파이썬

강현주

1. 파이썬 배우기

파이썬 소개와 IDE

1.1 파이썬 소개

1.1.1 파이썬이란

- 1989년 파이썬의 창시자 귀도 반 로섬(Guido van Rossum)이 크리스마스에 휴가를 보내며 개발
- BBC에서 방영되던 Monty Python's Flying Circus 라는 TV프로그램명을 따서 이름을 지음

오픈소스 + 영어와 유사한 문법 + 컴파일 필요 없음 + 간단하고 배우기 쉽다

프로그래밍 입문자에게 최적의 언어

1.1 파이썬 소개

1.1.2 파이썬의 특징

Simple & Easy to learn

Free & Open source

High-level language

Interpreted

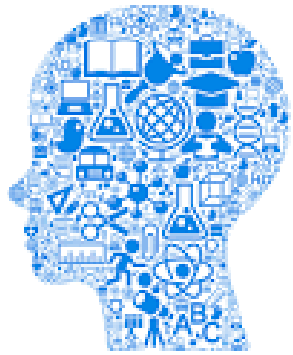
Object Oriented

Embeddable

Portable

Extensible

Dynamic type



교육용, 데이터 과학 분야에 적합한 언어

1.1 파이썬 소개

1.1.3 파이썬의 특징

- Interpreted Language vs Compiled Language

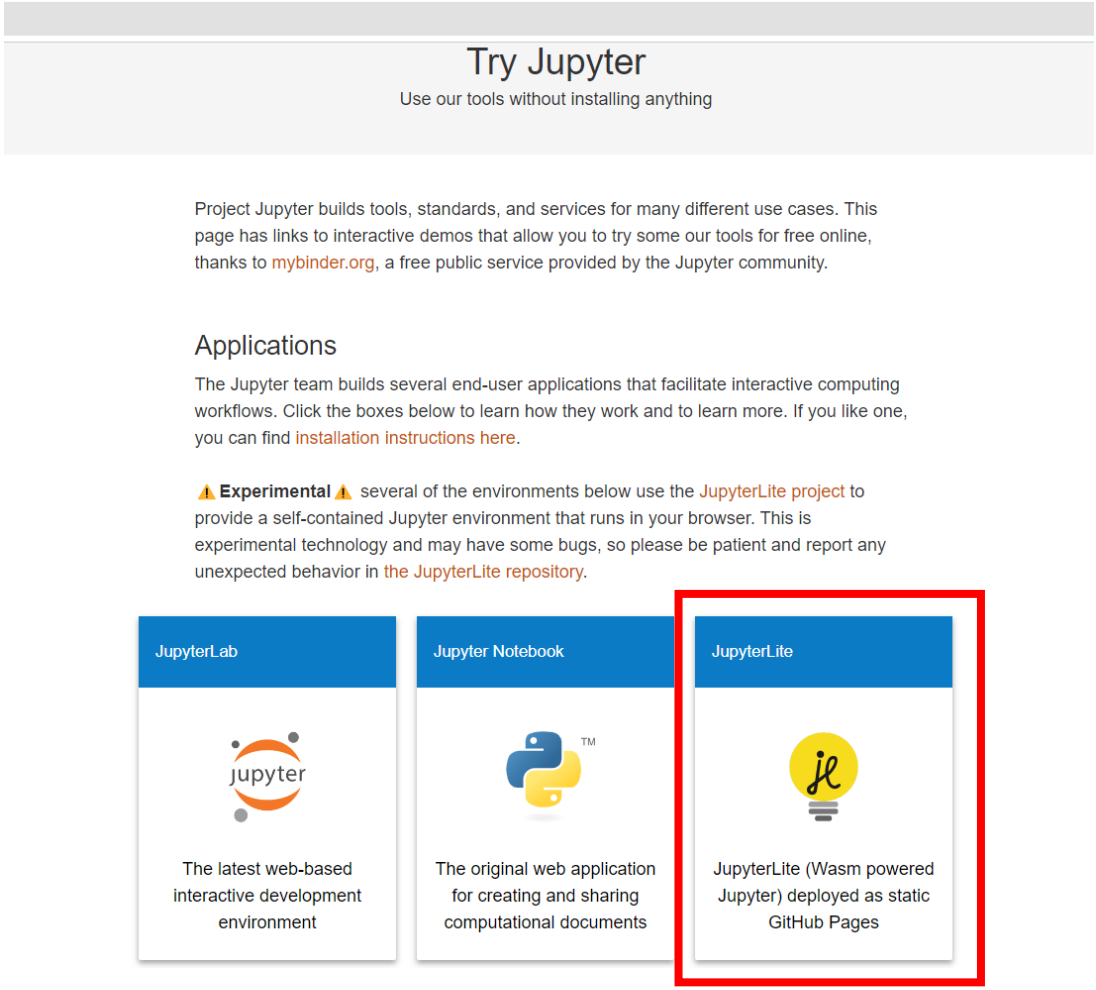
Interpreted Language	VS	Compiled Language
별도의 번역 과정 없이 소스코드를 실행시점에 해석 처리	**방식**	소스코드를 기계어로 먼저 번역
간단히 작성 가능 메모리를 적게 필요	**장점**	실행 속도가 빠름
실행 속도가 느림	**단점**	한번에 많은 기억장소가 필요함
파이썬, 스칼라, Perl, GW-Basic 등	**사용 언어**	C, 자바, C++, C# 등

PC성능이 향상됨에 따라 인터프리터 언어 선호도 증가

1.2 파이썬 IDE

1.2.1 파이썬 개발환경

- Jupyter Notebook : https://jupyter.org/try
- 설치 없이 웹 환경에서 파이썬 프로그래밍



1.2 파이썬 IDE

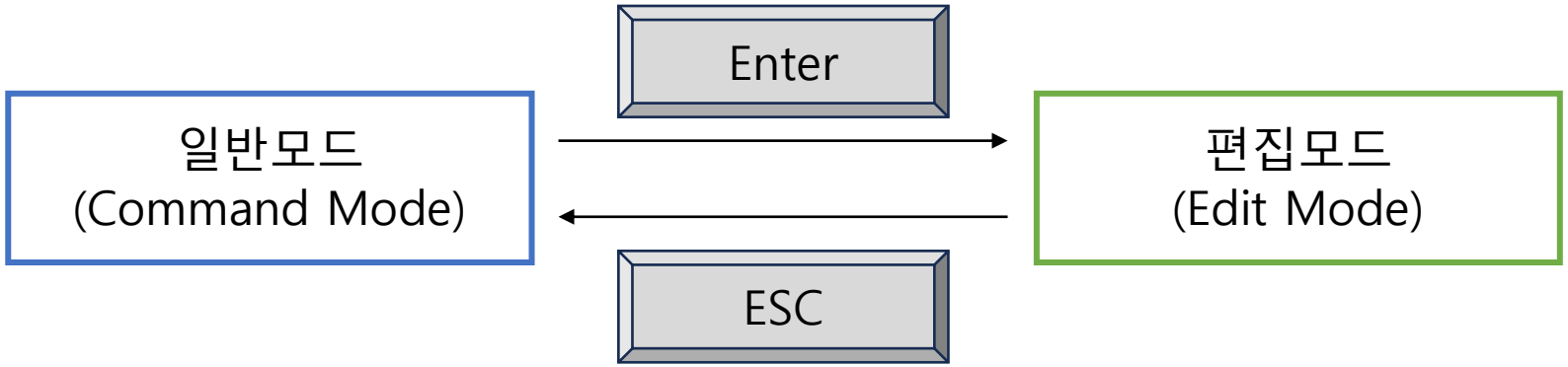
1.2.1 파이썬 개발환경

- Jupyter Notebook : https://jupyter.org/try
- 설치 없이 웹 환경에서 파이썬 프로그래밍

1.2 파이썬 IDE

1.2.2 파이썬 개발환경

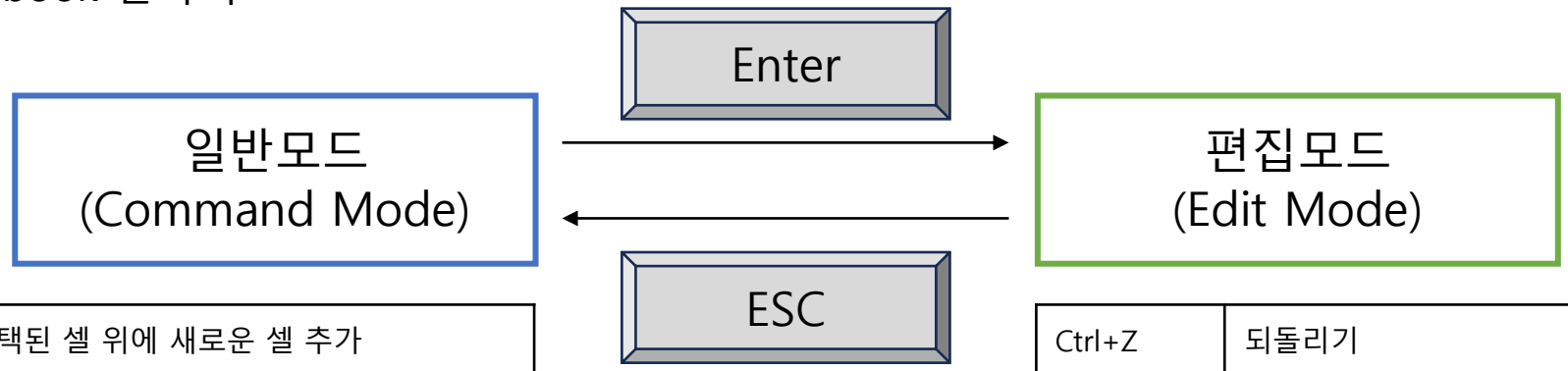
- Jupyter Notebook : https://jupyter.org/try
- 일반모드(Command Mode) : 셀 선택, 셀 추가, 셀 삭제
- 편집모드(Edit Mode) : 코드 작성, 셀 편집



1.2 파이썬 IDE

1.2.2 파이썬 개발환경

- Jupyter Notebook 단축키



A	선택된 셀 위에 새로운 셀 추가
B	선택된 셀 아래에 새로운 셀 추가
X	선택된 셀 잘라내기
C	선택된 셀 복사하기
V	선택된 셀 아래에 붙여넣기
Shift+V	선택된 셀 위에 붙여넣기
D D	선택된 셀 삭제하기
Z	셀 삭제 취소
Ctrl+Enter	선택된 셀 실행
Shift+Enter	선택된 셀 실행 후 아래의 셀로 이동
Alt+Enter	선택된 셀 실행 후 아래에 새로운 셀 추가

Ctrl+Z	되돌리기
Ctrl+Y	되돌리기 취소
Ctrl+A	셀 내용 전체 선택
Ctrl+X	선택한 내용 잘라내기
Ctrl+C	선택한 내용 복사하기
Ctrl+V	선택한 내용 붙여넣기
Ctrl+Home	셀 맨위로 이동하기
Ctrl+End	셀 맨아래로 이동하기
Ctrl+D	현재 줄 삭제
Tab	자동완성
Shift+Tab	도움말 툴팁

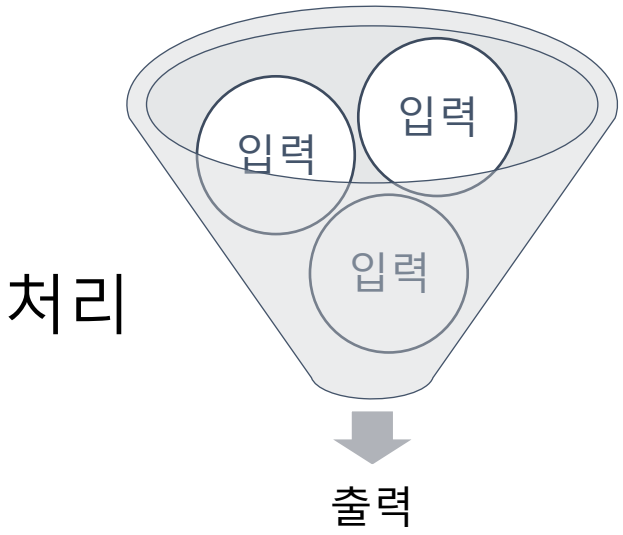
2. 파이썬 배우기

변수와 문자열

2. 파이썬 변수와 문자열

2.1.0 파이썬 예제 프로그램

- 프로그램이란?
  - 컴퓨터에게 내리는 일련의 명령



- **입력**을 받는다 → 입력을 **처리**한다 → **출력**을 만들어낸다

2. 파이썬 변수와 문자열

2.1.0 들여쓰기 (Indentation)

- 파이썬은 프로그래밍 언어의 기본 들여쓰기를 이례적으로 활용

예제코드 count.py

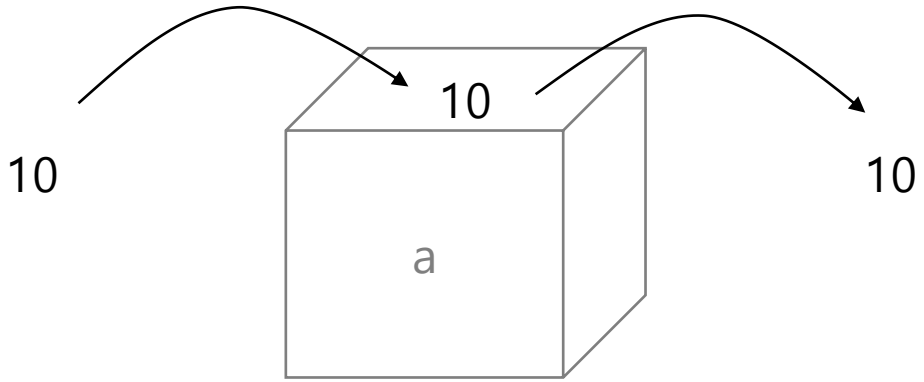
```
for i in range(1,10):
    print(i)
```

- C 기반의 언어 → 코드 블록 구분자로 '{' 와 '}'를 사용
- 파이썬 → 들여쓰기 수준으로 코드 블록을 구분 (통상적으로 기본 공백 네칸)
- 'IndentationError : unexpected indent' 에러 → 제대로 들여쓰기 되지 않은 경우 발생

2. 파이썬 변수와 문자열

2.1.1 메모리와 변수

- 메모리 : 입력값 또는 프로그램 자체를 기억하는 장치
- 변수 : 메모리 내에 입력 받은 내용을 보관하는 장소
- 변수명 : 보관할 장소를 만들어 이름을 주고, 다시 사용하고 싶을 때 같은 이름을 사용



2. 파이썬 변수와 문자열

2.1.2 변수 사용하기

예제코드

```
a=123
b=12.34
c="Hello"
d='Hello'
e=True
```

- 대입 연산자(=) 를 사용하여 이름에 값을 할당
- 변수 타입 선언할 필요 없이, 바로 값을 할당

- 숫자형 – 정수형과 소수점 숫자형(실수형)
- 문자열 상수 – 큰따옴표(“”)나 작은따옴표(‘’)를 사용해서 정의
- 논리 상수 - True , False 대소문자 구분

- 변수이름은 소문자로 시작, 변수이름이 두 단어 이상이면 밑줄로 단어 사이를 연결
  예) x, total, number_of_chars

2. 파이썬 변수와 문자열

2.1.2 변수 출력하기

예제코드

```
x=10
print(x)
```

결과확인

```
10
```

- 파이썬 2에서는 print 명령어에 괄호가 없어도 사용 가능
- 파이썬 3에서는 반드시 괄호 사용

2. 파이썬 변수와 문자열

2.1.3 사용자 입력 받기

예제코드

```
x=input("숫자를 입력하세요 :")
print(x)
```

결과확인

```
숫자를 입력하세요 :25
25
```

- 파이썬 2에서는 input 대신 raw_input 명령어를 사용

2. 파이썬 변수와 문자열

2.1.4 산술 연산 (+, -, *, /, %, **)

예제코드

```
tempC=27
tempF=(tempC*9)/5+32
print(tempF)
```

결과확인

```
80.6
```

예제코드

```
print(2**8) # 2^8은 2**8로 표현
```

결과확인

```
256
```

- % : 나머지 연산자
- ** : 거듭제곱 연산자

- Math 모듈 : 유용한 수학 연산 함수 사용 가능 (https://wikidocs.net/21116)

2. 파이썬 변수와 문자열

2.1.5 문자열 만들기 (대입연산자, 문자열 상수 이용)

예제코드

```
s="abc def"
print(s)
```

결과확인

```
abc def
```

예제코드

```
s="Isn't it cold?"
print(s)
```

결과확인

```
Isn't it cold?
```

- 문자열 안에 큰따옴표나 작은따옴표가 포함되어야 할 경우, 서로 다른 부호를 사용
- 이스케이프 문자 – 문자열 내부에 탭(₩t), 개행 문자(₩n) 등 특수문자가 포함될 때 사용

2. 파이썬 변수와 문자열

2.1.6 문자열 합치기 (연결연산자 + 사용)

예제코드 connect_string.py

```
s1 = "abc"
s2 = "def"
s = s1 + s2
print(s)
```

결과확인

```
abcdef
```

예제코드

```
s = "abc" + str(123)
print(s)
```

결과확인

```
abc123
```

- 파이썬은 숫자를 문자열로 자동 변환하는 기능을 지원하지 않음

  → 문자열로 연결하기 전에 각각의 요소를 문자열로 변환해야 함

2. 파이썬 변수와 문자열

2.1.7 숫자와 문자열 변환 (casting)

예제코드 casting_types.py

```
print(str(123))
```

결과확인

```
123
```

예제코드

```
print(int("-123"))
```

결과확인

```
-123
```

예제코드

```
print("00123.45")
```

결과확인

```
00123.45
```

예제코드

```
print(float("00123.45"))
```

결과확인

```
123.45
```

예제코드

```
print(int("1001",2))
```

결과확인

```
9
```

예제코드

```
print(int("AFF0",16))
```

결과확인

```
45040
```

2. 파이썬 변수와 문자열

2.1.8 문자열 다루기

예제코드

```
print(len("abcdef"))
```

결과확인

```
6
```

예제코드

```
s="abcdefghi"
print(s.find("def"))
```

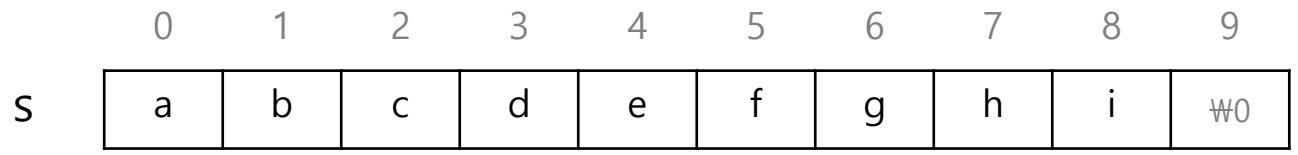
결과확인

```
3
```

- find() 함수 → 전체 문자열에서 문자의 위치는 0부터 시작
  → 찾으려는 문자열이 존재하지 않을 경우, -1 반환
- 포함된 모든 항목을 찾아 교체하려면 replace() 함수 사용

2. 파이썬 변수와 문자열

2.1.9 문자열 다루기 (슬라이싱)



예제코드

```
s="abcdefghi"
print(s[1:5])    # 1의 위치부터 (5-1)의 위치까지의 문자열
print(s[:5])     # 처음부터 (5-1)의 위치까지의 문자열
print(s[3:])     # 3의 위치부터 끝까지의 문자열
print(s[-3:])    # 끝에서 3번째부터 끝까지의 문자열
print(s[3:-3])   # 3위치부터 (끝에서 3번째-1)의 위치까지의 문자열
```

결과확인

```
bcde
abcde
defghi
ghi
def
```

2.1.10 문자열 다루기

예제코드

```
s="Every cat has his day"
print(s.replace("cat", "dog"))
```

결과확인

```
Every dog has his day
```

- 대소문자, 공백 포함 여부를 구분하여 정확히 일치하는 문자열만 검색한다.

2.1.11 문자열 다루기

예제코드

```
print("aBcDe".upper())
print("aBcDe".lower())

s="aBcDe"
print(s.upper())
print(s.lower())
print(s)
```

결과확인

```
ABCDE
abcde
ABCDE
abcde
aBcDe
```

- 문자열을 변경하는 함수 대부분은 원본 문자열을 변경하지 않고, 원본 문자열의 복사본을 수정해 반환한다.

2. 파이썬 변수와 문자열

2.1.12 수치형식 사용하기 format( )

예제코드

```
x=1.2345678
print("x={:.2f}".format(x))      # 소수 두자리 소수로 표현
print("x={:7.2f}".format(x))     # 앞에 공백 세개 추가됨
```

결과확인

```
x=1.23
x=   1.23
```

2.1.13 날짜형식 사용하기 (https://docs.python.org/3/library/string.html#formatspec)

예제코드

```
from datetime import datetime
d = datetime.now()

print(d)
print("{:%Y-%m-%d %H:%M:%S}".format(d))
```

결과확인

```
2020-09-30 23:04:52.526429
2020-09-30 23:04:52
```

2.1.14 주석코드

예제코드

```
# 한 줄 주석입니다


"""
여러 줄에 걸친 주석입니다
"""
```

- 프로그램 동작에는 영향을 주지 않지만, 개발과 유지보수를 쉽게 해줌
- 단축키 : Ctrl + /

3. 파이썬 배우기

조건문과 반복문

3. 파이썬 조건문과 반복문

3.1.1 조건문

- if문

예제코드

```
x = 101
if x > 100:    # 조건의 결과가 참(True)인 경우, 들여쓰기 라인 실행
    print("x is bigger than 100")
```

결과확인

```
x is bigger than 100
```

3. 파이썬 조건문과 반복문

3.1.2 조건문

- if ~ else 문

예제코드

```
x = 101
if x > 100:
    print("x is bigger than 100")
else:
    print("x is smaller than 100")

print("Bye")
```

결과확인

```
x is bigger than 100
Bye
```

3. 파이썬 조건문과 반복문

3.1.3 조건문

- if ~ elif ~ else 문

예제코드

```
x = 8
if x > 100:
    print("x is too big")
elif x < 10:
    print("x is too small")
else:
    print("x is normal")

print("Bye")
```

결과확인

```
x is too small
Bye
```

3. 파이썬 조건문과 반복문

3.2.1 연산자

- 값 비교하기 (비교연산자 < , > , <= , >= , == , !=)

예제코드

```
print("10!=20 :", 10!=20)
print("10!=10 :", 10!=10)
print("10>=10 :", 10>=10)
print("10>=11 :", 10>=11)
print("10==10 :", 10==10)
```

결과확인

```
10!=20 : True
10!=10 : False
10>=10 : True
10>=11 : False
10==10 : True
```

예제코드

```
print("aa < ab :", 'aa' < 'ab')
print("aa < aaa :", 'aa' < 'aaa')
print("aaa < aa :", 'aaa' < 'aa')
print("a < A :", 'a' < 'A')
print("A < a :", 'A' < 'a')
```

결과확인

```
aa < ab : True
aa < aaa : True
aaa < aa : False
a < A : False
A < a : True
```

- != 는 <> 연산자와 동일하게 동작
- 문자열은 사전에서 검색하는 순서로 비교됨 (대문자 < 소문자)

3. 파이썬 조건문과 반복문

3.2.2 연산자

- 논리 연산자 (and, or, not)

예제코드

```
x = 17
if x>=10 and x<=20:
    print('x is between 10 and 20')
```

결과확인

```
x is between 10 and 20
```

3. 파이썬 조건문과 반복문

3.3.1 반복문

- for ~ in 문

예제코드

```
for i in range(1,11):    # 1부터 (11-1)까지
    print(i)
```

결과확인

```
1
2
3
4
5
6
7
8
9
10
```

3. 파이썬 조건문과 반복문

3.3.2 반복문

- while 문

예제코드

```
i = 1
while i != 11:
    print(i)
    i += 1
```

결과확인

```
1
2
3
4
5
6
7
8
9
10
```

3. 파이썬 조건문과 반복문

3.3.3 반복문

- 반복문 나가기 (break 문)

예제코드

```
i = 1
while i != 11:
    print(i)
    i += 1
    if i == 5:
        break
```

결과확인

```
1
2
3
4
```

4. 파이썬 배우기

함수와 구조적 타입

4. 파이썬 함수와 구조적 타입

4.1.1 함수

- 함수 정의하기 (def 함수명:)

예제코드

```
# 1부터 5까지 출력하는 함수 정의
def count_to_5():
    for i in range(1,6):
        print(i)

# 함수 호출
count_to_5()
```

결과확인

```
1
2
3
4
5
```

예제코드

```
# 1부터 n까지 출력하는 함수 정의
def count_to_n(n):
    for i in range(1,n+1):
        print(i)

# 함수 호출
count_to_n(5)
```

결과확인

```
1
2
3
4
5
```

- 동일한 코드를 반복해서 입력하는 것을 피하고, 다양한 위치에서 재사용

4.1.2 함수

- 함수 정의하기 (def 함수명:)

예제코드

```
# 매개변수 기본값
def count_to_n(n=5):
    for i in range(1,n+1):
        print(i)

# 함수 호출
count_to_n()
```

결과확인

```
1
2
3
4
5
```

예제코드

```
# 매개변수에 값을 넣을 경우
def count_to_n(n=5):
    for i in range(1,n+1):
        print(i)

# 함수 호출
count_to_n(7)
```

결과확인

```
1
2
3
4
5
6
7
```

4.1.3 함수

- 함수 정의하기 (def 함수명:)

예제코드

```
# 두개 이상의 매개변수가 필요한 함수
def count(start_n=1, end_n=5):
    for i in range(start_n,end_n+1):
        print(i)

# 함수 호출
count()
```

결과확인

```
1
2
3
4
5
```

예제코드

```
# 두개 이상의 매개변수가 필요한 함수
def count(start_n=1, end_n=5):
    for i in range(start_n,end_n+1):
        print(i)

# 함수 호출시 매개변수 한 개만 입력한 경우
count(3)
```

결과확인

```
3
4
5
```

4.1.4 함수

- return 문

예제코드

```
def make_polite(sentence):
    return sentence + ', please'

print(make_polite("Pass the pencil"))
```

결과확인

```
Pass the pencil, please
```

  - 함수가 값을 반환해야 하는 경우 return 문 사용

4. 파이썬 함수와 구조적 타입

4.2.1 리스트(list)

- 리스트 생성하기

예제코드

```
a = [40, 'Joy', 28, False, 10.8]
b = []

print(a)
print(b)
```

결과확인

```
[40, 'Joy', 28, False, 10.8]
[]
```

- 리스트는 여러 값의 모음을 순서대로 저장하고, 저장한 위치를 통해 접근할 수 있는 자료구조
- '['와 ']' 사이에 초기값을 넣어 생성
- 리스트의 각각의 요소가 동일한 타입을 가질 필요는 없음
- 빈 리스트를 만든 후에 요소 추가 가능

4.2.2 리스트(list)

- 리스트 요소에 접근하기

	0	1	2	3	4
a	40	'Joy'	28	False	10.8

예제코드

```
a = [40, 'Joy', 28, False, 10.8]
print(a)

a[1]='Kang'
print(a)
```

결과확인

```
[40, 'Joy', 28, False, 10.8]
[40, 'Kang', 28, False, 10.8]
```

4.2.3 리스트(list)

- 리스트의 길이 파악하기 len( )

예제코드

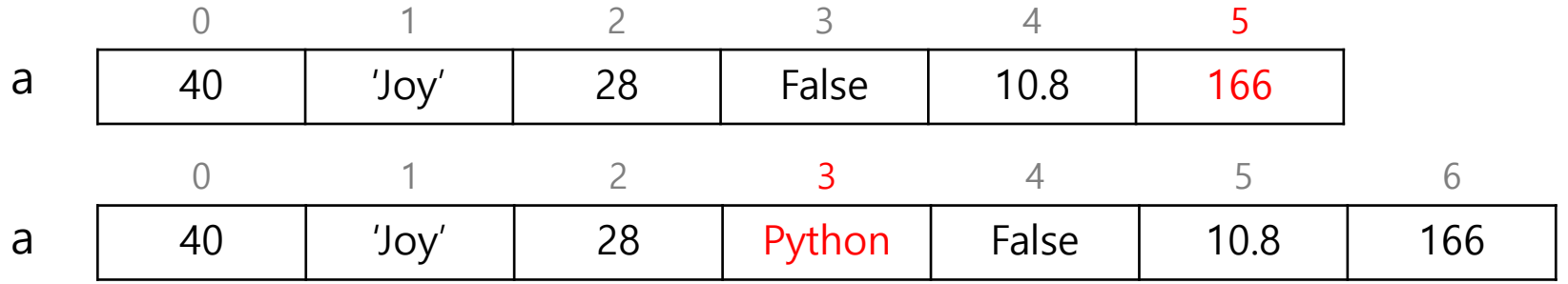
```
a = [40, 'Joy', 28, False, 10.8]

print(len(a))
```

결과확인

```
5
```

4.2.4 리스트(list)

- 리스트에 새로운 요소 추가하기 append( ) , insert( )



예제코드

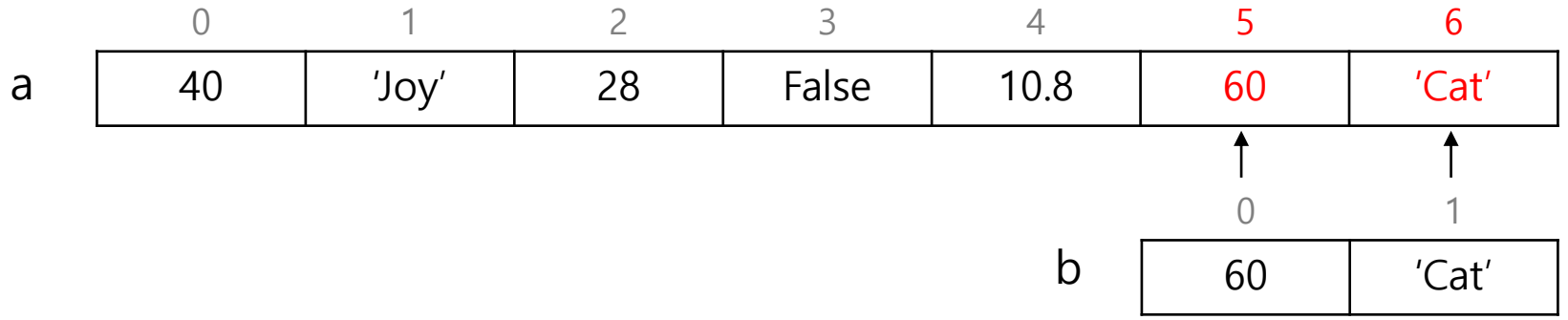
```
a = [40, 'Joy', 28, False, 10.8]
a.append(166)              #리스트의 끝에 새로운 요소 하나를 추가
print(a)

a.insert(3, 'Python')    #리스트의 특정 위치에 새로운 요소를 추가
print(a)
```

결과확인

```
[40, 'Joy', 28, False, 10.8, 166]
[40, 'Joy', 28, 'Python', False, 10.8, 166]
```

4.2.5 리스트(list)

- 리스트에 새로운 요소 추가하기 extend( )



예제코드

```
a = [40, 'Joy', 28, False, 10.8]
b = [60, 'Cat']

a.extend(b)              #리스트의 끝에 또 다른 리스트의 모든 요소를 추가
print(a)
```

결과확인

```
[40, 'Joy', 28, False, 10.8, 60, 'Cat']
```

4.2.6 리스트(list)

- 리스트에서 요소 제거하기 pop( )

	0	1	2	3	4
a	40	'Joy'	28	False	10.8

예제코드

```
a = [40, 'Joy', 28, False, 10.8]

print(a.pop())  # 리스트에서 제거된 값을 반환
print(a)        # pop으로 마지막 요소가 제거된 후의 리스트

print(a.pop(0)) # 리스트에서 특정 위치의 값을 제거
print(a)        # 특정 위치의 요소가 제거된 후의 리스트
```

결과확인

```
10.8
[40, 'Joy', 28, False]
40
['Joy', 28, False]
```

4.2.7 리스트(list)

- 문자열 분리해서 리스트에 넣기 split( )

예제코드

```
print("abc def ghi".split())
print("ab-c d-ef gh-i".split('-'))
print("a/b/c/d/e/f//g/h/i".split('/'))
```

결과확인

```
['abc', 'def', 'ghi']
['ab', 'c d', 'ef gh', 'i']
['a', 'b', 'c', 'd', 'e', 'f', '', 'g', 'h', 'i']
```

- 구분자(delimiter)를 이용하여 문자열을 분리한 후 각각의 요소로 구분하여 리스트에 넣음
- 파일에서 데이터를 가져오거나 특정 문자로 문자열을 분리할 때 유용함
- 매개변수가 없으면 공백(space)문자로 분리

4.2.8 리스트(list)

- 리스트 요소 반복하기

예제코드

```
a = [40, 'Joy', 28, False, 10.8]
for x in a:
    print(x)
```

결과확인

```
40
Joy
28
False
10.8
```

- 반복할 때마다 x는 리스트 순서대로 각 요소에 해당하는 값을 갖는다.

4.2.9 리스트(list)

- 리스트 열거하기 enumerate( )

예제코드

```
a = [40, 'Joy', 28, False, 10.8]
for (i,x) in enumerate(a):
    print(i, x)
```

결과확인

```
0 40
1 Joy
2 28
3 False
4 10.8
```

예제코드

```
a = [40, 'Joy', 28, False, 10.8]
for i in range(len(a)):
    print(i, a[i])
```

결과확인

```
0 40
1 Joy
2 28
3 False
4 10.8
```

4.2.10 리스트(list)

- 리스트 정렬하기 sort( )

예제코드

```
a = ['dog', 'cat', 'rabbit', 'duck']
a.sort()
print(a)
```

결과확인

```
['cat', 'dog', 'duck', 'rabbit']
```

예제코드

```
import copy

a = ['dog', 'cat', 'rabbit', 'duck']
b = copy.copy(a)
print('b = ', b)

b.sort()
print('b = ', b)

print('a = ', a)
```

결과확인

```
b =  ['dog', 'cat', 'rabbit', 'duck']
b =  ['cat', 'dog', 'duck', 'rabbit']
a =  ['dog', 'cat', 'rabbit', 'duck']
```

4.2.11 리스트(list)

- 리스트 나누기

예제코드

```
a_list=["a", "b", "c", "d"]
print(a_list)
print("[1:3] = ",a_list[1:3])
print("[:3] = ",a_list[:3])
print("[3:] = ",a_list[3:])
print("[-2:] = ",a_list[-2:])
print("[:-2] = ",a_list[:-2])
print("[1:-2] = ",a_list[1:-2])
```

결과확인

```
['a', 'b', 'c', 'd']
[1:3] =  ['b', 'c']
[:3] =  ['a', 'b', 'c']
[3:] =  ['d']
[-2:] =  ['c', 'd']
[:-2] =  ['a', 'b']
[1:-2] =  ['b']
```

4.2.12 리스트(list)

- 리스트에 함수 적용하기 (list comprehensions)

예제코드

```
list_a = ['abc','def','ghi','jkl']
list_b = [x.upper() for x in list_a]

print(list_b)
print(list_a)
```

결과확인

```
['ABC', 'DEF', 'GHI', 'JKL’]
['abc', 'def', 'ghi', 'jkl']
```

- 리스트의 각 요소에 함수를 적용하고 결과를 수집할 때 리스트 내장(list comprehensions) 기능을 사용
- 단, 하나의 리스트 내장구문 안에 다른 내장구문을 중첩 사용 불가

4. 파이썬 함수와 구조적 타입

4.3.1 사전(dictionary)

- 사전(dictionary) 구조체 만들기

예제코드

```
phone_num = {'Emily':'010-234-5678',
             'Eric':'010-345-6789',
             'Ellie':'010-456-7890'
            }

print(phone_num)
```

phone_num

Key:Emily	010-234-5678
Key:Eric	010-345-6789
Key:Ellie	010-456-7890

결과확인

```
{'Emily': '010-234-5678', 'Eric': '010-345-6789', 'Ellie': '010-456-7890'}
```

- 키와 값을 연결하는 lookup table로 사전 전체를 검색하지 않고 키를 이용해 효과적으로 값을 검색
- 키는 일반적으로 문자열 사용, 숫자 또는 다른 데이터 타입 사용 가능
- 값은 다른 사전을 포함하여 어떠한 데이터 타입을 사용해도 상관 없음

4.3.2 사전(dictionary)

- 사전(dictionary) 구조체 만들기

예제코드

```
dic_a = {'Key1': 1, 'key2':'value2'}
dic_b = {'b_Key1':dic_a}
print(dic_a)
print(dic_b)
```

결과확인

```
{'Key1': 1, 'key2': 'value2'}
{'b_Key1': {'Key1': 1, 'key2': 'value2'}}
```

- 키는 일반적으로 문자열 사용, 숫자 또는 다른 데이터 타입 사용 가능
- 값은 다른 사전을 포함하여 어떠한 데이터 타입을 사용해도 상관 없음

4.3.3 사전(dictionary)

- 사전(dictionary) 요소에 접근하기

예제코드

```
phone_num={'Emily':'010-234-5678',
           'Eric':'010-345-6789',
           'Ellie':'010-456-7890'
           }
print(phone_num['Emily'])
phone_num['Emily']='010-123-4567'
print(phone_num)
```

결과확인

```
010-234-5678
{'Emily': '010-123-4567', 'Eric': '010-345-6789', 'Ellie': '010-456-7890'}
```

- 키 값을 사용 접근해서 새로운 값을 추가하거나 기존 값 수정 가능
- 해당키가 존재하지 않으면 새로운 요소가 자동으로 추가, 존재하면 기존값이 새 값으로 수정됨

4.3.4 사전(dictionary)

- 사전의 요소 제거하기

예제코드

```
phone_num={'Emily':'010-234-5678',
           'Eric':'010-345-6789',
           'Ellie':'010-456-7890'
           }
phone_num.pop('Emily')
print(phone_num)
```

결과확인

```
{'Eric': '010-345-6789', 'Ellie': '010-456-7890'}
```

4.3.5 사전(dictionary)

- 사전 요소 반복하기

예제코드

```
phone_num={'Emily':'010-234-5678',
           'Eric':'010-345-6789',
           'Ellie':'010-456-7890'
           }

for name in phone_num:
    print(name)

for item in phone_num.items():
    print(item)

for name, num in phone_num.items():
    print(name+":"+num)
```

결과확인

```
Emily
Eric
Ellie
('Emily', '010-234-5678')
('Eric', '010-345-6789')
('Ellie', '010-456-7890')
Emily:010-234-5678
Eric:010-345-6789
Ellie:010-456-7890
```

4. 파이썬 함수와 구조적 타입

4.4.1 튜플(tuple)

- 하나 이상의 값을 반환하는 함수 에서 많이 사용

예제코드

```
# 절대온도를 입력받아 섭씨온도와 화씨온도 리턴
def calculate_temperatures(kelvin):
    celsius = kelvin - 273
    fahrenheit = celsius * 9 / 5 + 32
    return celsius, fahrenheit

c,f = calculate_temperatures(340)
print(c)
print(f)
```

결과확인

```
67
152.6
```

- 튜플은 리스트와 비슷하지만 값을 변경할 수 없고, 대괄호[ ]가 아닌 소괄호( ) 로 둘러쌓임 (괄호 생략가능)
- 한 개 또는 두개의 값을 반환할 때는 튜플을 사용
- 데이터가 복잡한 경우 데이터가 포함된 클래스를 정의하고 인스턴스를 반환하는게 좋음

QnA

References

Do it! 점프투 파이썬 (박응용, 이지스퍼블리싱)

라즈베리파이 쿡북 (사이먼 몽크, 한빛미디어)

헬로! 파이썬 프로그래밍 쉽고 재미있게 프로그래밍 배우기 (워렌 산데, 위키북스)

https://docs.python.org/3/reference/index.html

https://www.w3schools.com/python/default.asp