|  |  |
| --- | --- |
| 문법 | 설명 |
| s.find(x) | x의 첫 번째 위치를 반환. 없으면, -1 반환 |
| s.index(x) | x의 첫 번째 위치를 반환. 없으면, 오류 반환 |
| s.startswith(x) | 문자열이 x로 시작하면 True 반환, 아니면 False\*\*반환 |
| s.endswith(x) | 문자열이 x로 끝나면 True반환, 아니면 False반환 |
| s.isalpha() | 알파벳 문자 여부  단순 알파벳X 유니코드 상 letter 한국어도 포함) |
| s.isspace() | 공백 여부 |
| s.isupper() | 대문자 여부 |
| s.islower() | 소문자 여부 |
| s.istitle() | 타이틀 형식 여부 |

|  |  |
| --- | --- |
|  |  |
| s.isdecimal() | 문자열이 0~9까지의 수로 이루어져 있는가? |
| s.isdigit() | 문자열이 숫자로 이루어져 있는가? |
| s.isnumeric() | 문자열을 수로 볼 수 있는가? |

|  |  |
| --- | --- |
|  |  |
| s.replace(old, new, [,count]) | 바꿀 대상 글자를 새로운 글자로 바꿔서 반환count 지정: 해당 갯수만큼만 시행 |
| s.strip([chars]) | 공백이나 특정 문자 제거 |
| s.lstrip([chars]) | 문자열의 왼쪽으로 공백이나 모든 조합 제거 |
| s.rstrip([chars]) | 문자열의 오른쪽 공백이나 모든 조합 제거 |
| s.split(sep=None, maxsplit=-1) | 공백이나 특정 문자를 기준으로 분리  리스트로 반환 |
| 'separator'.join([iterable]) | 구분자로 iterable을 합침 |
| s.capitalize() | 가장 첫번째 글자를 대문자로 변경 |
| s.title() | 문자열 내 띄어쓰기 기준으로 각 단어의 첫글자는 대문자로, 나머지는 소문자로 변환  어포스트로피(')나 공백 이후를 대문자로 |
| s.upper() | 모두 대문자로 변경 |
| s.lower() | 모두 소문자로 변경 |
| s.swapcase() | 대, 소문자 서로 변경 |

리스트

|  |  |
| --- | --- |
|  |  |
| L.append(x) | 리스트 마지막에 항목 x를 추가  L[len(L) : ] = [x]와 동일 |
| L.insert(i, x) | 리스트 인덱스 i에 항목 x를 삽입  i가 리스트 길이보다 큰 경우 마지막에 삽입L.insert(len(L), x)는 L.append(x) 와 같다. |
| L.remove(x) | 리스트 가장 왼쪽에 있는 항목(첫 번째)x를 제거, 항목 존재X->Value Error |
| L.pop() | 리스트 가장 오른쪽 항목(마지막) 반환 후 제거 |
| L.pop(i) | 리스트 인덱스 i에 있는 항목 반환 후 제거 |
| L.clear() | 리스트의 모든 항목 삭제 |
| L.extend(iterable) | 순회형의 모든 항목들을 리스트 끝에 추가  (+=과 같은 기능)  L[len(L) : ] = iterable과 동일 |
| L.index(x, start, end) | 리스트 항목 중 가장 왼쪽에 있는 x의 인덱스를 반환 항목 존재X -> Value Error |
| L.reverse() | 리스트를 거꾸로 원본 list를 변형시키고 None 리턴 정렬X 원본 순서를 뒤집고 수정 |
| L.sort() | 리스트를 정렬, 원본 list를 변형시키고 None 리턴 reverse 옵션 존재 |
| L.count(x) | 리스트에서 항목 x가 몇 개 존재하는지 갯수 반환 |

튜플

|  |  |
| --- | --- |
|  |  |
| T.index(x[, start[, end]]) | 튜플에 있는 항목 중 x의 첫번째 인덱스 반환항목 존재X -> ValueError |
| T.count(x) | 항목 x가 몇 개 존재하는지 갯수 반환 |

셋

|  |  |
| --- | --- |
|  |  |
| s.copy() | 셋의 얕은 복사본을 반환 |
| s.add(x) | 항목 x가 셋s에 없다면 추가 |
| s.pop() | 셋 s에서 랜덤하게 항목 반환, 제거 set이 비어 있을 경우 KeyError |
| s.remove(x) | 항목 x를 셋 s에서 삭제  항목 존재하지 않을 경우 KeyError |
| s.discard(x) | 항목 x가 셋s에 있는 경우, x 삭제 (에러가 안남) |
| s.update(t) | 셋 t에 있는 모든 항목 중 셋 s에 없는 항목 추가  반드시 iterable 데이터 구조 전달해야 함 |
| s.clear() | 모든 항목 제거 |
| s.isdisjoint(t) | 셋s가 셋t의 서로 같은 항목을 하나라도 갖고 있지 않은 경우 True 반환 -> 서로소인 경우 |
| s.issubset(t) | 셋s가 셋t의 하위 셋인 경우, True반환 |
| s.issuperset(t) | 셋s가 셋t의 상위 셋인 경우, True반환 |

딕셔너리

|  |  |
| --- | --- |
|  |  |
| d.clear() | 모든 항목 제거 |
| d.copy() | 딕셔너리d의 얕은 복사본 반환 |
| d.keys() | 딕셔너리d의 모든 키를 담은 뷰 반환 |
| d.values() | 딕셔너리d의 모든 값을 담은 뷰 반환 |
| d.items() | 딕셔너리d의 모든 키-값의 쌍을 담은 뷰 반환 |
| d.get(k) | 키k의 값을 반환, 키k가 딕셔너리d에 없을 경우 None 반환 |
| d.get(k, v) | 키k의 값을 반환, 키k가 딕셔너리d에 없을 경우 v를 반환 |
| d.setdefault(k[,default]) | 키k의 값을 반환, 키k가 딕셔너리d에 없을 경우 default값을 갖는 key를 삽입 후 default 반환 |
| d.pop(k) | 키k의 값을 반환하고 삭제  키k가 딕셔너리d에 없을 경우 KeyError |
| d.pop(k, v) | 키k의 값을 반환하고 삭제  키k가 딕셔너리d에 없을 경우 v를 반환 |
| d.update([other]) | 딕셔너리 d의 값을 매핑하여 업데이트  other는 다른 딕셔너리나 key/value쌍으로 되어있는 모든 iterable 사용 가능 |