

# CS101 - Objects: Creation and Attributes

## Lecture 9

School of Computing  
KAIST

## Last week we learned

- Files
  - ▶ Reading from a file
  - ▶ Writing to a file
- break
- continue

## Last week we learned

- Files
  - ▶ Reading from a file
  - ▶ Writing to a file
- break
- continue

## This week we will learn

- Objects
  - ▶ Object creation
  - ▶ Object attributes

# Blackjack

There are 52 cards.

Each card has a face and a suit.



- The suits
  - ▶ clubs
  - ▶ spades
  - ▶ hearts
  - ▶ diamonds
- The faces
  - ▶ 2
  - ▶ 3
  - ▶ ...
  - ▶ 10
  - ▶ Jack
  - ▶ Queen
  - ▶ King
  - ▶ Ace

# Blackjack



The value of a card is the number for a number card, 11 for an Ace, and 10 for Jack, Queen, and King.

We can represent cards as a tuple (face, suit, value).

If `card` is a card, then `card[0]` is the face, `card[1]` is the suit, and `card[2]` is the value.

## Cards as tuples

Computing the value of a hand:

```
def hand_value(hand):  
    total = 0  
    for card in hand:  
        total += card[2]  
    return total
```

Printing a card nicely:

```
def card_string(card):  
    article = "a "  
    if card[0] in [8, "Ace"]:  
        article = "an "  
    return article + str(card[0]) + " of " + card[1]
```

# Cards as tuples

Computing the value of a hand:

```
def hand_value(hand):  
    total = 0  
    for card in hand:  
        total += card[2]  
    return total
```

Printing a card nicely:

```
def card_string(card):  
    article = "a "  
    if card[0] in [8, "Ace"]:  
        article = "an "  
    return article + str(card[0]) + " of " + card[1]
```

Easy to make mistakes

- What does `card[2]` mean?
- What if somebody creates a card (`"Ace", "Spades", 5`)?

# Cards as objects

Let us define a new object type with attributes for face, suit, and value:

```
class Card(object) :  
    """A Blackjack card."""  
    pass  
  
card = Card()           # Create Card object  
card.face = "Ace"      # Set attributes of the card  
card.suit = "Spades"  
card.value = 11
```



# Cards as objects

Let us define a new object type with attributes for face, suit, and value:

```
class Card(object):  
    """A Blackjack card."""  
    pass
```

```
card = Card()           # Create Card object  
card.face = "Ace"      # Set attributes of the card  
card.suit = "Spades"  
card.value = 11
```

card has a user-defined type:

```
>>> type(card)  
<class `__main__.Card'>
```

# Cards as objects

Computing the value of a hand:

```
def hand_value(hand):  
    total = 0  
    for card in hand:  
        total += card.value  
    return total
```

Printing a card nicely:

```
def card_string(card):  
    article = "a "  
    if card.face in [8, "Ace"]:  
        article = "an "  
    return article+str(card.face)+" of "+card.suit
```

# Cards as objects

Computing the value of a hand:

```
def hand_value(hand):  
    total = 0  
    for card in hand:  
        total += card.value  
    return total
```

Printing a card nicely:

```
def card_string(card):  
    article = "a "  
    if card.face in [8, "Ace"]:  
        article = "an "  
    return article+str(card.face)+" of "+card.suit
```

Getting rid of mistakes

- ~~What does `card[2]` mean?~~
- ~~What if somebody creates a card ("Ace", "Spades", 5)?~~

## Two or more cards

We can create many cards via Card class

```
card1 = Card()
card1.face = "Ace"
card1.suit = "Spades"
card1.value = 11
card2 = Card()
card2.face = 2
card2.suit = "Clubs"
card2.value = 2

...

>>> print(card_string(card1))
an Ace of Spades
>>> print(card_string(card2))
a 2 of Clubs
```

# Objects are mutable

There is one big difference between tuples and Card objects:  
Card objects are mutable:

```
card = Card()  
card.face = "Ace"  
card.suit = "Spades"  
card.value = 11  
# ... AND LATER ...  
card.suit = "Hearts"
```

# Journey of Chicken



An animation by Jeong-eun Yu and Geum-hyeon Song (2010 Freshmen).

# Journey of Chicken



An animation by Jeong-eun Yu and Geum-hyeon Song (2010 Freshmen).

Three Layer objects: hen, chick1, chick2 (all chickens).

Each chicken has body, wing, eye, and beak.

The hen also has two red dots on the head.



An animation by Jeong-eun Yu and Geum-hyeon Song (2010 Freshmen).

Three Layer objects: hen, chick1, chick2 (all chickens).

Each chicken has body, wing, eye, and beak.

The hen also has two red dots on the head.

The hen and the chicks are exactly the same. The hen is larger and white.



The simplest method to make similar objects is to write the code once, and copy & paste it (with the necessary modifications).

The simplest method to make similar objects is to write the code once, and copy & paste it (with the necessary modifications).

Disadvantage: When you find a bug, you have to debug all copies of the code. It is not easy to change the appearance of all the chickens at once.

The simplest method to make similar objects is to write the code once, and copy & paste it (with the necessary modifications).

Disadvantage: When you find a bug, you have to debug all copies of the code. It is not easy to change the appearance of all the chickens at once.

Let's try to implement the chicken as an object:

```
class Chicken (object) :  
    """Graphic representation of a chicken."""  
pass
```

The simplest method to make similar objects is to write the code once, and copy & paste it (with the necessary modifications).

Disadvantage: When you find a bug, you have to debug all copies of the code. It is not easy to change the appearance of all the chickens at once.

Let's try to implement the chicken as an object:

```
class Chicken (object) :  
    """Graphic representation of a chicken."""  
    pass
```

Our chicken will have attributes layer, body, wing, eye, and beak.

# Chicken objects

The function *make\_chicken* creates a chicken object, positioned at (0, 0).

# Chicken objects

The function *make\_chicken* creates a chicken object, positioned at (0, 0).

```
def make_chicken(hen = False):  
    layer = Layer()  
    if hen:  
        body = Ellipse(70, 80)  
        body.setFillColor("white")  
    else:  
        body = Ellipse(40, 50)  
        body.setFillColor("yellow")  
        body.move(0, 10)  
    body.setBorderColor("yellow")  
    body.setDepth(20)  
    layer.add(body)  
    # similar for wing, eye, beak, dots
```

# Chicken objects

Finally we create and return the Chicken object:

# Chicken objects

Finally we create and return the Chicken object:

```
def make_chicken(hen = False) :  
    # ... see previous page  
  
    ch = Chicken()  
    ch.layer = layer  
    ch.body = body  
    ch.wing = wing  
    ch.eye = eye  
  
    # return the Chicken object  
return ch
```



# Using chickens

We use Chicken objects by accessing their attributes:

We use Chicken objects by accessing their attributes:

```
hen = make_chicken(True)
chick1 = make_chicken()
chick1.layer.move(120, 0)

herd = Layer()
herd.add(hen.layer)
herd.add(chick1.layer)
herd.move(600, 200)

chick2 = make_chicken()
chick2.layer.move(800, 200)
```