파이썬 프로그래밍 기초 - 1

최도진

목차

학습목표

- 데이터 분석에 필요한 파이썬 프로그래밍 기초를 이해한다.
- 파이썬 프로그래밍의 구성요소와 작성 방법 및 실행 방법을 안다.
- 데이터 분석에 주로 사용하는 파이썬 라이브러리를 안다.

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 = 1 + 2j
>>> a.real
1.0
>>> a.imag
2.0
>>> a.conjugate()
(1 - 2j)
>>> abs(a)
2.23606797749979
>>> a = 0o12
>>> a
10
>>> a = 0x12A
>>> a
298
```

- 논리형

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

- 사용 가능 연산자

```
>>> 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
```

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

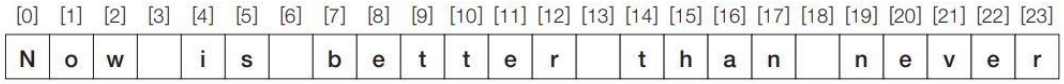

그림 4-2 문자열과 인덱스

- 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 연산자

```
>>> a.find('y')
1
>>> a.find('p')
-1
>>> 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
```

find = string
index =string/list/tuple

```
>>> 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

- 사용 가능 연산자

```
>>> 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

• 사용 가능 연산자

```
>>> 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

• 사용 가능 연산자

```
>>> 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

• 사용 가능 연산자

```
>>> 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

▪ if문

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

```
if 조건식:
    실행할 명령문 1

실행할 명령문 2
```

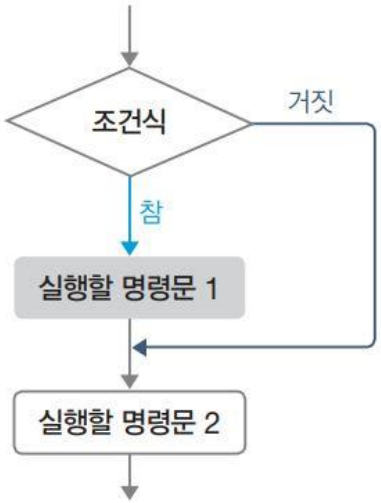

그림 4-4 if문의 구조와 제어 순서도

04. 조건문과 반복문

■ 조건문

▪ if-else문

- 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
```

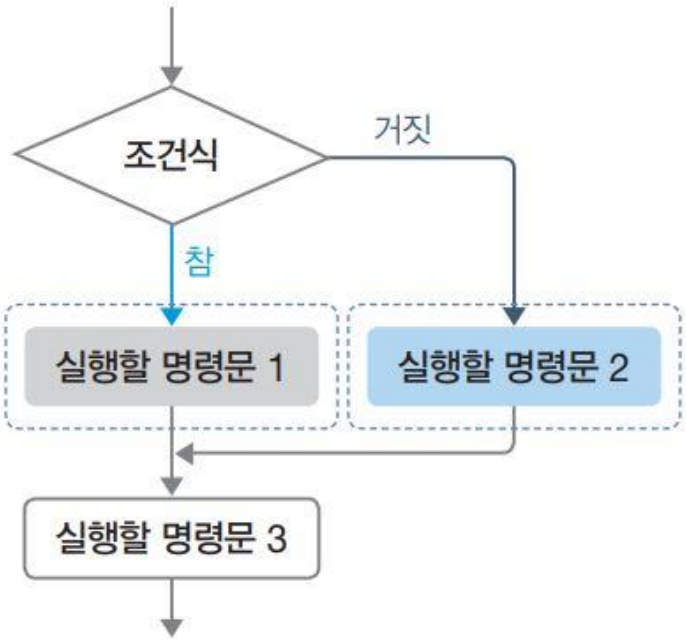

그림 4-5 if-else문의 구조와 제어 순서도

04. 조건문과 반복문

■ 조건문

▪ if-elif-else문

- 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
```

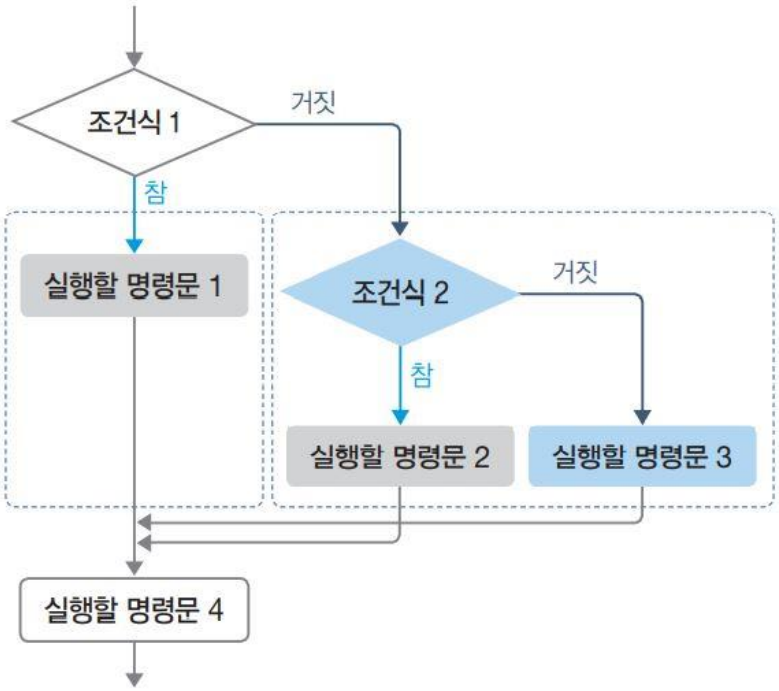

그림 4-6 if-elif-else문의 구조와 제어 순서도

04. 조건문과 반복문

■ 조건문

▪ 조건식

• 사용 가능 연산자

```
>>> 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

- for문

```
>>> 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.!
```

- while문

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


Q&A