빅데이터 분석

1. 파이썬의 특징

- **플랫폼 독립적인 언어** : 어떤 운영체제든 상관없이 사용할 수 있는 언어를 말한다
- **인터프리터 언어** : 컴파일러 언어와 달리, 소스코드 자체가 바로 실행되는 특징이 있는 언어이다

  이로 인해 속도는 느리지만, 굉장히 간편하게 사용할 수 있다
- **객체 지향 언어** : 해당 프로그램이 해결해야 할 문제의 구성요소를 요소별로 정의한 뒤, 각 요소의 기능(매서드)과 정보(속성)를 정의하여 요소들을 결합하고, 프로그램을 작성하는 방식이다
- **동적 타이핑 언어** : 프로그램의 실행 시점에서 각 프로그램 변수의 타입을 결정하는 언어이다

2. 변수 (variable)

- **2-1 변수란 ?**
  - 프로그래밍에서 변수는 어떠한 값을 저장하는 장소라는 뜻으로 사용된다
  - 변수에 값이 저장되는 공간을 메모리라고 한다
  - 변수에 값을 넣으라고 선언하는 순간, 물리적으로 메모리 어딘가에 물리적인 공간을 확보할 수 있게 운영체제와 파이썬 인터프리터가 협력하여 메모리 저장 위치를 할당한다

    이 위치를 메모리 주소라고 한다
- **2-2 변수명 선언**
  - 언더스코어(_)은 변수 이름에 들어갈 수 있지만, 나머지 특수 문자는 불가능하다
  - 변수 이름에 숫자를 사용할 수 있지만, 첫 글자로 숫자는 사용할 수 없다
  - 공백은 변수 이름에 들어갈 수 없다
  - 특별한 의미가 있는 예약어는 사용할 수 없다

- **2-3 기본 자료형**
  - **정수형 (integer type)** : 자연수를 포함해 값의 영역이 정수로 한정된 값
  - **실수형 (floating-point type)** : 값이 문자로 출력되는 자료형
  - **문자형 (string type)** : 값이 문자로 출력되는 자료형
  - **불린형 (boolean type)** : 논리형으로, 참 또는 거짓을 표현할 때 사용

자료형	설명	예	선언 형태
정수형	양수의 정수	1, 2, 3, 100, -9	data = 1
실수형	소수점이 포함된 실수	10.2, -9.3, 9.0	data = 9.0
문자형	따옴표에 들어가 있는 문자형	abc, a20abc	data = ‘abc’
불린형	참 또는 거짓	True, False	data = True

- **2-4 산술 연산자**

연산자	기능	문법
+	덧셈	a + b

-	뺄셈	a - b
*	곱셈	a * b
/	나눗셈	a / b
//	버림 나눗셈	a // b
%	나머지	a % b
**	거듭제곱	a ** b
@	행렬 곱셈	a @ b
+	양수 부호	+a
-	음수 부호	-a
+=	덧셈 후 할당	a += b
-=	뺄셈 후 할당	a -= b
*=	곱셈 후 할당	a *= b
/=	나눗셈 후 할당	a /= b
//=	버림 나눗셈 후 할당	a //= b
%=	나머지 연산 후 할당	a %= b
**=	거듭제곱 후 할당	a **= b
@=	행렬 곱셈 후 할당	a @= b

- **2-5 반환**

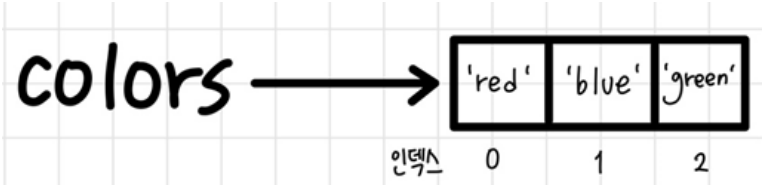
int()	실수 → 정수
float()	정수 → 실수
str()	정수, 실수 → 문자열
type()	자료형 확인

3. 화면 입출력과 리스트

- **3-1 리스트의 개념**
  - 하나의 변수에 여러 값을 할당하는 자료형
  - 대괄호[]에 자료를 쉼표로 구분하여 입력
  - 대괄호[] 내부에 넣는 자료를 요소(element)라고 부름
  - 리스트는 하나의 자료형으로만 저장하지 않고, 다양한 자료형을 포함할 수 있음

```
colors = ['red', 'blue', 'green']
```

- **3-2 인덱싱과 슬라이싱**
  - 인덱싱 : 문자열 내부의 문자 하나를 선택하는 연산자



- 슬라이싱 : 문자열의 특정 범위를 선택할 때 사용하는 연산자
  (시작 인덱스 이상 마지막 인덱스 미만)

변수명[시작 인덱스:마지막 인덱스]

입력	출력
“KE901”[0:1]	K
“KE901”[0:2]	KE
“KE901”[2:5]	901
“KE901”[:]	KE901
“KE901”[:3]	KE9

- **3-3 리스트 추가 & 삭제**

함수	기능	응례
append()	새로운 값을 기존 리스트의 맨 끝에 추가	color.append(‘white’)
extend()	새로운 리스트를 기존 리스트에 추가 (덧셈 연산과 같은 효과)	color.extend([’black’, ‘purple’])
insert()	기존 리스트의 i번째 인덱스에 새로운 값을 추가, i번째 인덱스를 기준으로 뒤쪽의 인덱스는 하나씩 밀림	color.insert(0. ‘orange’)
remove()	리스트 내의 특정 값을 삭제	color.remove(‘white’)
del	특정 인덱스 값을 삭제	del color[0]

- 3-4 이차원 리스트
  - 이차원 리스트에 인덱싱하여 값에 접근하기 위해서는 대괄호 2개를 사용한다

```
kor_score = [49, 79, 20, 100, 80]
math_score = [43, 59, 85, 30, 90]
eng_score = [49, 79, 48, 60, 100]
midterm_score = [kor_score, math_score, eng_score]
print(midterm_score)

--------------------------------------------------------------------
[[49, 79, 20, 100, 80], [43, 59, 85, 30, 90], [49, 79, 48, 60, 100]]
```

  - 이차원 리스트를 행렬로 본다는 [0]은 행, [2]는 열을 뜻한다
    즉, [0]은 kor_score, [2]는 인덱스를 뜻하며 실행결과 20을 화면에 출력한다

```
print(midterm_score[0][2])
----------------------------
20
```

4. 조건문과 반복문

- 비료 연산자 (또는 조건 연산자)

비교 연산자	비교 상태	설명
x < y	~보다 작은	x가 y보다 작은지 검사
x > y	~보다 큼	x가 y보다 큰지 검사
x == y	같음	x와 y의 값이 같은지 검사
x is y	같음 (메모리 주소)	x와 y의 메모리 주소가 같은지 검사
x != y	같지 않음	x와 y의 값이 같지 않은지 검사
x is not y	같지 않음 (메모리 주소)	x와 y의 메모리 주소가 같지 않은지 검사
x >= y	크거나 같음	x가 y보다 크거나 같은지 검사
x <= y	작거나 같음	x가 y보다 작거나 같은지 검사

5. 자료구조

- 파이썬에서의 자료구조

자료구조명	특징
스택 (stack)	나중에 들어온 값을 먼저 나갈 수 있도록 해주는 자료구조 (last in first out)
큐 (queue)	먼저 들어온 값을 먼저 나갈 수 있도록 해주는 자료구조 (first in first out)
튜플 (tuple)	리스트와 같지만, 데이터의 변경을 허용하지 않는 자료구조
세트 (set)	데이터의 중복을 허용하지 않고, 수학의 집합 연산을 지원하는 자료구조
딕셔너리 (dictionary)	전화번호부와 같이 키(key)와 값(value) 형태의 데이터를 저장하는 자료구조, 여기서 키값은 다른 데이터와 중복을 허용하지 않음
collections 모듈	위에 열거된 여러 자료구조를 효율적으로 사용할 수 있도록 지원하는 파이썬 내장(built-in)모듈

```
a = 777  # 변수 a에 정수 777을 할당한다
b = 777  # 변수 b에 정수 777을 할당한다

# a와 b의 값이 같은지 비교하는 코드임
# 값은 같지만 a와 b는 서로 다른 객체이므로
# a is b는 False가 된다
print(a == b, a is b)
------------------------
True False
```

```
a = 3.5  # 변수 a에 3.5 할당
b = int(3.5)  # 변수 b에 3.5를 정수형으로 변환한 값인 3 할당
print(a*((a / b) * 2))  # a를 (a/b)*2로 계산한 값을 곱하여 출력
print(((a-b) * a) // b)  # (a-b) * a를 b로 나눈 몫을 출력
b = (((a-b) * a) % b)
------------------------
8.166... 0.0
```

```
a = 3.5  # 변수 a에 3.5 할당
b = int(3.5)  # 변수 b에 3.5를 정수형으로 변환한 값인 3 할당
print(a * ((a / b) * 2))  # a를 (a/b)*2로 계산한 값을 곱하여 출력
print(((a - b) * a) // b)  # (a-b) * a를 b로 나눈 몫을 출력
b = (((a - b) * a) % b)  # (a-b) * a를 b로 나눈 나머지를 b에 할당
------------------------
8.166666666666668
0.0
```

```
a = True
------------------------
```

```
bool
```

```
a = 10.6
b = 10.5
print(a * b)
type(a + b)
-------------------------
111.3 float
```

```
a = "3.5"
b = 4
print(a * b)
-------------------------
3.53.53.53.5
```

```
a = '3'
b = float(a)
print(b ** int(a))
-------------------------
27.0
```

```
a = [0, 1, 2, 3, 4]  # a라는 리스트를 생성하고 요소를 초기화함
print(a[:3], a[:-3])  # a의 처음부터 3번째 요소까지와 마지막에서 3번째 요소까지 출력
-------------------------
[0, 1, 2] [0, 1]
```

```
a = [0, 1, 2, 3, 4]  # a라는 리스트를 생성하고 요소를 초기화함
print(a[::-1])  # a 리스트의 역순으로 출력
-------------------------
[4, 3, 2, 1, 0]
```

```
# 리스트 first, second, third를 생성하고 요소를 초기화함
first = ["egg", "salad", "bread", "soup", "canafe"]
second = ["fish", "lamb", "pork", "beef", "chicken"]
third = ["apple", "banana", "orange", "grape", "mango"]

# 리스트 order를 생성하고 first, second, third 리스트를 요소로 추가함
order = [first, second, third]

# 리스트 john을 생성하고 다음과 같은 값을 할당함
# - order 리스트의 첫 번째 요소 중 마지막 두 개의 값을 제외한 리스트
# - second 리스트의 인덱스 1부터 3씩 증가하며 추출한 리스트
# - third 리스트의 첫 번째 요소
john = [order[0][:-2], second[1::3], third[0]]

# john 리스트에서 인덱스 2에 해당하는 값을 삭제함
del john[2]

# john 리스트에 order 리스트의 세 번째 요소 중 첫 번째 요소를 추가함
john.extend([order[2][0:1]])

# john 리스트를 출력함
print(john)
-------------------------------------------------------------
[['egg', 'salad', 'bread'], ['lamb', 'chicken'], ['apple']]
```

```
# 리스트 list_a를 생성하고 요소를 초기화함
list_a = [3, 2, 1, 4]

# 리스트 list_a를 정렬한 결과를 리스트 list_b에 할당함
# - sort()는 리스트를 정렬하는 메서드로, 리스트 자체를 변경하며 반환값이 없음
```

```
list_b = list_a.sort()

# list_a는 정렬된 결과가 저장된 상태이며, list_b는 None 값을 가짐
print(list_a, list_b)
------------------------
[1, 2, 3, 4] None
```

```
# 리스트 a와 b를 생성하고 요소를 초기화함
a = [5, 7, 3]
b = [3, 9, 1]

# 리스트 a와 b를 연결하여 리스트 c를 만듦
c = a + b

# 리스트 c를 정렬한 결과를 리스트 c에 저장함
# - sort()는 리스트를 정렬하는 메서드로, 리스트 자체를 변경하며 반환값이 없음
# - 따라서 c.sort()의 반환값은 None
c = c.sort()

# c는 이제 None 값을 가지게 됨
print(c)----------------
None
```

```
fruits = ['apple', 'banana', 'cherry', 'grape', 'orange', 'strawberry', 'melon']  # fruits 리스트를 생성하고 요소를 초기화함
print(fruits[-3:], fruits[1::3])  # fruits 리스트의 뒤에서부터 3번째 요소부터 끝까지와, 1번째부터 3칸씩 건너뛴 요소들을 출력함
-------------------------------------------------------------------------------
['orange', 'strawberry', 'melon'] ['banana', 'orange']
```

```
num = [1, 2, 3, 4]
print(num * 2)
-------------------
[1, 2, 3, 4, 1, 2, 3, 4]
```

```
country = ["Korea", "Japan", "China"]  # country 리스트를 생성하고 요소를 초기화함
capital = ["Seoul", "Tokyo", "Beijing"]  # capital 리스트를 생성하고 요소를 초기화함
index = [1, 2, 3]  # index 리스트를 생성하고 요소를 초기화함
country[3][1] = index[1:]  # IndexError가 발생함. country 리스트의 4번째 요소의 2번째 인덱스에 index 리스트의 2번째 요소부터 끝까지 할당함
print(country)  # 결과를 출력함
-------------------------------------
error
```

```
a = [7, 4, 2, 1]  # a 리스트 생성하고 요소를 초기화함
b = a  # a 리스트를 참조하는 b 리스트 생성
c = [7, 4, 2, 1]  # 새로운 리스트 c를 생성하고 요소를 초기화함
a is b  # a와 b가 같은 객체인지 확인(True 출력됨)
a is c  # a와 c가 같은 객체인지 확인(False 출력됨)
-------------------
True False
```

```
fruit = 'apple'
if fruit == 'Apple':
  fruit = 'Apple'
elif fruit == 'fruit':
  fruit = 'fruit'
else:
  fruit = fruit
print(fruit)
----------------------
apple
```

```
num = ['12', '34', '56']  # 문자열 요소를 가지는 리스트 num 생성
for i in num:  # num 리스트의 모든 요소에 대해서 반복
    i = int(i)  # i를 int 타입으로 변환하고 새로운 값을 할당
print(num)  # 변화 없는 원래 num 리스트 출력
-------------------------
['12', '34', '56']
```

```
num = 0  # 변수 num을 0으로 초기화함
i = 1  # 변수 i를 1로 초기화함

while i < 8:  # i가 8보다 작을 때까지 반복문을 실행함
  if i % 3 == 0:  # i를 3으로 나눈 나머지가 0이면
    break  # 반복문을 종료함
  i += 1  # i에 1을 더해줌
  num += i  # num에 i를 더해줌

print(num)  # num을 출력함
-------------------
5
```

```
result = 0  # 결과값 변수 초기화
for i in range(5, -5, -2):  # 5부터 -5까지 -2씩 감소하면서 반복문 실행
  if i < -3:  # i가 -3보다 작으면
    result += 1  # result에 1을 더함
  else:  # i가 -3보다 크거나 같으면
    result -= 1  # result에 1을 뺌
print(result)  # 결과값 출력 (-3)
-------------------------
-5
```

```
num = ""  # 빈 문자열을 변수 num에 할당
for i in range(10):  # 0부터 9까지 반복문 실행
  if i <= 5 and (i % 2) == 0:  # i가 5 이하이고 2로 나눈 나머지가 0인 경우 건너뛰기
    continue
  elif i is 7 or i is 10:  # i가 7 또는 10인 경우 건너뛰기
    continue
  else:  # 위 조건들을 모두 만족하지 않으면
    num = str(i) + num  # 변수 num에 문자열 i와 num을 연결한 값을 할당
print(num)  # num 출력
------------------------------
986531
<>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
<ipython-input-3-7a981573be12>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
  elif i is 7 or i is 10:
<ipython-input-3-7a981573be12>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
  elif i is 7 or i is 10:
```

```
score_list = [5, 10, 15, 20, 25, 30]  # 점수 리스트 생성하고 초기화
sum_of_score = 0  # 점수의 합을 저장할 변수를 0으로 초기화
i = 0  # while 문을 위한 반복 변수 i를 0으로 초기화
while i < len(score_list):  # 리스트의 길이만큼 반복
  if i % 2 == 0:  # 인덱스가 홀수일 때
    sum_of_score += score_list[i]  # 현재 점수를 sum_of_score에 더함
  i += 1  # i에 1을 더해 다음 인덱스로 이동
print(sum_of_score)  # 홀수 번째 점수들의 합을 출력
----------------------------------------
45
```

```
list_1 = [[1, 2], [3], [4, 5, 6]]  # 2차원 리스트를 생성하고 초기화함
a, b, c = list_1  # 각 리스트 요소를 a, b, c 변수에 대입함
list_2 = a + b + c  # 리스트 a, b, c를 합쳐서 list_2에 대입함
```

```
print(list_2)  # list_2 출력
-----------------------------------
[1, 2, 3, 4, 5, 6]
```

```
def rain(colors):  # 함수 rain을 정의하고 colors라는 매개변수를 정의함
  colors.append("purple")  # 매개변수 colors에 "purple" 요소를 추가함
  colors = ["green", "blue"]  # colors 리스트를 ["green", "blue"]로 새로 정의함
  return colors  # colors 리스트를 반환함

rainbow = ["red", "orange"]  # rainbow 리스트를 생성하고 초기화함
print(rain(rainbow))  # rainbow 리스트를 함수 rain에 인수로 전달하고 반환된 결과를 출력함
----------------------------
['green', 'blue']
```

```
def function(value):
  print(value ** 3)  # 주어진 값(value)을 세제곱하여 출력

print(function(2))  # 2의 세제곱인 8을 출력하고, 반환값이 없으므로 None을 출력함
---------------------
8
None
```

```
a = 11
b = 9

print('a' + 'b')
--------------------
ab
```

```
fact = "Python is funny"
print(str(fact.count('n') + fact.find('n') + fact.rfind('n')))
----------------------------------------------------------------------
21
```

```
school_name = 'busan software meister high school'

for i in school_name:
  if i == 'busan':
    i = i.upper()
# 주어진 문자열 'busan software meister high school'에서 for 루프를 이용하여 각 문자를 하나씩 순회한다.

print(school_name)
# 문자 'busan'을 대문자로 변환하기 위해서는 조건문으로 해당 문자를 체크해야한다.
# 하지만 이 경우에는 문자열 전체를 찾아서 변경하는 것이 불가능하다.
# 따라서 조건문에서 해당 문자를 찾더라도 대문자로 변경되지 않는다.
# 따라서 원래의 문자열인 'busan software meister high school'이 그대로 출력된다.
-----------------------------------------------------------
busan software meister high school
```

```
a = '10'  # 문자열 '10'을 변수 a에 할당
b = '5-2'.split('-')[1]  # 문자열 '5-2'를 '-' 기준으로 분리하고, 분리된 리스트에서 인덱스 1을 선택하여 변수 b에 할당

print(a * 3 + b)  # a를 세 번 반복한 문자열과 b를 연결하여 출력
---------------------------
1010102
```

```
# 함수 정의
def add_number(add_list):
  add_list += [1,5,8]

# 리스트 초기화
```

```
list = [0, 1, 3, 5, 6, 7]

# 함수 호출
add_number(list)

# 리스트 출력
print(list)

# 중복을 제거한 set 출력
print(set(list))
-------------------------------
[0, 1, 3, 5, 6, 7, 1, 5, 8]
{0, 1, 3, 5, 6, 7, 8}
```