

# 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

- 타입을 확인하려면 => type() 함수

코드

```
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'>
```

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

코드

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

```
<class 'list'>
<class 'tuple'>
<class 'dict'>
<class 'set'>
```

데이터 형 변경

# 데이터 형 변경

3.8  
'1000'  
True  
False  
'a'  
'5.6'

정수형으로



int()

3  
1000  
1  
0  
오류  
오류



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

# 데이터 형 변경

3  
'1000'  
True  
False  
'a'  
'3.5'

실수형으로



float()

3.0  
1000.0  
1.0  
0.0  
오류  
3.5



# 데이터 형 변경

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

# 데이터 형 변경

3  
5.5  
True  
False

문자열로



`str()`

'3'  
'5.5'  
'True'  
'False'

코드

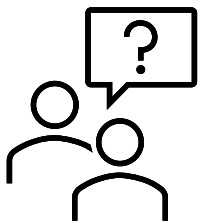
```
print(int(1.5))  
print(float(3))  
print(bool(1))  
print(str(10))
```

```
1  
3.0  
True  
10
```

divmod() : 몫과 나머지, pow() : 제곱, round() : 반올림  
sum() : 합계, max() : 최대값, min() : 최소값

코드

```
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  
정확하지  
않은  
이유

$0.1$	
$\times 2$	
<hr/>	
$0.2$	$0$
$\times 2$	
<hr/>	
$0.4$	$0$
$\times 2$	
<hr/>	
$0.8$	$0$
$\times 2$	
<hr/>	
$1.6$	$1$
$0.6$	
$\times 2$	
<hr/>	
$1.2$	$1$

10진수 :  $0.1$

2진수 :  $0.00011\dots$

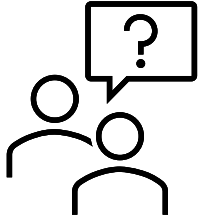


10진수 :  $0.25$

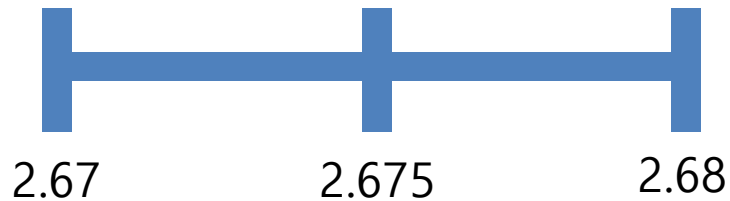
2진수 :  $0.01$



$0.25$	
$\times 2$	
<hr/>	
$0.5$	$0$
$\times 2$	
<hr/>	
$1.0$	$1$



`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 0.00499999999999999893

2.68-2.675의 값은 0.0050000000000000003375

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

## 코드

```
from decimal import Decimal
a=Decimal("1.1")
b=Decimal("2.2")
print(a+b)
```

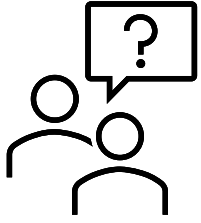


- 내장 모듈 decimal의 Decimal클래스를 사용
- Decimal클래스는 기본적으로 소수점이 28자리인 고정 소수점 연산을 제공
- 필요에 의해 자릿수를 더 늘릴 수 있다.

```
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 블록을 지정한 후 Ctrl키를 누르면서 클릭하면 된다.



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

- + : 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")
```


```
a2b  
-----  
True  
False
```

문자열은 문자의 순서를 나타내는 번호(index)가 붙는다.

```
s="hello"  
  0  1  2  3  4  
 -5 -4 -3 -2 -1
```

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

`s = "hello"`

`s[0]`       `h`

`s[-1]`      `o`

# 슬라이싱(slicing)-여러 개 선택 가능

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




시작 인덱스~종료 인덱스-1

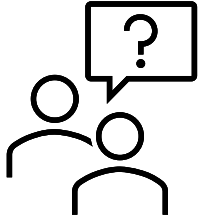
`s = "banana"`

`s[0:3]`  `ban`  
0~2

`s[::2]`  `bnn`  
0, 2, 4

`s[::-1]`  `ananab`  
-1,-2,-3,-4,-5,-6

문자	b	a	n	a	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
```

- `split()` : 지정한 문자로 분리
- `join()` : 추가할 문자열.`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()



코드

```
s="hello world"  
print(s.split())
```

```
['hello', 'world']
```

```
s="test.py"  
print(s.split('.'))
```

```
['test', 'py']
```

```
s="abcd"  
print('/'.join(s))
```

```
a/b/c/d
```

# 찾는 문자의 개수/찾는 문자의 위치

코드

```
s="environment!"
```

```
print(s.count('e'))
```

2

```
print(s.find('v'))
```

2

```
print(s.find('s'))
```

-1

```
print(s.startswith('f'))
```

False

```
print(s.endswith('!'))
```

True



index() : find() 와 동일한 함수.

차이점 => find()는 찾는 문자열이 없으면 -1, index()는 없으면 오류 반환.



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))
```

코드

```
s=" environment!"
```

```
print(s.strip())
```

environment!

```
print(s.strip('!'))
```

environment

```
print(s.upper())
```

ENVIRONMENT!



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
```

- `ord()` : 지정한 문자의 유니 코드값
- `chr()` : 코드값에 해당하는 문자
- `eval()` : 수식을 문자열로 전달하면 계산 결과 반환

코드

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

```
65  
A
```

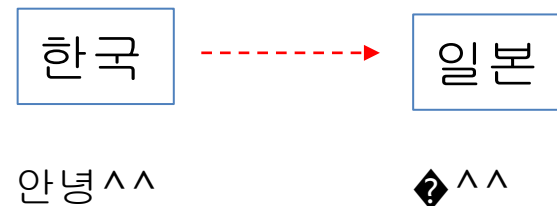
```
expr=input("수식 입력 :")  
print(eval(expr))
```

```
수식 입력: 10/2+5  
10.0
```

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

## ■ 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( $2^7$ )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. **똥양궁**

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.



현재 시스템의 인코딩 확인

```
import sys  
print(sys.stdin.encoding)
```

"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)`