본 페이지는 편집 시 유의사항을 담고 있습니다.

페이지 수 오류를 방지하기 위해 실제 편집 작업 시에는 지워 주시기 바랍니다.

이 파일은 B5로 제작을 원하는 저자분들께서 사용하시면 됩니다.

편집 과정중 꽉차게 들어가는 이미지, 배경색상이 있을 경우 하얀색 종이가 보이지 않도록 이미지, 색상 크기 반영하셔야 하는 점 알려드립니다.

이파일은 실제 제작시 필요한 사방여백 3mm가 포함된 크기입니다.(188x263mm)

상하좌우 3mm는 실제 제작시 재단되어 반영되지 않으니 참고해주세요!

**어린 왕자**(제목을 적어주세요.)

**어린 왕자(제목을 적어주세요)**

**발 행**｜2023년 00월 00일

**저 자**｜생텍쥐 페리(저자명, 필명을 적어주세요)

**펴낸이**｜한건희

**펴낸곳**｜주식회사 부크크

**출판사등록**｜2014.07.15(제2014-16호)

**주 소**｜서울특별시 금천구 가산디지털1로 119 SK트윈타워 A동 305호

**전 화**｜1670-8316

**이메일**｜info@bookk.co.kr

**ISBN**｜979-11-410-0000-0

**www.bookk.co.kr**

**ⓒ 셍텍쥐 페리 2023**

본 책은 저작자의 지적 재산으로서 무단 전재와 복제를 금합니다.

어

린

왕

자

Evan, Sarah 지음

**CONTENT**

내용

[Chapter 1. 집필 동기 9](#_Toc130723838)

[1. 이 책을 쓰게 된 동기 9](#_Toc130723839)

[A. 강사 입장에서 9](#_Toc130723840)

[B. 수강생 입장에서 9](#_Toc130723841)

[2. 무엇을 배울 수 있는가 9](#_Toc130723842)

[Chapter 2. 개발환경설정 10](#_Toc130723843)

[1. Github 회원가입 10](#_Toc130723844)

[2. Streamlit 회원가입 10](#_Toc130723845)

[3. Git 설치 10](#_Toc130723846)

[4. Python 설치 10](#_Toc130723847)

[5. Visual Studio Code 설치 10](#_Toc130723848)

[6. 가상환경 Virtualenv 설치 10](#_Toc130723849)

[7. requirements.txt 파일 작성 10](#_Toc130723850)

[8. Streamlit, Hello World 10](#_Toc130723851)

[Chapter 3. Python 기본문법 11](#_Toc130723852)

[1. Python Basic Tutorial 11](#_Toc130723853)

[A. Python 프로그래밍 시작 11](#_Toc130723854)

[B. 주석처리 11](#_Toc130723855)

[C. Variables 12](#_Toc130723856)

[D. 사칙연산 15](#_Toc130723857)

[E. 비교 연산자Comparison Operators 16](#_Toc130723858)

[F. 논리 연산자Logical Operators 17](#_Toc130723859)

[G. String 19](#_Toc130723860)

[2. Pandas 25](#_Toc130723861)

[3. Matplotlib & Seaborn 25](#_Toc130723862)

[4. Plotly 25](#_Toc130723863)

[5. Scikit-Learn 25](#_Toc130723864)

[Chapter 4. Streamlit 26](#_Toc130723865)

[1. Slider bar 26](#_Toc130723866)

[2. Select bar 26](#_Toc130723867)

[3. Tab 26](#_Toc130723868)

[4. Checkbox 26](#_Toc130723869)

[Chapter 5. 공공데이터 수집 27](#_Toc130723870)

[1. 서울열린데이터광장 27](#_Toc130723871)

[2. 공공데이터포털 27](#_Toc130723872)

[Chapter 6. 부동산 실거래가 대시보드 28](#_Toc130723873)

[1. 데이터 수집 28](#_Toc130723874)

[2. Home 화면 꾸미기 28](#_Toc130723875)

[3. 탐색적 자료분석 화면 꾸미기 28](#_Toc130723876)

[4. 머신러닝 모형 만들기 28](#_Toc130723877)

[5. Prediction 화면 꾸미기 28](#_Toc130723878)

[6. OpenAI 챗봇 구현하기 28](#_Toc130723879)

[Chapter 7. 배포 29](#_Toc130723880)

[1. secrets.toml 29](#_Toc130723881)

[2. 배포 29](#_Toc130723882)

[3. 마무리 29](#_Toc130723883)

[인용 자료 30](#_Toc130723884)

작가의 말 242

레옹 베르트에게 이 책을 바칩니다.

이 책을 어른에게 바친 데 대해 어린이들에게 용서를 바랍니다.

나에게는 그럴 만한 중요한 이유가 있습니다.

그것은 무엇보다도 그 사람은 이 세상에서 나와 가장 친한

친구이기 때문라는 점입니다. 그리고 그는 무엇이든지

알아들을 수 있으며 어린이들을 위한 책까지도 다 이해한다는 점입니다.

세 번째 이유는 그가 프랑스에 살고 있는데

그 곳에서 추위와 굶주림에 떨고 있다는 사실입니다.

그는 위로받아야 할 처지에 있는 것입니다.

그래도 이 모든 이유들이 부족하다면 예전 어린 시절의

그에게 이 책을 바치겠습니다. 어른들도 모두 한때는 어린이였으니까요.

(물론 그것을 기억하는 어른은 별로 없지만.)

그래서 바치는 글을 이렇게 고쳐 씁니다.

'어린 시절의 레옹 베르트에게 이 책을 바칩니다

# Chapter 1. 집필 동기

## 이 책을 쓰게 된 동기

### 강사 입장에서

### 수강생 입장에서

## 무엇을 배울 수 있는가

# Chapter 2. 개발환경설정

## Github 회원가입

### Streamlit 회원가입

### Git 설치

### Python 설치

### Visual Studio Code 설치

### 가상환경 Virtualenv 설치

### requirements.txt 파일 작성

### Streamlit, Hello World

# Chapter 3. Python 기본문법

## Python Basic Tutorial

### Python 프로그래밍 시작

프로그래밍의 첫번째 시작은 “Hello World!”를 출력하는 것에서부터 시작한다. Python에서는 다음과 같이 print()함수를 사용한다.

|  |
| --- |
| print("Hello, Streamlit!")  [결과]  Hello, Streamlit! |

### 주석처리

주석처리는 크게 1줄 주석처리와 다중 주석처리가 존재한다. 다음 코드를 통해 확인해 본다. 아래 코드를 실행하면 주석 처리된 곳은 생략하고, print( ) 함수만 실행되는 것을 확인할 수 있다.

|  |
| --- |
| # 한 줄 주석 처리  """  abc  다중 주석 처리  1234556  """  print("Hello, Streamlit!")  [결과]  Hello, Streamlit! |

### Variables

프로그래밍에서 변수Variables를 저장하는 방법은 크게 4가지(Camel case, Pascal case, Kebab case, Snake case) 방식으로 구분한다. 원 단어가 임의의 학생을 저장하는 변수 my student가 있다고 가정한다. 각자 원하는 방식을 사용하도록 한다. 다음 예시를 통해서 관련 코드를 익히도록 한다.

|  |
| --- |
| myStudent = “sarah” # Camel Case  MyStudent = “sarah” # Pascal Case  my-student = “sarah” # Kebab Case  my\_student = “sarah” # Snake Case |

Python에서 Data Structures를 말할 때, Primitive 자료형과 Non-Primitive 자료형으로 구분한다 (Jaiswal, 2017).[[1]](#footnote-1)

Table 1. Python Data Structures

|  |  |
| --- | --- |
| Primitive | Non-Primitive |
| Integer  Float  String  Boolean | Array  List  Tuple  Dictionary  Set  File |

여기에서 List는 크게 아래와 같이 Linear와 Non-Linear와 구분할 수 있다.

Table 2. List Data Types[[2]](#footnote-2)

|  |  |
| --- | --- |
| Linear | Non-Linear |
| Stacks  Queues | Graphs  Trees |

#### Integer

Integer는 정수형을 말하며, 4, 5, -1과 같은 숫자를 의미한다. 각자의 나이를 입력하고 출력하는 코드를 작성한다.

|  |
| --- |
| num\_int = 30  print(num\_int)  print(type(num\_int))  [결과]  30  <class 'int'> |

#### Float

Float는 실수형을 말하며, 일반적으로 1.11또는 3.11과 같은 소수를 표현한다. 이번에는 본인의 키를 입력하고 출력하는 코드를 작성한다.

|  |
| --- |
| my\_height = 181.8  print(my\_height)  print(type(my\_height))  [결과]  181.8  <class 'float'> |

#### String

String은 문자열을 말하며, 알파벳, 단어 또는 기타 문자의 모음이다. Python에서는 한쌍의 작은 따옴표(‘’) 또는 큰 따옴표(“”) 안에 일련의 문자를 포함하여 문자열을 생성할 수 있다. 우선 다음 코드를 확인해본다. 임의의 글자를 입력하고 출력하는 코드를 작성한다. 문자열을 다루는 방법에 대해서는 추후에 한번 더 다룰 예정이다.

|  |
| --- |
| my\_name = "evan"  print(my\_name)  print(type(my\_name))  [결과]  evan  <class 'str'> |

#### Boolean

일반적으로 True(참) 또는 False(거짓)을 표현하는 데이터 타입이며, True는 1로, False는 0으로 변환되기도 한다. 해당 데이터 타입은 특히 조건식 및 비교식에 매우 유용하게 활용할 수 있다. 임의의 두개의 숫자를 비교하는 코드를 작성하고, 결괏값을 확인해본다.

|  |
| --- |
| x = 5  y = 3  z = x == y  print(z)  print(type(z))  [결과]  False  <class 'bool'> |

### 사칙연산

일반적으로 형변환이라고 부른다. 임의의 두개를 변수를 만들고 사칙연산을 만들어 보도록 한다. 이 때 주의해서 봐야 할 것은 나눗셈이다. 두 개의 정수를 나눌 때, 정수가 실수형으로 변한다는 것에 주의해야 한다.

|  |
| --- |
| x = 6  y = 3  print(x + y)  print(x - y)  print(x \* y)  print(x / y)  [결과]  9  3  18  2.0 |

이번에는 사칙연산을 통해 나머지, 몫, 제곱을 구하는 연산자를 확인한다.

|  |
| --- |
| x = 6  y = 3  print(x % y)  print(x // y)  print(x \*\* y)  [결과]  0  2  216 |

### 비교 연산자Comparison Operators

비교 연산자를 통해서 알고자 하는 것은 비교의 결괏값이 True인지, False인지 구분하기 위함이다. 우선 비교 연산자는 크게 6개가 존재 한다. 아래 표를 통해서 확인하도록 한다.

Table 3. Comparison Operators

|  |  |
| --- | --- |
| Operator | Meaning |
| A == B | A와 B가 같다 |
| A != B | A와 B가 같지 않다 |
| A > B | A가 B보다 크다 |
| A < B | A가 B보다 작다 |
| A >= B | A가 B보다 크거나 같다 |
| A <= B | A가 B보다 작거나 같다 |

Python 코드의 예시를 통해 살펴보도록 한다.

|  |
| --- |
| x = 10  y = 11  print(x == y)  print(x != y)  print(x > y)  print(x < y)  print(x >= y)  print(x <= y)  [결과]  False  True  False  True  False  True |

### 논리 연산자Logical Operators

논리 연산자는 주어진 조건식이 True인지 False인지 확인하는데 사용하며, 의사 결정 시에 유용하게 사용된다.

|  |
| --- |
| a = 3  b = 5  print((a > 2) and (b >= 6)  [결과]  False |

위 코드는 Python에서 논리 연산자를 사용하는 예제이다. 논리 연산자는 크게 and, or, not 3개가 존재한다. 논리 연산자의 기본적인 개념을 표로 정리하였다.

Table 4. Logical Operators[[3]](#footnote-3)

|  |  |  |
| --- | --- | --- |
| Operator | Example | Meaning |
| and | X and Y | X와 Y가 모두 True이면 True |
| or | X or Y | X와 Y중 한 개라도 True이면 True |
| not | not X | X가 False이면 True |

다음은 위 논리 연산자의 모든 결과를 보여주는 표이다.

Table 5. Truth Table

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | X and B | A or B |
| True | True | True | True |
| True | False | False | True |
| False | True | False | True |
| False | False | False | False |

위 표가 실제로 Python에서도 구현이 되는지 코드로 확인한다. 먼저 and 연산자를 구현한 코드이며, 결과는 양쪽이 모두 True일 때, True를 반환한다.

|  |
| --- |
| print(True and True)  print(True and False)  print(False and True)  print(False and False)  [결과]  True  False  False  False |

이번에는 or 연산자를 구현한 코드이며, 결과는 양쪽 중 하나만 True여도 True로 결과가 반환이 된다.

|  |
| --- |
| print(True or True)  print(True or False)  print(False or True)  print(False or False)  [결과]  True  True  True  False |

not 연산자를 사용하면 반대로 결괏값이 반환되는 것도 같이 확인한다.

|  |
| --- |
| print(not True)  print(not False)  [결과]  False  True |

### String

앞에서 String에 대해서 간단하게 다뤘지만, 일반적으로 문자열은 보다 자세히 다뤄야 할 정도로 매우 중요하다.

#### String Operators

우선 문자열에는 덧셈 연산자와 곱셈 연산자 사용이 가능하다. 덧셈 연산자는 두개의 문자를 이어서 내보내고, 곱셈 연산자는 숫자만큼 문자열이 반복된다. 코드를 통해 확인해본다. 먼저 덧셈 연산자는 아래와 같이 표현할 수 있다.

|  |
| --- |
| # 알아서 바꿔요 메롱!  a = "evan"  b = "loves"  c = "sarah"  letter = a + " " + b + " " + c + " forever"  print(letter)  [결과]  evan loves sarah forever |

이번에는 곱셈 연산자를 확인하는 코드를 작성한다.

|  |
| --- |
| x = "Love "  lyrics = x \* 5  print(lyrics)  [결과]  Love Love Love Love Love |

#### 인덱싱Indexing

인덱싱은 각각의 문자열 안에서 범위를 지정하는 것을 의미한다. 다음 이미지[[4]](#footnote-4)를 확인하면 문자열은 앞쪽에서는 0부터 시작하고, 뒤쪽에서는 -1부터 시작하는 것을 확인할 수 있다. 이 때, 기억해야 하는 것은 이번에 배울 인덱싱은 문자열, List, Tuple, array와 같은 자료형에도 동일하게 사용할 수 있다.



Figure 1. String Indexing

위 이미지를 토대로 간단하게 Streamlit Dashboard 문자열을 가지고 인덱싱을 진행하도록 한다 (독자도 임의대로 코드를 작성하여 실행하도록 해본다). 임의의 텍스트를 X 변수에 담고, X[인덱스번호]과 같은 형태로 코드를 실행하면 해당 인덱스번호의 글자가 조회된다.

|  |
| --- |
| X = "Streamlit Dashboard"  print(X[0]) # 0번째 글자 가져오기  print(X[5]) # 5번째 글자 가져오기  print(X[-1]) # -1번째 글자 가져오기  print(X[-3]) # -3번째 글자 가져오기  [결과]  S  m  d  a |

그런데, 만약 주어진 텍스트의 인덱스 번호를 초과하면 다음과 같은 에러가 발생이 된다.

|  |
| --- |
| print(X[100]) |

도표이(가) 표시된 사진

자동 생성된 설명

위 에러 메시지에서 주의 깊게 살펴봐야 하는 것은 ①『----> 1 print(X[100])』와 ②『**IndexError**: string index out of range』이다. ① 메시지는 어떤 코드에서 에러가 나타났는지를 표시해주는 것이고, ② 메시지는 에러코드의 내용이다. 간단히 요약하면, 1번째 print() 코드에서 **IndexError** 에러가 발생했다고 알려주는 것이다. 즉, X가 담고 있는 텍스트는 인덱스 번호가 100까지는 없기 때문에 조회할 것이 없다는 뜻이기도 하다. X 텍스트가 가지고 있는 인덱스 번호를 재 확인하여 입력하면 에러는 발생이 되지 않을 것이다.

#### 슬라이싱Slicing

인덱싱은 각 문자열의 문자를 한 개씩 추출한다면 슬라이싱은 범위를 지정하여 추출하는 것을 말한다. 즉, 기존 데이터에서 일부의 범위를 지정하여 하위 집합을 추출하는 것을 말한다. 이 슬라이싱의 문법은 일반적으로 [start:end:step]의 일반적인 패턴을 따르게 된다.

* start : 슬라이싱의 첫번째 인덱스이며, 해당 인덱스 번호부터 조회를 시작한다. 만약에, 생략을 하면 인덱스는 0번째부터 시작을 한다.
* end : 슬라이싱의 마지막 인덱스를 말한다. 그런데, 해당 인덱스 번호를 포함되지 않고 그 전까지만 조회가 된다. 만약에, 생략을 하면 해당 문자열의 전체 길이가 기본값으로 설정된다.
* step : 슬라이싱의 각 인덱스 사이에서 건너뛸 인덱스의 수를 말하며, 생략할 경우 기본값은 1이다.

주어진 예제를 확인하도록 한다. 이 때, 해당 인덱스 번호를 바꿔서 진행해서 원하는 대로 결과가 나오는지 확인한다.

|  |
| --- |
| greeting = "Hello Streamlit"  print(greeting[:])  print(greeting[6:])  print(greeting[:6])  print(greeting[3:8])  print(greeting[0:9:2])  [결과]  Hello Streamlit  Streamlit  Hello  lo St  HloSr |

인덱싱과 슬라이싱의 원리는 향후에 나오게 되는 List, Tuple 등 다른 자료형에도 그대로 사용이 되기 때문에, 반드시 기억하도록 한다.

#### 문자열 수정

주어진 문자열을 수정하는 코드를 작성한다. 일반적으로는 다음과 같은 코드 형태로 작성할 수 있다. 주어진 문자열 중 H를 A로 바꾸는 코드를 작성한다. 그런데 다음과 같은 에러가 난다.

|  |
| --- |
| greeting = 'Hello, Streamlit'  greeting[0] = 'A'  print(greeting) |
| **---------------------------------------------------------------------------**  **TypeError** Traceback (most recent call last)  **~\AppData\Local\Temp\ipykernel\_29108\2735535411.py** in <module>  1 greeting = **'Hello, Streamlit'**  **----> 2** greeting[0] = **'A'**  3 print(greeting)  **TypeError**: 'str' object does not support item assignment |

우선, 에러가 나는 이유는 문자열은 불변immutable 객체로 인식되기 때문이다. 불변 객체는 객체 생성 이후 내부의 상태가 변하지 않는 것을 말한다. 그렇다면, 새로운 객체를 만들려면 새로운 변수명을 만드는 방법으로 우회할 수 있다.

|  |
| --- |
| greeting = "Hello Streamlit"  new\_greeting = 'A' + greeting[1:]  print(new\_greeting)  [결과]  Aello, Streamlit |

#### String Methods

String 클래스 내부에는 다양한 메서드Methods들이 존재한다.[[5]](#footnote-5) 여기에서 자주 사용하는 여러 메서드들을 확인하도록 한다.

1. str.upper() : 영어 문자들을 모두 대문자로 변경한다.

|  |
| --- |
| sampleText = "hello, streamlit"  upper\_text = sampleText.upper()  print(upper\_text)  [결과]  HELLO, STREAMLIT |

2. str.upper() : 영어 문자들을 모두 소문자로 변경한다.

|  |
| --- |
| sampleText = "HELLO, STREAMLIT"  lower\_text = sampleText.lower()  print(lower\_text)  [결과]  hello, streamlit |

3. str.capitalize() : 영어 문자의 첫번째 글자만 대문자로 변경한다.

|  |
| --- |
| sampleText = "hello, streamlit"  cap\_text = sampleText.capitalize()  print(cap\_text)  [결과]  Hello, streamlit |

4. str.title() : 문자열 내의 각 단어들의 첫번째 글자만 대문자로 변경한다.

|  |
| --- |
| sampleText = "hello, streamlit"  title\_text = sampleText.title()  print(title\_text)  [결과]  Hello, streamlit |

5. str.strip() : 주어진 문자열의 앞뒤 공백이 있다면 모두 제거한다.

|  |
| --- |
| sampleText = " hello, streamlit "  stripped\_text = sampleText.strip()  print(stripped\_text)  [결과]  hello, streamlit |

6. str.replace(old, new) : 기존(old) 문자열을 새로운(new) 문자열로 바꾼다. 주어진 텍스트 hello에서 welcome으로 변경하는 코드를 작성한다.

|  |
| --- |
| sampleText = "hello, streamlit"  new\_text = sampleText.replace("hello", "welcome")  print(new\_text)  [결과]  welcome, streamlit |

7. str.split(sep=None, maxsplit=-1) : 문자열을 sep 파라미터에 입력한 분리 기준점으로 분리한 후 List[[6]](#footnote-6)로 반환한다.

|  |
| --- |
| sampleText = "hello, streamlit"  words = sampleText.split(sep=",")  print(words)  [결과]  ['hello', ' streamlit'] |

8. str.join(iterable) : List나 Tuple내의 값을 하나의 문자열로 변환한다.

|  |
| --- |
| words = ["hello", "streamlit"]  text = ", ".join(words)  print(text)  [결과]  hello, streamlit |

### List

파이썬에 존재하는 자료형으로 시퀀스Sequence 데이터를 다룬다. 시퀀스를 갖는 것은 데이터에 순서가 있다는 뜻이며, 순서가 존재하기 때문에 인덱스와 슬라이싱을 사용할 수 있다. 아래 코드는 주어진 객체가 시퀀스인지 확인하는 코드이다. Sequnece 클래스에 대한 설명은 본 책에서는 생략한다.[[7]](#footnote-7)

|  |
| --- |
| from collections.abc import Sequence  my\_num = 100  my\_list = [1, 2, 3]  my\_string = "hello"  is\_num = isinstance(my\_num, Sequence)  is\_list = isinstance(my\_list, Sequence)  is\_string = isinstance(my\_string, Sequence)  print(is\_num)  print(is\_list)  print(is\_string)  [결과]  False  True  True |

위 결과를 보면 알 수 있듯이 리스트와 문자열은 시퀀스 데이터로 판정을 받았지만, 수치형은 시퀀스 데이터가 아님을 알 수 있다. 그 외 다른 데이터 자료형을 정의하고, 테스트를 해보는 것은 독자에게 맡기도록 한다.

#### 리스트 생성

먼저 리스트를 생성하는 다양한 방법에 대해 살펴보도록 한다. 개발을 하면 종종임의의 빈 리스트Empty List를 생성할 때가 있기 때문에, 관련 코드는 한번씩 숙지하면 좋다. 마지막 변수 e에 저장된 리스트는 리스트 안에 또다른 리스트가 존재하는 것으로 중첩리스트Nested List라고 부르기도 한다.[[8]](#footnote-8)

|  |
| --- |
| a = []  b = list()  c = [2]  d = ['streamlit']  e = [1, 2, ['hello streamlit']]  print(a)  print(b)  print(c)  print(d)  print(e)  [결과]  []  []  [2]  ['streamlit']  [1, 2, ['hello streamlit']] |

#### 인덱싱 & 슬라이싱

이미 문자열에서 인덱싱과 슬라이싱에 관한 설명은 진행했기 때문에 여기에서는 생략하도록 한다.

## Pandas

## Matplotlib & Seaborn

## Plotly

## Scikit-Learn

# Chapter 4. Streamlit

## Slider bar

## Select bar

## Tab

## Checkbox

# Chapter 5. 공공데이터 수집

## 서울열린데이터광장

## 공공데이터포털

# Chapter 6. 부동산 실거래가 대시보드

## 데이터 수집

## Home 화면 꾸미기

## 탐색적 자료분석 화면 꾸미기

## 머신러닝 모형 만들기

## Prediction 화면 꾸미기

## OpenAI 챗봇 구현하기

# Chapter 7. 배포

## secrets.toml

## 배포

## 마무리

# 인용 자료

Jaiswal, S. (2017, Dec). *Python Data Structures Tutorial*. Retrieved from DataCamp: https://www.datacamp.com/tutorial/data-structures-python

1. Jaiswal, S. (2017, Dec). *Python Data Structures Tutorial*. Retrieved from DataCamp: https://www.datacamp.com/tutorial/data-structures-python [↑](#footnote-ref-1)
2. Python 코딩테스트에서는 Stacks, Queues, Graphs, Trees와 같은 개념을 알아야 풀 수 있는 문제들이 있어서, 해당 문법들을 공부할 필요가 있다. 본 책에서는 다루지 않는다. [↑](#footnote-ref-2)
3. 아래 표에서 X와 Y는 모두 조건식을 의미하며, 조건식의 결과가 True 또는 False인 상태를 의미한다. [↑](#footnote-ref-3)
4. https://www.learnbyexample.org/wp-content/uploads/python/String-Indexing.png [↑](#footnote-ref-4)
5. https://docs.python.org/3/library/stdtypes.html#string-methods [↑](#footnote-ref-5)
6. List와 Tuple, Dictionary와 같은 기초 문법은 다음 페이지부터 순차적으로 다룬다. [↑](#footnote-ref-6)
7. 더 읽어보기 : https://docs.python.org/3/library/collections.abc.html [↑](#footnote-ref-7)
8. 일반적으로 리스트에는 파이썬에서 존재하는 다양한 값을 저장할 수 있다. 숫자, 문자열 뿐만 아니라, 아직 배우지 않은 데이터프레임 객체, 함수, 클래스 등도 넣을 수 있어서, 활용범위가 매우 무궁무진하다. [↑](#footnote-ref-8)