CHAPTER_03_Data Structures

```
파이썬의 데이터구조는 3가지가 있다
(1) List
(2) Tuple
(3) Dictionary
```

> **Data Struture(자료구조) 란?**
>
> 데이터를 구조화하고 싶을 때 사용하는 것이다.

01 Methods

- `Method`는 데이터에 결합된 `function`이다.

```
name = "pubao"

print(name.upper())
print(name.replace("u", "🐓"))
print("pubao".endswith("o"))
```

> **🤔 Functon과 Method 차이**
>
> - method는 데이터 뒤에 결합/연결된 function이다.
> - 함수와 데이터와 결합되어 있다면 메소드라 부르고 그렇지 않다면 함수라고 부른다.
> - `print(name.upper())` : upper `method`가 name의 데이터를 가져와서 대문자로 바꿔주고 있다.

02. Lists

```
day_of_week = ["Mon", "Tue", "Wed", "Thu", "Fri"]
```

🧅 프로그래밍 언어에서 `modify(수정하다)`는 `mutate(변화시키다)`는 의미이다. 🧅

- 리스트와 결합된 메소드

```
print(day_of_week.count("Wed"))
day_of_week.clear()
day_of_week.reverse()
day_of_week.append("Sat")
```

- 리스트 장점

> 데이터 가공에 도움을 주는 메소드들을 사용이 가능하다.

- 리스트에 있는 특정 아이템에 접근할 수 있는 방법

  > 대괄호안에 내가 접근하고 싶은 아이템의 인덱스를 넣어주면 된다.

  ```
  day_of_week = ["Mon", "Tue", "Wed", "Thu", "Fri"]
  print(day_of_week[2])
  ```

03. Tuples

> 튜플은 대괄호`[]` 대신 소괄호`()`를 사용한다

```
days = ("Mon", "Tue", "Wed")
```

- 튜플과 리스트의 차이점

> 튜플은 불변성(immutable)을 가진다 즉, 튜플은 변경하지 못한다.
>
> 메소드도 count(), index() 2개 뿐이다.
>
> 튜플은 불변하기 때문에 만들면 그 내용을 바꿀 수 없다.

04. Dicts

> 리스트는 대괄호`[]`, 튜플은 소괄호`()`, 딕셔너리는 중괄호`{}` 사용
>
> 딕셔너리는 `키-값` 쌍으로 이뤄져 있다

- `player`라는 딕셔너리를 만들어서 이 안에 `name, age, alive` 라는 속성 작성한 코드

```
player = {
    'name': 'pubao',
    'age': 8,
    'alive': True,
    'fav_food': ['🍒', '🥕']
}

print(player.get('age'))
```

- `딕셔너리`의 용도와 `리스트`의 용도는 아예 다르다
  - 숫자 목록, todo 목록, 어떤 목록이 있다면 그건 `리스트`나 `튜플`이 될 수 있다.
  - `딕셔너리`는 많은 속성들을 가지고 있는 데이터를 만들 때 사용한다.

- 딕셔너리를 생성한 이후 데이터를 추가하는 방법

- 딕셔너리는 리스트가 변경 가능했던 것처럼 변경이 가능하다.

```
print(player)
print(player.pop('age'))
print(player)
```

```
player['fav_food'].append("🍝")
print(player.get('fav_food'))
print(player['fav_food'])
```

🍌 Recap

01. 메소드

- `메소드`는 데이터에 연결된, 즉 결합된 `function(함수)`이다. (데이터 안에 있는것)
- 기억 : 메소드는 `함수`이다 `소괄호`를 사용하면 실행한다는 뜻, 소괄호 안에 `argument`를 넣을 수도 있다.
- 만약`함수`가 독립적으로 사용된다면 `함수`라고 하지만 데이터에 결함된 함수는`메소드`라고 불린다.

```
print("pubao".endswith("a"))
```

02. 리스트

- 대괄호`[]` 를 사용하여 생성한다.
- `list`는 값들의 목록을 정렬할 수 있게 해준다.
- 특정한 아이템에 접근할 때는 `인덱스(index)`로 할 수 있다.
- 리스트는 mutabel(변겨 가능)한 값들의 연속체를 만들게 해준다.
  - (= 즉, 생성 이후에도 값 변경이 가능하다는 것을 의미한다)

03. 튜플

- 소괄호`()`를 사용하여 생성한다.
- 튜플은 불변성(immutable)을 가진다. (= 즉, 생성 이후에도 값 변경이 불가능한다는 것을 의미한다)

04. 딕셔너리

- 딕셔너리는 더 복잡한 데이터구조를 만들때 유용하게 사용할 수 있다.
- 딕셔너리는 키와 값으로 되어있는 백과사전을 생각하면 쉽다.
- 키는 백과사전에 있는 단어이고 값은 정의이다.

```
player = {
'name': 'pubao',
'age': 8,
'alive': True,
'friend': {
  'name' : "ouou",
  'fav_food' : ['🍕']
  }
}
```

---

05 For Loops

- `for` 반복문을 사용할 때 `for`는 각각의 `item`이 실행될 때 `placeholder`를 만드는 것을 허락해준다.
- `placeholder`의 이름이 꼭 each일 필요가 없다 (이름 마음대로 지정 가능)

```
websites = (
    "google.com",
    "airbnb.com",
    "twitter.com",
    "facebook.com"
)
```

> python에게 list의 각 item을 활용해서 자동으로 코드를 실행하는 방법
>
> => for 반복분(loop) 라는 것을 사용

```
for each in websites:
  # hello 가 4번 출력이 된다
  print("hello")
```

> `each` => 현재 실행중인 item에 접근하는 변수 이름은 마음대로 지정가능

```
for potato in websites:
  print("potato is equals to", potato)

# 출력
potato is equals to google.com
potato is equals to airbnb.com
potato is equals to twitter.com
potato is equals to facebook.com
```

🧅 흔한 관습 : 튜플이나 리스트를 만들 때 `복수형`을 사용한다. 🧅

06. URL Formatting

> 📢 URL
>
> : 만약 웹사이트 주소가 http로 시작하면 바로 이동하고
>
> : 만약 그렇지 않다면 https를 붙여줘야 한다.

```
websites = (
    "google.com",
    "airbnb.com",
    "https://twitter.com",
    "facebook.com",
    "https://tiktok.com"
)

for website in websites:
  print(website)
```

- `websiste` 변수는 차례대로 각 사이클에서 `리스트` 안의 `item`으로 바뀐다는 것을 잊지 말기

```
for website in websites:
  #  if는 오직 무언가가 true인지 false인지만 판단한다
  if website.startswith("https://"):
    print("good to go")
  else:
    print("we have to fix it")

# (2) website 가 https:// 로 시작하지 않는 경우에 집중하기
for website in websites:
  if not website.startswith("https://"):
    print("have to fix it")

# (3)  https:// 로 시작할 수 있도록 업데이트 하기
for website in websites:
  if not website.startswith("https://"):
    # string안에 변수를 넣는 방법
```

```
    website = f"https://{website}"
  print(website)
```

07 Requests

> - pypi : 다른 사람이 만든 프로젝트나 module을 모아둔 곳
>   - ex) artistic software module
>   - pynmation (python animation)
>   - pygoogle-image (the package downloads images from google images)
>   - imgae2face (얼굴 인식 패키지 face recogniton package)

- `Requests` (`Packages > requests@2.31.0`)

```
python -m pip install requests
```

🤔 Requests 란 뭘까 ?

: 구글로 이동하는 것이 request 이다.

: 내 브라우저는 google 서버에 request를 보내고 google 서버는 나한테 웹 사이트를 보내준다.

reqeusts에서 get 가져오기.

> `get`은 `function`인데 이동안 다음에 website를 가져오는 것.
>
> ```
> from reqeusts import get
> ```

08 Status Codes

🧅 `return` value는 `function`이 어떤 작업을 수행하고 어떤 값을 돌려주는 것🧅

```
for website in websites:
  if not website.startswith("https://"):
    website = f"https://{website}"
  response = get(website)
  print(response)
```

```
# Response [200] 가 출력된다
```

> 이 의미는 웹사이트가 성공적으로 응답했다는 뜻

- internet은 `http protocol` 에 기반한다고 말 할 수 있다.

- 컴퓨터들이 서로 소통하는 방식은 당연하게도 `HTTP request`이다.
- 그래서 `request`가 정상인지 아닌지를 알 수 있는 수단이 있어야 하는데 `request`의 결과를 확인하는 방법으로 `HTTP 코드`를 사용한다.
- `get function`이 `request`한 `response`는 다른 것도 가지고 있는데 예를 들면 상태 코드이다.

```
for website in websites:
  if not website.startswith("https://"):
    website = f"https://{website}"
  response = get(website)
  print(response.status_code)

# 200 이 출력된다
```

```
for website in websites:
  if not website.startswith("https://"):
    website = f"https://{website}"
  response = get(website)
  if response.status_code == 200:
    print(f"{website} is OK")
  else:
    print(f"{website} is not OK")
```

- ok 또는 fail 같은 응답으로 dictionary를 만들고 싶다면, results라는 dictionary 만들기
- results dictionary 안에 새로운 entry 추가하기

```
results = {}

for website in websites:
  if not website.startswith("https://"):
    website = f"https://{website}"
  response = get(website)
  if response.status_code == 200:
    results[website] = "OK"
  else:
    results[website] = "FAILED"

  print(results)
```