Step 9: Object-Oriented Programming

Instructor: Eunil Park (eunilpark@skku.edu)

SUNG KYUN KWAN UNIVERSITY

Last Week

SUNG KYUN KWAN UNIVERSITY

Key Points

1. 동적 프로그래밍
   I. 피보나치 수열
   II. 파스칼삼각형
   III. 기타 예시들

Contents

SUNG KYUN KWAN UNIVERSITY

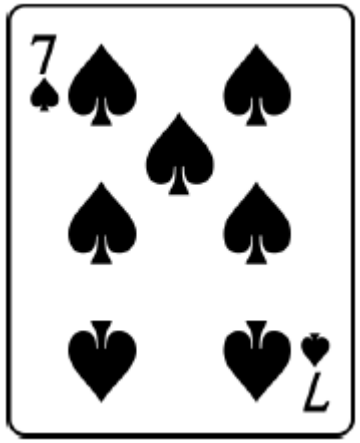
Today's Schedule

1. 객체와 클래스
2. 클래스 정의, 객체 생성
3. 생성메소드: 속성 및 활용
4. 메소드 정의 및 호출
5. 클래스 속성
6. 객체캡슐화
7. 클래스 소속 메소드

SUNG KYUN KWAN UNIVERSITY

객체와 클래스

- **객체(object)**: 우리의 사고 대상이 되는 모든 데이터/대상
- **속성(attribute)**: 객체 고유의 특성 또는 상태
- **행위(behavior)**: 객체가 고유로 갖고 있는 기능 또는 능력



- 객체(object): 놀이카드
- 속성(attribute): 특성 – '스페이드', '7', 상태 – '펼침'
- 행위(behavior): '카드 뒤집기'

SUNG KYUN KWAN UNIVERSITY

객체지향 프로그래밍

- 객체지향 프로그래밍(Object-Oriented Programming, OOP)
  - 객체를 중심으로 사고하여 프로그램을 작성하는 패러다임
  - 데이터를 모두 객체로 취급
  - 객체끼리 메시지를 주고받으며 프로그램이 작동

객체 (object)	
속성 (attribute)	행위 (behavior)
특성 / 상태	기능 / 능력
속성변수 (attribute)	메소드 (method)
변수	함수, 프로시저

속성변수	메소드

`suit = "Spade"`
`rank = "7"`
`face_up = True`
	`def flip():`
	`  face_up = not face_up`

02. 클래스 정의, 객체 생성

SUNG KYUN KWAN UNIVERSITY

객체, 클래스

- 객체(object): 메모리에 거주하는 실물(instance)
- 클래스(class): 실물 객체를 만들어 내는 일종의 형판(template)
  - 객체의 모든 것(속성과 행위)을 정해주는 청사진(blueprint)
  - 클래스 하나로 실물 객체를 몇 개이든 만들어 낼 수 있음

클래스 (class) 청사진 (blueprint) 형판 (template)	객체 (object) 실물 (instance)
1	n

02. 클래스 정의, 객체 생성

SUNG KYUN KWAN UNIVERSITY

클래스의 정의: class

```
class Card:
    pass
```

← pass는 아무 일도 하지 않는 명령

Card()를 호출하면 Card 클래스에서 정의한 실물 객체 생성

```
>> card = Card()
```

생성된 객체에 접근할 수 있는 변수 “card”를 통해 객체에 접근할 수 있음

03. 생성메소드: 속성 및 활용

SUNG KYUN KWAN UNIVERSITY

생성메소드

- 생성메소드(constructor)
  - 객체를 생성할 때 저절로 호출되어 실행
  - 새로 생성하는 객체 속성변수의 초기값설정
  - __init__ 메소드 사용

```
1  class Card:
2      def __init__(self, suit, rank, face_up):
3          self.suit = suit
4          self.rank = rank
5          self.face_up = face_up
```

클래스 내부에 정의하는 모든 메소드의 첫 파라미터는 "self"
"self"는 생성될 객체 자신을 가리키는 이름임

속성변수

속성변수는 클래스 내부 전역에서 사용할 수 있음

03. 생성메소드: 속성 및 활용

SUNG KYUN KWAN UNIVERSITY

객체 생성

>> card1 = Card(“Spade”, “7”, True)

>> card2 = Card(“Heart”, “Queen”, True)

⇒ 두개의 Card 객체인 “card1”과 “card2”가 각각 생성됨

객체 속성의 수정이 가능

>> card1.rank

‘7’

>> card1.rank = “Ace”

>> card1.rank

‘Ace’

이와 같이 객체 내부의 속성변수를 외부에 직접 노출 시키는 것은 보안상 바람직하지 않음

SUNG KYUN KWAN UNIVERSITY

파라미터의 기본값 지정

```
1 class Card:
2     def __init__(self, suit, rank, face_up=True):
3         self.suit = suit
4         self.rank = rank
5         self.face_up = face_up
```

```
>> card = Card("Spade", "7")
>> card.face_up
True
 >> card = Card("Spade", "7", False)
>> card.face_up
False
```

SUNG KYUN KWAN UNIVERSITY

객체의 간판문자열

- 간판문자열
  - 객체를 실행창에서 참조하면 보여주는 미리 정해놓은 형식의 문자열
  - 예: <__main__.Card object at 0x0000019ADB689198>
- 간판문자열의 정의: __str__ 메소드를 정의

```
 1  class Card:
...     ...
10      def __str__(self):
11          if self.__face_up:
12              return self.__suit + "." + self.__rank
13          else:
14              return "xxxxx" + "." + "xx"
...     ...
```

>> card = Card(“Spade”, “7”)

>> print(card)

Spade.7

04. 메소드 정의 및 호출

SUNG KYUN KWAN UNIVERSITY

flip 메소드

- flip 메소드 정의

메소드	실행 의미
flip()	PlayingCard 객체의 face_up 속성변수 논리값을 역으로 바꾼다.

```
1  class PlayingCard:
2      def __init__(self, suit, rank, face_up=True):
3          self.suit = suit
4          self.rank = rank
5          self.face_up = face_up
6
7      def flip(self):
8          self.face_up = not self.face_up
```

True -> False
False -> True

```
>> card = Card("Spade", "7")
>> card.flip()
```

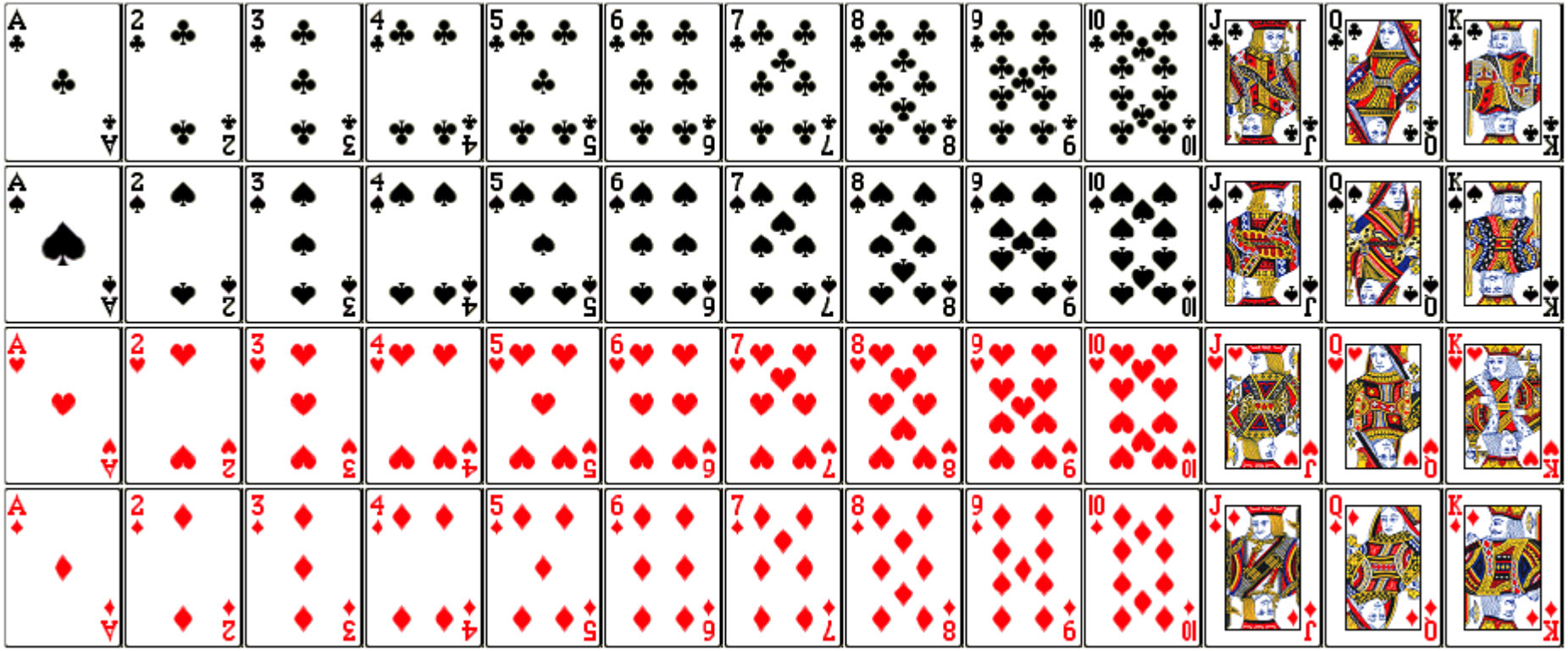
face_up: True
face_up: False

05. 클래스 속성

SUNG KYUN KWAN UNIVERSITY

클래스 속성

- 클래스 속성(class attribute): 모든 객체가 공유하는 속성
- 예: 놀이카드에서는 카드의 종류(4가지)와 계급(13가지)



05. 클래스 속성

SUNG KYUN KWAN UNIVERSITY

클래스 속성

```
1   class Card:
2       suits = ("Diamond", "Heart", "Spade", "Clover")
3       ranks = ("A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K")
4
5       def __init__(self, suit, rank, face_up=True):
6           self.suit = suit
7           self.rank = rank
8           self.face_up = face_up
9
10      def flip(self):
11          self.face_up = not self.face_up
```

튜플 같은 변경 불가능한 데이터 사용이 안전함

>> Card.suits

('Diamond', 'Heart', 'Spade', 'Clover')

>> card.suits

('Diamond', 'Heart', 'Spade', 'Clover')

>> Card.suits = ('D', 'H', 'S', 'C')

>> Card.suits

('D', 'H', 'S', 'C')

>> card.suits

('D', 'H', 'S', 'C')

>> card.suits = ('Diamond', 'Heart', 'Spade', 'Clover')

>> Card.suits

('D', 'H', 'S', 'C')

>> card.suits

('Diamond', 'Heart', 'Spade', 'Clover')

캡슐화

- 함수 캡슐화
  - 인수를 제공하고 결과를 받을 뿐 다른 부작용이 없어야 함
- 데이터 캡슐화(data encapsulation)
  - 객체의 속성변수를 외부에서 수정할 수 없게 해야 함
- 비공개(private) 속성변수
  - 속성변수 이름 앞에 '__'를 붙이면 됨
  - 클래스 내부에서는 접근 가능하지만 외부에서는 직접 접근이 불가능함

06. 객체캡슐화

SUNG KYUN KWAN UNIVERSITY

```
1  class Card:
2      __suits = ("Diamond", "Heart", "Spade", "Clover")
3      __ranks = ("A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K")
4
5      def __init__(self, suit, rank, face_up=True):
6          self.__suit = suit
7          self.__rank = rank
8          self.__face_up = face_up
9
10     def flip(self):
11         self.__face_up = not self.__face_up
```

```
>>> Card.__suits
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: type object 'Card' has no attribute '__suits'
>>> Card.__ranks
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: type object 'Card' has no attribute '__ranks'
```

```
>>> card = Card("Spade", "7")
>>> card.__suit
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Card' object has no attribute '__suit'
>>> card.__rank
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Card' object has no attribute '__rank'
>>> card.__face_up
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Card' object has no attribute '__face_up'
```

속성변수가 모두 차단됨!

SUNG KYUN KWAN UNIVERSITY

공개용 메소드 사용

```
1   class Card:
…       …
13      def suit(self):
14          return self.__suit
15
16      def rank(self):
17          return self.__rank
18
19      def face_up(self):
20          return self.__face_up
```

```
>>> card = Card("Spade", "7")
>>> card.suit()
'Spade'
>>> card.rank()
'7'
>>> card.face_up()
True
>>> card.flip()
>>> card.face_up()
False
>>> card.flip()
>>> card.face_up()
True
```

SUNG KYUN KWAN UNIVERSITY

프로퍼티 장식자

- 프로퍼티 장식자(@property)
  - 비공개 속성변수 참조를 메소드 호출과 구별할 수 있게 해줌
  - ()를 붙이지 않고 속성변수 참조가 가능
  - 예: card.suit() -> card.suit

```
1   class Card:
...     ...
12      @property
13      def suit(self):
14          return self.__suit
15
16      @property
17      def rank(self):
18          return self.__rank
19
20      @property
21      def face_up(self):
22          return self.__face_up
```

```
>>> card = Card("Spade", "7")
>>> card.suit
'Spade'
>>> card.rank
'7'
```

```
>>> card.suit()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object is not callable
>>> card.rank()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object is not callable
>>> card.face_up()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'bool' object is not callable
```

SUNG KYUN KWAN UNIVERSITY

캡슐화 정리

- 객체지향 프로그래밍에서 캡슐화를 잘 하는 것이 필요
  - 클래스를 만들 때 외부에 공개 하기로 정한 메소드만 공개하고 나머지는 모두 비공개로 설정
  - 이렇게 함으로써 외부 사용자는 공개된 메소드 호출을 통해서만 객체를 사용할 수 있음
  - 즉, 객체들끼리 대화와 소통은 메소드를 주고 받으면서 이루어지도록 함으로써, 객체 내부의 사유 데이터는 보호하면서 객체를 사용하게 하자는 것임

- (참고) 클래스 구현에서 네이밍 스타일 참고
  - 속성변수: 주로 데이터이므로 명사로 명명
  - 메소드: 행위를 주로 나타내므로 동사로 명명

07. 클래스 소속 메소드

SUNG KYUN KWAN UNIVERSITY

정적 메소드

- 정적 메소드(static method)
  - 클래스의 고유 기능을 정의하기 위한 메소드
  - 클래스 소속 메소드
  - 메소드 정의 앞에 “@staticmethod” 장식을 붙임
  - 객체에 속해 있지 않기 때문에 self 파라미터가 필요 없음
  - 객체가 없어도 클래스 이름으로 호출 가능. 그 외에는 일반 메소드와 호출하는 방식과 같음

```python
class Rectangle:
    count = 0  # 클래스 변수

    def __init__(self, width, height):
        self.width = width
        self.height = height
        Rectangle.count += 1

    # 인스턴스 메서드
    def calcArea(self):
        area = self.width * self.height
        return area

    # 정적 메서드
    @staticmethod
    def isSquare(rectWidth, rectHeight):
        return rectWidth == rectHeight

    # 클래스 메서드
    @classmethod
    def printCount(cls):
        print(cls.count)


# 테스트
square = Rectangle.isSquare(5, 5)
print(square)   # True

rect1 = Rectangle(5, 5)
rect2 = Rectangle(2, 5)
rect1.printCount()  # 2
```

SUNG KYUN KWAN UNIVERSITY

Docstring

- 코드의 가독성을 높이기 위해 코드를 설명하는 문서를 코드 내부에 삽입
  - 주석(comment): #
  - Python의 Docstring (문서 문자열)
- Docstring
  - 모듈, 함수, 클래스, 메소드 정의의 맨 앞에 기술하여 해당 정의를 설명.
  - """로 둘러싸서 표현
  - 따로 언급하지 않아도 해당 정의의 속성으로 귀속되어 __doc__ 라는 이름으로 지정됨

```
>>> card = Card("Spade", "7")
>>> print(card.__doc__)
defines Card class
>>> print(card.__init__.__doc__)
initializes a playing card object
      arguments:
      suit -- must be in suits
      rank -- must be in ranks
      face_up -- True or False (defaut True)

>>> print(card.__str__.__doc__)
returns its string representation
>>> print(card.flip.__doc__)
flips itself
```

07. 클래스 소속 메소드

SUNG KYUN KWAN UNIVERSITY

```
1  class Card:
2      """defines Card class"""
3      __suits = ("Diamond", "Heart", "Spade", "Clover")
4      __ranks = ("A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K")
5
6      def __init__(self, suit, rank, face_up=True):
7          """initializes a playing card object
8          arguments:
9          suit -- must be in suits
10         rank -- must be in ranks
11         face_up -- True or False (defaut True)
12         """
13         self.__suit = suit
14         self.__rank = rank
15         self.__face_up = face_up
16
17     def __str__(self):
18         """returns its string representation"""
19         if self.__face_up:
20             return self.__suit + "." + self.__rank
21         else:
22             return "xxxxx" + "." + "xx"
23
```

```
>>> card = Card("Spade", "7")
>>> print(card.__doc__)
defines Card class
>>> print(card.__init__.__doc__)
initializes a playing card object
        arguments:
        suit -- must be in suits
        rank -- must be in ranks
        face_up -- True or False (defaut True)

>>> print(card.__str__.__doc__)
returns its string representation
```

07. 클래스 소속 메소드

SUNG KYUN KWAN UNIVERSITY

```
24      def flip(self):
25          """flips itself"""
26          self.__face_up = not self.__face_up
27
28      @property
29      def suit(self):
30          """returns its suit value"""
31          return self.__suit
32
33      @property
34      def rank(self):
35          """returns its rank value"""
36          return self.__rank
37
38      @property
39      def face_up(self):
40          """returns its face_up value"""
41          return self.__face_up
```

```
>>> card = Card("Spade", "7")
>>> print(card.__doc__)
defines Card class
>>> print(card.__init__.__doc__)
initializes a playing card object
        arguments:
        suit -- must be in suits
        rank -- must be in ranks
        face_up -- True or False (defaut True)

>>> print(card.__str__.__doc__)
returns its string representation
>>> print(card.flip.__doc__)
flips itself
```

Today's Lessons!

SUNG KYUN KWAN UNIVERSITY

Summary

1. 객체와 클래스
2. 클래스 정의, 객체 생성
3. 생성메소드: 속성 및 활용
4. 메소드 정의 및 호출
5. 클래스 속성
6. 객체캡슐화
7. 클래스 소속 메소드

Thanks

Step 9: Object-Oriented Programming
Instructor: Eunil Park (eunilpark@skku.edu)