선발대 #1

- 타입별 메서드
- [이론] 프로세스와 스레드
- 파이썬 멀티 프로세싱(multiprocessing)과 멀티 스레드(threading)
- 과제 안내

문자열 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

count : 문자열 내에서 특정 문자가 몇 개나 있는지 세는 메서드

```
text = "Hello, World!"
count = text.count("l")
print(count)  # 3
```

문자열 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

find: 문자열 내에서 특정 문자열이 처음 나오는 위치를 찾아주는 메서드 (없을 경우 -1 return)

```
text = "Hello, World!"
position = text.find("World")
print(position)  # 7
```

문자열 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

index: 문자열 내에서 특정 문자열이 처음 나오는 위치를 찾아주는 메서드 (없을 경우 ValueError)

```
text = "Hello, World!"
try:
    position = text.index("World")
    print(position)  # 7
except ValueError:
    print("찾는 문자열이 없습니다.")
```

문자열 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

join: 특정 문자열을 기준으로 다른 문자열들을 합쳐주는 메서드

```
fruits = ["apple", "banana", "cherry"]
joined_fruits = ", ".join(fruits)
print(joined_fruits)  # "apple, banana, cherry"
```

문자열 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

upper: 문자열 내의 모든 소문자를 대문자로 바꾸는 메서드
lower: 문자열 내의 모든 대문자를 소문자로 바꾸는 메서드

```
text = "Hello, World!"
uppercase_text = text.upper()
print(uppercase_text)  # "HELLO, WORLD!"

lowercase_text = text.lower()
print(lowercase_text)  # "hello, world!"
```

문자열 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

replace: 문자열 내에서 특정 문자열을 다른 문자열로 바꾸는 메서드

```
text = "Hello, World!"
replaced_text = text.replace("World", "Python")
print(replaced_text)  # "Hello, Python!"
```

문자열 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

split: 문자열을 특정 문자를 기준으로 나누는 메서드(결과는 리스트 형태로 반환)

```
text = "apple,banana,cherry"
fruits = text.split(",")
print(fruits)  # ['apple', 'banana', 'cherry']
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

len: 리스트의 길이를 반환하는 내장 함수

```
numbers = [1, 2, 3, 4, 5]
print(len(numbers))  # 5
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

  del: 리스트에서 특정 요소를 삭제하는 연산자

```
numbers = [1, 2, 3, 4, 5]
del numbers[2]
print(numbers)  # [1, 2, 4, 5]
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

append: 리스트의 맨 뒤에 새로운 요소를 추가하는 메서드

```
numbers = [1, 2, 3, 4, 5]
numbers.append(6)
print(numbers)  # [1, 2, 3, 4, 5, 6]
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

sort: 리스트를 오름차순으로 정렬하는 메서드

```
numbers = [3, 2, 4, 1, 5]
numbers.sort()
print(numbers)  # [1, 2, 3, 4, 5]
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

reverse: 리스트의 요소 순서를 반대로 뒤집는 메서드

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

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

index: 리스트에서 특정 요소의 인덱스를 반환하는 메서드

```
fruits = ['apple', 'banana', 'cherry']
print(fruits.index('banana'))  # 1
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

insert: 리스트의 특정 위치에 요소를 삽입하는 메서드

```
numbers = [1, 2, 3, 4, 5]
numbers.insert(2, 10)
print(numbers)  # [1, 2, 10, 3, 4, 5]
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

  remove: 리스트에서 특정 요소를 제거하는 메서드

```
numbers = [1, 2, 3, 4, 5]
numbers.remove(3)
print(numbers)  # [1, 2, 4, 5]
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

pop: 리스트에서 마지막 요소를 빼낸 뒤, 그 요소를 삭제하는 메서드

```
numbers = [1, 2, 3, 4, 5]
numbers.pop(3)
print(numbers)  # [1, 2, 3, 5]
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

count: 리스트에서 특정 요소의 개수를 세는 메서드

```
numbers = [1, 2, 3, 3, 4, 5]
print(numbers.count(3))  # 2
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

  extend: 리스트를 확장하여 새로운 요소들을 추가하는 메서드

```
numbers = [1, 2, 3]
numbers.extend([4, 5, 6])
print(numbers)  # [1, 2, 3, 4, 5, 6]
```

리스트 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

  += 연산자를 사용해서도 구현할 수도

```
numbers = [1, 2, 3]
numbers += [4, 5, 6]
print(numbers)  # [1, 2, 3, 4, 5, 6]
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

  딕셔너리 초기화

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

딕셔너리 초기화

```
# 빈 딕셔너리 만들기
empty_dict = {}

# 초기화할 딕셔너리 만들기
my_dict = {"apple": 1, "banana": 2, "orange": 3}

print(my_dict)
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

딕셔너리 쌍 추가

```
my_dict = {"apple": 1, "banana": 2, "orange": 3}

my_dict["grape"] = 4
print(my_dict)  # {"apple": 1, "banana": 2, "orange": 3, "grape": 4}
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

del: 딕셔너리에서 특정 요소를 삭제

```
my_dict = {"apple": 1, "banana": 2, "orange": 3}

del my_dict["apple"]
print(my_dict)  # {"banana": 2, "orange": 3}
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

  딕셔너리에서 특정 Key에 해당하는 Value를 얻는 방법 (딕셔너리에 Key가 없는 경우, KeyError)

```
my_dict = {"apple": 1, "banana": 2, "orange": 3}

print(my_dict["banana"])  # 2
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

keys: 딕셔너리에서 모든 Key를 리스트로 만들기

```
my_dict = {"apple": 1, "banana": 2, "orange": 3}

key_list = list(my_dict.keys())
print(key_list)  # ["apple", "banana", "orange"]
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

values: 딕셔너리에서 모든 Value를 리스트로 만들기

```
my_dict = {"apple": 1, "banana": 2, "orange": 3}

value_list = list(my_dict.values())
print(value_list)  # [1, 2, 3]
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

items: 딕셔너리의 모든 키와 값을 튜플 형태의 리스트로 반환

```
person = {'name': 'John', 'age': 30, 'gender': 'male'}

items = person.items()
print(items)   # dict_items([('name', 'John'), ('age', 30), ('gender', 'male')])
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

clear: 딕셔너리의 모든 요소를 삭제

```
person = {'name': 'John', 'age': 30, 'gender': 'male'}

person.clear()
print(person)   # {}
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

get: 딕셔너리에서 지정한 키에 대응하는 값을 반환 (딕셔너리에 Key가 없는 경우, None 반환)

```python
person = {'name': 'John', 'age': 30, 'gender': 'male'}

name = person.get('name')
print(name)   # John

email = person.get('email')
print(email)  # None

# 기본값을 설정할 수 있음
email = person.get('email', 'unknown')
print(email)  # unknown
```

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

get: 딕셔너리에서 지정한 키에 대응하는 값을 반환

request.POST['key']

vs

request.POST.get('key')

딕셔너리 메서드

- 지금부터 하는 모든 내용은 전부 다 암기해주셔야 합니다

  in: 해당 키가 딕셔너리 안에 있는지 확인

```
person = {'name': 'John', 'age': 30, 'gender': 'male'}

print('name' in person)   # True
print('email' in person)  # False
```

프로세스와 스레드

- .

파이썬 멀티 프로세싱(multiprocessing)과 멀티 스레드(threading)

- 파이썬으로 프로세스 다루기 (multiprocessing)

https://github.com/kangtegong/self-learning-cs/blob/main/process/process_python.md

파이썬 멀티 프로세싱(multiprocessing)과 멀티 스레드(threading)

- 파이썬으로 스레드 다루기 (threading)

https://github.com/kangtegong/self-learning-cs/blob/main/thread/thread_python.md

과제 안내

- 위 모든 예제 코드를 직접 작성하여 깃허브에 푸쉬해주세요
  - 타입별 메서드
  - 파이썬 멀티 프로세싱(multiprocessing)과 멀티 스레드(threading)
- 변수 이름은 자신의 영문 이름으로 제출해주세요 (chatGPT 방지)

- 다음 주 예고
  - coroutine & asyncio
  - 파이썬 패키지 관리 기법
- 선발대 강의 시간 예고: 4/19 17:00

fin

Q & A

Email: tegongkang@gmail.com
Github: github.com/kangtegong
Youtube: youtube.com/@kangminchul