

# Lecture 02: Python Basics 1

---

**Hyeryung Jang**

**(hyeryung.jang@dgu.ac.kr)**

AI Department, Dongguk University

# Syllabus: Today's Topic

---

Week	Topics
1	Introduction to Data Science, Environment Set-up
<b>2</b>	<b>Python Basics 1</b>
3	Python Basics 2
4	Python for Data Analysis: NumPy
5	Python for Data Analysis: Pandas 1
6	Python for Data Analysis: Pandas 2
7	Python for Data Analysis: Web Crawling
8	Midterm Exam
9	Python for Data Visualization: Basics
10	Python for Data Visualization: Advanced
11	Machine Learning with Python: Supervised Learning
12	Machine Learning with Python: Unsupervised Learning
13	Machine Learning with Python: Recommender System
14	Project Presentation
15	Final Exam

## Python Programming for Data Science



- 데이터 사이언스를 위한 파이썬 프로그래밍 **환경 구축**
- 파이썬 라이브러리를 활용하여 **다양한 데이터를 효율적으로 가공, 변환, 처리, 분석하고 시각화하는 프로그래밍 방법**
- 대표적인 머신러닝 알고리즘을 사용하여 데이터의 정보를 분석/표현하고 **의미 있는 지식, 통찰력을 추출하여 의사결정하는 기술 및 구현 능력**

# Python

---

- Python is ...
- Powerful... and fast
- Plays well with others
- Runs everywhere
- Is friendly & easy to learn
- Is Open



# Python

---

- “Hello” 출력 방법

C

```
# include "stdio.h"
int main() {
    printf("Hello \n");
}
```

Java

```
public class Hi {
    public static void main (String [] args) {
        System.out.println("Hello");
    }
}
```

Python

```
print("Hello")
```

# Python Basics: Types, String Operations

---

# Common Types in Python

---

- **Numeric:** integers, float, ...
- **Sequence:** list, tuple, ...
- **True/False:** bool
- **Text:** string
  
- `type(x)`: x 의 자료형 확인하기

x	type(x)
11	int
2.75	float
"Hello"	str

# Common Types in Python

---

- Type Casting
  - `float(2): 2.0`
  - `int(1.3): 1`
  - `int('5'): 5`
  - `int('b'): error!`
  - `str(1): "1"`
  - `int(True): 1`
  - `int(False): 0`



# Common Types in Python

---

- Dynamic Typing

C

```
# include "stdio.h"
int main() {
    int x = 3;
    int y = 5;
    printf("%d\n", x+y);
    y = 7.5;
    printf("%d\n", x+y);
}
```

Python

```
x = 3
y = 4
print(x+y)
y = 7.5
print(x+y)
```

# String Functions (문자열 자료형)

---

- A string is a sequence of characters contained within ' ' or " "

`string = "Hello World"`      `'Hello World'`

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- `string[0]`? `string[4]`? `string[9]`?
- `string[-1]`? `string[-5]`? `string[-7]`?
- **Indexing** (인덱싱): 0부터 숫자를 센다
- Negative indexing: 뒤에서부터 세는 것

# String Functions

---

- **Slicing** (슬라이싱): `string[시작:끝]`

**string = "Hello World"**

<b>H</b>	<b>e</b>	<b>l</b>	<b>l</b>	<b>o</b>		<b>W</b>	<b>o</b>	<b>r</b>	<b>l</b>	<b>d</b>
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- `string[0:3]`: 'Hel',
- `string[:7]`: 'Hello W'
- `string[4:-2]`: 'o Wor'
- **`string[1:4]+string[-5:9]`: ??**

# String Functions

---

- **Striding:** string[시작:끝:간격]

string = "Hello World"

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- string[0:7:2]: 'HloW'
- string[::3]: 'HIWI'

# String Functions

---

- **Length** of string: `len(string)`

**string = "Hello World"**

<b>H</b>	<b>e</b>	<b>l</b>	<b>l</b>	<b>o</b>		<b>W</b>	<b>o</b>	<b>r</b>	<b>l</b>	<b>d</b>
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- `len(string)`: **11**
- Cannot change the value of the string: **`string[2] = 'k': error!`**

# String Functions: 문자열 관련 함수들

---

string = "Hello World"

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- **.count**: 문자 개수 세기
  - string.count('l'): 3
- **.find**: 위치 알려주기
  - string.find('o'): 4
  - string.find('el'): 1

# String Functions: 문자열 관련 함수들

---

string = "Hello World"

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- **.replace**: 문자열 바꾸기
  - string.replace('World', 'Guys'): 'Hello Guys'
- **.split**: 문자 나누기
  - string.split(): 공백 기준으로 문자열 나누기: ['Hello', 'World']
- **.strip**: 양쪽 공백 지우기

# Python Basics: Key Data Structures

---



# Lists (리스트 자료형)

---

- Ordered sequences

**L = ['Hello World', 3, 2.5, ['Data', 'Science']]**

- List\_name = [ element 1, element 2, ...]
  - a = []
  - b = [1, 2, 'Data']
  - c = ['Data', 'Science']
  - d = [1, 2, ['Data', 'Science']]
- 어떠한 자료형도 포함시킬 수 있음
  - int, float, string, ... and list!
- 값을 바꿀 수 있음: mutable

# Lists

---

- Indexing, Slicing, Striding, ....

- L[0]: 'Hello World'

- L[1]: 3

- L[2]: 2.5

- L[-1]: ['Data', 'Science']

- L[-1][1]: 'Science'

- L[1:3]: [3, 2.5]

- L[-3:]: [3, 2.5, ['Data', 'Science']]

**L = ['Hello World', 3, 2.5, ['Data', 'Science']]**

index	(Neg) index	element
0	-4	'Hello World'
1	-3	3
2	-2	2.5
3	-1	['Data', 'Science']

# Lists

---

`L = ['Hello World', 3, 2.5, ['Data', 'Science']]`

- `L1 = L + ['Python', 4001]`
- `L1 = ['Hello World', 3, 2.5, ['Data', 'Science'], 'Python', 4001]`

index	(Neg) index	element
0	-6	'Hello World'
1	-5	3
2	-4	2.5
3	-3	['Data', 'Science']
4	-2	'Python'
5	-1	4001

# Lists: Mutable

---

- List는 값을 수정하거나 삭제 가능함

**`L = ['Hello World', 3, 2.5, ['Data', 'Science']]`**

- `L[2] = 1.4058` → `L = ['Hello World', 3, 1.4058, ['Data', 'Science']]`
- **`del L[x]`**: x번째 element 삭제
  - `del L[2]` → `L = ['Hello World', 3, ['Data', 'Science']]`

# Lists Functions: 리스트 관련 함수들

---

`L = ['Hello World', 3, 2.5, ['Data', 'Science']]`

- **.append**: element 추가
- `L.append('Python') → L = ['Hello World', 3, 2.5, ['Data', 'Science'], 'Python']`
- **.insert(a,b)**: a번째 element에 b를 추가L
- `L.insert(3, 'insert!!') → L = ['Hello World', 3, 2.5, 'insert!!', ['Data', 'Science']]`

- **.sort**: element 정렬 `a = [1, 7, 4, 2]`
- `a.sort() → a = [1,2,4,7]`

- **.index, .count, .pop, .remove, .extend, ...**

# Lists (리스트 자료형)

---

- Ordered sequences

**L = ['Hello World', 3, 2.5, ['Data', 'Science']]**

- List\_name = [ element 1, element 2, ...]
  - a = []
  - b = [1, 2, 'Data']
  - c = ['Data', 'Science']
  - d = [1, 2, ['Data', 'Science']]
- 어떠한 자료형도 포함시킬 수 있음
  - int, float, string, ... and list!
- 값을 바꿀 수 있음: mutable

# Tuples (튜플 자료형)

---

- Ordered sequence

**T = ('Hello World', 3, 2.5, ['Data', 'Science'])**

- List의 값은 생성, 삭제, 수정이 가능
- Tuple의 값은 수정 불가능함
- del T[1]? → error!
- T[0] = 'Change' ? → error!
-

# Tuples

---

- Indexing, Slicing, Striding, +, len, ...

**T = ('Hello World', 3, 2.5, ['Data', 'Science'])**

- T + ('Python', 4001): ('Hello World', 3, 2.5, ['Data', 'Science'], 'Python', 4001)
- **T[3] = [ 'Change', 'Science'] ??**
- **T[3][0] = 'Change' ??**
- → T = ('Hello World', 3, 2.5, ['Change', 'Science'])



# Dictionaries (딕셔너리 자료형)

---

- Type of 'Collection' in Python
- Dictionary\_name = {Key1:Value1, Key2:Value2, ... }

## List

`L = ['Hello World', 3, 2.5, ['Data', 'Science']]`

index	element
0	'Hello World'
1	3
2	2.5
3	['Data', 'Science']

## Dictionary

`D = {'name': 'Kim', 'birth': ['March', 9], 'age': 27}`

key	value
'name'	'Kim'
'birth'	['March', 9]
'age'	27

# Dictionaries

---

**D = {'name': 'Kim', 'birth': ['March', 9], 'age': 27}**

- Keys: immutable and unique
- Values: can be mutable and duplicates
- **D[Key] = Value**: Dictionary pair 추가하기
- **D[1] = 'value1' → D = {'name': 'Kim', 'birth': ['March', 9], 'age': 27, 1: 'value1'}**
- **del D[x]**: key가 x 인 pair 삭제하기
- **del D[1] → D = {'name': 'Kim', 'birth': ['March', 9], 'age': 27}**

# Dictionaries

---

**D = {'name': 'Kim', 'birth': ['March', 9], 'age': 27}**

- List, Tuple의 값에 접근할 때에는 index를 사용
- Dictionary의 값에 접근할 때에는 key를 사용
- D['name']: 'Kim'
- D['birth']: ['March', 9]

# Dictionaries

---

**D = {'name': 'Kim', 'birth': ['March', 9], 'age': 27}**

- Key 목록 확인하기
- **D.keys():** dict\_keys(['name', 'birth', 'age'])
  - 'birth' in D → True
- Value 목록 확인하기
- **D.values():** dict\_values(['Kim', ['March', 9], 27])

key	value
'name'	'Kim'
'birth'	['March', 9]
'age'	27

- Item (Key, Value) 목록 확인하기
- **D.items():** dict\_items([('name', 'Kim'), ('birth', ['Marth', 9]), ('age', 27)])

# In this lesson, you have learned:

---

- Python Basics:
- Types: integer, float, string, ...
- Standard string operations
  - indexing, slicing, striding, ...
  - ... replace, split, ...
- Important data structures in Python:



---

# Thank you!

Any Questions?

[hyeryung.jang@dgu.ac.kr](mailto:hyeryung.jang@dgu.ac.kr)

---