[Python] 코딩테스트에서 알고리즘 구현에 자주 사용되는 함수 50선!

작성 : Zeromini

디스코드 채널(취업폭격기 제로미니 IT 취업 공부방) : https://discord.gg/yDWrPjqv

1. `len()`: 시퀀스(리스트, 튜플, 딕셔너리 등)의 길이를 반환합니다.

```
arr = [1, 2, 3, 4, 5]
print(len(arr))  # 출력: 5
```

2. `max()`: 가장 큰 요소를 반환합니다.

```
arr = [1, 2, 3, 4, 5]
print(max(arr))  # 출력: 5
```

3. `min()`: 가장 작은 요소를 반환합니다.

```
arr = [1, 2, 3, 4, 5]
print(min(arr))  # 출력: 1
```

4. `sum()`: 시퀀스의 모든 요소를 합한 값을 반환합니다.

```
arr = [1, 2, 3, 4, 5]
print(sum(arr))  # 출력: 15
```

5. `sorted()`: 시퀀스를 정렬하여 반환합니다.

```
arr = [5, 3, 4, 1, 2]
print(sorted(arr))  # 출력: [1, 2, 3, 4, 5]
```

6. `list.append()`: 리스트에 요소를 추가합니다.

```
arr = [1, 2, 3]
arr.append(4)
print(arr)  # 출력: [1, 2, 3, 4]
```

7. `list.extend()`: 리스트에 다른 리스트를 추가합니다.

```
arr1 = [1, 2, 3]
arr2 = [4, 5, 6]
arr1.extend(arr2)
print(arr1)  # 출력: [1, 2, 3, 4, 5, 6]
```

8. `list.remove()`: 리스트에서 첫 번째로 나타나는 특정 값을 제거합니다.

```
arr = [1, 2, 3, 2]
arr.remove(2)
print(arr)  # 출력: [1, 3, 2]
```

9. `list.pop()`: 리스트의 특정 인덱스 요소를 제거하고 반환합니다.

```
arr = [1, 2, 3]
arr.pop(1)
print(arr)  # 출력: [1, 3]
```

10. `list.insert()`: 리스트의 특정 위치에 요소를 삽입합니다.

```
arr = [1, 2, 3]
arr.insert(1, 4)
print(arr)  # 출력: [1, 4, 2, 3]
```

11. `str.split()`: 문자열을 분리하여 리스트를 반환합니다.

```
s = "Hello, World!"
print(s.split())  # 출력: ['Hello,', 'World!']
```

12. `str.join()`: 시퀀스의 모든 문자열을 합쳐 하나의 문자열로 반환합니다.

```
arr = ['Hello', 'World']
print(' '.join(arr))  # 출력: 'Hello World'
```

13. `str.replace()`: 문자열에서 특정 부분을 다른 문자열로 대체합니다.

```
s = "Hello, World!"
print(s.replace('World', 'Python'))  # 출력: 'Hello, Python!'
```

14. `str.find()`: 문자열에서 특정 부분의 인덱스를 반환합니다.

```
s = "Hello, World!"
print(s.find('World'))  # 출력: 7
```

15. `str.startswith()`: 문자열이 특정 문자열로 시작하는지 확인합니다.

```
s = "Hello, World!"
print(s.startswith('Hello'))  # 출력: True
```

16. `str.endswith()`: 문자열이 특정 문자열로 끝나는지 확인합니다.

```
s = "Hello, World!"
print(s.endswith('!'))  # 출력: True
```

17. `str.format()`: 문자열 내에 변수를 삽입합니다.

```
s = "Hello, {}!"
print(s.format('Python'))  # 출력: 'Hello, Python!'
```

18. `range()`: 일련의 숫자를 생성합니다.

```
for i in range(5):
    print(i)  # 출력: 0, 1, 2, 3, 4
```

19. `for loop`: 반복문을 사용합니다.

```
for i in range(5):
    print(i)  # 출력: 0, 1, 2, 3, 4
```

20. `if statement`: 조건문을 사용합니다.

```
x = 10
if x > 5:
    print("x는 5보다 큽니다.")  # 출력: 'x는 5보다 큽니다.'
```

21. `dict.get()`: 딕셔너리에서 특정 키의 값을 가져옵니다.

```
d = {'key': 'value'}
print(d.get('key'))  # 출력: 'value'
```

22. `dict.keys()`: 딕셔너리의 모든 키를 반환합니다.

```
d = {'key1': 'value1', 'key2': 'value2'}
print(list(d.keys()))  # 출력: ['key1', 'key2']
```

23. `dict.values()`: 딕셔너리의 모든 값을 반환합니다.

```
d = {'key1': 'value1', 'key2': 'value2'}
print(list(d.values()))  # 출력: ['value1', 'value2']
```

24. `dict.items()`: 딕셔너리의 모든 키-값 쌍을 반환합니다.

```
d = {'key1': 'value1', 'key2': 'value2'}
print(list(d.items()))  # 출력: [('key1', 'value1'), ('key2', 'value2')]
```

25. `set.add()`: 세트에 요소를 추가합니다.

```
s = {1, 2, 3}
s.add(4)
print(s)  # 출력: {1, 2, 3, 4}
```

26. `set.remove()`: 세트에서 요소를 제거합니다.

```
s = {1, 2, 3}
s.remove(2)
print(s)  # 출력: {1, 3}
```

27. `set.intersection()`: 두 세트의 교집합을 반환합니다.

```
s1 = {1, 2, 3}
s2 = {2, 3, 4}
print(s1.intersection(s2))  # 출력: {2, 3}
```

28. `set.union()`: 두 세트의 합집합을 반환합니다.

```
s1 = {1, 2, 3}
s2 = {2, 3, 4}
print(s1.union(s2))  # 출력: {1, 2, 3, 4}
```

29. `set.difference()`: 두 세트의 차집합을 반환합니다.

```
s1 = {1, 2, 3}
s2 = {2, 3, 4}
print(s1.difference(s2))  # 출력: {1}
```

30. `map()`: 모든 요소에 함수를 적용합니다.

```
def square(x):
    return x ** 2
numbers = [1, 2, 3, 4, 5]
print(list(map(square, numbers)))  # 출력: [1, 4, 9, 16, 25]
```

31. `filter()`: 함수의 조건을 만족하는 요소만 반환합니다.

```
def is_even(x):
    return x % 2 == 0
```

```
numbers = [1, 2, 3, 4, 5]
print(list(filter(is_even, numbers)))  # 출력: [2, 4]
```

32. `lambda`: 익명 함수를 생성합니다.

```
square = lambda x: x ** 2
print(square(5))  # 출력: 25
```

33. `zip()`: 두 개의 이터러블을 묶어줍니다.

```
names = ["Alice", "Bob"]
ages = [25, 30]
print(list(zip(names, ages)))  # 출력: [('Alice', 25), ('Bob', 30)]
```

34. `enumerate()`: 인덱스와 값을 함께 반환합니다.

```
names = ["Alice", "Bob", "Charlie"]
for i, name in enumerate(names):
    print(f"{i}: {name}")
    # 출력:
    # 0: Alice
    # 1: Bob
    # 2: Charlie
```

35. `list comprehension`: 리스트를 빠르게 생성합니다.

```
squares = [x**2 for x in range(10)]
print(squares)  # 출력: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

36. `str.lower()`: 문자열을 소문자로 변환합니다.

```
s = "Hello, World!"
print(s.lower())  # 출력: "hello, world!"
```

37. `str.upper()`: 문자열을 대문자로 변환합니다.

```
s = "Hello, World!"
print(s.upper())  # 출력: "HELLO, WORLD!"
```

38. `str.strip()`: 문자열의 양쪽 끝에서 공백을 제거합니다.

```
s = "  Hello, World!  "
print(s.strip())  # 출력: "Hello, World!"
```

39. `str.isdigit()`: 문자열이 숫자인지 확인합니다.

```
s = "1234"
print(s.isdigit())  # 출력: True
```

40. `list.reverse()`: 리스트의 순서를 반대로 뒤집습니다.

```
arr = [1, 2, 3, 4, 5]
arr.reverse()
print(arr)  # 출력: [5, 4, 3, 2, 1]
```

41. `any()`: 어떤 요소라도 참이면 True를 반환합니다.

```
arr = [True, False, True]
print(any(arr))  # 출력: True
```

42. `all()`: 모든 요소가 참이면 True를 반환합니다.

```
arr = [True, False, True]
print(all(arr))  # 출력: False
```

43. `type()`: 객체의 타입을 반환합니다.

```
s = "Hello, World!"
print(type(s))  # 출력: <class 'str'>
```

44. `isinstance()`: 객체가 특정 타입인지 확인합니다.

```
s = "Hello, World!"
print(isinstance(s, str))  # 출력: True
```

45. `globals()`: 현재의 전역 심볼 테이블을 딕셔너리로 반환합니다. 이 딕셔너리는 항상 현재 모듈에 대한 내용을 나타냅니다.

```
print(globals())  # 출력: 전역 심볼 테이블
```

46. `locals()`: 현재의 지역 심볼 테이블을 딕셔너리로 반환합니다. 이 딕셔너리는 현재 범위에 대한 내용을 나타냅니다.

```
def some_function():
    local_var = "I'm local!"
    print(locals())

some_function()  # 출력: {'local_var': "I'm local!"}
```

47. `del()`: 변수를 삭제합니다.

```
var = "I'm going to be deleted"
print(var)  # 출력: "I'm going to be deleted"
del var
print(var)  # 오류: var가 정의되지 않았습니다.
```

48. `eval()`: 문자열로 표현된 파이썬 표현식을 평가하고 결과를 반환합니다.

```
expression = "2 + 3 * 4"
print(eval(expression))  # 출력: 14
```

49. `re.match()`, `re.search()`, `re.findall()`: 정규 표현식을 사용하여 문자열에서 패턴을 찾습니다.

```
import re
s = "Hello, World!"
print(re.match(r"Hello", s))  # 출력: <re.Match object; span=(0, 5), match='Hello'>
print(re.search(r"Wor", s))   # 출력: <re.Match object; span=(7, 10), match='Wor'>
print(re.findall(r"o", s))    # 출력: ['o', 'o']
```

50. `list.index()`: 리스트에서 특정 값의 인덱스를 반환합니다.

```
lst = ['a', 'b', 'c']
print(lst.index('b'))
```