빅데이터 처리 및 응용

제 3 강
파이썬의 개요

김 대 경
한양대학교 응용수학과

파이썬의 개요

차 례

- 파이썬 기초
- 파이썬 프로그래밍
  - 파이썬 프로그래밍의 구조와 제어문
  - 파이썬 프로그래밍의 입력과 출력
  - 파이썬 프로그래밍의 핵심: 클래스, 모듈

▶ 파이썬 프로그래밍의 구조와 제어문

자료형을 바탕으로 if, while, for 등의 제어문을 이용하여 프로그램의 구조를 만듦.

- ❖ if문
- ❖ while문
- ❖ for문

❖ if문

if문의 기본 구조

```
if 조건문:
    수행할 문장1
    수행할 문장2
...
else:
    수행할 문장A
    수행할 문장B
...
```

- 주의 사항
  - ✓ “if 조건문” 또는 “else” 뒤에 반드시 콜론 (:)을 붙임.
  - ✓ “if 조건문” 또는 “else” 아래 모든 문장에 들여쓰기를 해줌. (4칸 들여쓰기)

```
>>> money = 1
>>> if money:
...     print("택시를 타고가라!")
... else:
...     print("걸어가라!")
...
택시를 타고가라!
```

```
>>> if money:
... print("택시를 타고가라!")
... else:
...     print("걸어가라!")
...
  File "<input>", line 2
    print("택시를 타고가라!")
        ^
IndentationError: expected an
indented block
```

[조건문]

- 조건문은 참과 거짓을 판단하는 문장임.
- 자료형의 참과 거짓

자료형	참	거짓
숫자	0이 아닌 숫자	0
문자열	"abc"	" "
리스트	[1,2,3]	[ ]
튜플	(1,2,3)	( )
딕셔너리	{"a" : "b"}	{ }

- 비교 연산자

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x <= y	x가 y보다 작거나 같다
x >= y	x가 y보다 크거나 같다

- and, or, not

연산자	설명
x or y	x와 y 둘 중에 하나만 참이어도 참이다
x and y	x와 y가 모두 참이어야 참이다
not x	x가 거짓이면 참이다

```
>>> money = 2000
>>> if money >= 3000:
...     print("택시를 타고가라!")
... else:
...     print("걸어가라!")
...
걸어가라!
```

```
>>> money = 2000
>>> card = 1
>>> if money >= 3000 or card:
...     print("택시를 타고가라!")
... else:
...     print("걸어가라!")
...
택시를 타고가라!
```

- in, not in

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
>>> 1 in [1,2,3]
True
>>> 1 not in [1,2,3]
False
```

```
>>> 'a' in ('a','b','c')
True
>>> 'j' not in 'python'
True
```

예제

- "만약 주머니에 돈이 있으면 택시를 타고, 없으면 걸어가라"

```
>>> pocket = ['paper','cellphone','money']
>>> if 'money' in pocket:
...     print("택시를 타고가라!")
... else:
...     print("걸어가라!")
...
택시를 타고가라!
```

- pass
  - ✓ 조건문에서 아무 일도 하지 않게 설정하고자 하는 경우

예제

- "주머니에 돈이 있으면 가만히 있고 주머니에 돈이 없으면 카드를 꺼내라"

```
>>> pocket = ['paper','cellphone','money']
>>> if 'money' in pocket:
...     pass
... else:
...     print("카드를 꺼내라!")
...
```

- elif
  - ✓ else if의 합성

예제

- "주머니에 돈이 있으면 택시를 타고, 주머니에 돈은 없지만 카드가 있으면 택시를 타고, 돈도 카드도 없으면 걸어가라"

```
>>> pocket = ['paper','cellphone']
>>> card = 1
>>> if 'money' in pocket:
...     print("택시를 타고가라!")
... elif card:
...     print("택시를 타고가라!")
... else:
...     print("걸어가라!")
...
택시를 타고가라!
```

❖ while문

while문의 기본 구조

```
while 조건문:
    수행할 문장1
    수행할 문장2
    수행할 문장3
...
```

- 주의 사항
  - ✓ “while 조건문” 뒤에 반드시 콜론 (:)을 붙임.
  - ✓ “while 조건문” 아래 모든 문장에 들여쓰기를 해줌 (4칸 들여쓰기)

예제

- 여러 가지 선택지 중 하나를 선택하여 입력받는 예제

```
>>> prompt = """
... 1.Add
... 2.Del
... 3.List
... 4.Quit
...
... Enter number:"""
```

```
>>> number = 0
>>> while number != 4:
...     print(prompt)
...     number = int(input())
...
1.Add
2.Del
3.List
4.Quit

Enter number:
>? 1

1.Add
2.Del
3.List
4.Quit
```

```
Enter number:
>? 3

1.Add
2.Del
3.List
4.Quit
```

```
Enter number:
>? 4

>>>
```

- break
  - ✓ while문 강제로 빠져나가기

예제

- 커피 자판기 예제

```
>>> coffee = 10
>>> money = 300
>>> while money:
...     print("돈이 입력되었으니 커피를 줍니다.")
```

```
...     coffee = coffee -1
...     print("남은 커피는 %d개 입니다." %coffee)
...     if not coffee:
...         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
...         break
...
돈이 입력되었으니 커피를 줍니다.
남은 커피는 9개 입니다.
돈이 입력되었으니 커피를 줍니다.
남은 커피는 8개입니다.
.............................
돈이 입력되었으니 커피를 줍니다.
남은 커피는 1개 입니다.
돈이 입력되었으니 커피를 줍니다.
남은 커피는 0개 입니다.
커피가 다 떨어졌습니다. 판매를 중지합니다.
```

- continue
  - ✓ 조건에 맞지 않은 경우 맨 처음으로 돌아가기

예제

- 1부터 10까지의 숫자 중에서 홀수만 출력하는 while문 작성

```
>>> while a<10:
...     a = a+1
...     if a%2 == 0: continue
...     print(a)
...
1
3
5
7
9
```

- 무한 루프

```
>>> while True:
...     print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")
...
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Traceback (most recent call last):
  File "<input>", line 2, in <module>
KeyboardInterrupt
```

❖ for문

for문의 기본 구조

```
for 변수 in 리스트(또는 튜플, 문자열):
    수행할 문장1
    수행할 문장2
...
```

- 전형적인 for문

```
>>> test_list = ['one','two','three']
>>> for i in test_list:
...     print(i)
...
one
two
three
```

```
>>> a = [(1,2), (3,4), (5,6)]
>>> for (first, last) in a:
...     print(first+last)
...
3
7
11
```

- for문의 응용

예제1

- "총 5명의 학생이 처음 시험을 보았는데 시험점수가 60점이 넘으면 합격이고 그렇지 않으면 불합격이다. 합격인지 불합격인지 보여주시오"

```
# marks1.py
marks = [90, 25, 67, 45, 80]
number = 0 #학생에게 붙여 줄 번호
for mark in marks:
    number = number + 1
    if mark >=60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```

```
C:\Python\python.exe C:/Users/home/marks1.py
1번 학생은 합격입니다.
2번 학생은 불합격입니다.
3번 학생은 합격입니다.
4번 학생은 불합격입니다.
5번 학생은 합격입니다.

Process finished with exit code 0
```

예제2

- 총 5명의 수험생이 논술 시험을 보았는데 시험점수가 60점 이상인 수험생에게 합격 축하 메시지를 보내고 나머지 수험생에게는 아무런 메시지를 전하지 않는 프로그램을 에디터를 이용하여 작성하라.

```
# marks2.py
marks = [90, 25, 67, 45, 80]
number = 0 #학생에게 붙여 줄 번호
for mark in marks:
    number = number + 1
    if mark < 60: continue
    print("%d번 수험생 축하합니다. 합격입니다." % number)
```

```
C:\Python\python.exe C:/Users/home/marks2.py
1번 수험생 축하합니다. 합격입니다.
3번 수험생 축하합니다. 합격입니다.
5번 수험생 축하합니다. 합격입니다.

Process finished with exit code 0
```

- for와 함께 자주 사용하는 range함수
  - ✓ for문은 숫자 리스트를 자동으로 만들어 주는 range함수와 함께 사용되는 경우가 많음.

```
>>> a = range(10)
>>> a
range(0, 10)  ⇦ 0,1,2, ... ,9
```

```
>>> sum = 0
>>> for i in range(1,11):
...     sum = sum + i
...
>>> print(sum)
55
```

- 리스트 안에 for문 포함하기

```
>>> a = [1,2,3,4]
>>> result = []
>>> for num in a:
...     result.append(num*3)
...
>>> print(result)
[3, 6, 9, 12]
```

```
>>> result = [num*3 for num in a]
>>> print(result)
[3, 6, 9, 12]
```

```
>>> result = [x*y for x in range(2,10) for y in range(1,10)]
>>> print(result)
[2, 4, 6, 8, 10, 12, 14, 16, 18, 3, 6, 9, 12, 15, 18, 21, 24, 27,
4, 8, 12, 16, 20, 24, 28, 32, 36, 5, 10, 15, 20, 25, 30, 35, 40,
45, 6, 12, 18, 24, 30, 36, 42, 48, 54, 7, 14, 21, 28, 35, 42, 49,
56, 63, 8, 16, 24, 32, 40, 48, 56, 64, 72, 9, 18, 27, 36, 45, 54,
63, 72, 81]
```

▶ 파이썬 프로그래밍의 입력과 출력

함수, 입력과 출력, 파일 처리 방법을 다룸.
(입출력은 프로그래밍의 설계와 관련이 있음)

- ❖ 함수
- ❖ 사용자 입력과 출력
- ❖ 파일 읽고 쓰기

❖ 함수

왜 함수를 사용하는가?

- 같은 내용이 반복적으로 사용되는 가치 있는 부분을 한 뭉치로 묶어서 어떤 입력 값을 주었을 때 어떤 결과값을 돌려주는 객체를 구성하기 위함.
- 프로그램의 흐름을 잘 파악하기 위함.

파이썬 함수의 구조

```
def 함수명(입력 인수):
    수행할 문장1
    수행할 문장2
    ...
    return 결과값
```

예제

- 두 수를 더하는 함수

```
>>> def sum(a,b):
...     return a+b
...
>>> c = sum(3,4)
>>> c
7
```

- 입력값이 없는 함수

```
>>> def say():
...     return 'Hi!'
...
>>> a = say()
>>> a
'Hi!'
```

- 결과값이 없는 함수

```
>>> def sum(a, b):
...     print("%d, %d의 합입니다." %(a, b))
...     c = a+b
...
>>> a = sum(1,2)
1, 2의 합입니다.
>>> print(a)
None
```

- 입력값도 결과값도 없는 함수

```
>>> def say():
...     print('Hi!')
...
>>> a = say()
Hi!
>>> print(a)
None
```

- 여러 개의 입력값을 받는 함수 만들기

```
def 함수명(*입력변수):
    수행할 문장1
    수행할 문장2
    ...
    return 결과값
```

예제

- 여러 개의 입력값을 모두 더하는 함수 만들기.

```
>>> def sum_many(*args):
...     sum = 0
...     for i in args:
...         sum = sum + i
...     return sum
...
```

```
>>> a = sum_many(1,2,3)
>>> print(a)
6

>>> a = sum_many(1,2,3,4,5)
>>> print(a)
15
```

```
>>> def sum_mul(choice, *args):
...     if choice == "sum":
...         result = 0
...         for i in args:
...             result = result + i
...     elif choice == "mul":
...         result = 1
...         for i in args:
...             result = result*i
...     return result
...
>>> r = sum_mul('sum', 1,2,3,4,5)
>>> print(r)
15
>>> r = sum_mul('mul', 1,2,3,4,5)
>>> print(r)
120
```

- return의 특성
  - ✓ 함수가 return문을 받는 즉시 결과값을 돌려준 다음 함수를 빠져나감

```
>>> def sum_mul(a,b):
...     return a+b
...     return a*b
...
>>> r = sum_mul(3,4)
>>> print(r)
7
```

```
>>> def say_nick(nick):
...     if nick == "바보":
...         return
...     print("나는 %s입니다!" %nick)
...
>>> say_nick("천재")
나는 천재입니다!
>>> say_nick("바보")
>>>
```

- 입력인수에 초깃값 미리 설정하기

```
>>> def say_myself(name, old, man=True):
...     print("나의 이름은 %s입니다." %name)
...     print("나이는 %d살입니다." %old)
...     if man:
...         print("남자입니다.")
...     else:
...         print("여자입니다.")
...
>>> say_myself("홍길동", 27, True)
나의 이름은 홍길동입니다.
나이는 27살입니다.
남자입니다.
>>> say_myself("홍길동", 27, False)
나의 이름은 홍길동입니다.
나이는 27살입니다.
여자입니다.
```

- 함수 안에서 선언된 변수의 효력 범위
  - ✓ 함수 안에서 사용되는 변수는 함수 밖의 변수 이름들과 독립적인 전혀 상관없는 변수임.

```
>>> a = 1
>>> def vartest(a):
...     a = a +1
...     return a
...
```

```
>>> vartest(a)
2
>>> print(a)
1
```

  - ✓ 함수 안에서 함수 밖의 변수를 변경하는 방법

```
>>> a = 1
>>> def vartest(a):
...     a = a +1
...     return a
...
>>> a = vartest(a)
>>> print(a)
2
```

```
>>> a = 1
>>> def vartest():
...     global a
...     a = a +1
...
>>> vartest()
>>> print(a)
2
```

❖ 사용자 입력과 출력

사용자 입력 ⇒ 처리(프로그램, 함수 등) ⇒ 출력

사용자 입력

- input(), input("질문 내용")의 사용

```
>>> a = input()
Life is too short!
>>> print(a)
'Life is too short!'
```

```
>>> number = input("숫자를 입력하시오:")
숫자를 입력하시오:3
>>> print(number)
3
```

print 더 알기

- 큰따옴표("")로 둘러싸인 문자열은 + 연산과 동일

```
>>> print("Life" "is" "too" "short!")
Lifeistooshort!
>>> print("Life"+"is"+"too"+"short!")
Lifeistooshort!
```

- 문자열 띄어쓰기는 콤마로 함.

```
>>> print("Life","is","too","short!")
Life is too short!
```

❖ 파일 읽고 쓰기

파일 객체 = open(파일 이름, 파일 접근 모드)

파일 접근 모드	설명
r	읽기 모드 – 파일을 읽기만 할 때 사용
w	쓰기 모드 – 파일에 내용만 쓸 때 사용
a	추가 모드 – 파일의 마지막에 새로운 내용을 추가할 때 사용

파일 생성하기

```
>>> f = open("새파일.txt", 'w')
>>> f.close()
```

```
C:\Users\home\untitled> dir

2020-09-07  오후 05:51    <DIR>          .
2020-09-07  오후 05:51    <DIR>          ..
2020-09-07  오후 05:54    <DIR>          .idea
2020-08-28  오후 03:24               423 hello.py
2020-09-06  오후 04:18               283 marks1.py
2020-09-07  오후 04:10                40 marks2.py
2020-08-27  오후 12:21    <DIR>          venv
2020-09-07  오후 05:51                 0 새파일.txt
               4개 파일                 746 바이트
               4개 디렉터리  843,352,977,408 바이트 남음
```

```
>>> f = open("C:/Users/home/untitled/새파일.txt", 'w')
>>> f.close()
```

- 파일을 쓰기 모드로 열어 출력값 적기

```
>>> f = open("새파일.txt", 'w')
>>> for i in range(1,6):
...     data = "%d번째 줄입니다.\n" %i
...     f.write(data)
...
>>> f.close()
```

```
C:\Users\home\untitled> dir

 ..............................................
2020-09-07  오후 06:02                85 새파일.txt
               4개 파일                 831 바이트
               4개 디렉터리  843,352,768,512 바이트 남음
```

```
C:\Users\home\untitled> type 새파일.txt
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
```

- 프로그램의 외부에 저장된 파일을 읽기
  (1) readline() 함수 이용하기
  (2) readlines() 함수 이용하기
  (3) read() 함수 이용하기

(1) readline() 함수 이용하기

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

>>> f.close()
```

```
>>> f = open("새파일.txt",'r')
>>> while True:
...     line = f.readline()
...     if not line: break
...     print(line)
...
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
>>> f.close()
```

(2) readlines() 함수 이용하기

```
>>> f = open("새파일.txt",'r')
>>> lines = f.readlines()
>>> for line in lines:
...     print(line)
...
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
>>> f.close()
```

```
>>> print(lines)
['1번째 줄입니다.\n',
'2번째 줄입니다.\n',
'3번째 줄입니다.\n',
'4번째 줄입니다.\n',
'5번째 줄입니다.\n',]
```

(3) read() 함수 이용하기

```
>>> f = open("새파일.txt",'r')
>>> data = f.read()
>>> print(data)
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
>>> f.close()
```

- 파일에 새로운 내용 추가하기

```
>>> f = open("새파일.txt",'a')
>>> for i in range(6,11):
...     data = "%d번째 줄입니다.\n" % i
...     f.write(data)
...
>>> f.close()
```

```
C:\Users\home\untitled> type 새파일.txt
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
```

- with문과 함께 사용하기
  - ✓ 자동으로 close하기

```
>>> f = open("foo.txt", "w")
>>> f.write("Life is too short!")
>>> f.close()
```

```
>>> with open("foo.txt", "w") as f:
...     f.write("Life is too short!")
...
```