Python Cheatsheet

find ( )

```
>>> string = "ABC"

>>> string.find('A')
0
>>> string.find('D')
-1
```

index ( )

```
>>> string = "ABC"

>>> string.index('A')
0
>>> string.index('D')
ValueError: substring not found
```

replace ( )

```
>>> string = "ABC"

>>> string.replace("A", "M")
MBC
```

join ( )

```
>>> li = ['A', 'B', 'C', 'D']

>>> ''.join(li)
ABCD
```

```
>>> '-'.join(li)
A-B-C-D
```

split ( )

```
>>> str = "A-B-C"

>>> str.split('-')
['A', 'B', 'C']
```

bool ( )

```
>>> bool(0)
False
>>> bool(1)
True

>>> bool('')
False
>>> bool(' ')
True

>>> bool([])
False
>>> bool([0])
True
```

자료형 변환

```
>>> int(9.7)
9
>>> float('10')
10.0
>>> str(13)
'13'
```

print ( )

```
>>> for i in range(1, 10): print(i, end = ',')
1,2,3,4,5,6,7,8,9
>>> print(1, 2, 3, 4, 5, 6, 7, 8, 9, sep = ',')
1,2,3,4,5,6,7,8,9
```

List

- Ordered → Indexing 가능
- Slicing 가능
- List 연산 가능

```
>>> li = ['A', 'B', 'C', 'D', 'E']

>>> li[2] # Indexing
'C'
>>> li[:3] # Slicing
['A', 'B', 'C']
>>> li0 = [0, 1, 2]

>>> li1 = [3, 4, 5]

>>> li0 + li1 # List 연산
[0, 1, 2, 3, 4, 5]
>>> li0 * 3 # List 연산
[0, 1, 2, 0, 1, 2, 0, 1, 2]
```

```
>>> li = ['E', 'D', 'C', 'A']

>>> li.append('D')
['E', 'D', 'C', 'A', 'B']
>>> li.sort()
['A', 'B', 'C', 'D', 'E']
```

```
>>> li.reverse()
['E', 'D', 'C', 'B', 'A']
```

```
>>> li = [1, 2, 3, 4]
>>> li.append(6)
[1, 2, 3, 4, 6]
>>> li.insert(4, 5)
[1, 2, 3, 4, 5, 6]
>>> li.remove(5)
[1, 2, 3, 4, 6]
>>> li.pop() # [1, 2, 3, 4]
6
>>> del li[:3] # [1, 2, 3]
```

Tuple

- Ordered → Indexing 가능
- Slicing 가능
- **Tuple 연산 가능**
- Item assign / delete 불가

```
>>> tu0 = (1,)
>>> tu1 = (1, 2, 3)
>>> tu2 = 1, 2, 3
>>> tu3 = tu1 + tu2
>>> print(tu3)
(1, 2, 3, 1, 2, 3)
>>> tu4 = tu1 * 2
>>> print(tu4)
(1, 2, 3, 1, 2, 3)
```

Dictionary (or 연관 배열 or Hash)

- Unordered → Indexing 불가 (Key → Value를 바로 얻음)
- Key
  - **CANNOT** be modified
  - **CANNOT** be duplicated (중복될 경우 선행된 Key는 무시됨)
  - **CANNOT** be a List type
- Value
  - Can be modified
  - Can be duplicated

```
>>> dic = {'A': 1, 'B': 2, 'C': 3}
>>> dic['D']
KeyError: 'D'
>>> dic.get('D')
None
>>> dic.keys()
dict_keys(['A', 'B', 'C'])
>>> type(dic.keys()) # list가 아닌 dict_keys 객체로 반환
<class 'dict_keys'>
```

Set

- Unordered → Indexing 불가
- **CANNOT** be duplicated

Unpacking

n = List | Tuple의 item 개수
m = 대입하는 변수의 개수 일 때

$$n = m$$

가 성립해야 함.

```
>>> a, b = 1, 2
>>> a
1
>>> b
2
>>> a, b = 1
TypeError: cannot unpack non-iteraable int object
```

Identity

- Unordered → Indexing 불가
- **CANNOT** be duplicated

```
>>> import copy from copy
>>> list1 = [1, 2, 3]
>>> list2 = [1, 2, 3]
>>> list3 = copy(list1)
>>> list4 = list1

>>> list1 == list2
True
>>> list1 is list2
False

>>> list1 == list3
True
>>> list1 is list3
False
```

```
>>> list1 == list4
True
>>> list1 is list4
True
```

```
>>> n = 256
>>> n is 256 # -5 ≤ n ≤ 256인 정수는 별도로 저장해 두었다가 공유한다
True
>>> m = 257
>>> m is 257
False
>>> s = 'ABCDEFG'
>>> s is 'ABCDEFG' # 간단한 문자열은 별도로 저장해 두었다가 공유한다
True
>>> p = 1.0
>>> p is 1.0
False
>>> q = 1.0
>>> r = q # 대입 연산자 사용 시 객체를 새로 생성하는 대신 *공유한다*
>>> r is q
True
```

if ( )

- 제어문 → 조건문에 해당 (`if ( )` 문 `elif ( )` 문 `match ( )` 문)
- 여러 가지 조건을 검사할 때에는 `if ( )` 문 대신 `elif ( )` | `match ( )` 문을 사용해야 함.
- `elif ( )` 문은 개수에 제한 없이 사용 가능함.

```
>>> score = 60

>>> if score >= 60: result = "True"
>>> else: result = "False"

>>> result = "True" if score >= 60 else "False"
```

while ( )

- 제어문 → 반복문에 해당 (`while ( )` 문 `for ( )` 문)
- 반복 횟수가 명확하지 않을 때 사용.
- 조건을 만족할 경우 프로그램을 종료하고자 할 때 사용.

```
>>> a = 0
>>> while True:
... if a == 10: break
... print(a)
... a += 1
0 1 2 3 4 5 6 7 8 9
```

for ( )

- 제어문 → 반복문에 해당 (`while ( )` 문 `for ( )` 문)
- 반복 횟수를 정확히 알고 있을 때 사용함.
- 반복 횟수가 변하지 않을 때 사용함.
- `in` | `not in` 은 List | Tuple | 문자열에서 사용 가능함

```
>>> for i in range(10):
...             print(i)
0 2 4 6 8

>>> for i in range(10, 0, -2):
...     print(i)
10 8 6 4 2

>>> for i in range(1, 11, 2):
...             print(i)
1 3 5 7 9
```

if ( ) / for ( ) with List Comprehension

- List Comprehension: 간결한 표현이 가능 but 가독성이 떨어질 수 있음

```
>>> marks = [90, 25, 67, 45, 80]
>>> result = ["PASS" if mark >= 60 else "FAIL" for mark in
marks]
>>> result
['PASS', 'FAIL', 'PASS', 'FAIL', 'PASS']

>>> scores = [98, 65, 77, 82, 56, 87, 92]
>>> grades = ['A' if s >= 90 else 'B' if s >= 80 else 'C' if s
>= 70 else 'D' if s >= 60 else 'F' for s in scores]
>>> grades
['A', 'D', 'C', 'B', 'F', 'B', 'A']
```

match - case with if ( )

```
>>> a = 135
>>> match a:
...     case a if a < 100:
...             print('smaller than')
...     case 100:
...             print('is')
...     case a if a > 100:
...             print('bigger than')
>>> print(a)
bigger than
100
```

continue

- 제어문 → 특수 제어문에 해당 (`break` `continue`)

```
>>> for i in range(1, 10):
...             if i % 3 print(i)
1 2 4 5 7 8

>>> for i in range(1, 10):
...             if i % 3 == 0: continue
...             print(i)
1 2 4 5 7 8
```