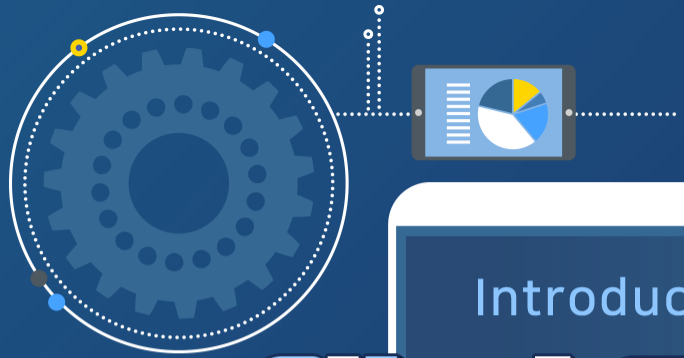
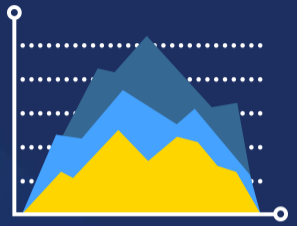


Introduction To Data Analysis

데이터분석입문



Lecture 04. 파이썬 자료형과 변수

인공지능소프트웨어학과 **강환수 교수**

학습개요

- 파이썬 자료형
  - int, float, str, bool, list, tuple, set, dict, frozenset
- 변수와 키워드
- 모임 자료형
  - list, tuple, set, dict, frozenset
- 수정가능과 수정 불가능

학습목표

- 파이썬 자료형을 이해하고 활용할 수 있다.
- 변수와 키워드를 이해하고 사용할 수 있다.
- 모임자료형을 사용할 수 있다.
- 수정가능과 수정불가능을 이해하고 활용할 수 있다.

LESSON 01

파이썬 자료형과 리터럴

현재 사용 파이썬 점검

코드 셀

- 파이썬 코드

텍스트 셀

- 마크다운 형식

셀 실행 방식

- shift + Enter : 실행 후 다음 셀로 이동, 다음 셀이 없으면 생성
- ctrl + Enter : 현재 셀만 실행
- alt + Enter : 현재 셀 실행 후 다음 셀 생성

현재 사용 파이썬 점검

```
In [1]: import sys
        sys.version

Out[1]: '3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.191
        6 64 bit (AMD64)]'

In [2]: import site
        site.getsitepackages()

Out[2]: ['C:\\ProgramData\\anaconda3-23-09',
         'C:\\ProgramData\\anaconda3-23-09\\Lib\\site-packages']
```

파이썬 자료형과 리터럴

수 관련 자료형

- int, float, complex

```
In [3]: # %% 자료형 개요
        print(20, type(-7))
        print(-3.67, type(-3.18))
        print(0.34e1, 0.34E2)
        print(64.6784e-1, 123.34E-2)
        print(3 + 4j, type(5 - 4.3j))
```

```
20 <class 'int'>
-3.67 <class 'float'>
3.4 34.0
6.46784 1.2334
(3+4j) <class 'complex'>
```

문자열

첨자로 인덱싱

In [1]:
```
# %% 문자열 str
print('Python is powerful.')
print("파이썬은 쉬워요.")
# print('파이썬은 강력해요.")
# print("파이썬은 확장성이 좋아요.')
```

```
Python is powerful.
파이썬은 쉬워요.
```

In [2]:
```
print('''Beautiful is better than ugly.
Explicit is better than implicit.''')
```

```
Beautiful is better than ugly.
Explicit is better than implicit.
```

In [3]:
```
print('"python"')
print("'java'")
```

```
"python"
'java'
```

In [4]:
```
print('py'[0], 'py'[1])
```

```
p y
```

논리형

bool

```
In [5]: # %% bool
        print(3 > 4, type(3 < 4))

        False <class 'bool'>

In [6]: print(bool(0), bool(0.0), bool(''))
        print(bool(1), bool(0.1), bool('b'))

        False False False
        True True True

In [7]: # %% literals & variable
        print(True, False)
        # 3000으로 인식하지 못하며 출력문 내부에서 출력 자료의 구분자 쉼표로 인식
        print(3,000) # 출력 자료 3과 0으로 인식

        True False
        3 0
```

정수 리터럴 표현

10진수 표현

2진수, 8진수, 16진수 표현

```
In [8]: # %% various radix literals
        print(10, 0b10, 0B11) # ob oB 2진수
        print(11, 0o11, 0O12) # 0o 8진수
        print(12, 0x12, 0X13) # 0x 0X 16진수
        print(1_000, 1_000_000, 1_00) # 수에서 _

        10 2 3
        11 9 10
        12 18 19
        1000 1000000 100
```

주석

- # 이후
- ''' 여러 줄 주석
- '''

In [8]:
```
# %% comments

# 이후 줄 종료까지 주석
print('comments') # 이후 줄 종료까지 주석

''' 여러줄 주석에 활용
    수: int float complex
    문자열: str
    시퀀스형: list tuple
    매핑형: dict
    집합형: set frozenset
'''
```

```
comments
```

Out[8]:
```
' 여러줄 주석에 활용₩n    수: int float complex₩n    문자열: str₩n    시퀀스형:
list tuple₩n    매핑형: dict₩n    집합형: set frozenset₩n'
```

LESSON 02

변수와 모임 자료형

변수

필요한 자료를 저장하는 공간

In [9]:
```
# %%
print('파이썬은 1990년에 개발되었다.')
print('자바는 1995년에 개발되었다.')
```

```
파이썬은 1990년에 개발되었다.
자바는 1995년에 개발되었다.
```

In [10]:
```
prog_lang = '파이썬은'
devp_year = 1990
print(prog_lang, devp_year, '년에 개발되었다.')

prog_lang = '자바는'
devp_year = 1995
print(prog_lang, devp_year, '년에 개발되었다.')
```

```
파이썬은 1990 년에 개발되었다.
자바는 1995 년에 개발되었다.
```

변수 식별자의 조건과 키워드

영문자, _, 숫자

- 숫자로 시작은 불가능
- 다른 문자 불가능

키워드

- 언어에서 이미 정한 예약어
- 식별자로 사용 불가능

```
In [13]: # 2026_worldcup = '미국 캐나다 멕시코'
         worldcup_2026 = '미국 캐나다 멕시코'
         print('2026년 월드컵 개최국:', worldcup_2026)

         # total@money = 300000
         # name&person = 'tom'

         2026년 월드컵 개최국: 미국 캐나다 멕시코
```

키워드 보기

키워드와 내장 함수 help()

In [9]:
```
import keyword
print(keyword.kwlist)
print(len(keyword.kwlist))
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'clas
s', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with', 'yield']
35
```

In [10]:
```
# %%
help('keywords')
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

리스트

모든 항목의 목록

In [16]:
```
# %% list tuple
lst = [10, 20, 30]
print(lst, type(lst))
```

```
[10, 20, 30] <class 'list'>
```

In [17]:
```
lst = [1, 3.92, 3 > 4, 'list']
print(lst)
lst.append('py')
print(lst)
```

```
[1, 3.92, False, 'list']
[1, 3.92, False, 'list', 'py']
```

In [18]:
```
lst[0]
lst[4]
print(lst)
lst[1] = 'java'
print(lst)
```

```
[1, 3.92, False, 'list', 'py']
[1, 'java', False, 'list', 'py']
```

튜플

- 항목 추가 삭제 등 불가능
- 인덱싱 기능

In [19]:
```
# %% list tuple
tpl = (10, 20, 30)
print(tpl, type(tpl))
```

```
(10, 20, 30) <class 'tuple'>
```

In [20]:
```
tpl = 1, 3.92, 3 > 4, 'tuple'
print(tpl, type(tpl))

print(tpl[0])
print(tpl[3])
```

```
(1, 3.92, False, 'tuple') <class 'tuple'>
1
tuple
```

In [21]:
```
# tpl[1] = 100
# tpl.append('py') # 오류발생
```

사전

{키:값} 쌍의 목록, 인덱싱 불가능, 키로 참조

```
In [22]: # %% dict
         mart = {'라면':1200, '음료수':1500, '과자':800}
         print(mart, type(mart))
```

```
{'라면': 1200, '음료수': 1500, '과자': 800} <class 'dict'>
```

```
In [23]: mart['라면']
         mart['음료수']

         print(mart)
         mart['과자'] = 900
         print(mart)

         print(mart)
         mart['삼각김밥'] = 1700
         print(mart)

         mart.keys()
         mart.values()
```

```
{'라면': 1200, '음료수': 1500, '과자': 800}
{'라면': 1200, '음료수': 1500, '과자': 900}
{'라면': 1200, '음료수': 1500, '과자': 900}
{'라면': 1200, '음료수': 1500, '과자': 900, '삼각김밥': 1700}
```

```
Out[23]: dict_values([1200, 1500, 900, 1700])
```

집합

수정 가능, 중복 불허, 인덱싱 불가능

In [24]:
```
# %% set
basket = {'apple', 'orange', 'apple', 'pear'}
print(basket)

basket.add('banana')
print(basket)

# basket[0] # 오류
```

```
{'orange', 'apple', 'pear'}
{'orange', 'apple', 'banana', 'pear'}
```

In [25]:
```
myset = {1, 2, 3.4}
print(myset)
myset.add(True)
print(myset)

# myset.add([1, 2]) # 오류
```

```
{1, 2, 3.4}
{1, 2, 3.4}
```

frosenset

집합 set의 변형

- set: 수정 가능
- frozenset: 수정 불가능

```
In [26]: # %% frosenset
         pl = frozenset({'python', 'kotlin'})
         pl = frozenset(('python', 'kotlin'))
         pl = frozenset(['python', 'kotlin'])
         print(pl, type(pl))

         # pl.add('java')  # 오류

         frozenset({'python', 'kotlin'}) <class 'frozenset'>
```

내장 함수 id(obj)

obj의 메모리 주소

- 동일한 객체 의미: 메모리 주소 값인 id가 동일

In [27]:
```
# %% mutable vs immutable
help(id)
```

```
Help on built-in function id in module builtins:

id(obj, /)
    Return the identity of an object.

    This is guaranteed to be unique among simultaneously existing objects.
    (CPython uses the object's memory address.)
```

수정 불가능

Immutable

- int, float, str, tuple

정수 값이 같으면 id가 같은 특징

```
In [11]: print(id(2), id(1+1), id(3-1))

         140712818611016 140712818611016 140712818611016

In [12]: data = 10
         print(id(data))
         data = 10 + 1
         print(id(data))

         140712818611272
         140712818611304
```

변수 data의 값이 수정된 것이
아니라 다른 메모리에 이동되어
다시 값을 저장: 수정된 것이 아님

수정 가능

Mutable

- list, set, dict

```
In [31]: lst = [1, 2, 3]
         print(id(lst))

         lst.append(4)
         print(id(lst))
```

변수 lst는 변하지 않고 내용인 목록이 수정됨

```
2613030810944
2613030810944
```

수정불가능 immutable

‘hash() 호출이 가능하다’: immutable 의미

In [32]:
```
help(hash)
```

```
Help on built-in function hash in module builtins:

hash(obj, /)
    Return the hash value for the given object.

    Two objects that compare equal must also have the same hash value, but the
    reverse is not necessarily true.
```

In [33]:
```
s, a, x, t = 'py', 10, 3.4, (1, 2, 3)
print(hash(s), hash(a))
print(hash(x), hash(t))
```

```
4475000099362023540 10
922337203685477379 529344067295497451
```

수정가능 mutable

‘hash() 호출이 불가능하다’: mutable 의미

- 목록, 사전, 집합

```
In [34]: print(hash([1, 2, 3]))
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 print(hash([1, 2, 3]))

TypeError: unhashable type: 'list'

In [35]: print(hash({'one':1}))
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 print(hash({'one':1}))

TypeError: unhashable type: 'dict'

In [36]: print(hash({1, 2, 3}))
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 print(hash({1, 2, 3}))

TypeError: unhashable type: 'set'
```

SUMMARY

학습정리

파이썬 자료형

- 정수형
  - int, float, complex
- 문자열과 논리형
  - str, bool
- 모임 자료형
  - list, tuple, set, frozenset, dict

변수와 키워드

- 식별자로 이름을 지정 생성
  - data = 10
- 문법에 사용되도록 이미 정한 예약어
  - help('keywords')

수정가능(mutable) vs 수정불가능(immutable)

- Mutable
  - list, set, dict
- Immutable
  - int, float, str, tuple, frozenset