Bult-In함수

03

파이썬 내장함수를 설명한다. 파이썬 사용설명서 내용을 정리한 수준이다.

내장함수 & 참_거짓

내장함수

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

```
print('길이= % 2d, 최대= % 4.2f, , 최대= % 4.2f ' % (len(x), max(x), min(x)))
```

```
길이=  4, 최대=  11.00, , 최대= -3.00
```

```
x=[1,-3,11,9]
sorted(x,reverse=True)   #sorted by size(reverse)
```

```
[11, 9, 1, -3]
```

```
str(x)   #convert to string
```

```
'[1, -3, 11, 9]'
```

bool(), _len() - 참 혹은 거짓

- constants defined to be false: `None` and `False`.
- zero of any numeric type: `0`, `0.0`, `0j`, `Decimal(0)`, `Fraction(0, 1)`
- empty sequences and collections: `''`, `()`, `[]`, `{}`, `set()`, `range(0)`

Boolen 연산자

Operation	Result
`x or y`	if *x* is false, then *y*, else *x*
`x and y`	if *x* is false, then *x*, else *y*
`not x`	if *x* is false, then `True`, else `False`

Operation	Meaning
`<`	strictly less than
`<=`	less than or equal
`>`	strictly greater than
`>=`	greater than or equal
`==`	equal
`!=`	not equal
`is`	object identity
`is not`	negated object identity

```
In [53]: not 5<2
Out[53]: True

In [56]: (5<2) or (3<5)
Out[56]: True

In [57]: (5<2) and (3<5)
Out[57]: False
```

Numeric Type

Numeric (only) int, float, numeric

Operation	Result
`x + y`	sum of *x* and *y*
`x - y`	difference of *x* and *y*
`x * y`	product of *x* and *y*
`x / y`	quotient of *x* and *y*
`x // y`	floored quotient of *x* and *y*
`x % y`	remainder of `x / y`
`-x`	*x* negated
`+x`	*x* unchanged
`abs(x)`	absolute value or magnitude of *x*
`int(x)`	*x* converted to integer
`float(x)`	*x* converted to floating point
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.
`c.conjugate()`	conjugate of the complex number *c*
`divmod(x, y)`	the pair `(x // y, x % y)`
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*

Operation	Result
`math.trunc(x)`	*x* truncated to `Integral`
`round(x[, n])`	*x* rounded to *n* digits, rounding half to even. If *n* is omitted, it defaults to 0.
`math.floor(x)`	the greatest `Integral` <= *x*
`math.ceil(x)`	the least `Integral` >= *x*

`float.`**`is_integer`**`()`

Return `True` if the float instan

```
>>> (-2.0).is_integer()
True
>>> (3.2).is_integer()
False
```

Sequence

Sequence operation

Operation	Result
`x in s`	`True` if an item of *s* is equal to *x*, else `False`
`x not in s`	`False` if an item of *s* is equal to *x*, else `True`
`s + t`	the concatenation of *s* and *t*
`s * n` or `n * s`	equivalent to adding *s* to itself *n* times
`s[i]`	*i*th item of *s*, origin 0
`s[i:j]`	slice of *s* from *i* to *j*
`s[i:j:k]`	slice of *s* from *i* to *j* with step *k*
`len(s)`	length of *s*
`min(s)`	smallest item of *s*
`max(s)`	largest item of *s*
`s.index(x[, i[, j]])`	index of the first occurrence of *x* in *s* (at or after index *i* and before index *j*)
`s.count(x)`	total number of occurrences of *x* in *s*

Range examples:

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

```
txt='역사는 더디다. 그러나 인간이 소망하는 희망의 등불은 쉽게 꺼지지 않는다'
'인간' in txt
```

```
True
```

```
'인간' not in txt
```

```
False
```

```
len(txt) #문자 총길이
```

```
40
```

```
txt[3:8]  #0=시작, 3~7위치 4개단어
```

```
' 더디다.'
```

```
txt.count('인간')  #지정단어 개수
```

```
1
```

Operation	Result
`s[i] = x`	item *i* of *s* is replaced by *x*
`s[i:j] = t`	slice of *s* from *i* to *j* is replaced by the contents of the iterable *t*
`del s[i:j]`	same as `s[i:j] = []`
`s[i:j:k] = t`	the elements of `s[i:j:k]` are replaced by those of *t*
`del s[i:j:k]`	removes the elements of `s[i:j:k]` from the list
`s.append(x)`	appends *x* to the end of the sequence (same as `s[len(s):len(s)] = [x]`)
`s.clear()`	removes all items from *s* (same as `del s[:]`)
`s.copy()`	creates a shallow copy of *s* (same as `s[:]`)
`s.extend(t)` or `s += t`	extends *s* with the contents of *t* (for the most part the same as `s[len(s):len(s)] = t`)
`s *= n`	updates *s* with its contents repeated *n* times
`s.insert(i, x)`	inserts *x* into *s* at the index given by *i* (same as `s[i:i] = [x]`)
`s.pop([i])`	retrieves the item at *i* and also removes it from *s*
`s.remove(x)`	remove the first item from *s* where `s[i]` is equal to *x*
`s.reverse()`	reverses the items of *s* in place

```
num=[1,-3,11,9]
num[0]=-7
num
```

```
[-7, -3, 11, 9]
```

```
num[0:2]=[] #del num[0:2] 동일
num
```

```
[11, 9]
```

```
num.append(-7)   #마지막에 지정한 -7 추가
num
```

```
[11, 9, -7]
```

```
num.reverse()
num
```

```
[-7, 9, 11]
```

```
num.remove(9)   #지정한 숫자,문자 제일 먼저 나온 값 제거
num
```

```
[-7, 11]
```

```
num.insert(0,1)   #지정한 위치 0에 1을 삽입
num
```

```
[1, -7, 11]
```

Text Sequence String(문자열)

str.Method()

```
s=str('i AM invincible. 나는 생각한다. 고로 존재한다')
```

len(s)

33

예제 사용되는 문자열 s의 길이는 33이다.

capitalize() 첫 글자 대문자

```
s.capitalize()
```

'I am invincible. 나는 생각한다. 고로 존재한다'

casefold() 모든 글자 소문자

```
s.casefold()
```

'i am invincible. 나는 생각한다. 고로 존재한다'

center(크기, '채움문자') 문자열은 중앙 배열한다.

문자열을 크기 40인 문자열 가운데 배열, 예제와 달리 채움문자 없는 경우에는 공백으로 채워진다.

```
s.center(40,'*')
```

'***i AM invincible. 나는 생각한다. 고로 존재한다****'

count('찾는문자') 문자열 중 찾는 단어 개수

```
In [34]: s.count('다')
Out[34]: 2
```

encode() str.encode(encoding="utf-8")

- 문자열을 정해진 형식으로 인토딩한다.

endswith('지정단어', 시작포지션, 끝포지션)

- 지정단어로 끝나는지 여부를 False, True로 반환한다.
- 끝 포지션은 -1이 된다. 즉 s 문자열 위치 0에서 2까지 'i A' 3글자의 끝자리가 A인지 알게 된다.

```
s.endswith('다')
```

True

```
s.endswith('A',0,3)
```

True

expandtabs() Sets the tab size of the string

```
>>> '01\t012\t0123\t01234'.expandtabs()
'01      012     0123    01234'
>>> '01\t012\t0123\t01234'.expandtabs(4)
'01  012 0123    01234'
```

find('찾는문자',시작위치,끝위치) 찾는 문자 존재한다면 제일 먼저 나타난 위치 반환, 없다면 -1 반환

rfind() find() 동일하나 마지막 위치 반환, 없다면 -1 반환

```
s.find('AM')
2
```

```
s.find('다')
23
```

```
s.find('를')
-1
```

```
>>> 'Py' in 'Python'
True
```

index('찾는문자',시작위치,끝위치)

rindex() index() 동일하나 마지막 위치 반환

- find()와 동일하나 없는 경우에는 오류가 반환

```
s.index("를")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-48-38c239b6ef4c> in <module>()
----> 1 s.index("를")

ValueError: substring not found
```

isalnum() 문자열에 알파벳와숫자 연결 포함여부(True, False) 반환

```
s1='age58' ; s2='age 58'
print(s1.isalnum(), '***' ,s2.isalnum())
```

```
True *** False
```

isalpha() 문자열에 알파벳만 있는지 여부 (True, False) 반환

```
s1='age58' ; s2='age 58'
print(s1.isalpha(), '***' ,s2.isalpha())
```

```
False *** False
```

isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case

ljust(문자열길이) 지정길이만큼 문자열을 왼쪽 정렬 결과 반환

rjust(지정길이) 지정길이만큼 문자열을 우쪽 정렬 결과 반환

```
s.ljust(40)
```

```
'i AM invincible. 나는 생각한다. 고로 존재한다      '
```

lower() 문자열을 소문자 결과로 반환한다.

upper() 문자열을 대문자 결과로 반환한다.

```
s.upper()
```

```
'I AM INVINCIBLE. 나는 생각한다. 고로 존재한다'
```

lstrip('제외시작단어') 문자열을 왼쪽 정렬하고 제외시작단어로 시작하는 단어는 제외한 결과 반환.

```
txt = "     banana     "
print("Of all fruits, ", txt.lstrip(), "is my favorite")
```

```
Of all fruits,  banana     is my favorite
```

```
txt = ",,,,,ssaaww.....banana"
x = txt.lstrip(",.asw")
print(x)
```

```
banana
```

partition('분리단어') 문자열을 분리단어가 처음 나타난 곳에서 분리되어 tuple 형식으로 저장된다. 만약 분리단어가 없다면 원 문자열과 동일하다.

```
s.partition('AM')
```

```
('i ', 'AM', ' invincible. 나는 생각한다. 고로 존재한다')
```

rpartition('분리단어') partition과 동일하나 지정단어 마지막 위치에서 분리된다.

```
txt = "I could eat bananas all day, bananas are my favorite fruit"
x = txt.rpartition("bananas")
print(x)
```

```
('I could eat bananas all day, ', 'bananas', ' are my favorite fruit')
```

```
txt = "I could eat bananas all day, bananas are my favorite fruit"
x = txt.partition("bananas")
print(x)
```

```
('I could eat ', 'bananas', ' all day, bananas are my favorite fruit')
```

replace('원단어','대체단어') 원 단어가 대체단어로 바뀐다.

```
s.replace('나는', '우리는')
```

```
'i AM invincible. 우리는 생각한다. 고로 존재한다'
```

split('지정단어',최대개수) 지정단어로 분리하여 리스트로 저장된다. 최대개수는 분리 개수를 지정한다. (없다면 모두 분리)

rsplit() 위와 동일하나 오른쪽부터 분리한다.

```
s.split('AM')
```

```
['i ', ' invincible. 나는 생각한다. 고로 존재한다']
```

'지정단어' 없다면 공백으로 분리

```
s.split()
```

```
['i', 'AM', 'invincible.', '나는', '생각한다.', '고로', '존재한다']
```

최대개수는 지정단어 사용 회수이다. r은 오른쪽부터 시작한다.

```
txt = "apple, banana, cherry"
txt.split(", ", 1)
```

```
['apple', 'banana, cherry']
```

```
txt = "apple, banana, cherry"
txt.rsplit(", ", 1)
```

```
['apple, banana', 'cherry']
```

splitlines() 문자열 리스트를 라인 수로 나누어 리스트에 저장한다.

Representation	Description
`\n`	Line Feed
`\r`	Carriage Return
`\r\n`	Carriage Return + Line Feed
`\v` or `\x0b`	Line Tabulation
`\f` or `\x0c`	Form Feed
`\x1c`	File Separator
`\x1d`	Group Separator
`\x1e`	Record Separator
`\x85`	Next Line (C1 Control Code)
`\u2028`	Line Separator
`\u2029`	Paragraph Separator

```
txt = "Thank you for the music \n Welcome to the jungle"
print(txt.splitlines())
```

```
['Thank you for the music ', ' Welcome to the jungle']
```

라인 브레이커(\n) 저장한다.

```
txt = "Thank you for the music \n Welcome to the jungle"
print(txt.splitlines(True))
```

```
['Thank you for the music \n', ' Welcome to the jungle']
```

strip() 문자열 양쪽 공백 모두 없앤 결과 반환

rstrip() 오른쪽 공백만 없앤 결과 반환

```
txt = "     banana     "
print("of all fruits", txt.strip(), "is my favorite")
```

```
of all fruits banana is my favorite
```

```
txt = "     banana     "
print("of all fruits", txt.rstrip(), "is my favorite")
```

```
of all fruits     banana is my favorite
```

startswith('지정단어') 문자가 지정단어로 시작하는지 여부

```
s.startswith('i')
```

```
True
```

```
s.startswith('I')
```

```
False
```

swapcase() 대문자->소문자, 소문자->대문자 변환

```
txt = "Hello My Name Is PETER"
txt.swapcase()
```

```
'hELLO mY nAME iS peter'
```

title() 모든 단어 대문자

```
txt = "hello my name is peter"
txt.title()
```

```
'Hello My Name Is Peter'
```

zfill(길이) 문자열을 지정한 길이만큼되도록 앞에 0을 넣는다. 부호가 있는 경우에는 +, - 뒤에 넣는다.

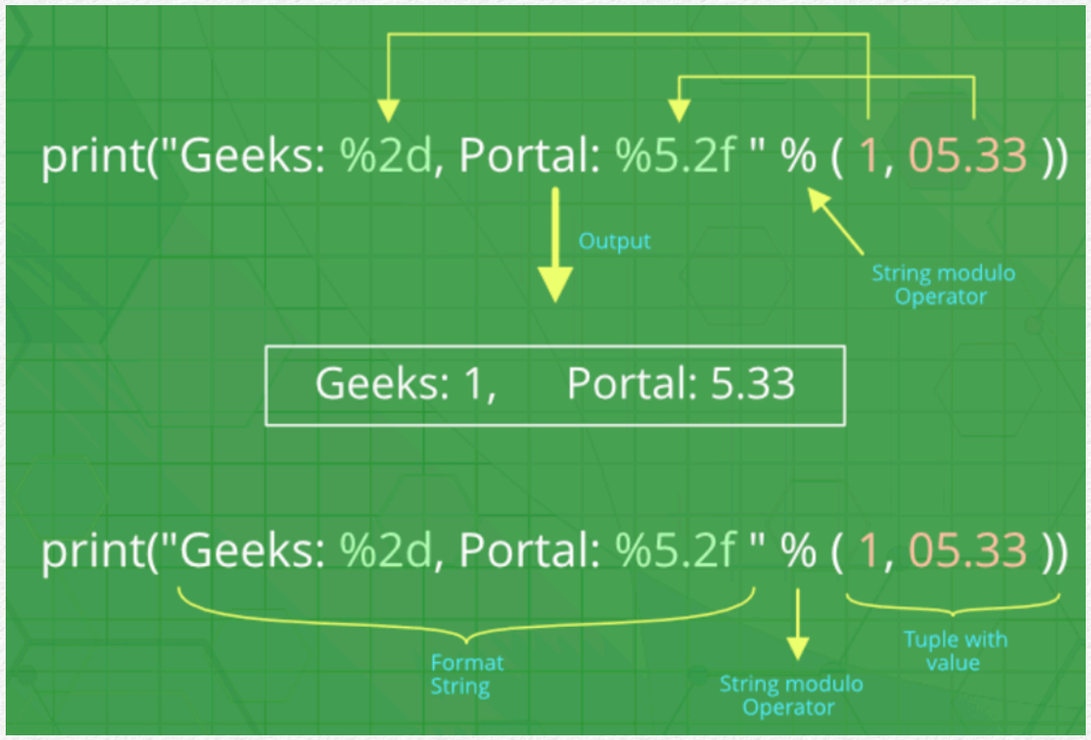
```
s='-137'
s.zfill(7)
```

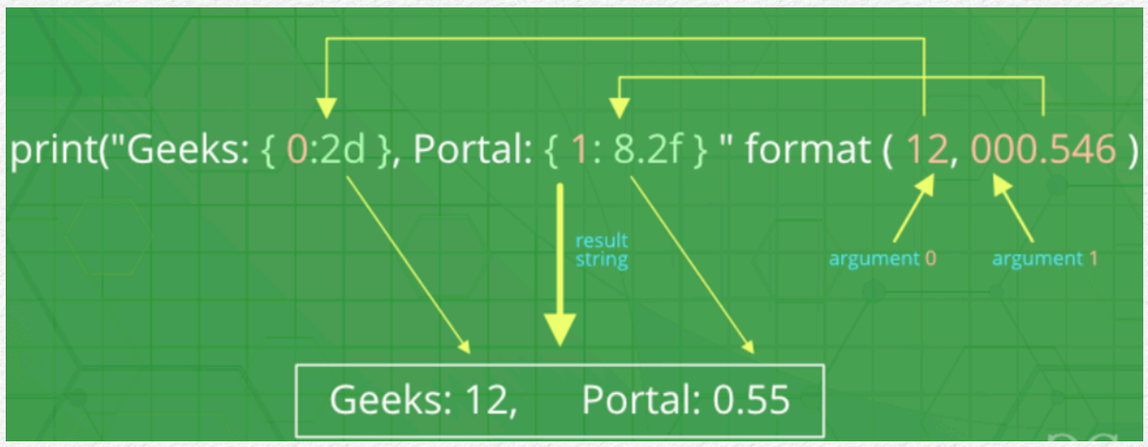
```
'-000137'
```

```
s='wolfpack'
s.zfill(10)
```

```
'00wolfpack'
```

Print Style





Conversion	Meaning
`'d'`	Signed integer decimal.
`'i'`	Signed integer decimal.
`'o'`	Signed octal value.
`'u'`	Obsolete type – it is identical to `'d'`.
`'x'`	Signed hexadecimal (lowercase).
`'X'`	Signed hexadecimal (uppercase).
`'e'`	Floating point exponential format (lowercase).
`'E'`	Floating point exponential format (uppercase).
`'f'`	Floating point decimal format.
`'F'`	Floating point decimal format.
`'g'`	Floating point format. Uses lowercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
`'G'`	Floating point format. Uses uppercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
`'c'`	Single character (accepts integer or single character string).
`'r'`	String (converts any Python object using `repr()`).
`'s'`	String (converts any Python object using `str()`).
`'a'`	String (converts any Python object using `ascii()`).
`'%'`	No argument is converted, results in a `'%'` character in the result.

```
str = "I love geeksforgeeks"

print (str.center(40, '#'))
```

```
##########I love geeksforgeeks##########
```

```
print (str.ljust(40, '-'))
```

```
I love geeksforgeeks----------------------
```

```
# print integer and float value
print("Geeks : % 2d, Portal : % 5.2f" %(1, 05.333))
```

```
Geeks :  1, Portal :  5.33
```

```
# print exponential value
print("% 10.3E"% (356.08977))
```

```
 3.561E+02
```

```
# print octal value
print("% 7.3o"% (25))
```

```
    031
```

```
s='my age'; n=58
print('내 나이는 ',n, '살',s)
```

```
내 나이는  58 살 my age
```

```
print('{0} and {1}'.format('Geeks', 'Portal'))
```

```
Geeks and Portal
```

```
# combining positional and keyword arguments
print('Number one portal is {0}, {1}, and {other}.'.format('Geeks', 'For', other ='Geeks'))
```

```
Number one portal is Geeks, For, and Geeks.
```

```
# using format() method with number
print("Geeks :{0:2d}, Portal :{1:8.2f}".format(12, 00.546))
```

```
Geeks :12, Portal :    0.55
```

```
# Changing positional argument
print("Second argument: {1:3d}, first one: {0:7.2f}".format(47.42, 11))
```

```
Second argument:  11, first one:   47.42
```

```
print("Geeks: {a:5d},  Portal: {p:8.2f}".format(a = 453, p = 59.058))
```

```
Geeks:   453,  Portal:    59.06
```

```
# using format() in dictionary
tab = {'geeks': 4127, 'for': 4098, 'geek': 8637678}
print('Geeks: {0[geeks]:d}; For: {0[for]:d}; ' 'Geeks: {0[geek]:d}'.format(tab))
```

```
Geeks: 4127; For: 4098; Geeks: 8637678
```