

자료형

숫자형 (Number)

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
복소수	1 + 2j, -3j
8진수	0o34, 0o25
16진수	0x2A, 0xFF

- 연산자

- 사칙연산자 : +, -, *, /
- ** : x^y
- % : 나머지 연산자
- // : 소수점 자리 버리는 연산자
- +=, -=, *=, /=
- <, >, <=, >=, ==, !=
- and, or, not

숫자형 (Number)

- 숫자 출력 포맷

Syntax	Output
<code>print('I have %d cats' % 6)</code>	I have 6 cats
<code>print('I have %3d cats' % 6)</code>	I have 6 cats
<code>print('I have %03d cats' % 6)</code>	I have 006 cats
<code>print('I have %f cats' % 6)</code>	I have 6.000000 cats
<code>print('I have %.2f cats' % 6)</code>	I have 6.00 cats
<code>print("I have {0:d} cats".format(6))</code>	I have 6 cats
<code>print("I have {0:3d} cats".format(6))</code>	I have 6 cats

- 여러 개 다른 출력

```
print("I have %d cats and %d dogs"% (5,6))
```

```
print("Here are the numbers!" + W
```

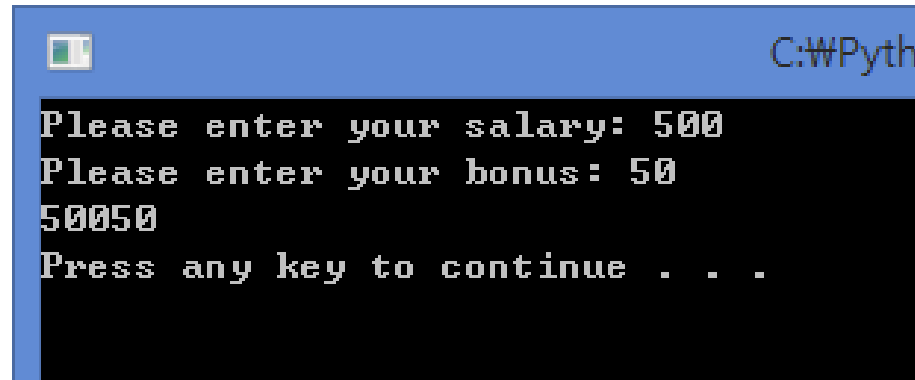
```
    " The first is{0:d} The second is{1:4d} ".format(7,8))
```

숫자형 (Number)

- 사용자로부터 숫자 입력 받기

```
salary = input("Please enter your salary: ")
bonus = input("Please enter your bonus: ")
payCheck = salary + bonus
print(payCheck)
```

- input 함수는 String을 반환함



```
C:\WPyth
Please enter your salary: 500
Please enter your bonus: 50
50050
Press any key to continue . . .
```

- 형변환 함수

int(value)	converts to an integer
long(value)	converts to a long integer
float(value)	converts to a floating number (i.e. a number that can hold decimal places)
str(value)	converts to a string

숫자형 (Number)

- 형변환 하기

```
salary = input("Please enter your salary: ")
bonus = input("Please enter your bonus: ")
payCheck = salary + bonus
payCheck = float(salary) + float(bonus)
print(payCheck)
```

- round 함수

- round(data, 2)
- data를 소수점 2자리 까지 반올림하여 나타냄

- type 함수

- 변수의 데이터형을 확인하는 함수

```
print(type(payCheck))
```

문자열(String)

- 문자열은 (') 과 (")를 이용하여 나타냄

```
print('Hickory Dickory Dock! The mouse ran up the clock')
```

```
print("Hickory Dickory Dock! The mouse ran up the clock")
```

```
print("It's a beautiful day in the neighborhood")
```

```
print('It\'s a beautiful day in the neighborhood')
```

- 주의 할 것

```
print('It's a beautiful day in the neighborhood')
```

- 개행문자 사용가능

```
print('Hickory Dickory Dock!\nThe mouse ran up the clock')
```

문자열(String)

- Tripple quotes (`'''`), (`"""`)
 - 여러줄에 해당하는 문자열 처리할 때 사용
 - #과 함께 주석처리에도 사용됨

```
print("""Hickory Dickory Dock!  
The mouse ran up the clock""")
```

```
print(' 'Hickory Dickory Dock!  
The mouse ran up the clock' ')
```

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

문자열(String)

- 문자열 연산

- + 문자열 연결

```
firstName = input("What is your first name? ")
lastName = input("What is your last name? ")
print("Hello " + firstName + " " + lastName)
```

- * 문자열 반복

```
print("="*10)
```

- 문자열 인덱싱

- 0부터 시작, 끝은 -1 색인 값을 가짐

```
name="greenjoa"
print(name[0])
print(name[-1])
```


문자열(String)

- 문자열 연산

- 문자열 슬라이싱

- 문자열의 일부분을 가져오기
 - `0 <= name < 3`
 - `name[0], name[1], name[2]`
 - `name[:5]` : 처음부터 지정된 범위까지
 - `name[5:]` : 지정된 위치부터 끝까지

```
name="greenjoa"  
print(name[0:3])
```

```
info = "201012345greenjoa"  
sid = info[:9]  
sname = info[9:]  
print(sid + " " + sname)
```

- 문자열 수정(??)

- 문자열은 상수로 수정할 수 없음

```
a="python"  
a[1] = 'y'
```



```
a="python"  
a=a[:1]+'y'+a[2:]  
print(a)
```

문자열(String)

- 문자열 포맷

```
a = "I eat %s apples." % "five"  
print(a)
```

- 오른쪽 정렬

```
a = "I eat %10s apples." % "five"
```

- 왼쪽 정렬

```
a = "I eat %-10s apples." % "five"
```

```
a = "Error is %d%%." % 98  
print(a)
```

Error is 98%.

문자열(String)

- 문자열 포맷

- 이름으로 치환하기

"I ate {number} apples. so I was sick for {day} days.".format(number=10, day=3)

- 정렬하기

좌측 정렬 : "{0:<10}".format("hi")

우측 정렬 : "{0:>10}".format("hi")

가운데 정렬 : "{0:^10}".format("hi")

- 공백 채우기

"{0:=^10}".format("hi")

문자열(String)

- 문자열 함수

- 문자열 길이 : len(s)
- 각종 함수

```
message = 'Hello world'
print(message.lower())
print(message.upper())
print(message.swapcase())
print(message.find('world'))
print(message.index('o'))
print(message.count('o'))
print(message.capitalize())
print(message.replace('Hello', 'Hi'))
```

문자열(String)

- 문자열 함수

- 문자열 삽입(join)

```
a= ","  
a.join('abcd')
```

```
'a,b,c,d'
```

- 왼쪽/오른쪽/모두 공백 및 개행문자 지우기(lstrip / rstrip/strip)

```
a = " hi"      a = "hi "  
a.lstrip()     a.strip()
```

- 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']
```

List

- 여러 개의 데이터를 저장하기 위한 자료형
 - 어떠한 자료형도 포함시킬 수 있음
- 일반적으로 정보를 **단위**로 저장함
 - 서로 다른 값들은 다른 리스트로 만드는 것이 좋음
 - 이름, 주소, 전화번호 각기 다른 리스트로 만듦
- 앞에서 액세스하는 **인덱스는 0**부터 , 끝에서 액세스 인덱스 -1부터

```
a = []  
b = [1, 2, 3]  
c = ['Life', 'is', 'too', 'short']  
d = [1, 2, 'Life', 'is']  
e = [1, 2, ['Life', 'is']] #e[2][0]
```

b+c ??

b*3 ??

```
guests = []
```

```
score = []
```

```
guests = ['a','b','c','d']  
score = [78,85,62,49,98]
```

```
print(guests[0])  
print(score[3])
```

```
print(guests[-1])  
print(score[-3])
```

List

- 리스트 변경

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

```
guests[0]='greenjoa'
```

```
guests[1]=['greenjoa1','greenjoa2']
```

```
guests[1:2]=['greenjoa1','greenjoa2']
```

```
guests.insert(2, 'e') #2번 인덱스에 'e' 삽입
```

```
guests.append('greenjoa2') # 마지막에 추가됨
```

```
print(guests[-1])
```

```
guests.remove('c')
```

```
guests[1:2]=[]
```

```
del guest[0]
```

```
print(guests.index('c')) # 데이터가 없으면 에러 발생
```

List

- 리스트 멤버 액세스

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

```
for steps in range(4) :  
    print(guests[steps])
```

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

```
nEntries = len(guests)  
for steps in range(nEntries) :  
    print(guests[steps])
```

```
for guest in guests :  
    print(guest)
```

```
scores = [85,62,63,45,90]
```

```
scores.sort()  
scores.reverse()
```


List

- 리스트 요소 액세스

- pop() : 리스트의 맨 마지막 요소를 돌려주고 그 요소는 삭제
- pop(x) : x번째 요소를 돌려주고 삭제

```
scores = [85,62,63,45,90]
num = scores.pop()
num = scores.pop(2)
```

- count(x) : x의 개수를 돌려주는 함수

```
scores = [85,63,63,45,90]
num = scores.count(63)
print(num)
```

- extend(x) : 리스트 x를 더하는 함수

```
scores = [85,63,63,45,90]
scores.extend([50,60])
print(scores)
```

Tuple

- 리스트와 유사한 성질은 가지는 자료형
 - 차이점
 - 리스트는 []로 둘러 싸지만, 튜플은 ()로 이용함
 - 리스트는 값을 생성, 삭제, 수정이 가능하지만, 튜플은 그 값을 변화시킬 수 없음

t1 = ()

t2 = (1,) # t2 = (1)과 다름

t3 = (1,2,3)

t4 = 1,2,3

t5 = ('a', 'b', ('ab', 'cd'))

Tuple

- 튜플 조작하기
 - 리스트와 마찬가지로 **인덱싱 가능** : `a[0]`
 - **슬라이싱** : `a[1:]`
 - 더하기 : `a+b` → 두개의 튜플을 합하기
 - 곱하기 : `a*3` → `a` 튜플을 3번 반복하기
 - **변경 연산시 에러 발생함을 주의 할 것**

딱 그만큼

Dictionary

- Key 와 Value를 한 쌍으로 갖는 자료형
 - Key로 Value값을 찾아낼 수 있는 자료형
 - Key는 중복되지 않도록 할 것
 - Key는 변할 수 없는 값으로 해야함. 튜플은 되고, 리스트는 안됨

```
dic1={}
dic2 = dict()
dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
a = {1: 'hi'}
b = { 'a': [1,2,3]}

print(dic['name'])
```

- 데이터 추가, 삭제하기

```
a = {1: 'a'}
a[2] = 'b'
a['name'] = 'pey'
a[3] = [1,2,3]

del a['name']
```

Dictionary

- 리스트 만들기

```
a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}  
b=a.keys()  
print(b)
```

```
b=list(a.keys()) # 리스트 반환
```

```
a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}  
b=a.values()  
print(b)
```

```
b=list(a.values())# 리스트 반환
```

Dictionary

- Key, Value 쌍 얻기

- 튜플로 묶은 값을 반환함

```
a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
```

```
b=a.items()
```

```
print(b)
```

- 지우기

- **a.clear()**

- Key로 value 얻기

- **a.get('name')**

- 없는 키값일 경우 None을 반환함
- a['name']은 오류를 발생시킴

- **a.get('foo','bar')**

- 'foo'라는 키가 없을 경우 디폴트 값인 'bar'를 반환함

- 해당 Key가 있는지 조사 : 'name' **in** a

Sets

- 집합 데이터를 처리하는 자료형
 - 중복을 허용하지 않음
 - 순서가 없음

```
s1 = set([1,1,2,3,4,5,2,3,6])  
a=list(s1)  
print(a)
```

- 교집합/합집합/차집합

```
s1 = set([1,2,3,4,5,6])  
s2 = set([4,5,6,7,8,9])  
s3 = s1 & s2  
s4 = s1.intersection(s2)  
s5 = s1 | s2  
s6 = s1.union(s2)  
s7 = s1 - s2  
s8 = s1.difference(s2)
```

Sets

- 추가/삭제

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

```
s1.add(4)
```

```
s1.update([4,5,6]) #여러 개의 값을 추가할 때
```

```
s1.remove(2)
```


참과 거짓

- 자료형에도 참과 거짓이 있음
 - 문자열, 리스트, 튜플, 딕셔너리 등의 값이 비어 있으면 거짓, 아니면 참

값	참 or 거짓
"python"	참
""	거짓
[1, 2, 3]	참
[]	거짓
()	거짓
{}	거짓
1	참
0	거짓
None	거짓