Ch6. 사용자 정의 데이터 객체: 데이터와 코드의 결합

- summarized by 전승일(2014/11/02)
- source code: http://www.headfirstlabs.com/books/hfpython/
- support site: http://python.itcarlow.ie

켈리 감독이 새로운 형식의 파일을 갖고 돌아왔습니다

- 선수 기록 앞에 선수 이름과 생일을 추가 함

```
Sarah Sweeney,2002-6-17,2:58,2.58,2:39,2-25,2-55,2:54,2.18,2:55,2:55,2:22,2-21,
```

- 이에 따른 기존 코드 변경: 176 new data format
- 이 코드의 문제점: 선수의 수가 많아 지면 중복된 코드가 증가 함, 한 선수의 데이터가 코드 내에서 분리되어 있음

연관된 데이터를 표현하기 위한 Dictionary

- key: value 형태로 연관된 데이터를 하나로 모음 (데이터의 구조화)

```
a_dict = { "key1" : "value1", "key2" : "value2" }
a_dict['key3'] = 'value3'
```

- dict를 이용하여 기존 코드 수정: 182 dict
- 추가 개선 포인트: 187 dict returning function

```
1. 곧바로 dict를 만든다.
```

```
2. dict를 만드는 코드를 get_coach_data() 함수 안으로 이동하고, list 대신 dict를 반환한다.
3. 상위 3개의 기록을 찾는 코드를 get_coach_data() 함수 안으로 이동한다.
4. 메인 코드에서 새 버전의 get_coach_data() 함수를 사용하도록 수정한다
```

Class는 코드와 데이터를 한데 묶습니다

- Class를 사용하면 복잡도를 줄일 수 있습니다
- 복잡도가 줄어들면 버그도 적어집니다.
- 버그가 적으면 유지보수 하기도 좋습니다

Class 정의

- p.226 클래스 개념도 참조
- method, attribute, instance

Class를 사용해서 Class 정의하기

```
class Athlete:
    def __init__(self):
        # The code to initialize an "Athlete" object.

a = Athlete()
b = Athlete()
```

self의 중요성

```
a = Athlete() ---> Athlete.__init__(a)
# target 식별자가 self 인자에 대입됩니다
```

- class에 정의된 method는 모든 instance object에 의해 공유됩니다

모든 method의 첫 번째 인자는 self입니다

- p.229 "class Athlete" 참조

```
d = Athlete("Holy Grail") ---> Athlete.__init__(d, "Holy Grail")

d.how_big()               ---> Athlete.how_big(d)
```

- class를 사용하여 기존 코드 변경하기: 196 using Class

Python 내장 list에서 상속 받기

- Athlete Class가 List와 매우 비슷합니다
- 내장된 클래스(list)에 사용자 정의 속성을 추가할 수 있음: inheritance
- 상속(inheritance)을 하게 되면 기존 class의 method를 그대로 사용할 수 있는 장점이 있음
- inheritance를 이용하여 기존 코드 변경하기: 210 inheritance

Additional Information