빅데이터 처리 및 응용

제 2 강
파이썬의 개요

김 대 경
한양대학교 응용수학과

파이썬의 개요

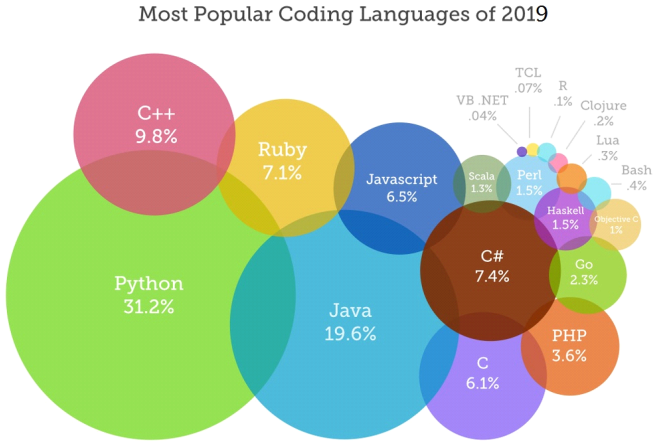
차 례

■ 파이썬 기초

▶ 파이썬 소개

파이썬이란?

- 파이썬(Python)은 1991년 귀도 반 로섬(Guido van Rossum)이 발표한 인터프리티드 객체지향 언어임.
- 배우기 쉽고 활용도가 높아 학교, 기업에서 많이 사용되고 있음
- 구글, 페이스북, 미 항공 우주국 등이 파이썬을 이용해 서비스를 구축
- 구글이 개발한 프로그램의 60%가 파이썬을 이용함
- 빅데이터/인공지능 관련 라이브러리에 활용



파이썬의 특징

- 인간의 사고 체계를 그대로 컴퓨터에게 지시하는 객체지향 프로그래밍 언어
- 문법이 쉽고 어순이 영어 구문과 유사하여 빠르게 학습할 수 있음
- 자료 구조의 표현이 매우 간결하고, 다양한 자료구조를 지원함
- 다양한 플랫폼에서 사용 가능하며, 오픈소스로 제공됨
- 모듈이나 패키지가 다양한 형태로 방대하게 제공됨
- 간결함의 철학으로 만들어져서 개발속도 빠르고, 유지보수가 용이하여 생산성이 높음.

컴퓨터 프로그램언로서 파이썬의 특징

- 동적언어(스크립트언어)
- (대화식) 인터프리터(interpreted) 언어
- 고급, 고수준(high level) 언어
- C, C++, Fortran 코드와 통합이 쉬움.

파이썬이 적합하지 않은 경우

- 대규모(large scale) 과학계산
- 동시 다발적인 멀티스레드(multi-thread)를 처리하는 경우
- CPU에 집중된 많은 스레드를 처리하는 경우

파이썬 설치

- 파이썬 공식 홈페이지
  (https://www.python.org/downloads/)
- 파이썬 에디터
  - ✓ 에디트 플러스 (Edit+)
  - ✓ 파이참 (PyCham)
  - ✓ 노트패드++ (Notepad++)
  - ✓ 주피터 노트북 (Jupyter Notebook)

- 파이썬 패키지 배포판
  - ✓ Anaconda (https://www.anaconda.com/)
  - ✓ Enthought Canopy (https://www.enthought.com/product/canopy)

데이터 분석을 위한 파이썬 라이브러리

- Numpy
- Scipy
- SymPy
- Pandas
- Matplotlib
- Seaborn
- StatsModels
- Pgmpy

▶ 파이썬 기본 문법

파이썬 기초 문법 맛보기

사칙연산

- 변수에 숫자를 대입한 후 사칙연산

```
>>> a=2, b=3

>>> a+b
5

>>> a*b
6

>>> a/b
0.6666666666666666

>>> a**b
8
```

```
>>> a=23; b=3

>>> a/b
7.666666666666667

>>> a//b
7

>>> a%b
2
```

파이썬 연산자

- 산술 연산을 위한 7가지 연산자

+	덧셈
-	뺄셈
*	곱하기
**	거듭 제곱
/	나누기
//	나누기 연산 후 소수점 이하의 수를 버리고, 정수 부분의 수만 구함
%	나누기 연산 후 몫이 아닌 나머지를 구함

출력

- 변수에 문자 대입한 후 출력

```
>>> a = "Hello Python!"

>>> print(a)
Hello Python!
```

조건문 if

- 간단한 조건문 형식

```
>>> a = 3

>>> if a>1:
...     print("a is greater than 1")
...
a is greater than 1
```

✓ ▭ : Space Bar 4칸 들여쓰기

✓ ↵ : Enter↵ 입력

반복문 for

- for문을 이용한 반복 실행

```
>>> for a in [1,2,3]:
...     print(a)
...
1
2
3
```

반복문 while

- while문을 이용한 반복 실행

```
>>> i = 0
>>> while i < 3:
...     i = i + 1
...     print(i)
...
1
2
3
```

파이썬 프로그램 파일

- 파이썬 프로그램 파일의 확장자명은 py로 함.

```
# hello.py
"""
Date : 2020-09-01
This is just a test!!!
"""
print("Hello~! Big Data Course")
```

```
C:\Python\python.exe C:/Users/home/hello.py
Hello~! Big Data Course

Process finished with exit code 0
```

▶ 파이썬 프로그램의 자료형

자료형은 프로그래밍을 할 때 쓰이는 숫자, 문자열, 리스트, 집합 등 자료 형태로 사용하는 모든 것을 뜻하며, 프로그램의 기본이자 핵심 단위가 됨.

- 자료형 분류
  - ✓ 숫자
  - ✓ 문자열
  - ✓ 리스트
  - ✓ 튜플
  - ✓ 딕셔너리
  - ✓ 집합

❖ 숫자 자료형

- 숫자 형태로 이루어진 자료형

항목	파이썬 사용 예
정수	123, -345, 0
실수	123.45, -123.45, 3.4e10, 3.4e-10
복소수	1+2j, -3j
2진수	0b101, 0b111
8진수	0o177, 0o057
16진수	0x2A, 0xFF

```
>>> 0o77 + 0xFF + 0b100000 + 100
450
```

복소수 자료형

- 복소수 자료형에는 몇 가지 유용한 함수가 있음

```
>>> a = 1 + 2j

>>> b = 3 - 4j

>>> a + b
(4 - 2j)

>>> a.real
1.0

>>> a.imag
2.0

>>> a.conjugate()
(1 - 2j)

>>> abs(a)
2.23606797749979
```

❖ 문자열 자료형

- 문자, 단어 등으로 구성된 문자들의 집합

문자열

- 파이썬에서 문자열은 큰따옴표 (" ")로 둘러싸임

```
>>> "Hello~! Big Data Course"
>>> "a"
>>> "123"
```

- 문자열 만들기

```
>>> a = 'Hi'
>>> b = "Hi"
>>> c = '''Hi'''
>>> d = """Hi"""
```

✓ 문자열을 표현하는 방법은 4가지가 있음.

✓ 문자열을 둘러싸는 기호

- 문자열 안에 작은따옴표나 큰따옴표를 포함시키기

```
>>> a = "Python's course is Big Data"
>>> a
Python's course is Big Data

>>> b = '"Python is very easy." he says.'
>>> b
"Python is very easy." he says.

>>> c = 'Python\'s course is Big Data'
>>> c
Python's course is Big Data

>>> d = "\"Python is very easy.\" he says."
>>> d
"Python is very easy." he says.
```

- 여러 줄인 문자열을 변수에 대입하기

```
Life is too short
You need python
```

```
a = "Life is too short\nYou need python"
```

✓ 문자열의 줄을 바꾸기 위한 이스케이프 코드 '\n'

```
b = '''
Life is too short
You need python
'''
```

```
c = """
Life is too short
You need python
"""
```

문자열 연산

- 문자열의 연산자에는 +, *, [], [:], % 등이 있음.

[문자열 결합]

- 문자열 더하기 또는 연결하기 (Concatenation)

```
>>> a = "Python"
>>> b = " is fun!"
>>> a + b
'Python is fun!'
```

- 문자열 곱하기

```
>>> a = "Python"
>>> a*2
'PythonPython'
```

[문자열 인덱싱과 슬라이싱]

- 문자열 인덱싱 또는 색인 (Indexing)

```
>>> a = "Life is too short!"
```

L	i	f	e		i	s		t	o	o		s	h	o	r	t	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

```
>>> a = "Life is too short!"
>>> a[3]
'e'
>>> a[12]
's'
>>> a[-1]
'!'
>>> a[-5]
'h'
```

- 문자열 슬라이싱 (Slicing)

```
>>> a = "Life is too short!"

>>> b = a[0]+a[3]+a[2]+a[8]
'Left'

>>> a[1:7]
'ife is'

>>> a[:7]
'Life is'

>>> a[12:]
'short!'

>>> a[8:-8]
'to'
```

```
>>> a = "20190903Rainy"

>>> date = a[:8]

>>> weather = a[8:]

>>> date
'20190903'

>>> weather
'Rainy'
```

[문자열 포매팅: 문자열 모듈로 연산자 '%']

- 숫자 바로 대입

```
>>> "I eat %d apples." %3
'I eat 3 apples.'
```

- 문자열 바로 대입

```
>>> "I eat %s apples." %"three"
'I eat three apples.'
```

- 2개 이상의 값 대입

```
>>> num = 10
>>> day = "three"
>>> "I eat %d apples, so I was sick for %s days." %(num, day)
'I eat 10 apples, so I was sick for three days.'
```

- 문자열 포맷 코드

코 드	설 명
%s	문자열(string)
%c	문자 1개(character)
%d	정수(integer)
%f	실수(floating-point number)
%o	8진수
%x	16진수
%%	문자 '%' 자체

[포맷 코드와 숫자 함께 사용]

- 정렬과 공백

```
>>> "%10s" %"hi"
'        hi'

>>> "%-10sjane" %"hi"
'hi        jane'
```

- 소수점 표현

```
>>> "%0.4f" %3.141592
'3.1416'

>>> "%10.4f" %3.141592
'    3.1416'
```

문자열 관련함수

[문자열의 메서드]

- 파이썬에서 메서드(method)는 일종의 함수이며 다음과 같은 형태로 호출됨.

객체.메서드(인자들)

- 문자 개수 세기 (count)

```
>>> a = "hobby"
>>> a.count('b')
2
```

- 위치 찾기1 (find)

```
>>> a = "my hobby"
>>> a.find('b')
5
```

- 위치 찾기2 (index)

```
>>> a = "my hobby"
>>> a.index('b')
5
```

- 문자열 삽입 (join)

```
>>> a = "12"
>>> a.join('abcd')
'a12b12c12d'
```

- 문자열 바꾸기 (replace)

```
>>> a = "Life is too short"
>>> a.replace("Life", "Your leg")
'Your leg is too short'
```

- 문자열 나누기 (split)

```
>>> a = "Life is too short"
>>> a.split()
['Life', 'is', 'too', 'short']
```

```
>>> a = "a:b:c:d"
>>> a.split(":")
['a', 'b', 'c', 'd']
```

❖ 리스트 자료형

리스트, 리스트 연산

```
리스트 명 = [요소1, 요소2, 요소3, ...]
```

```
>>> a = []
>>> b = [1,2,3]
>>> c = ['Life','is','too','short']
>>> d = [1,2,'Life','is']
>>> e = [1,2,['Life','is']]
>>> f = [a,b,c,d,e]
```

```
[[],
 [1, 2, 3],
 ['Life', 'is', 'too', 'short'],
 [1, 2, 'Life', 'is'],
 [1, 2, ['Life', 'is']]]
```

[리스트의 인덱싱과 슬라이싱]

- 리스트의 인덱싱

```
>>> a = [1,2,3]
>>> a[0]
1
>>> a[0]+a[2]
4
>>> a[-1]
3
```

```
>>> a = [1,2,3, ['a','b','c']]
>>> a[0]
1
>>> a[-1]
['a','b','c']
>>> a[3]
['a','b','c']
>>> a[3][0]
'a'
>>> a[3][1]
'b'
```

```
>>> a = [1,2,['a','b',['Life','is']]]
>>> a[2]
['a', 'b', ['Life', 'is']]
>>> a[2][2]
['Life', 'is']
>>> a[2][2][0]
'Life'
```

- 리스트의 슬라이싱

```
>>> a = [1,2,3]
>>> a[0:2]
[1, 2]

>>> a = "12345"
>>> a[0:2]
'12'
```

```
>>> a = [1,2,3,4,5]
>>> b = a[:2]
>>> c = a[2:]

>>> b
[1, 2]
>>> c
[3, 4, 5]
```

```
>>> a = [1,2,3,['a','b','c'],4,5]

>>> a[2:5]
[3, ['a', 'b', 'c'], 4]

>>> a[3][:2]
['a', 'b']
```

[리스트 연산]

- 리스트 더하기(결합하기), 반복하기

```
>>> a = [1,2,3]; b = [4,5,6]

>>> a+b
[1, 2, 3, 4, 5, 6]
```

```
>>> a = [1,2,3]

>>> a*3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

[리스트의 수정, 변경과 삭제]

- 리스트에서 하나의 값 수정하기

```
>>> a = [1,2,3]

>>> a[2] = 4
>>> a
[1, 2, 4]
```

- 리스트에서 연속된 값 수정하기

```
>>> a = [1,2,3]

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

```
>>> a = [1,2,3]

>>> a[1] = ['a','b','c']
>>> a
[1, ['a', 'b', 'c'], 3]
```

- [ ] 사용해 리스트 요소 삭제하기

```
>>> a = [1, 'a', 'b', 'c', 3]

>>> a[1:3] = []
>>> a
[1, 'c', 3]
```

```
>>> a = [1, 'a', 'b', 'c', 3]

>>> del a[1:3]
>>> a
[1, 'c', 3]
```

리스트 관련함수

- 리스트에 요소 추가 (append)

```
>>> a = [1,2,3]

>>> a.append(5)
>>> a
[1, 2, 3, 5]
```

```
>>> a = [1,2,3]

>>> a.append(['a','b'])
>>> a
[1, 2, 3, ['a', 'b']]
```

- 리스트 정렬 (sort)

```
>>> a = [1,4,3,5,2]

>>> a.sort()
>>> a
[1, 2, 3, 4, 5]
```

```
>>> a = ['d','a','c','b']

>>> a.sort()
>>> a
['a', 'b', 'c', 'd']
```

- 리스트 뒤집기 (reverse), 위치 반환 (index)

```
>>> a = ['a','b','c','d']

>>> a.reverse()
>>> a
['d', 'c', 'b', 'a']
```

```
>>> a = ['a','b','b','c','d']

>>> a.index('b')
1
>>> a.index('a')
0
```

- 리스트에 요소 삽입 (insert), 리스트 요소 제거 (remove)

```
>>> a = ['a','b','c','d']

>>> a.insert(0, 4)
>>> a
[4, 'a', 'b', 'c', 'd']
```

```
>>> a = ['a','b','b','c','d']

>>> a.remove('b')
>>> a
['a', 'b', 'c', 'd']
```

- 리스트에서 요소 끄집어내기 (pop), 리스트에 포함된 요소의 개수 세기 (count)

```
>>> a = ['a','b','b','c','d']
>>> a.pop()
'd'
>>> a
['a', 'b', 'b', 'c']

>>> a.pop(2)
'b'
>>> a
['a', 'b', 'c']
```

- 리스트에 포함된 요소의 개수 세기 (count)

```
>>> a = ['a','b','b','c','d']
>>> a.count(1)
0
>>> a.count('b')
2
```

- 리스트 확장 (extend)

```
>>> a = ['a','b','c']

>>> a.extend([4,5])

>>> a
['a', 'b', 'c', 4, 5]

>>> a.extend(a)

>>> a
['a', 'b', 'c', 4, 5, 'a', 'b', 'c', 4, 5]
```

❖ 튜플 자료형

튜플(tuple)은 리스트와 거의 비슷하며 다른 점은 다음과 같다.

- ✓ 리스트는 [ ]로 둘러싸지만, 튜플은 ( )로 둘러싼다.
- ✓ 리스트와 달리 튜플은 초기화 이후 편집이 불가함.

튜플, 튜플 연산

```
튜플 명 = (요소1, 요소2, 요소3, ...)
```

```
>>> t1 = ()
>>> t2 = (1,)
>>> t3 = (1,2,3)
>>> t4 = 1,2,3
>>> t5 = ('a','b',('ab','cd'))
```

- ✓ 원소의 개수가 1개인 튜플은 요소 뒤에 콤마 (,)를 반드시 붙여야함.

[튜플의 인덱싱과 슬라이싱]

- 튜플의 인덱싱과 슬라이싱

```
>>> t1 = (1,2,'a','b')

>>> t1[0]
1

>>> t1[2]
'a'
```

```
>>> t1 = (1,2,'a','b')

>>> t1[1:]
(2, 'a', 'b')

>>> t1[1:2]
(2,)
```

- 튜플의 더하기(결합하기), 반복하기

```
>>> t2 = (3,4)

>>> t1+t2
(1, 2, 'a', 'b', 3, 4)
```

```
>>> t2*3
(3, 4, 3, 4, 3, 4)
```

튜플을 사용하는 이유

- 튜플은 더 적은 공간을 사용함.
- 실수로 튜플의 항목이 손상될 염려가 없음.
- 튜플을 딕셔너리 키로 사용할 수 있음.
- 함수의 인자들은 튜플로 전달됨.

❖ 딕셔너리 자료형

딕셔너리(dictionary)는 Key와 Value의 쌍으로 이루어짐. 이러한 대응 관계를 나타내는 자료형을 연관배열(associative array) 또는 해시(hash)라고 함.

Key	Value
야구	배트, 볼, 글러브, 야구장
커피	아메리카노, 에스프레소, 라테

- 딕셔너리 자료형의 특징
  - ✓ 리스트나 튜플처럼 순차적으로 해당 요소값을 구하지 않고, Key를 통하여 Value를 얻음.
  - ✓ 인덱스가 없고, Key를 이용하여 각 항목들을 추가하고 Key에 대한 Value를 수정, 삭제함.
  - ✓ 순서가 없음.

딕셔너리 만들기

- 딕셔너리 변수 만들기

```
딕셔너리 명 = {키1:값1, 키2:값2, 키3:값3, ...}
```

```
>>> dic = {'name':'Kim','phone':'0109993232','birth':'0905'}

>>> {'김연아':'피겨스케이팅','류현진':'야구','박지성':'축구'}
{'김연아': '피겨스케이팅', '류현진': '야구', '박지성': '축구'}

>>> a = {1:'hi'}

>>> a = {'a':[1,2,3]}

>>> dict()
{}

>>> dict({1:'T', 0:'F'})
{1: 'T', 0: 'F'}
```

- 딕셔너리 쌍 추가, 요소 삭제하기

```
>>> a = {1:'a'}

>>> a['year'] = 2020

>>> a
{1: 'a', 'year': 2020}

>>> a[3] = 'bye'

>>> a
{1: 'a', 'year': 2020, 3: 'bye'}

>>> del a['year']

>>> a
{1: 'a', 3: 'bye'}
```

딕셔너리를 사용하는 방법

- 딕셔너리에서 Key를 사용해 Value얻기

```
>>> grade = {'pey':10,'julliet':99}

>>> grade['pey']
10

>>> grade['julliet']
99
```

- 딕셔너리 만들 때 주의 사항
  - ✓ Key는 고유한 값이므로 중복되는 Key 값을 설정하면, 하나를 제외한 나머지 것들이 모두 무시됨
  - ✓ Key에 리스트는 쓸 수 없음. (튜플은 쓸 수 있음!)

```
>>> {1:'a',2:'b',1:'c'}
{1: 'c', 2: 'b'}

>>> {[1,2]:'hi'}
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: unhashable type: 'list'

>>> {(1,2):'hi'}
{(1, 2): 'hi'}
```

딕셔너리 관련 함수

- Key 리스트 만들기 (keys)

```
>>> dic = {'name':'Kim','phone':'0109993232','birth':'0905'}

>>> dic.keys()
dict_keys(['name', 'phone', 'birth'])
```

- dict_keys 객체를 리스트로 변환하기.

```
>>> list(dic.keys())
['name', 'phone', 'birth']
```

- Value 리스트 만들기 (values)

```
>>> dic.values()
dict_values(['Kim', '0109993232', '0905'])
```

- Key, Value 쌍 얻기 (items)

```
>>> dic.items()
dict_items([('name','Kim'), ('phone','0109993232'), ('birth','0905')])
```

- Key, Value 쌍 모두 지우기 (clear)

```
>>> dic.clear()

>>> dic
{}
```

- 해당 Key가 딕셔너리 안에 있는지 조사하기 (in)

```
>>> 'name' in dic
True

>>> 'email' in dic
False
```

❖ 집합 자료형

원소의 값들의 순서에 상관없는 집합 자료형으로 집합론에 관련된 것들을 쉽게 처리하기 위해 만들어짐.

집합 만들기

```
>>> s1 = set([1,1,2,'a','a','b'])

>>> s1
{1, 2, 'a', 'b'}

>>> s2 = set("hello")

>>> s2
{'o', 'h', 'e', 'l'}

>> {1,2,'1'}
{1, 2, '1'}
```

- 집합 자료형의 특징
  - ✓ 중복을 허용하지 않음.
  - ✓ 순서가 없음.

집합 자료형 활용하는 방법

- 교집합, 합집합, 차집합 구하기

```
>>> s1 = set([1,2,3,4,5])

>>> s2 = set([4,5,6,7,8,9])

>>> s1 & s2
{4, 5}

>>> s1.intersection(s2)
{4, 5}

>>> s1 | s2
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
>>> s1.union(s2)
{1, 2, 3, 4, 5, 6, 7, 8, 9}

>>> s1 - s2
{1, 2, 3}

>>> s2 - s1
{8, 9, 6, 7}

>>> s1.difference(s2)
{1, 2, 3}
```

집합 자료형 관련 함수

- 값 1개 추가하기 (add)

```
>>> s1 = set([1,2,3])

>>> s1.add('a')

>>> s1
{'a', 1, 2, 3}
```

- 값 여러 개 추가하기 (update)

```
>>> s1 = set([1,2,3])

>>> s1.update([2,3,'a','b'])

>>> s1
{'a', 1, 2, 3, 'b'}
```

- 특정 값 제거하기 (remove)

```
>>> s1 = set([1,2,3])

>>> s1.remove(2)

>>> s1
{1, 3}
```

❖ 자료형의 참과 거짓

자료형에 값(원소)이 없으면 **거짓**이고 값(원소)이 있으면 **참**

```
>>> while a:
...     a.pop()
...     print(a)
...
[1, 2, 3]
[1, 2]
[1]
[]

>>> if []:
...     print("True")
... else:
...     print("False")
...
False
```

❖ 자료형의 값을 저장하는 공간, 변수

파이썬은 변수에 저장된 값을 스스로 판단하여 자료형을 알아냄.

변수 명 = 변수에 저장할 값

```
>>> a=1

>>> b="python“

>>> c=[1,2,3]

>>> type(a)
<class 'int'>

>>> type(b)
<class 'str'>

>>> type(c)
<class 'list'>
```

변수를 만드는 여러 가지 방법

```
>>> a, b = ('python', 'life')

>>> a
'python'

>>> b
'life'

>>> (a,b) = 'python', 1

>>> a
'python'

>>> b
1
```

```
>>> [a, b] = ['python', 1]

>>> a
'python'

>>> b
1

>>> a = b = 'python'

>>> a
'python'

>>> b
'python'
```

- 메모리에 생성된 변수 없애기 (del)

```
>>> a = 3

>>> del(a)

>>> a
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'a' is not defined
```

- 리스트를 변수에 넣고 복사하고자 할 때

```
>>> a = [1,2,3]

>>> b = a

>>> a[1] = 4
```

```
>>> a
[1, 4, 3]

>>> b
[1, 4, 3]
```

```
>>> a = [1, 2, 3]

>>> b = a[:]

>>> a[1]=4

>>> a
[1, 4, 3]

>>> b
[1, 2, 3]
```