

Lesson 6

데이터 타입(Data type)

파이썬 기본 자료형



목차

- 1. 데이터 타입이란?
- 2. 데이터 타입 변경
- 3. 문자열(String)
 - 문자열 연산자
 - 문자열 접근(인덱싱, 슬라이싱)
 - 문자열 관련 함수들
 - 인코딩과 디코딩

데이터 타입(Data type)이란?

데이터 종류

데이터 타입(Data type)



■ 타입을 확인하려면 => type() 함수

```
a = 10
b = 4.4
c = 4 + 3j
d="hello"
e=True
print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
```

```
정수는 10진, 2진(0b), 8진(0o), 16진(0x) 정수가 있다.
ex) 123, -50, 0xFF, 0o77, 0b1111
지수형(e를 포함하는 숫자)
ex) 3.14e5(=3.14*10<sup>5</sup>)
```

```
<class 'int'>
  <class 'float'>
  <class 'complex'>
  <class 'str'>
  <class 'bool'>
```

데이터 타입(Data type) 확인



- 하나의 변수에 여러 개 데이터를 저장할 수 있다.
- 리스트(list), 튜플(tuple), 딕셔너리(dictionary), 셋(set)

```
list=[1,2,3,4]
tuple=(1,2,3)
dict={"name":"anna", "age":29}
set={1,2,"ace"}
                            <class 'list'>
print(type(list))
                            <class 'tuple'>
print(type(tuple))
                            <class 'dict'>
print(type(dict))
                            <class 'set'>
print(type(set))
```

3.8

'1000'

True

False

'a'

'5.6'

정수형으로

int()

1000

0

오류

오류



int()함수로 문자열인 값을 변경하는 경우 그 값은 반드시 정수여야 한다.

3

'1000'

True

False

'a'

'3.5'

실수형으로

float()

3.0

1000.0

1.0

0.0

오류

3.5

3.3

'1000'

0

'a'

. .

불형(논리형)으로

bool()

True

True

False

True

False

Boolean: 영국의 수학자였던 조지 부울(George Boole)의 이름을 따서 만든 부울 데이터형 참(True)이나 거짓(False)만 저장

숫자 0, 빈 문자열(''), None, 빈 리스트([]) 등 값이 없는 것은 False

None :변수에 아무 값도 없다는 것을 나타내기 위해 사용되는 상수

3

5.5

True

False

문자열로

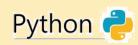
str()

'3'

'5.5'

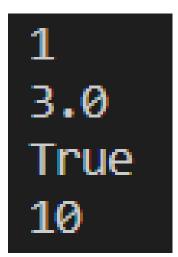
'True'

'False'

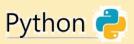


코드

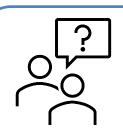
```
print(int(1.5))
print(float(3))
print(bool(1))
print(str(10))
```



숫자 관련 함수들



```
divmod() : 몫과 나머지, pow() : 제곱, round() : 반올림
 sum(): 합계, max(): 최대값, min(): 최소값
print(pow(2, 10))
   print(divmod(3, 2))
   list = [1, 2, 3, 4, 5]
   print(sum(list)) # 15
   print(max(list)) # 5
   print(min(list)) # 1
   print(0.1+0.2==0.3) # False
   print(0.5+0.25==0.75) # True
   print(round(1.55, 0)) # 2.0
   print(round(2.675, 2)) # 2.67
```



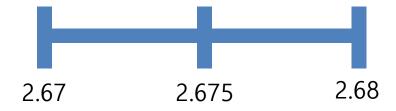
컴퓨터의 실수 연산이 정확 않은 이유

0.1 x 2	
0.2 x 2	0
<pre>0.4 x 2</pre>	0
0.8 x 2	0
1.6 0.6 x 2	1
1.2	1

```
10진수 : 0.1
10진수 : 0.25
2진수 : 0.01 ✔
   0.25
   x 2
    0.5
    1.0
```



round(2.675, 2) 가 2.68이 아닌 2.67이 되는 이유

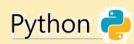


2.675은 2.67과 2.68 사이의 값이다.

2.675-2.67의 값은 0.0049999999999999893 2.68-2.675의 값은 0.0050000000000003375

2.67은 2.7과의 차이가 더 작으므로 더 가까운 쪽인 2.67이 결과로 나오게 된다.

decimal 모듈로 정확한 실수 연산



퀴드

```
from decimal import Decimal
a=Decimal("1.1")
b=Decimal("2.2")
                            -내장 모듈 decimal의 Decimal클래스를 사용
                            -Decimal클래스는 기본적으로 소수점이 28자리인 고정
print(a+b)
                            소수점 연산을 제공
                            -필요에 의해 자릿수를 더 늘릴 수 있다.
from decimal import *
num=Decimal("2.675")
rounded=num.quantize(Decimal("0.01"),rounding=ROUND UP)
print(rounded)
```



파이참에서 코드 내에서 모듈, 클래스, 또는 함수 이름을 선택한 후 바로 해당 정의를 열려면 블럭을 지정한 후 Ctrl키를 누르면서 클릭하면 된다.

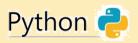
000

- 1.str(1+2+3) 의 결과는?
- 2.bool(0) 의 결과는?
- 3.bool("ace") 의 결과는?
- 4.bool([]) 의 결과는?

문자열(String)

따옴표(" " or ' ')로 묶인 모든 데이터

문자열 연산



+ : 연결

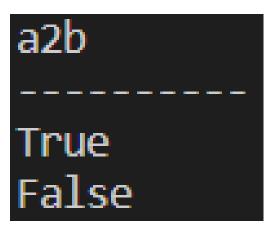
• * : 반복

• in : 지정한 값이 있으면 True

• not in : 지정한 값이 포함되어 있으면 False

코드

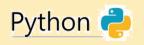
```
print('a'+'2'+'b')
print('-'*10)
print('h' in "happy")
print('h' not in "happy")
```



문자열은 문자의 순서를 나타내는 번호(index)가 붙는다.

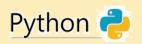


인덱싱(indexing)- 하나만 선택



- 문자열에서 특정 위치의 문자를 선택하는 것.
- 문자열[인덱스]
- 인덱스는 0부터 시작함. 인덱스를 뒤에서 지정하는 것도 가능.

슬라이싱(slicing)-여러 개 선택 가능



- 문자열에서 지정한 범위의 문자열을 추출.
- 문자열[시작 인덱스:종료 인덱스:증감값]





$$s[::2] \longrightarrow bnn$$

문자	b	a	n	а	n	a
index	0	1	2	3	4	5
index	-6	-5	-4	-3	-2	-1



1.문자열은 순차적으로 번호(index)가 붙는다. 이 번호의 시작은 []부터 시작한다.

```
2.아래의 결과는 무엇인가? [ ]
word="victory"
word[1:4]
```

도전!

[문제] 주민번호를 입력받아서 생년월일, 성별 기호, 맨뒤의 1문자 추출하는 프로그램 작성하기

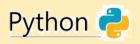
주민번호 입력: 031230-3123456

생년월일:031230

성별 기호:3

유효성 기호:6

문자열 관련 함수



- split() : 지정한 문자로 분리
- join() : 추가할문자열.join(기존문자열)
- count() : 찾는 문자의 개수
- find() : 찾는 문자의 위치
- startswith() : 지정한 문자열로 시작하면 True
- endswith() : 지정한 문자열로 끝나면 True
- replace() 메소드 : 지정한 문자를 다른 문자로 변경
- strip() : 양쪽 끝에 있는 문자열 제거. 인수가 없으면 공백을 제거함.
- upper() : 대문자로 변경

사용법 => 문자열.함수명()

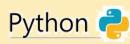
문자열 분리와 조인



코드

```
['hello', 'world']
s="hello world"
print(s.split())
s="test.py"
                        ['test', 'py']
print(s.split('.'))
s="abcd"
print('/'.join(s))
                        a/b/c/d
```

찾는 문자의 개수/찾는 문자의 위치



s="environment!"

```
print(s.count('e')) 2
                               index(): find() 와 동일한 함수.
print(s.find('v'))
print(s.find('s'))
print(s.startswith('f'))
                            False
print(s.endswith('!'))|
```



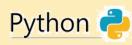
정규표현식(Regular Expression) 일정한 규칙(패턴)을 가진 문자열을 표현하는 방법. 특정한 규칙으로 된 문자열을 검색한 뒤 추출하거나 바꿀 때 또는 문자열이 정해진 규칙에 맞는지 판단할 때도 사용함. 이메일 형식 체크, 전화번호 유형 체크...

```
import re
s="environment!"
print(re.search("^e", s))
print(re.search("!$", s))
print(re.search("ron", s))
```

차이점 => find()는 찾는 문자열이

없으면 -1, index()는 없으면 오류 반환.

공백 제거, 대문자로 변경



코드

upper() : 대문자로 변환

lower() : 소문자로 변환

swapcase(): 대문자->소문자로, 소문자->대문자로

title() : 단어의 첫 글자만 대문자로 변환

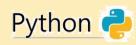
lstrip(): 왼쪽만 제거 rstrip(): 오른쪽만 제거

도전!

[문제] 이름을 영문으로 입력받고 각 단어의 첫 문자만 대문자로 변경하기

Enter your name:steve jobs
Steve Jobs

문자 코드, 코드 문자



• ord() : 지정한 문자의 유니 코드값

• chr() : 코드값에 해당하는 문자

• eval() : 수식을 문자열로 전달하면 계산 결과 반환

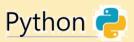
코드

```
print(ord('A'))
print(chr(65))
A
```

```
expr=input("수식 입력:")
print(eval(expr))
```

수식 입력: 10/2+5 10.0

인코딩(Encoding), 디코딩(Decoding)



- 컴퓨터에서는 모든 문자들에 번호를 붙여서 관리한다. 이런 번호를 문자 코 드라고 한다.
- 우리가 키보드로 'A'를 입력하면 A의 코드값인 65(01000001)로 변환되고 처리가 끝난 것을 사람이 인식할 수 있는 문자로 내보낸다.

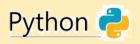
```
'A'-----> > 01000001 -----> 'A'
인코딩 디코딩
```

```
Dec Hx Oct Char
                                      Dec Hx Oct Html Chr Dec Hx Oct Html Chr Dec Hx Oct Html Chr
                                       32 20 040 6#32; Space 64 40 100 6#64; 6
                                                                              96 60 140 6#96;
0 0 000 NUL (null)
                                      33 21 041 4#33;
                                                            65 41 101 a#65; A
                                                                              97 61 141 4#97; 4
 1 1 001 SOH (start of heading)
                                                                              98 62 142 6#98; b
 2 2 002 STX (start of text)
                                      34 22 042 6#34;
                                                            66 42 102 B B
 3 3 003 ETX (end of text)
                                      35 23 043 4#35; #
                                                            67 43 103 4#67; C
                                                                              99 63 143 4#99;
                                                            68 44 104 @#68; D
 4 4 004 EOT (end of transmission)
                                      36 24 044 4#36;
                                                                             100 64 144 @#100; d
                                      37 25 045 % %
                                                            69 45 105 6#69; E 101 65 145 6#101; e
 5 5 005 ENQ (enquiry)
 6 6 006 ACK (acknowledge)
                                       38 26 046 4#38;
                                                            70 46 106 4#70; F
                                                                             102 66 146 6#102; f
 7 7 007 BEL (bell)
                                      39 27 047 4#39;
                                                            71 47 107 6#71; G
                                                                             103 67 147 6#103; g
 8 8 010 BS (backspace)
                                      40 28 050 (
                                                            72 48 110 6#72; H 104 68 150 6#104; h
 9 9 011 TAB (horizontal tab)
                                      41 29 051 6#41;
                                                            73 49 111 6#73; I
                                                                             105 69 151 6#105; 1
                                                            74 4A 112 @#74; J
10 A 012 LF (NL line feed, new line
                                     42 2A 052 4#42;
                                                                             106 6A 152 @#106; j
11 B 013 VT (vertical tab)
                                      43 2B 053 4#43; +
                                                            75 4B 113 6#75; K 107 6B 153 6#107; k
12 C 014 FF (NP form feed, new page) 44 2C 054 6#44;
                                                            76 4C 114 6#76; L 108 6C 154 6#108; L
13 D 015 CR (carriage return)
                                      45 2D 055 -
                                                            77 4D 115 6#77; M 109 6D 155 6#109; M
14 E 016 SO (shift out)
                                      46 2E 056 .
                                                            78 4E 116 6#78; N 110 6E 156 6#110; n
                                                           79 4F 117 4#79; 0 111 6F 157 4#111; 0
15 F 017 SI (shift in)
                                      47 2F 057 @#47;
                                      48 30 060 6#48;
                                                            80 50 120 P P
                                                                             112 70 160 p p
16 10 020 DLE (data link escape)
17 11 021 DC1 (device control 1)
                                                           81 51 121 6#81; Q 113 71 161 6#113; q
                                      49 31 061 6#49; 1
18 12 022 DC2 (device control 2)
                                      50 32 062 4#50; 2
                                                            82 52 122 6#82; R 114 72 162 6#114; r
19 13 023 DC3 (device control 3)
                                      51 33 063 6#51; 3
                                                            83 53 123 4#83; 5
                                                                             115 73 163 4#115; 8
                                                            84 54 124 6#84; T
20 14 024 DC4 (device control 4)
                                      52 34 064 6#52; 4
                                                                             116 74 164 4#116; t
21 15 025 NAK (negative acknowledge)
                                      53 35 065 4#53; 5
                                                            85 55 125 U U
                                                                             117 75 165 @#117; u
                                                            86 56 126 4#86; V 118 76 166 4#118; V
                                      54 36 066 4#54; 6
22 16 026 SYN (synchronous idle)
23 17 027 ETB (end of trans. block)
                                      55 37 067 4#55;
                                                           87 57 127 6#87; ₩ 119 77 167 6#119; ₩
24 18 030 CAN (cancel)
                                       56 38 070 4#56; 8
                                                            88 58 130 4#88; X 120 78 170 4#120; X
                                                            89 59 131 6#89; Y 121 79 171 6#121; Y
25 19 031 EM (end of medium)
                                      57 39 071 4#57; 9
                                       58 3A 072 4#58; :
                                                            90 5A 132 4#90; Z
26 1A 032 SUB (substitute)
                                                                             122 7A 172 6#122; Z
27 1B 033 ESC (escape)
                                      59 3B 073 &#59;;
                                                            91 5B 133 6#91; [
                                                                             123 7B 173 {
28 1C 034 FS (file separator)
                                      60 3C 074 4#60; <
                                                            92 5C 134 4#92; \
                                                                             124 7C 174 6#124;
29 1D 035 GS (group separator)
                                      61 3D 075 4#61; =
                                                           93 5D 135 6#93; 1 125 7D 175 6#125; }
                                                           94 5E 136 @#94; ^
                                                                             126 7E 176 @#126;
30 1E 036 RS (record separator)
                                      62 3E 076 > >
31 1F 037 US (unit separator)
                                      63 3F 077 4#63; ?
                                                           95 5F 137 6#95; _ 127 7F 177 6#127; DEL
```

```
Dec Hx Oct Html Chrl
                   Dec Hx Oct Html Chr
   40 100 &#64:
                             &#96:
                         140
                             &#97:
      101 A
                                    а
                             &#98:
      102 &#66:
                                    b
      103 C
                             &#99:
                                    С
      104 &#68:
                             &#100:
      105 E
                             e
70 46 106 &#70•
                  102 66 106 £#102•
```

https://namu.wiki/w/%EC%95%84%EC%8 A%A4%ED%82%A4%20%EC%BD%94%EB %93%9C

문자코드



- ASCII 코드 ASCII(American Standard Code for Information Interchange)
 - 미국 국립 표준 협회(ANSI)가 만든 7비트 코드
 - 최초의 문자열 코드이다.
 - 영문, 숫자, 기호 등 128(2⁷)개 문자 표현 가능



- ANSI 코드(=ASCII) 독일: ä ö ü, ß 프랑스: é à è
 - ASCII 확장판으로 8비트(256) 사용.
 - 한글은 8비트로 모두 나타낼 수 없다.

EUC-KR: 한글 지원을 위해 유닉스 계열에서 나온 완성형 코드. ANSI를 한국에서 확장한 것. 2바이트 사용. 똑약곳

CP949: MS가 EUC-KR을 개선, 확장해서 만든 완성형 코드.

■ 유니코드(Unicode)

- 전세계 모든 문자에 번호만 붙여놓은 코드
- 이걸 몇 비트를 사용해서 표현할지는 정해두지 않았음.
- 이 유니코드를 인코딩하는 방식이 UTF-8, UTF-16, UTF-32 가 있다.
- UTF-8: 1~4바이트 사용.



"abc".upper()



ord('a')

ord(), chr() 은 파이썬의 내장(Builtin) 함수이다.

앞서 사용해 본 print(), input() 등이 바로 파이썬 어디서든 바로 사용할 수 있는 내장 함수들이다. upper()같은 함수들은 사용하려면 앞에 문자열이 와야 한다.

다른 자료형도 같은 방식이다.

내장 함수 확인 방법 : dir(__builtins__)

문자열 함수 확인 방법: dir(str)