CES1017 Programming Fundamentals

Week 12: OOP 사례학습

- 파일 활용

Instructor: Eunil Park (pa1324@hanyang.ac.kr)

HANYANG UNIVERSITY

Contents

HANYANG UNIVERSITY

Today's Schedule

1. 사전(Dictionary)
2. 블랙잭 확장

01. 사전(Dictionary)

HANYANG UNIVERSITY

Dictionary

- 지금까지 공부한 문자열, 튜플, 리스트와 같은 시퀀스는 모두 정수 **위치번호(index)**를 키(key)로 사용

- 저절로 매겨진 정수 위치번호 대신 **키를 지정하여 사용하면** 편리한 경우 있음
  - 예1 (전화번호부)
    - 전화번호를 모아놓은 시퀀스
    - 전화번호를 찾을 때? ➔ 이름을 키를 사용하는 것이 편리
  - 예2 (영어사전)
    - 영어 단어와 그 의미의 쌍을 모아놓은 것
    - 영어 단어를 찾을 때? ➔ 단어를 키로 하여 의미를 찾는 것이 편리

- Python에서는 "**Dictionary**" 사용 가능

01. 사전(Dictionary)

HANYANG UNIVERSITY

사전 만들기

- 사전(Dictionary): 키와 값의 쌍을 모아놓은 것
  - 사전의 원소는 키와 표현식의 쌍으로 <키> : <표현식>으로 표현
  - { <키> : <표현식>, …, <키> : <표현식> }
  - 키로는 문자열 또는 정수를 쓸 수 있음
  - 사전은 지정한 키로 해당 값을 찾을 수 있으므로 위치번호를 키로 쓰는 리스트 또는 튜플과는 달리 나열된 순서는 중요하지 않음
  - 사전 내부에서 키를 중복하여 사용할 수 없음
- 예: 지갑에 들어 있는 지폐
  - cash = {'50000':2, '10000':7, '5000':0, '1000':3}
  - 왼쪽 문자열은 지폐의 종류를 표시하는 키이고, 오른쪽 정수는 해당 지폐의 장수를 나타냄
  - 즉, 50,000원 짜리 지폐 2장, 10,000원 짜리 지폐 7장, 5,000원 짜리 지폐 0장, 1,000원 짜리 지폐 3장

01. 사전(Dictionary)

HANYANG UNIVERSITY

사전에서 값 꺼내기

```
>>> cash = {'50000':2, '10000':7, '5000':0, '1000':3}
>>> cash['10000']
7
>>> cash['20000']
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    cash['20000']
KeyError: '20000'
>>>
```

[안전코딩]

```
if '20000' in cash:
    cash['20000']
```

01. 사전(Dictionary)

HANYANG UNIVERSITY

사전에서 값 꺼내기

사전 전체에서 키를 하나씩 꺼내 쓰기

```
for key in cash:
        print(key, "짜리가", cash[key], '장 있음')
```

01. 사전(Dictionary)

HANYANG UNIVERSITY

사전 고치기

#추가

```
>>> cash
{'50000': 2, '10000': 7, '5000': 0, '1000': 3}
>>> cash['100000'] = 3
>>> cash
{'50000': 2, '10000': 7, '5000': 0, '1000': 3, '100000': 3}
```

#교체

```
>>> cash['5000'] = 1
>>> cash
{'50000': 2, '10000': 7, '5000': 1, '1000': 3, '100000': 3}
>>>
```

01. 사전(Dictionary)

HANYANG UNIVERSITY

사전 고치기

#삭제

```
>>> cash
{'50000': 2, '10000': 7, '5000': 1, '1000': 3, '100000': 3}
>>> del cash['1000']
>>> cash
{'50000': 2, '10000': 7, '5000': 1, '100000': 3}
>>>
>>>
>>> del cash['1000']
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    del cash['1000']
KeyError: '1000'
>>>
```

[안전코딩]

if ‘1000’ in cash:

    del cash[‘1000’]

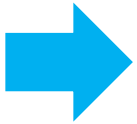
01. 사전(Dictionary)

HANYANG UNIVERSITY

정수 키 사용

키로 사용할 수 있는 값은 수정불가능한 값인 문자열, 정수, 불값, 튜플 등

```
>>> cash = {50000: 1, 10000: 7, 5000: 0, 1000: 3}
>>> cash
{50000: 1, 10000: 7, 5000: 0, 1000: 3}
>>> cash[10000]
7
>>> cash[100000]
Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    cash[100000]
KeyError: 100000
>>>
```



[안전코딩]

```
if 100000 in cash:
    cash[100000]
```

01. 사전(Dictionary)

HANYANG UNIVERSITY

사전 메소드

- 표준 라이브러리에 사전을 관리하는 메소드가 다양하게 있음
  - ‘4.10. Mapping Types — dict’
- 메소드를 호출하여 사용하는 방식
  - <사전>.<메소드이름>(<인수>,…,<인수>)

01. 사전(Dictionary)

HANYANG UNIVERSITY

사전 메소드 예제

- 변수.keys(): **key 들**을 모아놓은 리스트를 반환
- 변수.values(): **value 들**을 모아놓은 리스트를 반환
- 변수.items(): 쌍의 튜플를 모아놓은 리스트를 반환
- 변수.clear(): 모든 쌍들을 삭제
- 변수.get(s): **key 값으로 value 반환**

01. 사전(Dictionary)

HANYANG UNIVERSITY

예제

- 문자열에 있는 문자의 숫자를 종류별로 카운트하는 코드를 작성하라

- 출력 예

```
Please Enter your countable list:David Robert Joseph Beckham, OBE[4] (born 2 May 1975)
{'D': 1, 'a': 3, 'v': 1, 'i': 1, 'd': 1, ' ': 8, 'R': 1, 'o': 3, 'b': 2, 'e': 3, 'r': 2, 't': 1, 'J': 1, 's': 1, 'p': 1, 'h': 2, 'B': 2, 'c': 1, 'k': 1, 'm': 1, ',': 1, 'O': 1, 'E': 1, '[': 1, '4': 1, ']': 1, '(': 1, 'n': 1, '2': 1, 'M': 1, 'y': 1, '1': 1, '9': 1, '7': 1, '5': 1, ')': 1}
```

```
Please Enter your countable list:When mobile game industry is one of the biggest markets in this world
{'W': 1, 'h': 3, 'e': 7, 'n': 4, ' ': 12, 'm': 3, 'o': 4, 'b': 2, 'i': 6, 'l': 2, 'g': 3, 'a': 2, 'd': 2, 'u': 1, 's': 5, 't': 5, 'r': 3, 'y': 1, 'f': 1, 'k': 1, 'w': 1}
>>>
```

HANYANG UNIVERSITY

블랙잭 확장

- 파일로 게임 데이터 저장하기
  - 예: 블랙잭 게임에 플레이어 별로 게임 횟수, 승리 횟수, 취득한 칩의 개수 등
  - 게임을 종료한 후에도 기록이 영구히 남아있게 하고 추후에 다시 게임을 할 때 지속적으로 사용할 수 있도록 함

02. 블랙잭 확장

HANYANG UNIVERSITY

추가 요구사항

- 게임을 시작하기 전에 지금까지 게임을 몇 번 하여 몇 번 이겼는지 누적 승률을 보여줌
  - 누적승률: 이긴 횟수 나누기 게임 횟수(백분율), 비긴 경우 0.5회 이긴 것으로 간주
- 게임에서 이기면 받게 되는 칩도 누적하여 기록해두고 게임을 시작하면서 알려줌

You played 37 games and won 20.5 of them.
Your all-time winning percentage is 55.4%
You have 5 chips.

- 칩을 잃어서 부채가 있는 경우 다음과 같은 형식으로 알려줌

You have -3 chips.

- 게임이 끝나면 해당 세션 동안의 기록을 다음과 같은 형식으로 보여 줌

You played 5 games and won 4 of them

All-time Top 5
doh: 135 chips
didi: 36 chips
hy: 23 chips
who: 3 chips
dr : 2 chips

02. 블랙잭 확장

HANYANG UNIVERSITY

텍스트 파일에 저장형식

- 다음의 정보를 텍스트 파일 members.txt에 멤버 1인당 1줄 씩 저장
  - 이름 (name)
  - 게임시도 횟수 (tries)
  - 이긴 횟수 (wins)
  - 칩 보유 개수 (chips)

- 각 정보 사이에 쉼표를 두어 구분
  - doh,993,550,35
  - didi,130,55,10
  - hy,35,18,2
  - dr,18,8,0
  - who,34,18,0

HANYANG UNIVERSITY

load_members()

- 텍스트 파일 "members.txt"에서 한 줄씩 읽어서 다음과 같이 이름을 키로 하는 사전을 만들어 내줌
  - {"doh":(993,550,35),"didi":(130,55,10),"hy":(35,18,2),"dr":(18,8,0),"who":(34,18,0)}

```
1  def load_members():
2      file = open("members.txt", "r")
3      members = {}
4      for line in file:
5          name, tries, wins, chips = line.strip('\n').split(',')
6          members[name] = (int(tries), float(wins), int(chips))
7      file.close()
8      return members
```

사전 초기화

₩n 제거

',' 단위로 분리

사전에 3가지 정보 저장, key = name

02. 블랙잭 확장

HANYANG UNIVERSITY

store_members(member)

- 이름을 키로 하는 사전 members를 텍스트 파일 "members.txt"에 위와 같은 텍스트 파일 형식으로 씀
  - 이전에 이 파일에 저장되어 있던 정보는 모두 지움

```
1  def store_members(members):
2      file = open("members.txt", "w")
3      names = members.keys()
4      for name in names:
5          tries, wins, chips = members[name]
6          line = name + ',' + str(tries) + ',' + \
7                 str(wins) + "," + str(chips) + '\n'
8          file.write(line)
9      file.close()
```

각 key에 대해서

02. 블랙잭 확장

HANYANG UNIVERSITY

login()

```
1  def login():
2      """gets player's name and returns it (string)"""
3      name = input("Enter your name : (4 letters max) ")
4      while len(name) > 4:
5          name = input("Enter your name : (4 letters max) ")
6      members = load_members()
7      if name in members.keys():
8          tries = members[name][0]
9          wins = members[name][1]
10         print("Your played", tries, "games and won", wins, "of them")
11         winrate = 100 * wins / tries if tries > 0 else 0
12         print("Your all-time winning rate is", "{0:.1f}".format(winrate), "%")
13         chips = members[name][2]
14         print("You have", chips, "chips.")
15         return name, tries, wins, chips
16     else:
17         members[name] = (0,0,0)
18         return name, 0, 0, 0
```

ZeroDivisionError 방지

소수점 이하 자리수 지정하여 문자열로 바꿈

02. 블랙잭 확장

HANYANG UNIVERSITY

login() 필요 지식

ZeroDivisionError 방지

```
>> def divide(x,y) : return x/y
>> divde(3,0)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    divide(3,0)
  File "<pyshell#3>", line 2, in divide
    return x / y
ZeroDivisionError: division by zero
>> def divide(x,y) : return x/y if y > 0 else 0
>> divde(3,0)
```

실수출력포맷

```
>> 0.246
0.246
>> “{0:.2f}”.format(0.246)
‘0.25’
```

HANYANG UNIVERSITY

Top 5 보여주기

- show_top5()
  - 칩의 보유개수가 가장 많은 순으로 출력

```
1  def show_top5():
2      members = load_members()
3      print('-----')
4      sorted_members = sorted(members.items(),\
5                              key=lambda x: x[1][2],\
6                              reverse=True)
7      print("All-time Top 5")
8      rank = 1
9      for member in sorted_members[:5]:
10         chips = member[1][2]
11         if chips <= 0:
12             break
13         print(rank, '.', member[0], ':', chips)
14         rank += 1
```

칩의 개수 기준 역수 정렬

출력부분

02. 블랙잭 확장

HANYANG UNIVERSITY

lambda?

- lambda: 임의의 간단한 계산식/함수를 만들 때, 사용
- 예1) A와 B를 더할 때
  - 함수형태로 작성
  - lambda를 사용하여 작성

lambda?

- 그럼 lambda를 왜 사용할까?

  1. def 보다 간결하게 사용할 수 있기 때문
  2. def를 사용할 수 없는 곳에서도 사용할 수 있음

HANYANG UNIVERSITY

사전 정렬

```
dict  = {}
dict["Pink Floyd"] = ("Dark Side of the Moon", 1973)
dict["The Beatles"] = ("Abbey Road", 1969)
dict["Neil Young"] = ("Harvest", 1972)
print(dict)

print(sorted(dict))

print(sorted(dict.items()))
```

키를 기준으로 오름차순으로 정렬한 키의 리스트

키를 기준으로 오름차순으로 정렬한 키와 원소의 튜플쌍

[결과창]
{'Pink Floyd': ('Dark Side of the Moon', 1973), 'The Beatles': ('Abbey Road', 1969), 'Neil Young': ('Harvest', 1972)}

['Neil Young', 'Pink Floyd', 'The Beatles']

[('Neil Young', ('Harvest', 1972)), ('Pink Floyd', ('Dark Side of the Moon', 1973)), ('The Beatles', ('Abbey Road', 1969))]

sorted & sort

- sorted: 입력받은 시퀀스 자료형을 소트한 후, 리스트로 리턴하는 함수
- sort: 리스트 객체를 소트 (리턴형 없음)
- 예제:

HANYANG UNIVERSITY

사전 정렬

```
print(sorted(dict.items(), key = lambda x: x[1][1]))
```

음반의 발매년도 기준으로 정렬

key를 추가인수로 지정함 – 정렬의 기준으로 삼을 데이터 지정

lambda x: x[1][1] 에서 x는 함수의 형식파라미터, x[1][1]은 함수의 몸체임
예: ('The Beatles', ('Abbey Road', 1969))에서 x[1]은 ('Abbey Road', 1969), x[1][1]은 1969가 됨

```
print(sorted(dict.items(), key = lambda x: x[1][1], reverse=True))
```

내림차순으로 정렬

[결과창]
[('The Beatles', ('Abbey Road', 1969)), ('Neil Young', ('Harvest', 1972)), ('Pink Floyd', ('Dark Side of the Moon', 1973))]
[('Pink Floyd', ('Dark Side of the Moon', 1973)), ('Neil Young', ('Harvest', 1972)), ('The Beatles', ('Abbey Road', 1969))]

HANYANG UNIVERSITY

Top 5 보여주기

- show_top5()
  - 칩의 보유개수가 가장 많은 순으로 출력

```
1  def show_top5():
2      members = load_members()
3      print('-----')
4      sorted_members = sorted(members.items(),\
5                              key=lambda x: x[1][2],\
6                              reverse=True)
7      print("All-time Top 5")
8      rank = 1
9      for member in sorted_members[:5]:
10         chips = member[1][2]
11         if chips <= 0:
12             break
13         print(rank, '.', member[0], ':', chips)
14         rank += 1
```

칩의 개수 기준 역수 정렬

출력부분

02. 블랙잭 확장

HANYANG UNIVERSITY

블랙잭 알고리즘 정리

A. 환영인사를 프린트 한다.

```
print("Welcome to SMaSH Casino!")
```

B. members.txt 파일에서 멤버 데이터를 읽고 로그인 절차를 통해서 사용자이름, 게임시도 횟수, 이긴 횟수, 칩 보유개수, 전체멤버 사전 정보를 수집한다.

C. 잘 섞은 카드 1벌을 준비한다.

```
deck = fresh_deck()
```

D. 손님이 원하면 다음을 반복한다.

1. 카드를 1장씩 손님, 딜러, 손님, 딜러 순으로 배분한다.

```
dealer = []
player = []
card, deck = hit(deck)   # 1장 뽑아서
player.append(card)      # 손님에게 주고
card, deck = hit(deck)   # 1장 뽑아서
dealer.append(card)      # 딜러에게 주고
card, deck = hit(deck)   # 1장 뽑아서
player.append(card)      # 손님에게 주고
card, deck = hit(deck)   # 1장 뽑아서
dealer.append(card)      # 딜러에게 준다.
```

2. 딜러의 첫 카드를 제외하고 모두 보여준다.

```
print("My cards are:")
print(" ", "****", "**")
print(" ", dealer[1]["suit"], dealer[1]["rank"])
```

3. 손님의 카드를 보여준다.

```
show_cards(player, "Your cards are:")
```

4. 손님과 딜러의 카드 두 장의 합을 각각 계산한다.

```
score_player = count_score(player)
score_dealer = count_score(player)
```

5. 손님의 카드 두 장의 합 score_player가 21이면 블랙잭으로 손님이 이긴다. 점수 chips를 2 만큼 더한다.

02. 블랙잭 확장

HANYANG UNIVERSITY

블랙잭 알고리즘 정리

6. 손님의 카드 합이 21이 넘지 않는 한 손님이 원하면 카드를 더 준다. 21이 넘으면 손님이 버스트가 되어 딜러가 이기고 점수를 1 뺀다. A는 1 또는 11을 유리한 쪽으로 사용할 수 있어야 한다.
7. 손님이 21이 넘지 않았으면, 딜러의 카드 합을 계산하여 16 이하이면 16이 넘을때까지 무조건 카드를 받는다.
8. 딜러가 21이 넘으면 딜러가 버스트가 되어 손님이 이기고 1점을 더한다.
9. 둘 다 21이 넘지 않으면 합이 큰 쪽이 이긴다. 손님이 이기면 1점을 더하고, 딜러가 이기면 1점을 빼고, 비기면 점수 변동은 없다.
10. 더 할지 손님에게 물어봐서 그만하길 원하면 끝낸다.
11. 게임이 진행되는 동안 승패 횟수와 칩의 획득 개수를 추적하여, 게임이 끝난 뒤 결과를 멤버 사전에 적용하여 수정하고, `members.txt` 파일에 저장한다.
12. 해당 세션의 게임 결과를 다음과 같이 요약하여 보여준다.
    `You played 21 games and won 11 of them.`
13. 지금까지의 칩 최다 보유 멤버 5명을 보여준다.
    `show_top5(members)`

Today's Lessons!

HANYANG UNIVERSITY

Summary

1. 사전(Dictionary)
2. 블랙잭 확장

Thanks

Week 12: OOP 사례학습 – 파일 활용

Instructor: Eunil Park (pa1324@hanyang.ac.kr)