만 참이면 참	A 또는 B 기능을 만족
$\sim A$	A가 아님	A를 제외한 항목

1. 파이썬 핵심문법

2. 조건 판단하기

```
if [조건문]:  
    [조건문 참인경우 실행]  
else:  
    [조건문 불일치 시 실행]
```

Example

```
testModel = "UN40EN001"  
preFix = "LEDTV_"  
  
# preFix가 없으면 붙여라  
if testModel.count(preFix) == 0:  
    testModel = preFix + testModel  
else:  
    pass
```

```
testModel = "UN40EN001"  
preFix = "LEDTV_"  
  
if testModel.count(preFix) == 0:  
    testModel = preFix + testModel  
else:  
    pass  
  
testModel  
  
'LEDTV_UN40EN001'
```

1. 파이썬 핵심문법

2. 조건 판단하기



```
tvList = [ LEDTV_LEDTV_UN40EN001, UN40EN002,  
            LEDTV_UN40EN003, UN40EN004]
```

헉! tv목록앞에
접두어가 2개 붙은것도 있고...
한개 붙은것도있고..
안붙은것도 있네..
다중 조건 로직이 필요한데..

다중 조건문은 어떻게 사용할 수 있을까요?

1. 파이썬 핵심문법

2. 조건 판단하기

```
if [조건문1]:  
    [조건문1 참인경우 실행]  
elif [조건문2]:  
    [조건문2 참인경우 실행]  
elif [조건문3]:  
    [조건문3 참인경우 실행]  
.....  
else:  
    [앞 조건문 모두 불일치 시 실행]
```

Example

```
testModel = "LEDTV_LEDTV_UN40EN001"  
preFix = "LEDTV_"  
  
if testModel.count(preFix) >= 2:  
    testModel = testModel.replace(preFix,"")  
    testModel = preFix + testModel  
  
elif testModel.count(preFix) == 1:  
    pass  
  
else:  
    testModel = preFix + testModel  
  
testModel
```

```
tvList2 = [ 'UN40EN001', 'LEDTV_UN40EN002',  
            'LEDTV_LEDTV_UN40EN003', 'UN40EN004']
```

리스트에서 LEDTV_가 한 개만 앞에 붙도록 하세요
(* 대소문자 구분없이 전부 처리!)

힌트:

A = "haiteam_"

B = A*3

B <- haiteam_haiteam_haiteam_


```
nationList ['A01' , '한국' , 'A02' , '미국' , 'A03' , '프랑스']  
국가코드만 출력하세요
```

SCORE 변수를 80으로 선언 후
90점 이상 A, 80~90점 B, 이외 C 점수로
GRADE변수에 할당하는 프로그램을 구현하세요

```
tvList2 = [ 'UN40EN001', 'LEDTV_UN40EN002',  
'LEDTV_LEDTV_UN40EN003', 'ledtv_UN40EN004']
```

리스트에서 LEDTV_가 한 개만 앞에 붙도록 하세요
(* 대소문자 구분없이 전부 처리!)

힌트:

A = "haiteam_"

B = A*3

B <- haiteam_haiteam_haiteam_

1. 파이썬 핵심문법

3. 반복 프로세스 함수화



화면에는 소수점 2자리까지만 표현해야 하는데
소수점 처리 할때마다 코드가 반복되네...
따로 한번에 관리할 수 있을까?

```
a = 10  
b = 5.3  
displayDigit = a/b  
(1.8867924528301887)
```

소수점 반올림 프로세스

```
target = 1.8867924528301887  
step1 = target * 100    → 188.67924528301887  
step2 = int(step1 + 0.5) → 189  
step3 = step2/100       → 1.89
```

1. 파이썬 핵심문법

3. 반복 프로세스 함수화

```
def 함수명(입력변수#1, 입력변수#2,...):  
    return 결과값
```

기능: 입력값을 받아 반올림 수행
입력: inValue: 숫자형 값
반환: outvalue: 2자리수 반올림 결과 값

함수이름

입력변수

함수 헤더

```
def roundFunction( inValue ):
```

```
    step1 = inValue * 100
```

```
    step2 = int(step1 + 0.5)
```

```
    outvalue = step2 / 100
```

```
    return outvalue
```

while문은 조건식을
만족하는 동안 구문연산

함수내부 로직

반환변수 (return 필수 기입)

1. 파이썬 핵심문법

3. 반복 프로세스 함수화

```
def 함수명(입력변수#1, 입력변수#2,...):  
    return 결과값
```

기능: 입력값을 받아 반올림 수행

```
def roundFunction( inValue ):  
    step1 = inValue * 100  
    step2 = int(step1 + 0.5)  
    outvalue = step2 / 100  
    return outvalue
```

활용편

```
a = 10  
b = 5.3  
c = 1.7  
displayDigitOne = a/b  
displayDigitTwo = a/c  
answer1 = roundFunction(displayDigitOne)  
answer2 = roundFunction(displayDigitTwo)
```

1. 파이썬 핵심문법

3. 반복 프로세스 함수화

지역변수 / 전역변수 의 이해

전역변수

```
globalValue = 0.3
```

지역변수

```
def roundFunction( inValue ):  
    globalValue = 0.5  
    localValue = 0.5  
    step1 = inValue * 100  
    step2 = int(step1 + localValue)  
    outvalue = step2 / 100  
    return outvalue
```

지역범위

전역변수 값

```
print(globalValue)  
print(localValue)
```

에러발생

1. 파이썬 핵심문법

3. 반복 프로세스 함수화

2개 이상의 입력변수 활용

```
def roundFunction( inValue, option):  
    stdVal = 0.5  
    if(option==0): # round  
        stdVal = 0.5  
    elif(option==1): #ceil  
        stdVal = 1  
    elif(option==2): #floor  
        stdVal = 0  
    else:  
        pass  
    step1 = inValue * 100  
    step2 = int(step1 + stdVal)  
    outvalue = step2 / 100  
    return outvalue
```

활용편

a = 1.725

```
for i in range(0,3):  
    print(roundFunction(a,i))
```


함수 입력변수에 따라 소수점 2자리수, 3자리 등의 반올림
이 가능하도록 함수를 구현하세요
(예: roundFunction(12.2225,3) -> 3번째자리 반올림)

< 보기 >

```
def roundFunction(inValue, option):
```

```
    <로직 구현>
```

```
    outValue = inValue
```

```
    return outValue
```

```
#활용
```

```
inputValue = 12.2225
```

```
roundFunction(inputValue, 3)
```

```
=> 12.223
```

리스트를 인수로 받아서 최대, 최소값을 뺀
평균을 구하는 함수를 생성하세요.

문자열로 구성된 리스트의
합계를 구하는 함수를 생성하세요

제공된 dataset 폴더 내
"kopo_customerdata.csv" 데이터를 작성중인 코드와 같은
폴더에 저장한 후 "보기"코드를 활용하여 불러온 후
"TOTAL_AMOUNT" 최소, 최대값을 한개씩만 뺀
평균을 소수점 2째자리 까지 표시하세요 (반올림)

< 보기 >

```
import pandas as pd

customerData = pd.read_csv("./kopo_customerdata.csv")

customerTotalList = customerData["TOTAL_AMOUNT"].tolist()
```

1. 파이썬 핵심문법

추가 제어로직

break

반복문 탈출 (성능 향상 코드로 활용 많이 됨)

continue

반복문에서 다음 단계로 넘긴다 (반복대상 스킵)

pass

특정 오류 조건 스킵 시 활용 반복문과 관련 없음

```
model = ["MODEL01","MODEL02"]
attatedModel = []
for i in range(0, len(model) ):
    if( model[i] == "MODEL01" ):
        break
    attatedModel.append(model[i])
    print( model[i] )
```

print (attatedModel)

```
[ ]
```

```
model = ["MODEL01","MODEL02"]
attatedModel = []
for i in range(0, len(model) ):
    if( model[i] == "MODEL01" ):
        continue
    attatedModel.append(model[i])
    print( model[i] )
```

print (attatedModel)

```
MODEL02
[ 'MODEL02' ]
```

```
model = ["MODEL01","MODEL02"]
attatedModel = []
for i in range(0, len(model) ):
    if( model[i] == "MODEL01" ):
        pass
    attatedModel.append(model[i])
    print( model[i] )
```

print (attatedModel)

```
MODEL01
MODEL02
[ 'MODEL01' , 'MODEL02' ]
```

4차산업혁

지능(광문각, 김효관 교수)
www.youtube.com/hkcode

1. 파이썬 핵심문법

추가 제어로직

```
tvList = ["LEDTV_UN40EN001", "LEDTV_UN40EN002", "LEDTV_UN40EN003"]
mobileList = ["G3_MO001", "G3_MO002", "G3_MO003"]
speakerList = ["BL_SP001", "BL_SP002", "BL_SP003"]

productList = [tvList, mobileList, speakerList]
```

```
# 제품군 loop
for i in range (0, len(productList)):
    productlength = len(productList[i])
    # 제품 내 모델 loop
    for j in range (0, productlength):
        if (productList[i][j].split("_")[0] == "G3") & \
            (productList[i][j].split("_")[1] == "MO002"):
            break
    print(productList[i][j])
```

상위 반복문 한개만 탈출함

1. 파이썬 핵심문법

```
tvList = ["LEDTV_UN40EN001", "LEDTV_UN40EN002", "LEDTV_UN40EN003"]
mobileList = ["G3_MO001", "G3_MO002", "G3_MO003"]
speakerList = ["BL_SP001", "BL_SP002", "BL_SP003"]

productList = [tvList, mobileList, speakerList]
```

```
# 제품군 Loop
for i in range(0, len(productList)):
    productlength = productList[i]
    # 제품 내 모델 Loop
    for j in range(0, len(productlength)):
        if (productList[i][j].split("_")[0] == "G3") &\
            (productList[i][j].split("_")[1] == "MO002"):
            break
        print(productList[i][j])
```

```
LEDTV_UN40EN001
LEDTV_UN40EN002
LEDTV_UN40EN003
G3_MO001
BL_SP001
BL_SP002
BL_SP003
```

```
# 제품군 Loop
for i in range(0, len(productList)):
    productlength = productList[i]
    # 제품 내 모델 Loop
    for j in range(0, len(productlength)):
        if (productList[i][j].split("_")[0] == "G3") &\
            (productList[i][j].split("_")[1] == "MO002"):
            continue
        print(productList[i][j])
```

```
LEDTV_UN40EN001
LEDTV_UN40EN002
LEDTV_UN40EN003
G3_MO001
G3_MO003
BL_SP001
BL_SP002
BL_SP003
```

1. 파이썬 핵심문법

```
tvList = ["LEDTV_UN40EN001", "LEDTV_UN40EN002", "LEDTV_UN40EN003"]
mobileList = ["G3_MO001", "G3_MO002", "G3_MO003"]
speakerList = ["BL_SP001", "BL_SP002", "BL_SP003"]

productList = [tvList, mobileList, speakerList]
```

```
# 제품군 Loop
for i in range (0, len(productList)):
    productlength = productList[i]
    # 제품 내 모델 Loop
    for j in range (0, len(productlength)):
        if (productList[i][j].split("_")[0] == "G3") & \
            (productList[i][j].split("_")[1] == "MO002"):
            pass
        print(productList[i][j])
```

```
LEDTV_UN40EN001
LEDTV_UN40EN002
LEDTV_UN40EN003
G3_MO001
G3_MO002
G3_MO003
BL_SP001
BL_SP002
BL_SP003
```


1. 파이썬 핵심문법

예외처리



여러개의 csv파일을 읽어보자!

```
import pandas as pd
dataSum = []
for i in range(0, 3):
    dataSum.append(
        pd.read_csv("../dataset/kopo_product_volume"+str(i)+".csv")
```



kopo_product_volume2.csv



kopo_product_volume0.csv


중간에 빠진 데이터가 있으면 바로 에러가 나겠네..
에러가 있는경우 처리하는 방법은?


1. 파이썬 핵심문법

예외처리

```
dataSum = []  
for i in range(0, 3):  
    tf = pd.read_csv\  
        ("../dataset/kopo_product_volume"+str(i)+".csv")  
    dataSum.append(tf)
```

예외 발생 가능 코드

 kopo_product_volume2.csv

 kopo_product_volume0.csv

```
dataSum = []  
for i in range(0, 3):  
    tf = pd.read_csv\  
        ("../dataset/kopo_product_volume"+str(i)+".csv")  
    dataSum.append(tf)
```

-
FileNotFoundError

Traceback

1. 파이썬 핵심문법

예외처리

```
try:  
    [예외가 발생 가능한 로직]  
except 예외종류 as e:  
    [ 예외 발생시 로직 ]
```

```
dataSum = []  
for i in range(0, 3):
```

```
    try:
```

예외 발생
가능한 로직

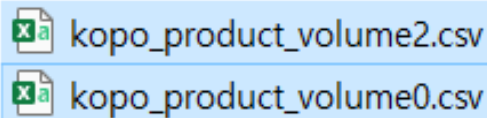
```
        { tf = pd.read_csvW  
          ("../dataset/kopo_product_volume"+str(i)+".csv")  
          dataSum.append(tf)
```

```
    except Exception as e:
```

예외
발생 시
로직

```
        print(e)
```

```
print(len(dataSum))
```



kopo_product_volume2.csv
kopo_product_volume0.csv

1. 파이썬 핵심문법

예외처리

```
dataSum = []
for i in range(0, 3):
    try:
        tf = pd.read_csv\
            ("../dataset/kopo_product_volume"+str(i)+".csv")
        dataSum.append(tf)
    except Exception as e:
        print(e)
print(len(dataSum))
```

```
[Errno 2] File b'../dataset/kopo_product_volume1.csv' does not exist:
b'../dataset/kopo_product_volume1.csv'
2
```



감사합니다