
알고리즘 실습

180913 - Python 이해
Data types, Flow control, Function, Class

오늘의 목표

- Data types
- Flow control
- Function
- Class

오늘의 목표

- ~~Data types~~
- ~~Flow control~~
- ~~Function~~
- ~~Class~~
- GAME!!

Feedback

지난 과제: Python 3 unittest

- 제출: 40 / 41 (97.56%)
 - 1명: .egg 제출로 채점 안 함 - 사실상 100%
- 질문: 17개
 - 평균 답장 시간: 느림!!
- 제출에 관련된 Feedback
 - 사이버캠퍼스에 파일 1개 업로드 가능!

사과 드립니다!!



지난 과제: Python 3 unittest

- 제출에 관련된 Feedback
 - 파일 이름 제대로 할 것
 - 잘못된 예시) AL_201550320_문현수_01.zip <- 이거 그대로였음
 - 보고서 이름 및 확장자: AL_201550320_문현수_01.pdf <- docx, hwp 등등 안 됨
 - 압축은 zip: .egg 등은 안 됨
 - venv 폴더 압축하지 말 것
 - 보고서 표지 제거: 대학원생 도와주세요.
 - 보고서에 최소한 주요 코드 및 실행 결과 스크린샷 필요: 카피 방지

지난 과제: Python 3 unittest

- 과제 관련된 Feedback
 - if __name__ 설명해달라!
 - self 설명해달라!
 - main 설명해달라!
 - indentation 설명해달라!



지난 과제: Python 3 unittest

- 프로그래밍 관련된 Feedback
 - 자료구조가 기억이 안 남

지난번에 까먹었었던 전달 사항

- 출석 / 지각 기준
- 과제, 예제 코드에 문제가 생겼을 때(TA가 늦게 발견하였을 때)!
- 딜레이 감점 기준

지난번에 까먹었었던 전달 사항

- 출석 / 지각 기준
- 과제, 예제 코드에 문제가 생겼을 때(TA가 늦게 발견하였을 때)!
- 딜레이 감점 기준
- 그리고 카피!!!

주의

진행할 내용이
상당히 많음!

13 Chocolate and 1 Chilli Game

BBC 다큐멘터리에서...

- 13 초콜릿 1 고추 게임
- 번갈아가며 1~3개의 초콜릿을 가져가야함
- 마지막에 고추를 가져가는 사람이 패배
- 필승 조건이 있음!



우리가 만들 것: 결과

- 이 예제를 통하여 우리가 배우게 되는 것
- Data types
- Flow control
- Function
- Class

게임을 시작합니다.

항아리에 13개의 초콜릿과 1개의 고추가 있습니다.

1~3개의 초콜릿 혹은 고추를 꺼낼 수 있으며 고추를 꺼내면 패배합니다.
시작!

■□□□□□□□□□□□□□

-----+ 1번 턴! -----+

몇 개의 초콜릿을 뺄까요?: 1

플레이어는 1개의 초콜릿을 꺼냈습니다.

■□□□□□□□□□□□□□

컴퓨터 고민중... 컴퓨터는 2개의 초콜릿을 꺼냈습니다.

■□□□□□□□□□□□□

-----+ 2번 턴! -----+

-----+ 4번 턴! -----+

몇 개의 초콜릿을 뺄까요?: 3

플레이어는 3개의 초콜릿을 꺼냈습니다.

■

플레이어 승리!

게임 준비: 항아리 - chocolate_jar_test.py

- 객체(클래스): 멤버 변수, 멤버 함수를 가진 변수
- 항아리:
 - 초콜릿 수를 나타낸 변수
 - 초콜릿을 꺼내는 동작
- 테스트 코드

```
1  import unittest
2
3  from chocolate_jar import ChocolateJar
4
5
6  class ChocolateJarTest(unittest.TestCase):
7      def test_create(self):
8          jar = ChocolateJar(13)
9          self.assertEqual(14, jar.chocolates)
10
11     def test_take(self):
12         jar = ChocolateJar(13)
13         jar.take_chocolate(3)
14         self.assertEqual(11, jar.chocolates)
15         jar.take_chocolate(2)
16         self.assertEqual(9, jar.chocolates)
17         jar.take_chocolate(1)
18         self.assertEqual(8, jar.chocolates)
19
20
21 if __name__ == '__main__':
22     unittest.main(verbosity=2)
```

게임 준비: 항아리 코드 - chocolate_jar.py

- 간단한 코드지만 굉장히 많은 내용이 담겨 있음
- 알아야 하는 것:
 - 클래스 생성, (상속,) 생성자, 기본값, **self**, 멤버 변수, 멤버 함수

```
1 class ChocolateJar(object):
2     def __init__(self, chocolates=13):
3         self.chocolates = chocolates+1
4
5     def take_chocolate(self, hands):
6         self.chocolates = self.chocolates - hands
```


사용자 입력 - chocolate.py

- 사용자가 빼고 싶은 양을 입력받아야 함
- Python 3 표준 입력: input()
- 알아야 하는 것: 자료형(문자열, 숫자형), 형변환, 문자열다루기

25
26
27
28

```
take = input('몇 개의 초콜릿을 뺄까요?: ')\ntake = int(take.strip())\nprint('플레이어는 {0}개의 초콜릿을 꺼냈습니다.'.format(take))\njar.take_chocolate(take)
```

컴퓨터 무작위 선택 - chocolate.py

- 컴퓨터가 1~3개를 무작위로 선택해야 함: 패배 고려한 선택
- 알아야 하는 것: 무작위 선택, 대기(슬립)

```
34 max_take = min(jar.chocolates - 1, 3)
35 take = random.randint(1, max_take)
36 print('컴퓨터 고민중...', end=' ')
37 time.sleep(take)
38 print('컴퓨터는 {0}개의 초콜릿을 꺼냈습니다.'.format(take))
39 jar.take_chocolate(take)
```

라운드 반복 - chocolate.py

- 마지막 고추를 꺼내는 플레이어가 나올 때까지 반복!
- 알아야하는 것: Flow control

```
15 def main():
16     jar = ChocolateJar(13)
17     print('게임을 시작합니다.')
18     print('항아리에 {0}개의 초콜릿과 1개의 고추가 있습니다.'.format(13))
19     print('1~3개의 초콜릿 혹은 고추를 꺼낼 수 있으며 고추를 꺼내면 패배합니다.')
20     print('시작!')
21     show_jar(jar)
22     turn = 1
23     while jar.chocolates > 0:
24         print('-----+ {0}번 턴! -----+'.format(turn))
```

항아리속 내용 표시 - chocolate.py

- 항아리에 있는 고추, 초콜릿의 수를 텍스트로 출력
 - ■□□□□□□□□□□□□□□□□
- 지정된 개수 만큼 반복할 땐 for가 편리
 - range()와 함께 사용하면 다양한 작업 가능

```
1 import time
2 import random
3
4 from chocolate_jar import ChocolateJar
5
6
7 def show_jar(jar):
8     chocolates = jar.chocolates
9     print('■', end='')
10    for index in range(0, chocolates-1):
11        print('□', end='')
12    print('')
```

더 나은 게임을 위하여...

- 2명의 사람 대결
- 초콜릿 개수 랜덤 or 입력
- 컴퓨터가 선공을 잡고 무조건 이김

전체 코드

- <https://drive.google.com/open?id=1NloxkHtu1IRpOWE5qLruC6tIV-RB0s3s>

```
1 import time
2 import random
3
4 from chocolate_jar import ChocolateJar
5
6
7 def show_jar(jar):
8     chocolates = jar.chocolates
9     print('■', end='')
10    for index in range(0, chocolates-1):
11        print('□', end='')
12    print('')
13
14
15 def main():
16     jar = ChocolateJar(13)
17     print('게임을 시작합니다.')
18     print('항아리에 {0}개의 초콜릿과 1개의 고추가 있습니다.'.format(13))
19     print('1~3개의 초콜릿 혹은 고추를 꺼낼 수 있으며 고추를 꺼내면 패배합니다.')
20     print('시작!')
21     show_jar(jar)
22     turn = 1
23     while jar.chocolates > 0:
24         print('----- {0}번 턴! -----'.format(turn))
25         take = input('몇 개의 초콜릿을 뺄까요?: ')
26         take = int(take.strip())
27         print('플레이어는 {0}개의 초콜릿을 꺼냈습니다.'.format(take))
28         jar.take_chocolate(take)
29         show_jar(jar)
30         if jar.chocolates == 1:
31             print('플레이어 승리!')
32             break
33
34         max_take = min(jar.chocolates - 1, 3)
35         take = random.randint(1, max_take)
36         print('컴퓨터 고민중...', end=' ')
37         time.sleep(take)
38         print('컴퓨터는 {0}개의 초콜릿을 꺼냈습니다.'.format(take))
39         jar.take_chocolate(take)
40         show_jar(jar)
41         if jar.chocolates == 1:
42             print('컴퓨터 승리!')
43             break
44         turn = turn+1
45
46
47 if __name__ == '__main__':
48     main()
```

Blackjack Game!

Blackjack

- 52장의 플레잉카드(포커카드)를 이용하여 노는 게임
- 핸드의 점수를 합하여 **21**을 맞추어야 함
 - 21이 넘으면 무조건 패배
 - 21보다 작으면(미만) 가장 가까운 사람이 승리
- <http://www.247blackjack.com/>



Blackjack을 선택한 이유

- 교재에 있음: 프로그래밍 실습 문제
- 13 Chocolate 1 Chilli보다 더욱 더 많은 것을 알아야 함
 - Data types
 - Flow control
 - Function
 - Class



설계

- 블랙잭 프로그램은 2개의 클래스가 필요함
- 카드 클래스
 - suit, rank를 가짐
- 딜러 클래스
 - 카드 객체 52개를 모아서 덱(카드 뭉치)을 구축
 - 사용자에게 카드를 내어줌

설계: 카드 클래스

- card_tests.py
- test_create()가 중요!
 - suit, rank가 제대로 들어갔는지 테스트
- __eq__()는 이해가 안 되면 무시

```
1 import unittest
2
3 from card import Card
4
5
6 class CardTest(unittest.TestCase):
7     def test_create(self):
8         suit = 'Hearts'
9         rank = 'Ace'
10        card1 = Card(suit, rank)
11        self.assertEqual((suit, rank), card1.get_value())
12
13    def test__eq__(self):
14        card1 = Card('Spades', 'Queen')
15        card2 = Card('Spades', 'Queen')
16        self.assertEqual(card1, card2)
17        card3 = Card('Hearts', 'Queen')
18        self.assertNotEqual(card1, card3)
19
20
21 if __name__ == '__main__':
22     unittest.main(verbosity=2)
```

설계: 딜러 클래스(일부)

- 덱을 제대로 생성했는가?
- 카드를 제대로 내어주는가?
- 점수 계산을 제대로 하는가?

```
21 ▶
22
23
24
25
26 ▶
27
28
29
30
31
32 ▶
33
34
35
36
37
```

```
def test__create_deck(self):
    dealer = Dealer()
    deck = dealer._create_deck()
    self.assertEqual(self.deck, deck)

def test_get_card(self):
    dealer = Dealer()
    for index in range(0, len(self.deck)):
        hand = dealer.get_card()
        self.assertIn(hand, self.deck)

def test_get_values_k10(self):
    hands = [Card('Hearts', 'King'),
             Card('Spades', '10')]
    dealer = Dealer()
    self.assertEqual(20, dealer.get_values(hands))
```

구현: 카드 클래스 - card.py

- `__init__()`
 - 인자로 들어온 `suit`, `rank`를 멤버 변수 `suit`, `rank`에 저장
- `get_value()`
 - 멤버 변수 `suit`, `rank`를 반환
 - 팁) `return var1, var2`
 - 2개가 같이 반환됨 (`tuple()`)

```
1 class Card(object):
2     def __eq__(self, other):
3         if (self.suit == other.suit) and (self.rank == other.rank):
4             return True
5         return False
6
7     def __repr__(self):
8         return '{0}, {1}'.format(self.suit, self.rank)
9
10    def __init__(self, suit, rank):
11        pass
12
13    def get_value(self):
14        pass
```

구현: 딜러 클래스 - dealer.py

- get_card()
 - 멤버변수 deck에서 하나를 뽑아서 반환
 - Tip) var_list.pop()은 리스트의 마지막 원소를 꺼내줌
- get_values()
 - list of card를 인자로 받음
 - 인자를 읽어가며 점수를 계산
 - 알고리즘!!) Ace는 1이 될 수도, 11이 될 수도 있음. 플레이어에게 유리하게 계산해야 함.

```
1 import random
2
3 from card import Card
4
5 SUIT = ['Spades', 'Diamonds', 'Hearts', 'Clubs']
6 RANK = ['Ace',
7         '2', '3', '4', '5', '6', '7', '8', '9', '10',
8         'Jack', 'Queen', 'King']
9 VALUE = {'Ace': 11,
10         '2': 2, '3': 3, '4': 4, '5': 5, '6': 6,
11         '7': 7, '8': 8, '9': 9, '10': 10,
12         'Jack': 10, 'Queen': 10, 'King': 10}
13
14
15 class Dealer(object):
16     def __init__(self):
17         self._create_deck()
18         random.shuffle(self.deck)
19
20     def _create_deck(self):
21         self.deck = list()
22         for suit in SUIT:
23             for rank in RANK:
24                 self.deck.append(Card(suit, rank))
25         return self.deck
26
27     def get_card(self):
28         pass
29
30     def get_values(self, cards):
31         pass
```

구현: 모든 테스트가 통과된 후 main()

- 혼자서 노는 블랙잭이 완성됨

```
-----+ 당신의 패 -----+
카드를 더 받을까요? (y/n): y
-----+ 당신의 패 -----+
Clubs Jack
카드를 더 받을까요? (y/n): y
-----+ 당신의 패 -----+
Clubs Jack
Spades 5
카드를 더 받을까요? (y/n): y
-----+ 당신의 패 -----+
Clubs Jack
Spades 5
Clubs 6
카드를 더 받을까요? (y/n): n
-----+ 당신의 점수 -----+
총점: 21
```

```
1 from dealer import Dealer
2
3
4 def main():
5     dealer = Dealer()
6     user_input = None
7     hands = list()
8
9     while user_input != 'n':
10        print('-----+ 당신의 패 -----+')
11        for hand in hands:
12            print('{0} {1}'.format(hand.suit, hand.rank))
13        user_input = input('카드를 더 받을까요? (y/n): ')
14        user_input = user_input.strip()
15        user_input = user_input.lower()
16        if user_input == 'y':
17            hands.append(dealer.get_card())
18
19        print('-----+ 당신의 점수 -----+')
20        print('총점: {0}'.format(dealer.get_values(hands)))
21
22
23 if __name__ == '__main__':
24     main()
```

기타 유용한 정보

실습 숙제 제출

- 숙제 제출 기한: 2018. 09. 19. 23:59:59
 - 실습 전 날
- 파일 제목: AL_학번_이름_02.zip
 - 파일 제목 다를 시 채점 안 합니다.
 - .egg 안 됨!

실습 숙제 제출할 것

- 2가지 파일을 제출
 - AL_학번_이름_숙제번호.zip
 - Pycharm을 사용했을 경우 Project 디렉터리에 .idea, venv 같은 디렉터리는 제외
 - Jupyter + IPython을 사용했을 경우 'File - Download as' 에서 .py 다운로드 가능
 - AL_학번_이름_숙제번호.pdf
 - 보고서는 무조건 .pdf
 - .hwp, .doc 등 채점 안 함

실습 보고서에 들어가야 할 것

- 목표(할 일)
- 과제를 해결하는 방법
 - 알아야 할 것
- 과제를 해결한 방법
 - 주요 소스코드: 굳이 소스코드 전체를 붙일 필요는 없음
- 결과화면
 - 결과화면 설명(해석)
- 보고서는 기본적으로 '내가 숙제를 했음'을 보이는 것
 - 지나치게 대충 작성하면 의심하게 됨

출석부 및 실습 점수가 궁금하다면?

- 출석부 및 실습 채점표
 - 수업 시작 후 30분까지 지각, 이후 결석
 - 실습 딜레이 1일당: -2점

질문이 생기면?

- 이름: 문현수
- 전공: 통신및보안
- 과정: 석박사통합과정 8학기
- 연구실: 데이터네트워크연구실(공5633)
- 메일: munhyunsu@cs-cnu.org
- 알고리즘은 함께 해결해가는 과목이므로 과감하게 연락
- 이메일로 처리가 안 되는 급한일: 문자/전화 등