



데이터 분석 with 파이썬

Chapter 02. 데이터 분석을 위한 파이썬 기초

목차

1. 변수와 상수
2. 연산자
3. 조건문과 반복문
4. 자료형
5. 함수

01

변수와 상수

01. 변수와 상수

I. 변수

프로그래밍에서 변수(Variable)는 데이터(Data, 값)를 저장하는 그릇.

[코드 2-1] 변수에 값 담기

```
#변수에 '철수'를 담기  
name = '철수'  
print(name)  
  
#변수에 '영희'를 담기  
name = '영희'  
print(name)
```

[코드 2-1] 실행결과

```
철수  
영희
```



그림 2-1 변수와 데이터

01. 변수와 상수

I. 변수

- 변수 사용 규칙
 - 새로운 변수를 정의할 때 반드시 변수에 값을 대입해야 함.

[코드 2-2] 값을 할당하지 않고 변수 정의

```
score
```

[코드 2-2] 실행결과

```
-----  
NameError Traceback (most recent call last)  
<ipython-input-2-d2d780e36333> in <module>  
----> 1 score  
NameError: name 'score' is not defined
```

01. 변수와 상수

I. 변수

- 변수 사용 규칙
 - 변수의 이름에서 첫 글자는 문자 또는 밑줄 문자(_)여야 함.

[코드 2-3] 숫자로 시작하는 변수명

```
2_name = '철수, 영희'
```

[코드 2-2] 실행결과

```
File "<ipython-input-3-42583fbcf0dc>", line 1
    2_name = '철수, 영희'
    ^
SyntaxError: invalid syntax
```

- 변수명에 길이 제한은 없지만 알아보기 쉬운 이름으로 정의해야 편리함.

01. 변수와 상수

I. 변수

- 컴퓨터의 동작 원리와 변수
 - 컴퓨터가 프로그램을 수행하려면 사용자가 지시한 작업에 대한 정보를 기록하여 저장한 후 필요할 때 불러내어 활용.

1번지	2번지	3번지				
		Hello				
				10		
				N번지	N+1번지	N+2번지

그림 2-2 주기억장치의 데이터 저장

- 일반적으로 사용자는 "3번지의 단어를 N번지에 저장된 수만큼 반복하여 말하십시오."와 같이 프로그램을 작성
- [그림 2-2]의 3번지에 '인사', N번지에 '반복 횟수'라고 이름 붙여 사용

'인사'의 값을 '반복 횟수'만큼 반복하여 말하십시오

01. 변수와 상수

I. 파이썬의 상수

- 상수(Constant) : 변하지 않고 항상 일정한 값인 수
- 파이썬 에서 상수처럼 사용하는 변수에 다른 값을 담는 일이 없도록 주의해야 함.

[코드 2-4] 상수처럼 사용하는 변수

```
PI = 3.1415
```

```
GRAVITY = 9.8
```

- 상수처럼 사용할 변수를 정의할 때 이 변수에는 본래 값 이외의 값을 할당하지 말라는 뜻으로 변수 이름에 대문자와 밑줄 문자(_)만 사용

02

연산자

02. 연산자

I. 산술 연산자

- 산술 연산자는 숫자인 값을 연산.

표 2-1 산술 연산자의 종류

연산자	연산
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
//	정수 몫
%	나머지
**	거듭제곱

[코드 2-5] 산술 연산자

```
a = 11
b = 4

print('덧셈 결과', a + b)
print('뺄셈 결과', a - b)
print('곱셈 결과', a * b)
print('나눗셈 결과', a / b)
print('나눗셈 정수 몫', a // b)
print('나눗셈 나머지', a % b)
print('거듭제곱 결과', a ** b)
```

[코드 2-5] 실행결과

```
덧셈 결과 15
뺄셈 결과 7
곱셈 결과 44
나눗셈 결과 2.75
나눗셈 정수 몫 2
나눗셈 나머지 3
거듭제곱 결과 14641
```

02. 연산자

II. 관계 연산자

- 관계 연산자는 두 값의 대소관계 또는 상등관계를 계산하는 연산자.
- 연산 결과는 참(True)과 거짓(False)의 논리값(Boolean value)으로 나타남.

표 2-2 관계 연산자의 종류

연산자	연산
>	왼쪽 값이 오른쪽 값보다 큰지 비교
>=	왼쪽 값이 오른쪽 값보다 크거나 같은지 비교
<	왼쪽 값이 오른쪽 값보다 작은지 비교
<=	왼쪽 값이 오른쪽 값보다 작거나 같은지 비교
==	왼쪽 값이 오른쪽 값과 같은지 비교
!=	왼쪽 값이 오른쪽 값과 같지 않은지 비교

[코드 2-6] 관계 연산자

```
a = 11
b = 4

print(a > b)
print(a >= b)
print(a < b)
print(a <= b)
```

[코드 2-6] 실행결과

```
True
True
False
False
```

02. 연산자

III. 논리 연산자

- 논리 연산자는 논리값 참(True)과 거짓(False)을 연산

표 2-3 논리 연산자의 종류

연산자	연산
and	논리곱(conjunction)
or	논리합(disjunction)
not	논리 부정

- 진리표(Truth Table)는 논리 연산의 결과를 정리한 표.

표 2-4 논리 연산의 결과 진리표

A	B	A and B	A or B	not a
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

02. 연산자

III. 논리 연산자

[코드 2-7] 논리 연산자

```
a = True  
b = False  
print(a and b)  
print(a or b)
```

[코드 2-7] 실행결과

```
False  
True
```

02. 연산자

IV. 멤버 연산자

- 멤버 연산자는 집합 연산에 사용하는 연산자로, 어떤 집합(객체)에 연산 대상의 포함 여부를 판단.

표 2-5 멤버 연산자의 종류

연산자	연산
in	왼쪽 값이 오른쪽 집합(객체) 안에 포함되었는지 판단
not in	왼쪽 값이 오른쪽 집합(객체) 안에 포함되지 않았는지 판단

[코드 2-8] 논리 연산자

```
a = 3
B = [1, 3, 5, 7, 9]

print(a in B)
print(a not in B)
```

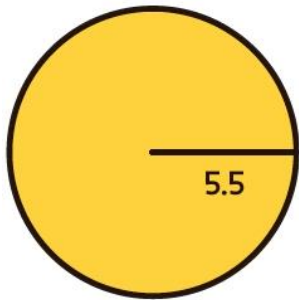
[코드 2-8] 실행결과

```
True
False
```

LAB. 파이썬의 연산자 사용

변수와 상수, 파이썬의 다양한 연산자를 이용하여 원의 면적 구하기, 두 삼각형의 크기 비교, 성적 구하기와 같은 산수 문제를 나타내 봅시다.

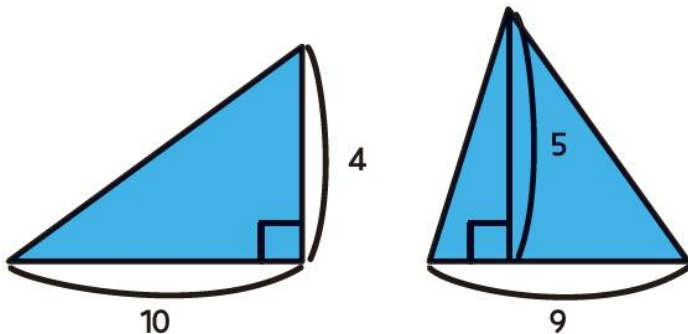
1. 반지름이 5.5인 원의 면적을 계산. 원주율은 3.14를 변수에 저장.



```
PI = 3.14  
5.5 * 5.5 * PI
```

94.985

2. 그림의 두 삼각형 A와 B의 면적이 같은지 판단하여 참 또는 거짓으로 출력



```
A = 10 * 4 * (1/2)  
B = 9 * 5 * (1/2)  
print(A == B)
```

False

LAB. 파이썬의 연산자 사용

변수와 상수, 파이썬의 다양한 연산자를 이용하여 원의 면적 구하기, 두 삼각형의 크기 비교, 성적 구하기와 같은 산수 문제를 나타내 봅시다.

3. 철수는 중간고사에서 국어 95점, 수학 88점, 영어 90점을 받았고, 과학 점수는 잊어버렸지만 네 과목 평균 점수는 92점이었습니다. 철수의 과학 점수는 몇 점일까요?

$92 * 4 - 95 - 88 - 90$

95

03

조건문과 반복문

03. 조건문과 반복문

I. 조건문

- 컴퓨터는 스스로 결정하는 능력이 없으므로 선택이 필요한 경우, 사용자가 조건문에서 조건을 정의.

표 2-6 조건문의 종류

조건문	동작
if	조건이 참이면 실행
if ~ else	조건이 참이면 if 이후만 실행하고 거짓이면 else 이후만 실행
if ~ elif ~ else	여러 조건을 if 와 elif에 각각 할당하여 조건이 참인 부분만 실행하고 만약 모두 거짓이면 else 이후만 실행

- If 조건문
 - if 조건문은 조건식이 참이면 명령을 실행

03. 조건문과 반복문

I. 조건문

- if 조건문은 조건식이 참이면 명령을 실행.

[코드 2-9] if 조건문

```
a = 5

if a == 5:
    print('Right!')
    print('a is 5')
if a == 3:
    print('Right!')
    print('a is 3')
if a != 3:
    print('Right!')
    print('a is not 3')
```

[코드 2-9] 실행결과

```
Right!
a is 5
Right!
a is not 3
```

03. 조건문과 반복문

I. 조건문

- 다중 조건문
 - 다중 조건문인 if ~ else 조건문은 하나의 조건식을 판별하여 두 블록 중 하나 실행.

[코드 2-10] else if 조건문

```
a = 5

if a == 5:
    print('Right!')
    print('a is 5')
else :
    print('a is not 5')

a = 3
if a == 5:
    print('Right!')
    print('a is 5')
else
    print('a is not 5')
```

[코드 2-10] 실행결과

```
Right!
a is 5
a is not 5
```

03. 조건문과 반복문

I. 조건문

- 다중 조건문
 - 다중 조건문인 if ~ elif ~ else 조건문은 여러 조건을 정의하고 어떤 조건이 참인지에 따라 다른 동작을 실행할 때 사용.
 - 서로 다른 조건을 if절과 elif절에 제시하여 차례로 판별

[코드 2-11] if ~ elif ~ else 조건문

```
a = 5
if a < 5:
    print('a is smaller than 5')
elif a > 5:
    print('a is larger than 5')
else:
    print('a is 5')
```

[코드 2-10] 실행결과

```
a is 5
```

03. 조건문과 반복문

II. 반복문

표 2-7 반복문의 종류

반복문	동작
while	조건이 참인 동안 반복 실행
for	집합에서 하나씩 세서 하나도 남지 않을 때까지 반복 실행

- while 반복문
 - while 반복문은 조건이 참인 동안 실행문을 반복 실행

[코드 2-12] while 반복문으로 구구단 5단 출력 [코드 2-12] 실행결과

```
a = 5
i = 1

#9번 반복하기
while i <= 9:
    print(str(a) + ' X ' + str(i) + ' = ' +
          str(i*a))
    i += 1
print('파이썬으로 구구단 5단을 계산할 수 있다!')
```

```
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
파이썬으로 구구단 5단을 계산할 수 있다!
```

03. 조건문과 반복문

II. 반복문

- for 반복문
 - 마치 주머니에서 동전을 하나씩 꺼내는 동작을 동전이 하나도 남지 않을 때까지 반복하는 것처럼 조건을 설정



그림 2-3 동전을 하나씩 반복하여 꺼내기

03. 조건문과 반복문

II. 반복문

- for 반복문

[코드 2-13] for 반복문으로 구구단 5단 출력

```
a = 5
for i in range(1,10):
    print(str(a) + ' X ' + str(i) + ' = ' + str(i*a))
print('while 조건문을 for 조건문으로 바꾸어 사용할 수 있다!')
```

[코드 2-13] 실행결과

```
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
while 조건문을 for 조건문으로 바꾸어 사용할 수 있다!
```


LAB. 음료 자판기

조건문과 반복문을 활용하여 음료 자판기 프로그램을 만들기

- 자판기는 반복하여 동작.
- 오렌지주스, 커피, 콜라를 각각 100원, 200원, 300원에 판매.
- 구매자에게 동전 액수와 주문번호(1.오렌지주스, 2.커피, 3.콜라)를 입력 받음.
- 입력 받은 액수보다 메뉴가 비싸면 "잔액이 부족합니다."라고 출력함.
- 메뉴가 잘못 입력되면 "없는 메뉴입니다. 다시 입력해 주세요."라고 출력하고 처음부터 다시 입력 받음.
- 자판기는 주문과 동시에 잔액을 알려주고 반환.



1. 반복하여 동작하는 자판기를 while 반복문으로 정의.

```
while True:
```

2. 사용자에게 메뉴를 알려주고 동전 액수와 메뉴를 입력 받음.

```
print('음료목록 1.오렌지주스(100원), 2.커피(200원), 3.콜라(300원)')
coin = int(input('동전을 넣으세요.'))
drink = int(input('음료를 고르세요.\n'))
```

LAB. 음료 자판기

조건문과 반복문을 활용하여 음료 자판기 프로그램을 만들기

3. 반복하여 동작하는 자판기를 while 반복문으로 정의.

```
if drink == 1:
    #오렌지주스 100원
    if coin >= 100:
        remain = coin - 100
        print('오렌지주스가 곧 제공됩니다.')
        print('거스름돈은 { }원입니다.'.format(remain))
    else:
        print('잔액이 부족합니다.')
```

LAB. 음료 자판기

조건문과 반복문을 활용하여 음료 자판기 프로그램을 만들기

4. elif문으로 커피를 선택했을 때와 콜라를 선택했을 때의 동작을 정의.

```
elif drink == 2:
    #커피 200원
    if coin >= 200:
        remain = coin - 200
        print('커피가 곧 제공됩니다.')
        print('거스름돈은 { }원입니다.'.format(remain))
    else:
        print('잔액이 부족합니다.')

elif drink == 3:
    #콜라 300원
    if coin >= 300:
        remain = coin - 300
        print('콜라가 곧 제공됩니다.')
        print('거스름돈은 { }원입니다.'.format(remain))
    else:
        print('잔액이 부족합니다.')
```

LAB. 음료 자판기

조건문과 반복문을 활용하여 음료 자판기 프로그램을 만들기

5. 잘못된 번호를 입력했을 때의 동작을 정의.

```
else:  
    #없는 번호  
    print('없는 메뉴입니다. 다시 입력해 주세요.')
```

6. 주문이 끝나면 거스름돈을 반환하고 자판기 잔액을 초기화.

```
coin = 0
```

04

자료형

04. 자료형

I. 숫자와 문자열

- 숫자(Numeric) 자료형은 말 그대로 숫자로 이루어진 자료형.
- 정수형과 실수형
 - 정수형은 수학에서의 정수와 동일하게 소수점 아래 값이 없는 숫자.
 - 실수형은 소수점이 포함된 숫자까지 표현 가능.

[코드 2-14] 숫자형 데이터

```
integer_1 = 3214
integer_2 = -128109
float_1 = -1.986214
float_2 = 123.e2

print(integer_1, type(integer_1))
print(integer_2, type(integer_2))
print(float_1, type(float_1))
print(float_2, type(float_2))

print(integer_1 / integer_2, type(integer_1 / integer_2))
```

[코드 2-14] 실행결과

```
3214 <class 'int'>
-128109 <class 'int'>
-1.986214 <class 'float'>
12300.0 <class 'float'>
-0.0250888010990640782 <class 'float'>
```

04. 자료형

I. 숫자와 문자열

- 문자열
 - 사전에 정의된 명령어나 따옴표와 같은 특수문자를 문자열로 사용하기 위해 확장 문자(Escape sequence)라고 부르는 역슬래시 기호(\)를 사용.

[코드 2-16] 확장 문자

```
string_5 = "This is String! \"따옴표\" 기호: !@#$%^"  
print(string_5)
```

[코드 2-15] 실행결과

```
This is String! "따옴표" 기호: !@#$%^
```

04. 자료형

I. 컬렉션 자료형

- 컬렉션 자료형은 데이터를 효율적으로 처리하기 위한 일종의 자료구조
- 튜플
 - 튜플(Tuple)은 순서가 있는 데이터의 목록.

[코드 2-17] 튜플

```
tuple_1 = 1, 2, 3, 4, 5
tuple_2 = ('가', '나', '다', '라', '마')
tuple_3 = '파이썬', 10000, False
tuple_4 = '파이썬', (10000, '만큼', '어려워'), False

print(tuple_1, type(tuple_1))
print(tuple_2, type(tuple_2))
print(tuple_3, type(tuple_3))
print(tuple_4, type(tuple_4))
```

[코드 2-17] 실행결과

```
(1, 2, 3, 4, 5) <class 'tuple'>
('가', '나', '다', '라', '마') <class 'tuple'>
('파이썬', 10000, False) <class 'tuple'>
('파이썬', (10000, '만큼', '어려워'), False)
<class 'tuple'>
```


04. 자료형

I. 컬렉션 자료형

- 튜플

- 튜플을 한번 할당하면 값을 변경하거나 삭제할 수 없으므로 주의.

[코드 2-18] 튜플 변경 및 삭제

```
tuple_1 = 1, 2, 3, 4, 5
tuple_1[2] = 100
print(tuple_1)
```

[코드 2-18] 실행결과

```
-----
TypeError Traceback (most recent call last)
<ipython-input-5-cb3969b988fd> in <module>
      1 tuple_1 = 1, 2, 3, 4, 5
----> 2 tuple_1[2] = 100
      3 print(tuple_1)

TypeError: 'tuple' object does not support item assignment
```

04. 자료형

I. 컬렉션 자료형

- 튜플
 - 튜플의 크기를 알고 있으면 편리함.

[코드 2-19] len() 함수

```
tuple_1 = 1, 2, 3, 4, 5  
print(len(tuple_1))
```

[코드 2-19] 실행결과

5

04. 자료형

I. 컬렉션 자료형

- 세트

- 세트(Set)는 데이터 중복을 허용하지 않으며 데이터 입력 순서는 중요하지 않음.

[코드 2-20] 세트 생성

```
set_1 = {1, 2, 3, '가', '나', '다', 1, 2}
set_2 = set({1, 2, 3, '가', '나', '다', 1, 2})
set_3 = set([1, 2, 3, '가', '나', '다', 3])

print(set_1)
print(set_2)
print(set_3)
```

[코드 2-20] 실행결과

```
{1, 2, 3, '가', '나', '다'}
{1, 2, 3, '가', '다', '나'}
{1, 2, 3, '가', '나', '다'}
```

04. 자료형

I. 컬렉션 자료형

- 세트

- 세트의 데이터는 유일하기 때문에 순서는 중요하지 않음.
(따라서 데이터의 순서가 예시와 다르게 출력될 수 있음.)

[코드 2-21] 세트 변경 및 복제

```
set_1 = {1, 2, 3, '가', '나', '다'}  
  
set_1.add('추가')  
print(set_1)  
set_1.remove('가')  
print(set_1)  
set_copy_1 = set_1.copy( )  
print(set_copy_1)  
set_copy_1.clear( )  
print(set_copy_1)
```

[코드 2-21] 실행결과

```
{1, 2, 3, '나', '다', '가', '추가'}  
{1, 2, 3, '나', '다', '추가'}  
{1, 2, 3, '나', '추가', '다'}  
set( )
```

04. 자료형

I. 컬렉션 자료형

- 리스트
 - 리스트(List)는 데이터를 다루기 편리하여 매우 자주 활용되는 컬렉션 자료형.
 - 파이썬의 내장함수로 리스트 데이터를 추가(Append), 삽입(Insert), 삭제(Remove), 정렬(Sort).

04. 자료형

I. 컬렉션 자료형

- 리스트

[코드 2-22] 리스트

```
#리스트 생성하기
list_1 = [1, 2, 3, 4, 5, 1, 3]
list_2 = [ ]
print(list_1)
print(list_2)
print(len(list_1))

#리스트 변경하기
list_1[3] = 9999
print(list_1)
list_1.append(100)
print(list_1)
list_1.remove(9999)
print(list_1)
list_1.insert(0,777)
print(list_1)

#리스트 복제하기
list_2 = list_1.copy( )
print(list_2)
```

[코드 2-21] 실행결과

```
[1, 2, 3, 4, 5, 1, 3]
[ ]
7
[1, 2, 3, 9999, 5, 1, 3]
[1, 2, 3, 9999, 5, 1, 3, 100]
[1, 2, 3, 5, 1, 3, 100]
[777, 1, 2, 3, 5, 1, 3, 100]
[777, 1, 2, 3, 5, 1, 3, 100]
```

04. 자료형

I. 컬렉션 자료형

- 튜플이나 리스트처럼 순서가 있는 컬렉션 자료형은 인덱싱하거나 슬라이싱할 수 있음.
 - 인덱싱(Indexing)은 리스트 요소 값에 인덱스로 접근하는 것.
 - 슬라이싱(Slicing)은 리스트의 연속한 요소 일부를 잘라 사용하는 것.

[코드 2-23] 리스트 데이터 정렬

```
list_1 = [897, 2, 1, 4, 99, 5.24, 17]
print(list_1)

#뒤집기
list_1.reverse( )
print(list_1)

#오름차순 정렬하기
list_1.sort( )
print(list_1)

#내림차순 정렬하기
list_1.sort(reverse=True)
print(list_1)
```

[코드 2-23] 실행결과

```
[897, 2, 1, 4, 99, 5.24, 17]
[17, 5.24, 99, 4, 1, 2, 897]
[1, 2, 4, 5.24, 17, 99, 897]
[897, 99, 17, 5.24, 4, 2, 1]
```

04. 자료형

I. 컬렉션 자료형

- 딕셔너리
 - 딕셔너리(Dictionary)는 단어 그대로 사전과 같은 자료형으로, 값(value)과 키(key)가 한 쌍을 이루어 요소가 되는 자료구조.
 - 키를 이용하여 쌍을 이루는 값에 접근할 수 있으므로 신속하게 값을 찾아내야 할 때 딕셔너리를 사용.

04. 자료형

I. 컬렉션 자료형

- 딕셔너리

[코드 2-24] 딕셔너리

```
#딕셔너리 생성하기
dict_1 = {'name': '홍길동', 'birth': 1990, 'addr': 'KR'}
print(dict_1)
print(dict_1['birth'])

#키와 값 추가하기
dict_1['weight'] = 60.5
dict_1['family'] = ['아빠', '엄마', '여동생']
print(dict_1)

#여러 키와 값을 동시에 추가하기
dict_1.update({'weight': 67.8, 'hobby': ['게임', '독서']})
print(dict_1)
```

#다음 페이지에 계속

[코드 2-24] 실행결과

```
{'name': '홍길동', 'birth': 1990, 'addr': 'KR'}
1990
{'name': '홍길동', 'birth': 1990, 'addr': 'KR',
'weight': 60.5, 'family': ['아빠', '엄마',
'여동생']}
{'name': '홍길동', 'birth': 1990, 'addr': 'KR',
'weight': 67.8, 'family': ['아빠', '엄마',
'여동생'], 'hobby': ['게임', '독서']}
```

#다음 페이지에 계속

04. 자료형

I. 컬렉션 자료형

- 딕셔너리

[코드 2-24] 딕셔너리

```
#딕셔너리 값 변경하기
dict_1['hobby'] = ['축구', '등산']
print(dict_1)

#데이터 삭제하기
del dict_1['weight']
del dict_1['birth']
del dict_1['addr']
print(dict_1)
```

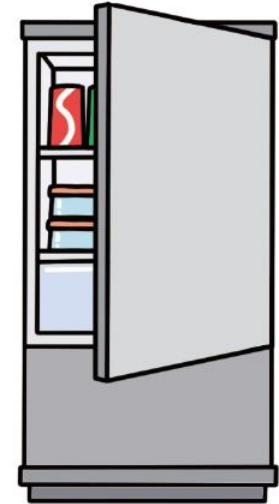
[코드 2-24] 실행결과

```
{'name': '홍길동', 'birth': 1990, 'addr':  
'KR', 'weight': 67.8, 'family': ['아빠',  
'엄마', '여동생'], 'hobby': ['축구', '등산']}  
{'name': '홍길동', 'family': ['아빠', '엄마',  
'여동생'], 'hobby': ['축구', '등산']}
```

LAB. 식재료 관리 프로그램

리스트를 활용하여 식재료 관리 프로그램 만들기

- '보관 식재료 출력' 메뉴를 선택하면 현재 냉장고에 보관 중인 식재료 목록을 출력.
- '식재료 추가' 메뉴를 선택하면 "추가할 식재료 이름을 입력하십시오" 출력하고 입력 받은 이름을 식재료 목록에 추가.
- '식재료 삭제' 메뉴를 선택하면 "삭제할 식재료를 입력하십시오" 출력하고 입력 받은 식재료가 냉장고에 있으면 삭제. 없으면 "식재료 재고 없음" 출력.
- '식재료 변경' 메뉴를 선택하면 "교환할 식재료를 입력하십시오" 출력하고, 입력 받은 식재료가 냉장고에 있으면 식재료를 하나 더 입력 받아서 변경. 없으면 "식재료 재고 없음" 출력.
- '종료' 메뉴를 선택하면 프로그램을 종료.



1. 식재료 목록을 리스트형으로 정의.

```
menu = 0
food = [ ]

while menu != 5:
    print('-'*10)
    print('''1.보관 식재료 출력
2.식재료 추가
3.식재료 삭제
4.식재료 변경
5.종료''')
    print('-'*10)
```

LAB. 식재료 관리 프로그램

리스트를 활용하여 식재료 관리 프로그램 만들기

2. `input()` 명령어로 메뉴를 선택하라고 안내.

```
menu = int(input('관리 메뉴를 선택하시오: '))
```

3. 메뉴 번호가 1이면 식재료 리스트를 출력.

```
if menu == 1:  
    print(food)
```

4. 메뉴 번호가 2이면 리스트에 원소를 추가하는 `input()` 명령어를 작성

```
elif menu == 2:  
    name = input('추가할 식재료를 입력하시오: ')  
    food.append(name)  
    print(food)
```

LAB. 식재료 관리 프로그램

리스트를 활용하여 식재료 관리 프로그램 만들기

5. `input()` 명령어로 메뉴를 선택하라고 안내.

```
elif menu == 3:
    eli_name = input('삭제할 식재료를 입력하시오: ')
    if eli_name in food:
        food.remove(eli_name)
    else:
        print('식재료 재고 없음')
```

6. 교환할 식재료 이름 문자열을 입력 받아 변수 `exch_name`에 할당, 식재료 목록에 `exch_name`과 일치하는 요소가 있으면 인덱스를 변수 `idx`에 할당, `idx` 인덱스로 요소에 접근해서 식재료 변경, 식재료 목록에 `exch_name`과 일치하는 요소가 없으면 '식재료 재고 없음'을 출력.

```
elif menu == 4:
    exch_name = input('교환할 식재료를 입력하시오: ')
    if exch_name in food:
        idx = food.index(exch_name)
        new_name = input('새로운 식재료를 입력하시오: ')
        food[idx] = new_name
    else:
        print('식재료 재고 없음')
```

LAB. 식재료 관리 프로그램

리스트를 활용하여 식재료 관리 프로그램 만들기

7. 메뉴 번호가 5이면 프로그램을 종료.

```
elif menu == 5:  
    break
```

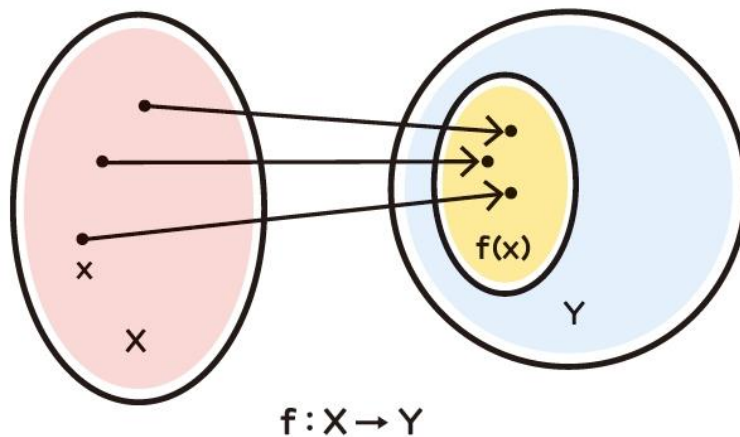
05

함수

05. 함수

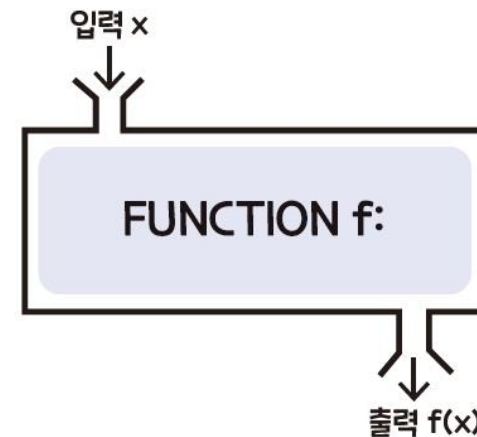
I. 함수

- 프로그래밍에서 함수(Function)는 기능을 정의한 가장 작은 단위.
- 파이썬에서의 함수는 수학에서의 함수와 비슷.



(a) 수학의 함수

그림 2-4 함수의 정의



(b) 파이썬 프로그래밍의 함수

05. 함수

I. 함수

[코드 2-25] len() 함수

```
list_1 = [1, 2, 3, 4, 5, 1, 3, 13, 41, 51]  
length = len(list_1)  
print(length)
```

[코드 2-25] 실행결과

10



그림 2-5 len() 함수

05. 함수

I. 함수

- 리스트에 포함된 모든 수를 더하고 결과를 출력하는 간단한 함수 만들기.
- def 뒤에 함수명으로 사용할 이름을 적고 괄호 안에 입력으로 사용할 데이터의 가상 이름을 작성
- 괄호 안 입력은 매개변수(parameter 또는 argument).

[코드 2-26] 합을 출력하는 함수 정의

```
def sum_list(a):  
    j = 0  
    for i in a:  
        j = j + i  
    print(j)
```

[코드 2-27] 합을 출력하는 함수 호출

```
list_a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
sum_list(list_a)
```

[코드 2-27] 실행결과

55

05. 함수

I. 함수

- `sum_list()` 함수는 함수가 계산 결과를 출력하기만 하고 작업을 종료.
- 함수를 호출한 지점에 계산 결과를 반환하는 `sum_list_r()` 함수 필요.

[코드 2-28] 합을 반환하는 함수 정의

```
def sum_list_r(a):  
    j = 0  
    for i in a:  
        j = j + i  
    return j
```

LAB. 식재료 관리 프로그램

사용자에게 숫자를 입력 받고 숫자가 소수(Prime number)인지 판별하는 프로그램 만들기

- '소수는 1보다 큰 자연수 중에서 1과 자기 자신을 제외한 자연수로 나누어 떨어지지 않는 수.
- 사용자가 입력한 값이 x이면 2부터 x-1까지 모든 자연수로 x를 나눕니다. 나누어 떨어지는 수가 하나라도 있으면 소수가 아니며, 없으면 소수입니다.



1. 소수를 판별할 함수의 이름을 정의.

```
def check_prime_num(x):
```

2. 반복문으로 2부터 입력 받은 수까지 확인하도록 명령

```
for i in range(2, x):
```

LAB. 식재료 관리 프로그램

사용자에게 숫자를 입력 받고 숫자가 소수(Prime number)인지 판별하는 프로그램 만들기

3. 반복문 블록 안에서 입력 받은 수가 i로 나누어 떨어지는지 확인.

```
if x % i == 0:  
    #x가 i로 나누어 떨어지면 실행하기  
    return False  
return True
```

4. 사용자에게 숫자를 입력 받는 input() 함수를 호출

```
number = int(input('판별할 자연수를 입력하세요:'))  
print(check_prime_num(number))
```

LAB. 식재료 관리 프로그램

사용자에게 숫자를 입력 받고 숫자가 소수(Prime number)인지 판별하는 프로그램 만들기

- 반복문 블록 안에서 입력 받은 수가 i로 나누어 떨어지는지 확인.

판별할 자연수를 입력하세요: 131071

True

판별할 자연수를 입력하세요: 30

False

실전분석 예금 이자 계산기

[문제]

대학을 졸업하고 취업에 성공한 난생이는 1년 동안 3천만 원을 모았습니다. 난생이는 모은 돈을 고금리 예금에 예치하기로 했습니다. 은행에서 연 5.1% 금리인 3년짜리 예금 상품에 가입했을 때, 함수를 이용하여 만기 시 수령할 원금과 이자를 계산해 봅시다.



[해결]

1. 예금은 원금에 대해 연간 이자를 적립하고 만기 시에 한꺼번에 지급하는 상품. 그러므로 원금에 연간 이율을 거치연수만큼 곱한 p' 가 복리 예금의 원리금.

$$p' = p * (1 + r)^n$$

p' : 원리금, p : 거치금액, r : 연간 이율, n : 거치 연수

실전분석 예금 이자 계산기

[해결]

2. 원리금을 계산하는 `interest_year()` 함수를 정의. 원금 `p`, 연간 이율 `r`, 거치 연수 `n`을 입력 받아 원리금 `result`를 반환하도록 함.

```
def interest_year(p, r, n):  
    return p * (1+r)**n
```

3. 난생이가 가입한 예금 상품의 원금, 연간 이율, 거치 연수를 변수에 할당하고 함수의 입력으로 사용

```
p = 30000000  
r = 0.051  
n = 3  
  
result = interest_year(p, r, n)
```

4. 원금 `p`와 이자 `result-p`를 각각 출력하는 출력문을 작성.

```
print('원금: {0}, 이자: {1}'.format(p, result-p))
```

```
원금: 30000000, 이자: 4828069.529999994
```


Thank You!