IT@COOKBOOK

데이터 과학 기반의 파이썬 빅데이터 분석

Chapter 04 파이썬 프로그래밍 기초

목차

01. 파이썬 시작하기

■ 파이썬과 빅데이터 분석

- 파이썬
  - 1990년 암스테르담의 귀도 반 로섬이 개발한 인터프리터 방식의 프로그래밍 언어
  - 귀도 반 로섬이 좋아하던 코미디 프로그램인 Monty Python' s Flying Circus에서 따온 것
  - 고대 신화에서 파르나소스 산 동굴에 살던 큰 뱀을 일컫는 말이라 파이썬 로고는 뱀
  - 파이썬은 쉽게 배울 수 있고 오픈 소스로 제공되어 무료로 사용할 수 있어 빅데이터 분석에 사용함

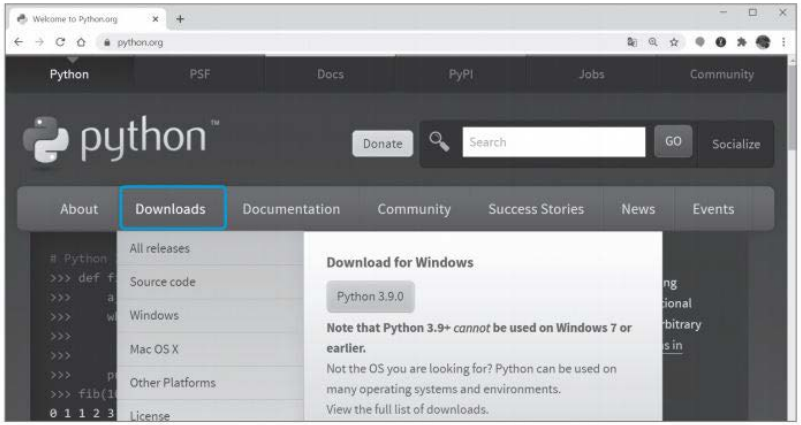
그림 4-1 파이썬 로고

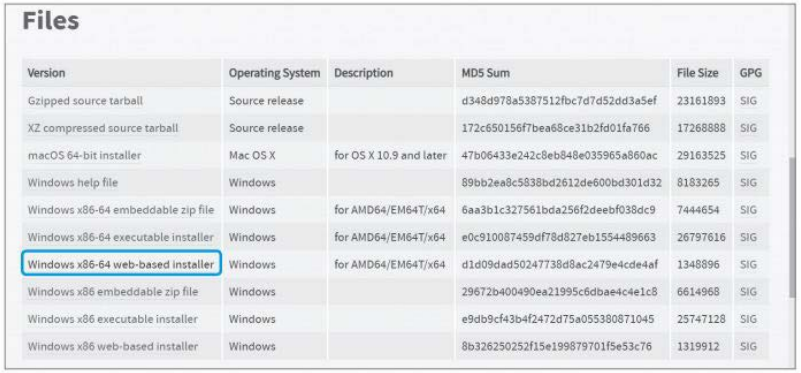
01. 파이썬 시작하기

■ 파이썬 설치와 실행

- 파이썬 설치하기(IDLE : Integrated Development and Learning Environment)

1. 설치 파일 다운로드하기





Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		d348d978a5387512fbc7d7d52dd3a5ef	23161893	SIG
XZ compressed source tarball	Source release		172c650156f7bea68ce31b2fd01fa766	17268888	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	47b06433e242c8eb848e035965a860ac	29163525	SIG
Windows help file	Windows		89bb2ea8c5838bd2612de600bd301d32	8183265	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	6aa3b1c327561bda256f2deebf038dc9	7444654	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	e0c910087459df78d827eb1554489663	26797616	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	d1d09dad50247738d8ac2479e4cde4af	1348896	SIG
Windows x86 embeddable zip file	Windows		29672b400490ea21995c6dbae4c4e1c8	6614968	SIG
Windows x86 executable installer	Windows		e9db9cf43b4f2472d75a055380871045	25747128	SIG
Windows x86 web-based installer	Windows		8b326250252f15e199879701f5e53c76	1319912	SIG

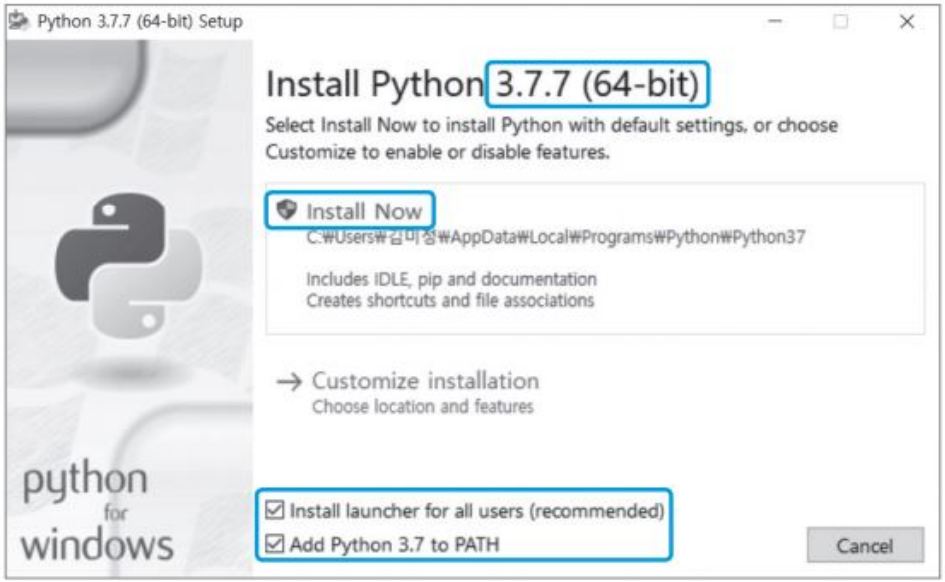
01. 파이썬 시작하기

■ 파이썬 설치와 실행

- 파이썬 설치하기

2. 설정 후 설치하기

2025년03월 =>3.13.2

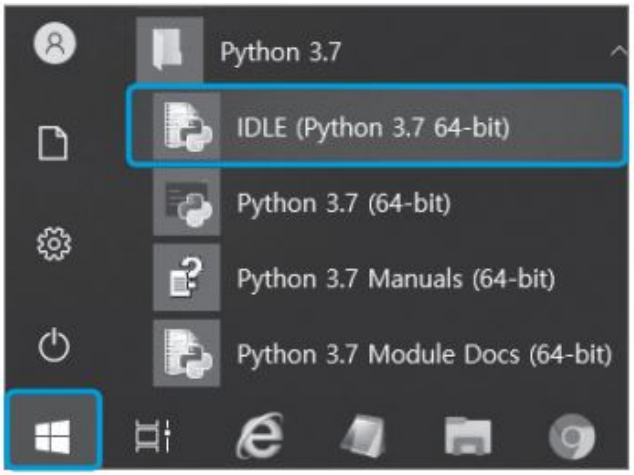


01. 파이썬 시작하기

■ 파이썬 설치와 실행

- 파이썬 작업 환경 실행하기

1. 시작창 → [Python 3.7] → [IDLE (Python 3.7 64-bit)] → 파이썬 작업 환경을 실행

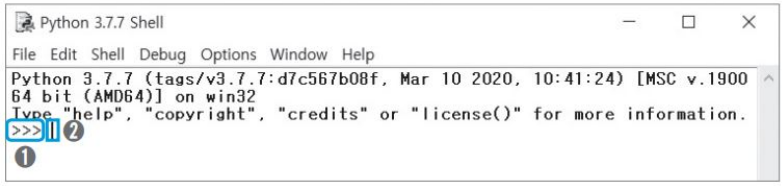


01. 파이썬 시작하기

■ 파이썬 설치와 실행

▪ 파이썬 작업 환경 실행하기

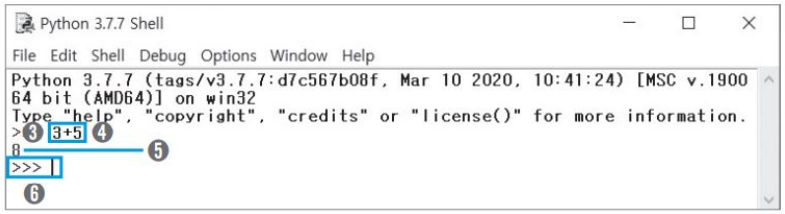
2. 파이썬 셸 창 확인하기



❶ 프롬프트: 명령어를 입력받을 준비가 되었음을 나타냄

❷ 커서: 입력한 명령어가 들어갈 위치

파이썬 셸 창의 기능을 알아보기 위해 '3+5'를 입력한 뒤 [Enter] 를 누름



❸ 명령어 입력하기: 키보드로 입력한 명령어가 프롬프트 다음에 있는 커서 위치에 표시

❹ [Enter] 누르기: 명령어 입력 상태를 끝내고 입력된 명령어를 실행

❺ 실행 결과가 파란색으로 나타난다. 에러가 발생하면 빨간색으로 메시지가 나타남

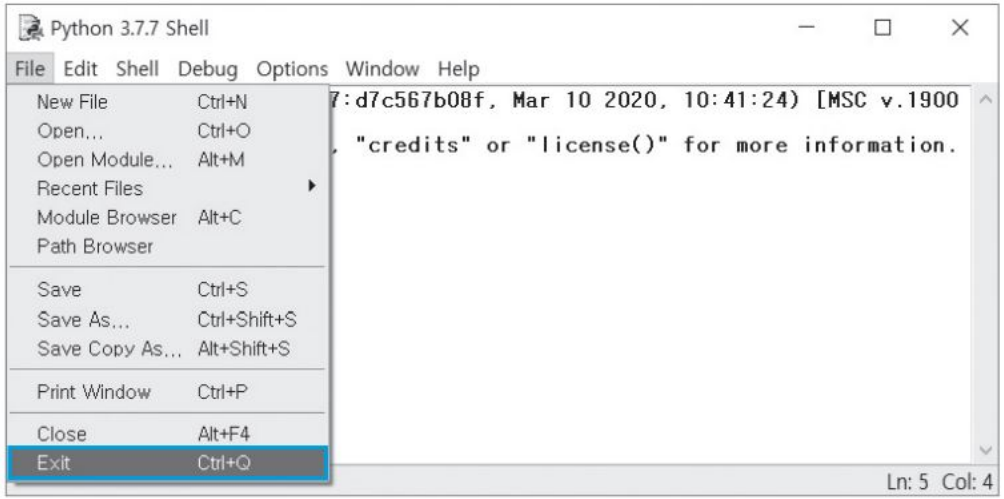
❻ 명령 실행이 끝나면 프롬프트와 커서가 다시 나타나서 다음 명령어를 입력할 준비가 되었음을 알림

01. 파이썬 시작하기

■ 파이썬 설치와 실행

- 파이썬 작업 환경 실행하기

3. 파이썬 셸 창 닫기



4. 실습 폴더 만들기

02. 변수와 객체

■ 변수

- 값을 저장하는 메모리 공간
- 파이썬에서는 변수를 미리 선언하지 않음
- 변수에 저장해서 사용하는 값의 자료형으로 변수의 자료형이 결정

■ 객체

- 변수 형태의 속성과 함수 형태의 메서드를 가진 것
- 각 객체는 자기 의 속성(내부 데이터)과 메서드(내부 연산)를 가짐
- 타 프로그래밍 언어와 달리 파이썬에서는 모든 변수와 자료형이 객체로 되어 있음

03. 자료형과 연산자

■ 기본 자료형

- 숫자형

```
>>> a = 123
>>> a = 12.34
```

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>> a - b
-1
>>> a * b
12
>>> a / b
0.75
>>> a ** b
81
>>> 2 ** 3
8
>>> a % b
3
>>> 7 % 3
1
>>> a // b
0
>>> 7 // 3
2
```

- 논리형

```
>>> b = True
>>> b
True
```

03. 자료형과 연산자

■ 그룹 자료형

▪ 문자열 자료형

- 한 개 이상의 문자로 구성된 문자 집합
- 작은따옴표('), 큰따옴표(") 또는 작은따옴표 3개(''')나 큰따옴표 3개(""")를 사용하여 나타냄

- 문자열의 각 문자는 인덱스를 이용하여 지정할 수 있음

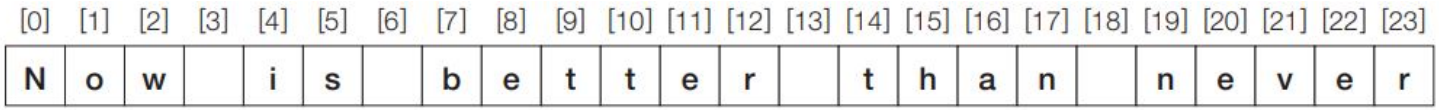


그림 4-2 문자열과 인덱스

- 인덱스의 범위를 이용하여 내부 문자열을 지정할 수도 있

그림 4-3 인덱스 범위

03. 자료형과 연산자

■ 그룹 자료형

▪ 문자열 자료형

- 문자열

```
>>> s1 = 'Hello Python'
>>> s1
'Hello Python'

>>> s2 = "Hello Python"
>>> s2
'Hello Python'

>>> s3 = '''Hello Python'''
>>> s3
'Hello Python'

>>> s4 = """Hello Python"""
>>> s4
'Hello Python'
```

- 사용 가능 연산자

```
>>> head = "Python"
>>> tail = " is fun"
>>> head + tail
'Python is fun'

>>> head * 2
'PythonPython'

>>> print("=" * 5)
=====

>>> a = "Now is better than never"
>>> a[0]
'N'
>>> a[4]
'i'
>>> a[-1]
'r'
>>> a[-2]
'e'
```

```
>>> b = a[0] + a[1] + a[2]
>>> b
'Now'

>>> a[0:3]
'Now'
>>> a[4:6]
'is'
>>> a[19:]
'never'
>>> a[:3]
'Now'
>>> a[:]
'Now is better than never'
>>> a[7:-11]
'better'

>>> a = "Python"
>>> a.count('p')
0
```

03. 자료형과 연산자

■ 그룹 자료형

▪ 문자열 자료형

- 사용 가능 함수

```
>>> a='python'
>>> a.find('y')
1
>>> a.find('p')
0
>>> a.index('y')
1
>>> a.index('p')
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    a.index('p')
ValueError: substring not found
```

```
>>> b = ","
>>> c = b.join('Abcd')
>>> c
'A, b, c, d'

>>> a.upper()
'PYTHON'
>>> a.lower()
'python'
>>> d = " py "
>>> d.lstrip()
'py '
 >>> d.rstrip()
' py'
>>> d.strip()
'py'

>>> a = 'Pithon'
>>> a[1] = 'y'
Traceback (most recent call last):
  File "<pyshell#81>", line 1, in <module>
    a[1] = 'y'
TypeError: 'str' object does not support item assignment
```

```
>>> a = "Python is difficult."
>>> a.replace("difficult", "funny")
'Python is funny.'

>>> a.split()
['Python', 'is', 'difficult.']

>>> b = "a, b, c, d"
>>> b
'a, b, c, d'
>>> b.split(',')
['a', 'b', 'c', 'd']
```

03. 자료형과 연산자

■ 그룹 자료형

▪ 리스트 자료형

```
>>> a = [1, 2, 3]
>>> b = ['Life', 'is', 'too', 'short']
>>> c = [1, 2, 'Life', 'is']
>>> d = [1, 2, [3, 4], ['Life', 'is']]
>>> d[0]
1
>>> d[2]
[3, 4]
>>> d[3]
['Life', 'is']
>>> d[3][-1]
'is'
>>> d[0:3]
[1, 2, [3, 4]]
>>> a + b
[1, 2, 3, 'Life', 'is', 'too', 'short']
>>> b[0] + " hi~ ^^;"
'Life hi~ ^^;'
>>> a[0] + " hi~ ^^;"
Traceback (most recent call last):
  File "<pyshell#88>", line 1, in <module>
    a[0] + ' hi~ ^^;'
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> a[2] = 99
>>> a
[1, 2, 99]
>>> a[1:2] = ['a', 'b', 'c']
>>> a
[1, 'a', 'b', 'c', 99]
>>> a[-1] = ['d', 'e', 'f']
>>> a
[1, 'a', 'b', 'c', ['d', 'e', 'f']]
>>> del a[-1]
>>> a
[1, 'a', 'b', 'c']
>>> a.append(5)
>>> a
[1, 'a', 'b', 'c', 5]

>>> b.sort()
>>> b
['Life', 'is', 'short', 'too']
>>> a = [3, 4, 1, 9]
>>> a.reverse()
>>> a
[9, 1, 4, 3]
>>> a.index(9)
0
>>> a.insert(0, 99)
>>> a
[99, 9, 1, 4, 3]
>>> a.remove(99)
>>> a
[9, 1, 4, 3]
>>> b = [1, 2, 3]
>>> b.pop()
3
>>> b
[1, 2]
>>> b.pop(0)
1
>>> b
[2]
>>> a = [2, 1, 0, 2, 3, 2, 4, 2]
>>> a.count(2)
4
```

03. 자료형과 연산자

■ 그룹 자료형

▪ 튜플 자료형

```
>>> t1 = (1, )
>>> t2 = (1, 2, 3)
>>> t3 = 1, 2, 3
>>> t4 = (1, 2, (3, 4), ('Life', 'is'))
>>> t4[0]
1
>>> t4[3][-1]
'is'
>>> t4[0:3]
(1, 2, (3, 4))
>>> t1 + t2
(1, 1, 2, 3)
>>> t1 + "hi~ ^^;"
Traceback (most recent call last):
  File "<pyshell#157>", line 1, in
<module>
    t1 + 'hi~ ^^;'
TypeError: can only concatenate tuple (not
"str") to tuple
```

```
>>> t2 * 3
(1, 2, 3, 1, 2, 3, 1, 2, 3)
>>> t2[2] = 99
Traceback (most recent call last):
 File "<pyshell#159>", line 1, in
<module>
 t2[2] = 99
TypeError: 'tuple' object does not support
item assignment
```

03. 자료형과 연산자

■ 그룹 자료형

▪ 딕셔너리 자료형

```
>>> dic = {'name':'Hong', 'phone':'01012345678', 'birth':'0814'}
>>> dic[1] = 'a'
>>> dic
{'name': 'Hong', 'phone': '01012345678', 'birth': '0814', 1: 'a'}
>>> dic['pet'] = 'dog'
>>> dic
{'name': 'Hong', 'phone': '01012345678', 'birth': '0814', 1:
'a', 'pet': 'dog'}
>>> del dic[1]
>>> dic
{'name': 'Hong', 'phone': '01012345678', 'birth': '0814',
'pet': 'dog'}
>>> dic['pet']
'dog'
>>> dic['name']
'Hong'
>>> dic.keys()
dict_keys(['name', 'phone', 'birth', 'pet'])
>>> list(dic.keys())
['name', 'phone', 'birth', 'pet']
```

```
>>> dic.values()
dict_values(['Hong', '01012345678', '0814',
'dog'])
>>> list(dic.values())
['Hong', '01012345678', '0814', 'dog']

>>> dic.items()
dict_items([('name', 'Hong'), ('phone',
'01012345678'),
('birth', '0814'), ('pet', 'dog')])
>>> dic.clear()
>>> dic
{}
```

03. 자료형과 연산자

■ 그룹 자료형

▪ 집합 자료형

```
>>> s1 = {1, 2, 'a', 5}
>>> s2 = set([1, 2, 3, 4, 5, 6])
>>> s2
{1, 2, 3, 4, 5, 6}
>>> s3 = set([4, 5, 6, 7, 8, 9])
>>> s3
{4, 5, 6, 7, 8, 9}
>>> s2 & s3
{4, 5, 6}
>>> s2.intersection(s3)
{4, 5, 6}
>>> s2 | s3
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> s2.union(s3)
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> s2 - s3
{1, 2, 3}
>>> s2.difference(s3)
{1, 2, 3}
>>> s3.difference(s2)
{8, 9, 7}
```

```
>>> s2.add(7)
>>> s2
{1, 2, 3, 4, 5, 6, 7}
>>> s2.update([6, 7, 8, 9, 10])
>>> s2
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
>>> s2.remove(7)
>>> s2
{1, 2, 3, 4, 5, 6, 8, 9, 10}
```

03. 자료형과 연산자

■ 그룹 자료형

▪ 그룹 자료형의 특징

표 4-1 그룹 자료형의 특징

인덱스 사용	그룹 자료형	원소값 변경	파이썬 코드

가능Ordered	문자열	X	>>> a = 'abc'
			>>> a
			'abc'

	리스트	O	>>> a = ['a', 'b', 'c']
			>>> a
			['a', 'b', 'c']

	튜플	X	>>> a = ('a', 'b', 'c')
			>>> a
			('a', 'b', 'c')

불가능Unordered	딕셔너리	O	>>> a = {'name':"Hong"}
			>>> a
			{'name':'Hong'}

	집합	O	>>> a = set(['a', 'b', 'c'])
			>>> a
			{'a', 'b', 'c'}

04. 조건문과 반복문

■ 조건문

- 조건문
  - 참 또는 거짓을 판별하는 조건식을 검사하여 결과값이 참인지 거짓인지에 따라 실행할 문장을 선택하여 처리하는 제어문

▪ if 문

- 단일 조건문
- 조건식을 검사하여 결과가 참이면 명령문 1을 수행
- 거짓이면 명령문 1은 건너뛰고 명령문 2를 수행

```
if 조건식:
    실행할 명령문 1

실행할 명령문 2
```

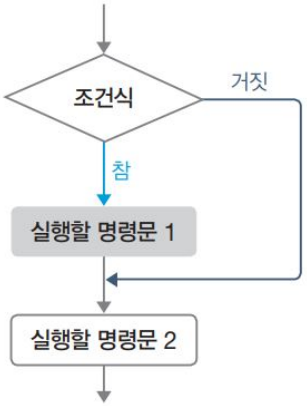


그림 4-4 if문의 구조와 제어 순서도

04. 조건문과 반복문

■ 조건문

▪ if-else 문

- 단일 조건문
- 조건식을 검사하여 결과가 참이면 명령문 1을 수행한 뒤 명령문 3을 수행
- 거짓이면 명령문 2를 수행한 뒤 명령문 3을 수행

```
if 조건식:
    실행할 명령문 1
else:
    실행할 명령문 2

실행할 명령문 3
```

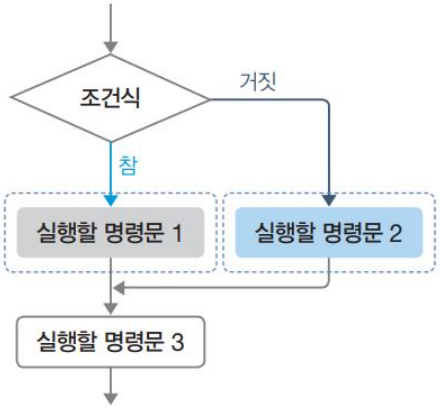


그림 4-5 if-else문의 구조와 제어 순서도

04. 조건문과 반복문

■ 조건문

▪ if-elif-else 문

- 다중 조건문
- 조건식 1이 거짓이면 elif 다음의 조건식 2를 검사(elif 키워드: else 와 if를 결합)
- 참이면 명령문 2를 수행한 뒤 명령문 4를 수행
- 거짓이면 명령문 3을 수행 한 뒤 명령문 4를 수행

```
if 조건식 1:
    실행할 명령문 1
elif 조건식 2:
    실행할 명령문 2
else:
    실행할 명령문 3

실행할 명령문 4
```

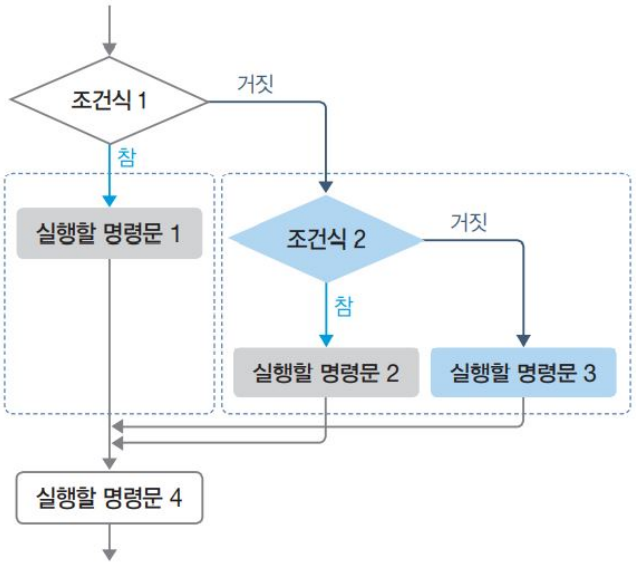


그림 4-6 if-elif-else문의 구조와 제어 순서도

04. 조건문과 반복문

■ 조건문

▪ 조건식

```
>>> x = 3
>>> y = 2
>>> x == y
False
>>> x != y
True
>>> x >= y
True
>>> money = 1300
>>> if money >= 1200 and money < 3500: [Enter]
        print("버스를 탈 수 있습니다.") [Enter]
        [Enter]
버스를 탈 수 있습니다.
```

```
>>> 1 in [1, 2, 3]
True
>>> x in [1, 2, 3]
True
>>> x not in [1, 2, 3]
False
>>> 'a' in ('a', 'b', 'c', 'd')
True
>>> 'i' not in 'Python'
True
>>> if money < 10: [Enter]
        pass [Enter]
else: [Enter]
        print("저금하자!") [Enter]
        [Enter]
저금하자!
```

04. 조건문과 반복문

■ 반복문

- for 문

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list: [Enter]
        x = i + '!' [Enter]
        print(x) [Enter]
        [Enter]
one!
two!
three!
>>> number = 0
>>> for score in [90, 25, 67, 45, 93]: [Enter]
        number += 1 [Enter]
        if score >= 60: [Enter]
            print("%d번 학생은 합격입니다." %number) [Enter]
        else: [Enter]
            print("%d번 학생은 불합격입니다." %number) [Enter]
            [Enter]
1번 학생은 합격입니다.
2번 학생은 불합격입니다.
3번 학생은 합격입니다.
4번 학생은 불합격입니다.
5번 학생은 합격입니다.!
```

- while 문

```
>>> i = 0
>>> while i < 5: [Enter]
        i += 1 [Enter]
        print('*' * i) [Enter]
        [Enter]
*
**
***
****
*****
```

05. 함수

■ 사용자 정의 함수

- 함수 정의

```
>>> def sum1(a, b): [Enter]
        x = a + b [Enter]
        return x [Enter]
[Enter]

>>> def sum2(*args): [Enter]
        x = 0 [Enter]
        for i in args: [Enter]
            x += i [Enter]
        return x [Enter]
[Enter]
```

- 함수 호출

```
>>> a = 5
>>> b = 3
>>> sum1(a, b)
8
>>> sum1(3, 5)
8
>>> sum2(1, 2, 3, 4, 5)
15
>>> sum2(2, 3.5, 10)
15.5
```

05. 함수

■ 내장 함수

- 함수 종류

abs (x)
all(iterable_x)
any(iterable_x)
chr(x)
ord(c)
dir(x)
divmod(a, b)
oct(x)
hex (x)
id(object)
int(x)
str(x)
list(x)

- 함수 테스트

```
>>> abs(-3.5)
3.5
>>> all([1, 2, 3, 4])
True
>>> all([4, -2, 0.0, 4])
False
>>> any([1, 2, 3, 4])
True
>>> any([4, -2, 0.0, 4])
True
>>> chr(97)
'a'
>>> chr(48)
'0'
>>> ord('a')
97
>>> ord('0')
48

>>> dir([1, 2, 3])
>>> dir({'1':'a'})
>>> dir(1)
```

```
>>> divmod(7, 3)
(2, 1)
>>> divmod(1.3, 0.2)
(6.0, 0.09999999999999998)
>>> oct(8)
'0o10'
>>> oct(234)
'0o352
>>> hex(16)
'0x10'
>>> hex(234)
'0xea'
>>> a = 3
>>> id(a)
1728080976
>>> int('3')
3
>>> str(3)
'3'
>>> list("Python")
['P', 'y', 't', 'h', 'o', 'n']
>>> list((1, 2, 3))
[1, 2, 3]
```

05. 함수

■ 내장 함수

- 함수 종류

tuple(x)

type(x)

Lambda

max (iterable_x)

min(iterable_x)

pow(x, y)

input()

range(x)

len(s)

sorted(iterable_x)

- 함수 테스트

```
>>> tuple("Python")
('P', 'y', 't', 'h', 'o', 'n')
>>> tuple([1, 2, 3])
(1, 2, 3)
>>> type("abc")
<class 'str'>
>>> type(a)
<class 'int'>
>>> sum = lambda a,b: a+b
>>> sum
<function <lambda> at 0x000002C826BABEA0>
>>> sum([3, 5])
8
>>> max([1, 4, 2, 8, 6])
8
>>> max("Python")
'y'
>>> min([1, 4, 2, 8, 6])
1
>>> min("Python")
'P'
```

```
>>> pow(2, 4)
16
>>> c = input()
21 [Enter]
>>> c
'21'
>>> c = input("정수를 입력하세요: ")
정수를 입력하세요: 21 [Enter]
>>> c
'21'
>>> range(5)
range(0, 5)
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> list(range (5, 10))
[5, 6, 7, 8, 9]
>>> list(range (5, 10, 2))
[5, 7, 9]
>>> len('Python')
6
>>> sorted([3, 0, 2, 1])  # x.sort()
[0, 1, 2, 3]
>>> sorted('Python')
['P', 'h', 'n', 'o', 't', 'y']
```

05. 함수

■ 모듈과 패키지

- 파이썬 코드

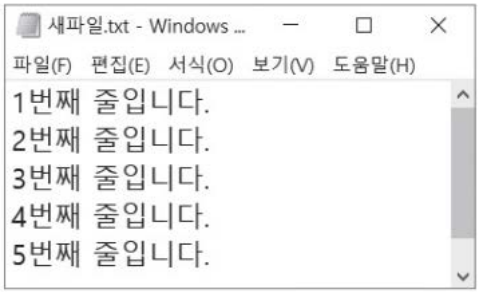
```
>>> Request('http://www.hanb.co.kr')
Traceback (most recent call last):
 File "<pyshell#0>", line 1, in <module>
 Request('http://www.hanb.co.kr')
NameError: name 'Request' is not defined

>>> import urllib.request
>>> urllib.request.Request('http://www.hanb.co.kr')
<urllib.request.Request object at 0x000001E5E0AE8390>
>>> import pandas
>>> pandas.DataFrame()
Empty DataFrame
Columns: []
Index: []

>>> from datetime import datetime
>>> datetime.now()
datetime.datetime(2018, 7, 25, 15, 42, 53, 119540)
```

06. 파일 처리

■ 파일 사용 모드

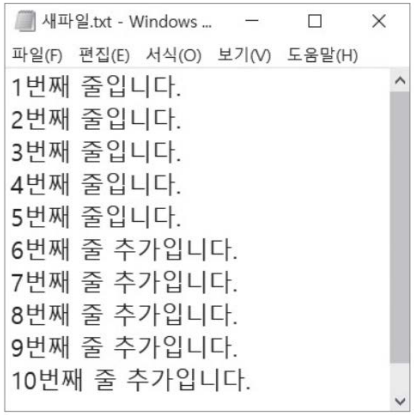
- 쓰기 모드

```
>>> f = open("D:/새파일.txt", 'w')
>>> f
<_io.TextIOWrapper name='D:/새파일.txt' mode='w' encoding='cp949'>
>>> f.close()
>>> f = open("D:/새파일.txt", 'w')
>>> for i in range(1, 6):
        data = "%d번째 줄입니다. ₩n"% i [Enter]
        f.write(data) [Enter]
        [Enter]
>>> f.close()
```



- 추가 모드

```
>>> f= open("D:/새파일.txt", 'a')
>>> for i in range (6, 11):
        data = "%d번째 줄 추가입니다. ₩n"% i [Enter]
        f.write(data) [Enter]
        [Enter]
>>> f.close()
```



06. 파일 처리

■ 파일 사용 모드

- 읽기 모드

```
>>> f= open("D:/새파일.txt", 'r')
>>> line = f.readline()
>>> print(line)
1번째 줄입니다.

>>> while True:
        line = f.readline() [Enter]
        if not line: break [Enter]
        print(line) [Enter]
        [Enter]
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄 추가입니다.
7번째 줄 추가입니다.
8번째 줄 추가입니다.
9번째 줄 추가입니다.
10번째 줄 추가입니다.
>>> f.close()
```

```
>>> f= open("D:/새파일.txt", 'r')
>>> lines = f.readlines()
>>> print(lines)
['1번째 줄입니다. ₩n', '2번째 줄입니다. ₩n', '3번째 줄
입니다. ₩n', '4번째 줄입니다. ₩n', '5번째 줄입니다.
₩n', '6번째 줄 추가입니다. ₩n', '7번째 줄 추가입니다.
₩n', '8번째 줄 추가입니다. ₩n', '9번째 줄 추가입니다.
₩n', '10번째 줄 추가입니다. ₩n']

>>> for line in lines:
        print(line) [Enter]
        [Enter]
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄 추가입니다.
7번째 줄 추가입니다.
8번째 줄 추가입니다.
9번째 줄 추가입니다.
10번째 줄 추가입니다.
>>> f.close()
```

06. 파일 처리

■ 파일 사용 모드

- 읽기 모드

```
>>> f= open("D:/새파일.txt", 'r')
>>> data = f.read()
>>> data
'1번째 줄입니다. ₩n2번째 줄입니다. ₩n3번째 줄입니다.
₩n4번째 줄입니다. ₩n5번째 줄입니다. ₩n6번째 줄 추가
입니다. ₩n7번째 줄 추가입니다. ₩n8번째 줄 추가입니다.
₩n9번째 줄 추가입니다. ₩n10번째 줄 추가입니다. ₩n'

>>> f.close()
>>> with open("D:/새파일.txt", 'w') as f:
        f.write("Now is better than never.") [Enter]
        [Enter]
>>> data = f.read()
Traceback (most recent call last):
 File "<pyshell#95>", line 1, in <module>
 data = f.read()
ValueError: I/O operation on closed file.
```

07. 데이터 분석을 위한 주요 라이브러리

■ library

- 라이브러리 설치 위치 보기
  import sys
  sys.path
- 라이브러리에서 함수 찾기
  - 프로그램(dirtest.py)
- import 문의 작동 원리
  - gcdlib.py, lcmlib.py 프로그램
  - gcdlcmtest.py 프로그램

#gcdlcmtest.py

```
from lcmlib import lcm
k=lcm(20,10)
print(k)
```

#lcmlib.py

```
import gcdlib

def lcm(a, b):
    g=gcdlib.gcd(a,b)
    x,y=a//g, b//g
    return x*y*g

if __name__ == '__main__':
    x,y=[int(n) for n in input('두수 입력:').split()[:2]]
    l=lcm(x,y)
    print("%d, %d 최소공배수= %d" %(x,y,l))
```

#gcdlib.py

```
def gcd(a, b):
    if a<b:
        a,b=b,a
    if a%b==0:
        return b
    return gcd(b, a%b)

if __name__ == '__main__':
    x,y=[int(n) for n in input('두수 입력:').split()[:2]]
    g=gcd(x,y)
    print("%d, %d 최대공약수= %d" %(x,y,g))
#print(__name__)
```

07. 데이터 분석을 위한 주요 라이브러리

numpy를 사용하는 이유

1. NumPy는 python에서 수학/과학 연산을 위한 다차원 배열 객체를 지원합니다.
2. NumPy의 다차원 배열 자료형인 ndarray는 scipy, pandas 등 다양한 파이썬 패키지의 기본 자료형으로 사용되기 때문에 딱히 이유없이 사용할 수 밖에 없습니다.
3. 또 For 문과 같이 반복적인 연산 작업을 배열단위로 처리해(= vevtorized operation), 효율적인 코딩이 가능합니다.

07. 데이터 분석을 위한 주요 라이브러리

■ numpy

- 파이썬 코드

```
import numpy as np
np.__version__
'1.19.2'

ar1 = np.array([1, 2, 3, 4, 5])
ar1
array([1, 2, 3, 4, 5])
type(ar1)
numpy.ndarray
ar2 = np.array([[10, 20, 30], [40, 50, 60]])
ar2
array([[10, 20, 30],
       [40, 50, 60]])
ar3 = np.arange(1, 11, 2)
ar3
array([1, 3, 5, 7, 9])
ar4 = np.array([1, 2, 3, 4, 5, 6]).reshape((3, 2))
ar4
array([[1, 2],
       [3, 4],
       [5, 6]])
ar5 = np.zeros((2, 3))
ar5
array([[0., 0., 0.],
       [0., 0., 0.]])
ar6 = ar2[0:2, 0:2]
ar6
array([[10, 20],
       [40, 50]])
ar7 = ar2[0, :]
ar7
array([10, 20, 30])
ar8 = ar1 + 10
ar8
array([11, 12, 13, 14, 15])
ar1 + ar8
array([12, 14, 16, 18, 20])
ar8 - ar1
array([10, 10, 10, 10, 10])
ar1 * 2
array([ 2, 4, 6, 8, 10])
ar1 / 2
array([0.5, 1. , 1.5, 2. , 2.5])
ar9 = np.dot(ar2, ar4)
ar9
array([[220, 280],
       [490, 640]])
```

07. 데이터 분석을 위한 주요 라이브러리

■ list vs numpy array 프로그램(숫자데이터 경우) (listvsnumpy.py)

```
# list

myarray = [1, 2, 3]
print (type(myarray))           # 출력 "<type>
#print (mynumpy.shape)          #
print (myarray[0], myarray[1], myarray[2]) # 출력 "1 2 3"
myarray[0] = 5                  # 요소를 변경
print (myarray)                 # 출력 "[5, 2, 3]"
print (sum(myarray))
print (myarray.sort())          # sort
print (myarray*2)

yourarray = [[1,2,3],[4,5,6]]   # 원소 2개인 배열 생성
#print (yournumpy.shape)        #
print (yourarray[0])
print (yourarray[0][0], yourarray[0][1], yourarray[1][0])

x = [[1,2],[3,4]]
y = [[5,6],[7,8]]
print (x + y)
#print (add(x, y))
```

```
# numpy
import numpy as np
mynumpy = np.array([1, 2, 3])  # rank가 1인 배열 생성
print (type(mynumpy))           # 출력 "<type>
print (mynumpy.shape)           # 출력 "(3,)"
print (mynumpy[0], mynumpy[1], mynumpy[2])   # 출력
mynumpy[0] = 5                  # 요소를 변경
print (mynumpy)                 # 출력 "[5, 2, 3]"
print (np.sum(mynumpy))
print (np.sort(mynumpy))        # sort
print (mynumpy*2)

yournumpy = np.array([[1,2,3],[4,5,6]])   # rank가 2인 배열 생성
print (yournumpy.shape)         # 출력 "(2, 3)"
print (yourarray[0])
print (yournumpy[0, 0], yournumpy[0, 1], yournumpy[1, 0])

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]], dtype=np.float64)
print (x + y)
print (np.add(x, y))
```

http://aikorea.org/cs231n/python-numpy-tutorial/#numpy

07. 데이터 분석을 위한 주요 라이브러리

■ pandas

- Series 자료형

```
import pandas as pd
pd.__version__
'1.0.3'
data1 = [10, 20, 30, 40, 50]
data1
[10, 20, 30, 40, 50]
data2 = ['1반', '2반', '3반', '4반', '5반']
data2
['1반', '2반', '3반', '4반', '5반']
sr1 = pd.Series(data1)
sr1
0   10
1   20
2   30
3   40
4   50
dtype: int64
sr2 = pd.Series(data2)
sr2
0   1반
1   2반
2   3반
3   4반
4   5반
dtype: object
```

```
sr3 = pd.Series([101, 102, 103, 104, 105])
sr3
0   101
1   102
2   103
3   104
4   105
dtype: int64
sr4 = pd.Series(['월', '화', '수', '목', '금'])
sr4
0   월
1   화
2   수
3   목
4   금
dtype: object
```

07. 데이터 분석을 위한 주요 라이브러리

■ pandas

• Series 자료형

```
sr5 = pd.Series(data1, index = [1000, 1001, 1002, 1003, 1004])
sr5
1000   10
1001   20
1002   30
1003   40
1004   50
dtype: int64
sr6 = pd.Series(data1, index = data2)
sr6
1반   10
2반   20
3반   30
4반   40
5반   50
dtype: int64
sr7 = pd.Series(data2, index = data1)
sr7
10   1반
20   2반
30   3반
40   4반
50   5반
dtype: object
```

```
sr8 = pd.Series(data2, index = sr4)
sr8
월   1반
화   2반
수   3반
목   4반
금   5반
dtype: object
sr8[2]
'3반'
sr8['수']
'3반'
sr8[-1]
'5반'
sr8[0:4]
월   1반
화   2반
수   3반
목   4반
dtype: object
sr8.index
Index(['월', '화', '수', '목', '금'], dtype = 'object')
sr8.values
array(['1반', '2반', '3반', '4반', '5반'], dtype = object)
```

07. 데이터 분석을 위한 주요 라이브러리

■ pandas

- Series 자료형

```
sr1 + sr3
0    111
1    122
2    133
3    144
4    155
dtype: int64
sr4 + sr2
0    월1반
1    화2반
2    수3반
3    목4반
4    금5반
dtype: object
```

07. 데이터 분석을 위한 주요 라이브러리

■ pandas

- DataFrame 자료형

pandas.DataFrame

판다스 DataFrame은 다음과 같은 생성자를 사용해서 만들어진다.

	pandas.DataFrame(data, index, columns, dtype, copy)
data	ndarray, series, map, lists, dict, 상수 그리고 또 다른 DataFrame까지 변수의 형태로 가질 수 있다.
index	row 레이블의 경우, 인덱스 값이 없으면 0-based index (np.arange(n))가 기본으로 사용된다.
columns	column 레이블의 경우, 인덱스 값이 없으면 0-based index (np.arange(n))가 기본으로 사용된다.
dtype	각 column 마다 데이터 유형을 나타낸다.
copy	이 속성의 기본값이 False인 경우 데이터 복사에 사용된다.

pandas 형 데이터 파악하는 방법 : https://dsbook.tistory.com/11

python 공부하기 좋은 웹사이트

■ 파이썬 - 기본을 갈고 닦자! - https://wikidocs.net/book/1553